Mathematics for Computer Science

revised June 15, 2010, 408 minutes

Eric Lehman
Google Inc.

F Tom Leighton
Massachusets Institute of Technology

Albert R Meyer
Massachusets Institute of Technology

Contents

Part I

Proofs

Mathematical Proofs

A proof is a method of establishing truth. What constitutes a proof differs among fields.

- *Legal* truth is decided by a jury based on allowable evidence presented at trial.
- *Authoritative* truth is specified by a trusted person or organization.
- *Scientific* truth[1] is confirmed by experiment.
- *Probable* truth is established by statistical analysis of sample data.
- *Philosophical* proof involves careful exposition and persuasion typically based on a series of small, plausible arguments. The best example begins with "Cogito ergo sum," a Latin sentence that translates as "I think, therefore I am." It comes from the beginning of a 17th century essay by the mathematician/philospher, René Descartes, and it is one of the most famous quotes in the world: do a web search on the phrase and you will be flooded with hits.

  Deducing your existence from the fact that you're thinking about your existence is a pretty cool and persuasive-sounding first axiom. However, with just a few more lines of argument in this vein, Descartes goes on to conclude that there is an infinitely beneficent God. Whether or not you believe in a beneficent God, you'll probably agree that any very short proof of God's existence is bound to be far-fetched. So even in masterful hands, this approach is not reliable.

Mathematics also has a specific notion of "proof."

Definition. A *formal proof* of a *proposition* is a chain of *logical deductions* leading to the proposition from a base set of *axioms*.

The three key ideas in this definition are highlighted: proposition, logical deduction, and axiom. These three ideas are explained in Chapter 1, and Chapter 2 describes some basic ways of organizing proofs.

0.0.1 Problems

Class Problems

Problem 0.1.
Identify exactly where the bugs are in each of the following bogus proofs.[2]

[1] Actually, only scientific *falsehood* can be demonstrated by an experiment —when the experiment fails to behave as predicted. But no amount of experiment can confirm that the *next* experiment won't fail. For this reason, scientists rarely speak of truth, but rather of *theories* that accurately predict past, and anticipated future, experiments.

[2] From Stueben, Michael and Diane Sandford. *Twenty Years Before the Blackboard*, Mathematical Association of America, ©1998.

(a) Bogus Claim: $1/8 > 1/4$.

Bogus proof.

$$\begin{aligned} 3 &> 2 \\ 3\log_{10}(1/2) &> 2\log_{10}(1/2) \\ \log_{10}(1/2)^3 &> \log_{10}(1/2)^2 \\ (1/2)^3 &> (1/2)^2, \end{aligned}$$

and the claim now follows by the rules for multiplying fractions. ■

(b) *Bogus proof*: $1¢ = \$0.01 = (\$0.1)^2 = (10¢)^2 = 100¢ = \$1$. ■

(c) Bogus Claim: If a and b are two equal real numbers, then $a = 0$.

Bogus proof.

$$\begin{aligned} a &= b \\ a^2 &= ab \\ a^2 - b^2 &= ab - b^2 \\ (a-b)(a+b) &= (a-b)b \\ a+b &= b \\ a &= 0. \end{aligned}$$

■

Problem 0.2.
It's a fact that the Arithmetic Mean is at least as large the Geometric Mean, namely,

$$\frac{a+b}{2} \geq \sqrt{ab}$$

for all nonnegative real numbers a and b. But there's something objectionable about the following proof of this fact. What's the objection, and how would you fix it?

Bogus proof.

$$\begin{aligned} \frac{a+b}{2} &\overset{?}{\geq} \sqrt{ab}, && \text{so} \\ a+b &\overset{?}{\geq} 2\sqrt{ab}, && \text{so} \\ a^2 + 2ab + b^2 &\overset{?}{\geq} 4ab, && \text{so} \\ a^2 - 2ab + b^2 &\overset{?}{\geq} 0, && \text{so} \\ (a-b)^2 &\geq 0 && \text{which we know is true.} \end{aligned}$$

The last statement is true because $a-b$ is a real number, and the square of a real number is never negative. This proves the claim. ■

Problem 0.3.
Albert announces that he plans a surprise 6.042 quiz next week. His students wonder if the quiz could be next Friday. The students realize that it obviously cannot, because if it hadn't been given before Friday, everyone would know that there was only Friday left on which to give it, so it wouldn't be a surprise any more.

So the students ask whether Albert could give the surprise quiz Thursday? They observe that if the quiz wasn't given *before* Thursday, it would have to be given *on* the Thursday, since they already know it can't be given on Friday. But having figured that out, it wouldn't be a surprise if the quiz was on Thursday either. Similarly, the students reason that the quiz can't be on Wednesday, Tuesday, or Monday. Namely, it's impossible for Albert to give a surprise quiz next week. All the students now relax, having concluded that Albert must have been bluffing.

And since no one expects the quiz, that's why, when Albert gives it on Tuesday next week, it really is a surprise!

What do you think is wrong with the students' reasoning?

0.1 Propositions

Definition. A *proposition* is a statement that is either true or false.

This definition sounds very general, but it does exclude sentences such as, "Wherefore art thou Romeo?" and "Give me an A!". But not all propositions are mathematical. For example, "Albert's wife's name is 'Irene' " happens to be true, and could be proved with legal documents and testimony of their children, but it's not a mathematical statement.

Mathematically meaningful propositions must be about well-defined mathematical objects like numbers, sets, functions, relations, etc., and they must be stated using mathematically precise language. We can illustrate this with a few examples.

Proposition 0.1.1. *2 + 3 = 5.*

This proposition is true.

A *prime* is an integer greater than one that is not divisible by any integer greater than 1 besides itself, for example, 2, 3, 5, 7, 11, ...

Proposition 0.1.2. *For every nonnegative integer, n, the value of $n^2 + n + 41$ is prime.*

Let's try some numerical experimentation to check this proposition. Let [3]

$$p(n) ::= n^2 + n + 41. \tag{1}$$

[3] The symbol ::= means "equal by definition." It's always ok to simply write "=" instead of ::=, but reminding the reader that an equality holds by definition can be helpful.

We begin with $p(0) = 41$ which is prime. $p(1) = 43$ which is prime. $p(2) = 47$ which is prime. $p(3) = 53$ which is prime. ... $p(20) = 461$ which is prime. Hmmm, starts to look like a plausible claim. In fact we can keep checking through $n = 39$ and confirm that $p(39) = 1601$ is prime.

But $p(40) = 40^2 + 40 + 41 = 41 \cdot 41$, which is not prime. So it's not true that the expression is prime *for all* nonnegative integers. The point is that in general you can't check a claim about an infinite set by checking a finite set of its elements, no matter how large the finite set.

By the way, propositions like this about *all* numbers or other things are so common that there is a special notation for it. With this notation, Proposition 1.5.1 would be

$$\forall n \in \mathbb{N}.\ p(n) \text{ is prime.} \tag{2}$$

Here the symbol $\forall$ is read "for all". The symbol $\mathbb{N}$ stands for the set of *nonnegative integers*, namely, 0, 1, 2, 3, ... (ask your TA for the complete list). The symbol "$\in$" is read as "is a member of" or simply as "is in". The period after the $\mathbb{N}$ is just a separator between phrases.

Here are two even more extreme examples:

Proposition 0.1.3. *$a^4 + b^4 + c^4 = d^4$ has no solution when a, b, c, d are positive integers.*

Euler (pronounced "oiler") conjectured this in 1769. But the proposition was proven false 218 years later by Noam Elkies at a liberal arts school up Mass Ave. The solution he found was $a = 95800, b = 217519, c = 414560, d = 422481$.

In logical notation, Proposition 1.5.2 could be written,

$$\forall a \in \mathbb{Z}^+\ \forall b \in \mathbb{Z}^+\ \forall c \in \mathbb{Z}^+\ \forall d \in \mathbb{Z}^+.\ a^4 + b^4 + c^4 \neq d^4.$$

Here, $\mathbb{Z}^+$ is a symbol for the positive integers. Strings of $\forall$'s like this are usually abbreviated for easier reading:

$$\forall a, b, c, d \in \mathbb{Z}^+.\ a^4 + b^4 + c^4 \neq d^4.$$

Proposition 0.1.4. *$313(x^3 + y^3) = z^3$ has no solution when $x, y, z \in \mathbb{Z}^+$.*

This proposition is also false, but the smallest counterexample has more than 1000 digits!

Proposition 0.1.5. *Every map can be colored with 4 colors so that adjacent[4] regions have different colors.*

This proposition is true and is known as the "*Four-Color Theorem*". However, there have been many incorrect proofs, including one that stood for 10 years in the late 19th century before the mistake was found. An extremely laborious proof was finally found in 1976 by mathematicians Appel and Haken, who used a complex computer program to categorize the four-colorable maps; the program left a couple

[4]Two regions are adjacent only when they share a boundary segment of positive length. They are not considered to be adjacent if their boundaries meet only at a few points.

of thousand maps uncategorized, and these were checked by hand by Haken and his assistants—including his 15-year-old daughter. There was a lot of debate about whether this was a legitimate proof: the proof was too big to be checked without a computer, and no one could guarantee that the computer calculated correctly, nor did anyone have the energy to recheck the four-colorings of thousands of maps that were done by hand. Finally, about five years ago, a mostly intelligible proof of the Four-Color Theorem was found, though a computer is still needed to check colorability of several hundred special maps (see
`http://www.math.gatech.edu/~thomas/FC/fourcolor.html`). [5]

Proposition 0.1.6 (Goldbach)**.** *Every even integer greater than 2 is the sum of two primes.*

No one knows whether this proposition is true or false. It is known as *Goldbach's Conjecture*, and dates back to 1742.

For a computer scientist, some of the most important things to prove are the "correctness" programs and systems —whether a program or system does what it's supposed to. Programs are notoriously buggy, and there's a growing community of researchers and practitioners trying to find ways to prove program correctness. These efforts have been successful enough in the case of CPU chips that they are now routinely used by leading chip manufacturers to prove chip correctness and avoid mistakes like the notorious Intel division bug in the 1990's.

Developing mathematical methods to verify programs and systems remains an active research area. We'll consider some of these methods later in the course.

0.2 Predicates

A *predicate* is a proposition whose truth depends on the value of one or more variables. Most of the propostions above were defined in terms of predicates. For example,

$$\text{“}n \text{ is a perfect square”}$$

is a predicate whose truth depends on the value of n. The predicate is true for $n = 4$ since four is a perfect square, but false for $n = 5$ since five is not a perfect square.

Like other propositions, predicates are often named with a letter. Furthermore, a function-like notation is used to denote a predicate supplied with specific variable values. For example, we might name our earlier predicate P:

$$P(n) ::= \text{“}n \text{ is a perfect square”}$$

Now $P(4)$ is true, and $P(5)$ is false.

[5]The story of the Four-Color Proof is told in a well-reviewed popular (non-technical) book: "Four Colors Suffice. How the Map Problem was Solved." *Robin Wilson*. Princeton Univ. Press, 2003, 276pp. ISBN 0-691-11533-8.

This notation for predicates is confusingly similar to ordinary function notation. If P is a predicate, then $P(n)$ is either *true* or *false*, depending on the value of n. On the other hand, if p is an ordinary function, like $n^2 + 1$, then $p(n)$ is a *numerical quantity*. **Don't confuse these two!**

0.3 The Axiomatic Method

The standard procedure for establishing truth in mathematics was invented by Euclid, a mathematician working in Alexandria, Egypt around 300 BC. His idea was to begin with five *assumptions* about geometry, which seemed undeniable based on direct experience. (For example, "There is a straight line segment between every pair of points.) Propositions like these that are simply accepted as true are called *axioms*.

Starting from these axioms, Euclid established the truth of many additional propositions by providing "proofs". A *proof* is a sequence of logical deductions from axioms and previously-proved statements that concludes with the proposition in question. You probably wrote many proofs in high school geometry class, and you'll see a lot more in this course.

There are several common terms for a proposition that has been proved. The different terms hint at the role of the proposition within a larger body of work.

- Important propositions are called *theorems*.
- A *lemma* is a preliminary proposition useful for proving later propositions.
- A *corollary* is a proposition that follows in just a few logical steps from a theorem.

The definitions are not precise. In fact, sometimes a good lemma turns out to be far more important than the theorem it was originally used to prove.

Euclid's axiom-and-proof approach, now called the *axiomatic method*, is the foundation for mathematics today. In fact, just a handful of axioms, called the axioms Zermelo-Frankel with Choice (*ZFC*), together with a few logical deduction rules, appear to be sufficient to derive essentially all of mathematics. We'll examine these in Chapter 5.

0.4 Our Axioms

The ZFC axioms are important in studying and justifying the foundations of mathematics, but for practical purposes, they are much too primitive. Proving theorems in ZFC is a little like writing programs in byte code instead of a full-fledged programming language —by one reckoning, a formal proof in ZFC that $2 + 2 = 4$ requires more than 20,000 steps! So instead of starting with ZFC, we're going to take a *huge* set of axioms as our foundation: we'll accept all familiar facts from high school math!

This will give us a quick launch, but you may find this imprecise specification of the axioms troubling at times. For example, in the midst of a proof, you may find yourself wondering, "Must I prove this little fact or can I take it as an axiom?" Feel free to ask for guidance, but really there is no absolute answer. Just be up front about what you're assuming, and don't try to evade homework and exam problems by declaring everything an axiom!

0.4.1 Logical Deductions

Logical deductions or *inference rules* are used to prove new propositions using previously proved ones.

A fundamental inference rule is *modus ponens*. This rule says that a proof of P together with a proof that P IMPLIES Q is a proof of Q.

Inference rules are sometimes written in a funny notation. For example, *modus ponens* is written:

Rule.

$$\frac{P, \quad P \text{ IMPLIES } Q}{Q}$$

When the statements above the line, called the *antecedents*, are proved, then we can consider the statement below the line, called the *conclusion* or *consequent*, to also be proved.

A key requirement of an inference rule is that it must be *sound*: any assignment of truth values that makes all the antecedents true must also make the consequent true. So if we start off with true axioms and apply sound inference rules, everything we prove will also be true.

There are many other natural, sound inference rules, for example:

Rule.

$$\frac{P \text{ IMPLIES } Q, \quad Q \text{ IMPLIES } R}{P \text{ IMPLIES } R}$$

Rule.

$$\frac{\text{NOT}(P) \text{ IMPLIES NOT}(Q)}{Q \text{ IMPLIES } P}$$

On the other hand,

Rule.

$$\frac{\text{NOT}(P) \text{ IMPLIES NOT}(Q)}{P \text{ IMPLIES } Q}$$

is not sound: if P is assigned **T** and Q is assigned **F**, then the antecedent is true and the consequent is not.

Note that a propositional inference rule is sound precisely when the conjunction (AND) of all its antecedents implies its consequent.

As with axioms, we will not be too formal about the set of legal inference rules. Each step in a proof should be clear and "logical"; in particular, you should state what previously proved facts are used to derive each new conclusion.

0.4.2 Patterns of Proof

In principle, a proof can be *any* sequence of logical deductions from axioms and previously proved statements that concludes with the proposition in question. This freedom in constructing a proof can seem overwhelming at first. How do you even *start* a proof?

Here's the good news: many proofs follow one of a handful of standard templates. Each proof has it own details, of course, but these templates at least provide you with an outline to fill in. We'll go through several of these standard patterns, pointing out the basic idea and common pitfalls and giving some examples. Many of these templates fit together; one may give you a top-level outline while others help you at the next level of detail. And we'll show you other, more sophisticated proof techniques later on.

The recipes below are very specific at times, telling you exactly which words to write down on your piece of paper. You're certainly free to say things your own way instead; we're just giving you something you *could* say so that you're never at a complete loss.

0.5 Proving an Implication

Propositions of the form "If P, then Q" are called *implications*. This implication is often rephrased as "P IMPLIES Q."

Here are some examples:

- (Quadratic Formula) If $ax^2 + bx + c = 0$ and $a \neq 0$, then
$$x = \left(-b \pm \sqrt{b^2 - 4ac}\right)/2a.$$
- (Goldbach's Conjecture) If n is an even integer greater than 2, then n is a sum of two primes.
- If $0 \leq x \leq 2$, then $-x^3 + 4x + 1 > 0$.

There are a couple of standard methods for proving an implication.

0.5.1 Method #1

In order to prove that P IMPLIES Q:

1. Write, "Assume P."
2. Show that Q logically follows.

Example

Theorem 0.5.1. *If $0 \leq x \leq 2$, then $-x^3 + 4x + 1 > 0$.*

Before we write a proof of this theorem, we have to do some scratchwork to figure out why it is true.

The inequality certainly holds for $x = 0$; then the left side is equal to 1 and $1 > 0$. As x grows, the $4x$ term (which is positive) initially seems to have greater magnitude than $-x^3$ (which is negative). For example, when $x = 1$, we have $4x = 4$, but $-x^3 = -1$ only. In fact, it looks like $-x^3$ doesn't begin to dominate until $x > 2$. So it seems the $-x^3 + 4x$ part should be nonnegative for all x between 0 and 2, which would imply that $-x^3 + 4x + 1$ is positive.

So far, so good. But we still have to replace all those "seems like" phrases with solid, logical arguments. We can get a better handle on the critical $-x^3 + 4x$ part by factoring it, which is not too hard:

$$-x^3 + 4x = x(2 - x)(2 + x)$$

Aha! For x between 0 and 2, all of the terms on the right side are nonnegative. And a product of nonnegative terms is also nonnegative. Let's organize this blizzard of observations into a clean proof.

Proof. Assume $0 \leq x \leq 2$. Then x, $2 - x$, and $2 + x$ are all nonnegative. Therefore, the product of these terms is also nonnegative. Adding 1 to this product gives a positive number, so:

$$x(2 - x)(2 + x) + 1 > 0$$

Multiplying out on the left side proves that

$$-x^3 + 4x + 1 > 0$$

as claimed. ■

There are a couple points here that apply to all proofs:

- You'll often need to do some scratchwork while you're trying to figure out the logical steps of a proof. Your scratchwork can be as disorganized as you like— full of dead-ends, strange diagrams, obscene words, whatever. But keep your scratchwork separate from your final proof, which should be clear and concise.

- Proofs typically begin with the word "Proof" and end with some sort of doohickey like □ or "q.e.d". The only purpose for these conventions is to clarify where proofs begin and end.

0.5.2 Method #2 - Prove the Contrapositive

An implication ("P IMPLIES Q") is logically equivalent to its *contrapositive*

$$\text{NOT}(Q) \text{ IMPLIES NOT}(P)$$

Proving one is as good as proving the other, and proving the contrapositive is sometimes easier than proving the original statement. If so, then you can proceed as follows:

1. Write, "We prove the contrapositive:" and then state the contrapositive.

2. Proceed as in Method #1.

Example

Theorem 0.5.2. *If r is irrational, then $\sqrt{r}$ is also irrational.*

Recall that rational numbers are equal to a ratio of integers and irrational numbers are not. So we must show that if r is *not* a ratio of integers, then $\sqrt{r}$ is also *not* a ratio of integers. That's pretty convoluted! We can eliminate both *not*'s and make the proof straightforward by considering the contrapositive instead.

Proof. We prove the contrapositive: if $\sqrt{r}$ is rational, then r is rational.

Assume that $\sqrt{r}$ is rational. Then there exist integers a and b such that:

$$\sqrt{r} = \frac{a}{b}$$

Squaring both sides gives:

$$r = \frac{a^2}{b^2}$$

Since a^2 and b^2 are integers, r is also rational. ■

0.5.3 Problems

Homework Problems

Problem 0.4.
Show that $\log_7 n$ is either an integer or irrational, where n is a positive integer. Use whatever familiar facts about integers and primes you need, but explicitly state such facts. (This problem will be graded on the clarity and simplicity of your proof. If you can't figure out how to prove it, ask the staff for help and they'll tell you how.)

0.6 Proving an "If and Only If"

Many mathematical theorems assert that two statements are logically equivalent; that is, one holds if and only if the other does. Here is an example that has been known for several thousand years:

> Two triangles have the same side lengths if and only if two side lengths and the angle between those sides are the same.

The phrase "if and only if" comes up so often that it is often abbreviated "iff".

0.6.1 Method #1: Prove Each Statement Implies the Other

The statement "P IFF Q" is equivalent to the two statements "P IMPLIES Q" and "Q IMPLIES P". So you can prove an "iff" by proving *two* implications:

1. Write, "We prove P implies Q and vice-versa."

2. Write, "First, we show P implies Q." Do this by one of the methods in Section 2.2.1.

3. Write, "Now, we show Q implies P." Again, do this by one of the methods in Section 2.2.1.

0.6.2 Method #2: Construct a Chain of Iffs

In order to prove that P is true iff Q is true:

1. Write, "We construct a chain of if-and-only-if implications."

2. Prove P is equivalent to a second statement which is equivalent to a third statement and so forth until you reach Q.

This method sometimes requires more ingenuity than the first, but the result can be a short, elegant proof.

Example

The *standard deviation* of a sequence of values $x_1, x_2, \ldots, x_n$ is defined to be:

$$\sqrt{\frac{(x_1 - \mu)^2 + (x_2 - \mu)^2 + \cdots + (x_n - \mu)^2}{n}} \tag{3}$$

where μ is the *mean* of the values:

$$\mu ::= \frac{x_1 + x_2 + \cdots + x_n}{n}$$

Theorem 0.6.1. *The standard deviation of a sequence of values $x_1, \ldots, x_n$ is zero iff all the values are equal to the mean.*

For example, the standard deviation of test scores is zero if and only if everyone scored exactly the class average.

Proof. We construct a chain of "iff" implications, starting with the statement that the standard deviation (2.1) is zero:

$$\sqrt{\frac{(x_1 - \mu)^2 + (x_2 - \mu)^2 + \cdots + (x_n - \mu)^2}{n}} = 0. \tag{4}$$

Now since zero is the only number whose square root is zero, equation (2.2) holds iff

$$(x_1 - \mu)^2 + (x_2 - \mu)^2 + \cdots + (x_n - \mu)^2 = 0. \tag{5}$$

Now squares of real numbers are always nonnegative, so every term on the left hand side of equation (2.3) is nonnegative. This means that (2.3) holds iff

Every term on the left hand side of (2.3) is zero. (6)

But a term $(x_i - \mu)^2$ is zero iff $x_i = \mu$, so (2.4) is true iff

Every x_i equals the mean.

■

0.7 Proof by Cases

Breaking a complicated proof into cases and proving each case separately is a useful, common proof strategy. Here's an amusing example.

Let's agree that given any two people, either they have met or not. If every pair of people in a group has met, we'll call the group a *club*. If every pair of people in a group has not met, we'll call it a group of *strangers*.

Theorem. *Every collection of 6 people includes a club of 3 people or a group of 3 strangers.*

Proof. The proof is by case analysis[6]. Let x denote one of the six people. There are two cases:

1. Among 5 other people besides x, at least 3 have met x.
2. Among the 5 other people, at least 3 have not met x.

Now we have to be sure that at least one of these two cases must hold,[7] but that's easy: we've split the 5 people into two groups, those who have shaken hands with x and those who have not, so one the groups must have at least half the people.

Case 1: Suppose that at least 3 people did meet x.

This case splits into two subcases:

Case 1.1: No pair among those people met each other. Then these people are a group of at least 3 strangers. So the Theorem holds in this subcase.

Case 1.2: Some pair among those people have met each other. Then that pair, together with x, form a club of 3 people. So the Theorem holds in this subcase.

[6]Describing your approach at the outset helps orient the reader.

[7]Part of a case analysis argument is showing that you've covered all the cases. Often this is obvious, because the two cases are of the form "P" and "not P". However, the situation above is not stated quite so simply.

This implies that the Theorem holds in Case 1.

Case 2: Suppose that at least 3 people did not meet x.

This case also splits into two subcases:

> **Case 2.1**: Every pair among those people met each other. Then these people are a club of at least 3 people. So the Theorem holds in this subcase.
>
> **Case 2.2:** Some pair among those people have not met each other. Then that pair, together with x, form a group of at least 3 strangers. So the Theorem holds in this subcase.

This implies that the Theorem also holds in Case 2, and therefore holds in all cases.

■

0.7.1 Problems

Class Problems

Problem 0.5.
If we raise an irrational number to an irrational power, can the result be rational? Show that it can by considering $\sqrt{2}^{\sqrt{2}}$ and arguing by cases.

Homework Problems

Problem 0.6.
For $n = 40$, the value of polynomial $p(n) ::= n^2 + n + 41$ 1.4, is not prime, as noted in Section 1.5. But we could have predicted based on general principles that no nonconstant polynomial, $q(n)$, with integer coefficients can map each nonnegative integer into a prime number. Prove it.

Hint: Let $c ::= q(0)$ be the constant term of q. Consider two cases: c is not prime, and c is prime. In the second case, note that $q(cn)$ is a multiple of c for all $n \in \mathbb{Z}$. You may assume the familiar fact that the magnitude (absolute value) of any nonconstant polynomial, $q(n)$, grows unboundedly as n grows.

0.8 Proof by Contradiction

In a *proof by contradiction* or *indirect proof*, you show that if a proposition were false, then some false fact would be true. Since a false fact can't be true, the proposition had better not be false. That is, the proposition really must be true.

Proof by contradiction is *always* a viable approach. However, as the name suggests, indirect proofs can be a little convoluted. So direct proofs are generally preferable as a matter of clarity.

Method: In order to prove a proposition P by contradiction:

1. Write, "We use proof by contradiction."

2. Write, "Suppose P is false."
3. Deduce something known to be false (a logical contradiction).
4. Write, "This is a contradiction. Therefore, P must be true."

Example

Remember that a number is *rational* if it is equal to a ratio of integers. For example, $3.5 = 7/2$ and $0.1111\cdots = 1/9$ are rational numbers. On the other hand, we'll prove by contradiction that $\sqrt{2}$ is irrational.

Theorem 0.8.1. *$\sqrt{2}$ is irrational.*

Proof. We use proof by contradiction. Suppose the claim is false; that is, $\sqrt{2}$ is rational. Then we can write $\sqrt{2}$ as a fraction n/d in *lowest terms*.

Squaring both sides gives $2 = n^2/d^2$ and so $2d^2 = n^2$. This implies that n is a multiple of 2. Therefore n^2 must be a multiple of 4. But since $2d^2 = n^2$, we know $2d^2$ is a multiple of 4 and so d^2 is a multiple of 2. This implies that d is a multiple of 2.

So the numerator and denominator have 2 as a common factor, which contradicts the fact that n/d is in lowest terms. So $\sqrt{2}$ must be irrational. ■

0.8.1 Problems

Class Problems

Problem 0.7.
Generalize the proof from lecture (reproduced below) that $\sqrt{2}$ is irrational, for example, how about $\sqrt[3]{2}$? Remember that an irrational number is a number that cannot be expressed as a ratio of two integers.

Theorem. *$\sqrt{2}$ is an irrational number.*

Proof. The proof is by contradiction: assume that $\sqrt{2}$ is rational, that is,

$$\sqrt{2} = \frac{n}{d}, \tag{7}$$

where n and d are integers. Now consider the smallest such positive integer denominator, d. We will prove in a moment that the numerator, n, and the denominator, d, are both even. This implies that

$$\frac{n/2}{d/2}$$

is a fraction equal to $\sqrt{2}$ with a smaller positive integer denominator, a contradiction.

> *Since the assumption that $\sqrt{2}$ is rational leads to this contradiction, the assumption must be false. That is, $\sqrt{2}$ is indeed irrational.* This italicized comment on the implication of the contradiction normally goes without saying, but since this is the first 6.042 exercise about proof by contradiction, we've said it.

To prove that n and d have 2 as a common factor, we start by squaring both sides of (2.8) and get $2 = n^2/d^2$, so

$$2d^2 = n^2. \tag{8}$$

So 2 is a factor of n^2, which is only possible if 2 is in fact a factor of n.

This means that $n = 2k$ for some integer, k, so

$$n^2 = (2k)^2 = 4k^2. \tag{9}$$

Combining (2.9) and (2.10) gives $2d^2 = 4k^2$, so

$$d^2 = 2k^2. \tag{10}$$

So 2 is a factor of d^2, which again is only possible if 2 is in fact also a factor of d, as claimed. ■

Problem 0.8.
Here is a different proof that $\sqrt{2}$ is irrational, taken from the American Mathematical Monthly, v.116, #1, Jan. 2009, p.69:

Proof. Suppose for the sake of contradiction that $\sqrt{2}$ is rational, and choose the least integer, $q > 0$, such that $\left(\sqrt{2} - 1\right) q$ is a nonnegative integer. Let $q' ::= \left(\sqrt{2} - 1\right) q$. Clearly $0 < q' < q$. But an easy computation shows that $\left(\sqrt{2} - 1\right) q'$ is a nonnegative integer, contradicting the minimality of q. ■

(a) This proof was written for an audience of college teachers, and is a little more concise than desirable at this point in 6.042. Write out a more complete version which includes an explanation of each step.

(b) Now that you have justified the steps in this proof, do you have a preference for one of these proofs over the other? Why? Discuss these questions with your teammates for a few minutes and summarize your team's answers on your whiteboard.

Problem 0.9.
Here is a generalization of Problem 2.7 that you may not have thought of:

Lemma 0.8.2. *Let the coefficients of the polynomial $a_0 + a_1x + a_2x^2 + \cdots + a_{n-1}x^{m-1} + x^m$ be integers. Then any real root of the polynomial is either integral or irrational.*

(a) Explain why Lemma 2.5.1 immediately implies that $\sqrt[m]{k}$ is irrational whenever k is not an mth power of some integer.

(b) Collaborate with your tablemates to write a clear, textbook quality proof of Lemma 2.5.1 on your whiteboard. (Besides clarity and correctness, textbook quality requires good English with proper punctuation. When a real textbook writer does this, it usually takes multiple revisions; if you're satisfied with your first draft, you're probably misjudging.) You may find it helpful to appeal to the following:
Lemma 0.8.3. *If a prime, p, is a factor of some power of an integer, then it is a factor of that integer.*

You may assume Lemma 2.5.2 without writing down its proof, but see if you can explain why it is true.

Homework Problems

Problem 0.10.
The fact that that there are irrational numbers a, b such that a^b is rational was proved in Problem 2.2. Unfortunately, that proof was *nonconstructive*: it didn't reveal a specific pair, a, b, with this property. But in fact, it's easy to do this: let $a ::= \sqrt{2}$ and $b ::= 2\log_2 3$.

We know $\sqrt{2}$ is irrational, and obviously $a^b = 3$. Finish the proof that this a, b pair works, by showing that $2\log_2 3$ is irrational.

0.9 *Good* Proofs in Practice

One purpose of a proof is to establish the truth of an assertion with absolute certainty. Mechanically checkable proofs of enormous length or complexity can accomplish this. But humanly intelligible proofs are the only ones that help someone understand the subject. Mathematicians generally agree that important mathematical results can't be fully understood until their proofs are understood. That is why proofs are an important part of the curriculum.

To be understandable and helpful, more is required of a proof than just logical correctness: a good proof must also be clear. Correctness and clarity usually go together; a well-written proof is more likely to be a correct proof, since mistakes are harder to hide.

In practice, the notion of proof is a moving target. Proofs in a professional research journal are generally unintelligible to all but a few experts who know all the terminology and prior results used in the proof. Conversely, proofs in the first weeks of a beginning course like 6.042 would be regarded as tediously long-winded by a professional mathematician. In fact, what we accept as a good proof later in the term will be different from what we consider good proofs in the first couple of weeks of 6.042. But even so, we can offer some general tips on writing good proofs:

State your game plan. A good proof begins by explaining the general line of reasoning, for example, "We use case analysis" or "We argue by contradiction."

Keep a linear flow. Sometimes proofs are written like mathematical mosaics, with juicy tidbits of independent reasoning sprinkled throughout. This is not good. The steps of an argument should follow one another in an intelligble order.

A proof is an essay, not a calculation. Many students initially write proofs the way they compute integrals. The result is a long sequence of expressions without explanation, making it very hard to follow. This is bad. A good proof usually looks like an essay with some equations thrown in. Use complete sentences.

Avoid excessive symbolism. Your reader is probably good at understanding words, but much less skilled at reading arcane mathematical symbols. So use words where you reasonably can.

Revise and simplify. Your readers will be grateful.

Introduce notation thoughtfully. Sometimes an argument can be greatly simplified by introducing a variable, devising a special notation, or defining a new term. But do this sparingly since you're requiring the reader to remember all that new stuff. And remember to actually *define* the meanings of new variables, terms, or notations; don't just start using them!

Structure long proofs. Long programs are usually broken into a hierarchy of smaller procedures. Long proofs are much the same. Facts needed in your proof that are easily stated, but not readily proved are best pulled out and proved in preliminary lemmas. Also, if you are repeating essentially the same argument over and over, try to capture that argument in a general lemma, which you can cite repeatedly instead.

Be wary of the "obvious". When familiar or truly obvious facts are needed in a proof, it's OK to label them as such and to not prove them. But remember that what's obvious to you, may not be —and typically is not —obvious to your reader.

Most especially, don't use phrases like "clearly" or "obviously" in an attempt to bully the reader into accepting something you're having trouble proving. Also, go on the alert whenever you see one of these phrases in someone else's proof.

Finish. At some point in a proof, you'll have established all the essential facts you need. Resist the temptation to quit and leave the reader to draw the "obvious" conclusion. Instead, tie everything together yourself and explain why the original claim follows.

The analogy between good proofs and good programs extends beyond structure. The same rigorous thinking needed for proofs is essential in the design of critical computer systems. When algorithms and protocols only "mostly work" due to reliance on hand-waving arguments, the results can range from problematic to catastrophic. An early example was the Therac 25, a machine that provided radiation therapy to cancer victims, but occasionally killed them with massive overdoses due to a software race condition. A more recent (August 2004) example involved a single faulty command to a computer system used by United and American Airlines that grounded the entire fleet of both companies— and all their passengers!

It is a certainty that we'll all one day be at the mercy of critical computer systems designed by you and your classmates. So we really hope that you'll develop the ability to formulate rock-solid logical arguments that a system actually does what you think it does!

Chapter 1

Propositions

1.1 Mathematical Statements

Definition. A *proposition* is a mathematical statement that is either true or false.

Being true or false doesn't sound like much of a limitation, but it does exclude statements such as, "Wherefore art thou Romeo?" and "Give me an *A*!".

Being "mathematical" is a more serious restriction. For example, "Albert's wife's name is 'Irene' " is a true statement, and you could prove it by presenting legal documents and the testimony of their children. But it isn't a proposition because it is not a *mathematical* statement. There is no mathematical definition of Albert or Irene, and statements about them are not part of mathematics. Propositions must be about well-defined mathematical objects like numbers, sets, functions, relations, etc., and they must be stated using mathematically precise language. Here are some simple examples.

Proposition 1.1.1. *$2 + 3 = 5$.*

This is a true proposition.

Proposition 1.1.2. *The binary representation of every nonnegative integer starts with a* `1`.

This is a false proposition. It could be fixed by ruling out the nonnegative integer zero. So the following proposition is true:

Proposition 1.1.3. *The binary representation of every positive integer starts with a* `1`.

1.1.1 Problems

Class Problems

Problem 1.1.
Albert announces that he plans a surprise 6.042 quiz next week. His students wonder if the quiz could be next Friday. The students realize that it obviously cannot,

because if it hadn't been given before Friday, everyone would know that there was only Friday left on which to give it, so it wouldn't be a surprise any more.

So the students ask whether Albert could give the surprise quiz Thursday? They observe that if the quiz wasn't given *before* Thursday, it would have to be given *on* the Thursday, since they already know it can't be given on Friday. But having figured that out, it wouldn't be a surprise if the quiz was on Thursday either. Similarly, the students reason that the quiz can't be on Wednesday, Tuesday, or Monday. Namely, it's impossible for Albert to give a surprise quiz next week. All the students now relax, having concluded that Albert must have been bluffing.

And since no one expects the quiz, that's why, when Albert gives it on Tuesday next week, it really is a surprise!

What do you think is wrong with the students' reasoning?

1.2 Compound Propositions

It is amazing that people manage to cope with all the ambiguities in the English language. Here are some sentences that illustrate the issue:

1. "You may have cake, or you may have ice cream."
2. "If pigs can fly, then you can understand the Chebyshev bound."
3. "If you can solve any problem we come up with, then you get an A for the course."
4. "Every American has a dream."

What *precisely* do these sentences mean? Can you have both cake and ice cream or must you choose just one dessert? If the second sentence is true, then is the Chebyshev bound incomprehensible? If you can solve some problems we come up with but not all, then do you get an A for the course? And can you still get an A even if you can't solve any of the problems? Does the last sentence imply that all Americans have the same dream or might some of them have different dreams?

Some uncertainty is tolerable in normal conversation. But when we need to formulate ideas precisely —as in mathematics and programming —the ambiguities inherent in everyday language can be a real problem. We can't hope to make an exact argument if we're not sure exactly what the statements mean. So before we start into mathematics, we need to investigate the problem of how to talk about mathematics.

To get around the ambiguity of English, mathematicians have devised a special mini-language for talking about logical relationships. This language mostly uses ordinary English words and phrases such as "or", "implies", and "for all". But mathematicians endow these words with definitions more precise than those found in an ordinary dictionary. Without knowing these definitions, you might sometimes get the gist of statements in this language, but you would regularly get misled about what they really meant.

Surprisingly, in the midst of learning the language of logic, we'll come across the most important open problem in computer science —a problem whose solution could change the world.

1.2.1 Propositions from Propositions

In English, we can modify, combine, and relate propositions with words such as "not", "and", "or", "implies", and "if-then". For example, we can combine three propositions into one like this:

If all humans are mortal **and** all Greeks are human, **then** all Greeks are mortal.

For the next while, we won't be much concerned with the internals of propositions —whether they involve mathematics or Greek mortality —but rather with how propositions are combined and related. So we'll frequently use variables such as P and Q in place of specific propositions such as "All humans are mortal" and "$2 + 3 = 5$". The understanding is that these variables, like propositions, can take on only the values **T** (true) and **F** (false). Such true/false variables are sometimes called *Boolean variables* after their inventor, George —you guessed it —Boole.

NOT, AND, OR

We can precisely define these special words using *truth tables*. For example, if P denotes an arbitrary proposition, then the truth of the proposition "NOT P" is defined by the following truth table:

P	NOT P
T	**F**
F	**T**

The first row of the table indicates that when proposition P is true, the proposition "NOT P" is false. The second line indicates that when P is false, "NOT P" is true. This is probably what you would expect.

In general, a truth table indicates the true/false value of a proposition for each possible setting of the variables. For example, the truth table for the proposition "P AND Q" has four lines, since the two variables can be set in four different ways:

P	Q	P AND Q
T	**T**	**T**
T	**F**	**F**
F	**T**	**F**
F	**F**	**F**

According to this table, the proposition "P AND Q" is true only when P and Q are both true. This is probably the way you think about the word "and."

There is a subtlety in the truth table for "P OR Q":

P	Q	P OR Q
T	**T**	**T**
T	**F**	**T**
F	**T**	**T**
F	**F**	**F**

The third row of this table says that "P OR Q" is true when even if *both* P and Q are true. This isn't always the intended meaning of "or" in everyday speech, but this is the standard definition in mathematical writing. So if a mathematician says, "You may have cake, or you may have ice cream," he means that you *could* have both.

If you want to exclude the possibility of having both having and eating, you should use "exclusive-or" (XOR):

P	Q	P XOR Q
T	**T**	**F**
T	**F**	**T**
F	**T**	**T**
F	**F**	**F**

IMPLIES

The least intuitive connecting word is "implies." Here is its truth table, with the lines labeled so we can refer to them later.

P	Q	P IMPLIES Q	
T	**T**	**T**	(tt)
T	**F**	**F**	(tf)
F	**T**	**T**	(ft)
F	**F**	**T**	(ff)

To experiment with this definition, we're going to examine a famous unsolved problem in mathematics called the Riemann Hypothesis.[1] For now, it doesn't matter what the Riemann Hypothesis happens to be; all that's important in the following example is it is a proposition and no one knows whether it is true or false.

So now let's look at the if-then proposition:

"If the Riemann Hypothesis is true, then $x^2 \geq 0$ for every real number x."

Is this if-then proposition true or false? The answer (that a mathematician would give, anyway) is that it's true! That's because it means that $P \longrightarrow Q$, where the

[1] We'll explain a little about the Riemann Hypothesis in a later chapter (see Definition 4.6.5). Whether or not the Riemann Hypothesis holds is generally considered to be one of the most important unolved problems is mathematics.

if-part, P, called the*hypothesis*, is "the Riemann Hypothesis is true," and the then-part, Q called the *conclusion*, is "$x^2 \geq 0$ for every real number x." Since the conclusion is definitely true, we're on either line (tt) or line (ft) of the truth table. Either way, the proposition as a whole is *true*.

One of our original examples demonstrates an even stranger side of implications.

"If pigs fly, then you can understand the Chebyshev bound."

(Don't take this as an insult to your understanding; we just need to figure out whether this proposition is true or false.) Curiously, the answer has *nothing* to do with whether or not you know anything about the Chebyshev bound. Pigs do not fly, so we're on either line (ft) or line (ff) of the truth table. In both cases, the proposition is *true*!

Sometimes it's worth emphasizing that an implication is true because its hypothesis is false, and we do this by saying it is *vacuously true*.

In contrast, here's an example of a false implication:

"If the moon shines white, then the moon is made of white cheddar."

Yes, the moon shines white. But, no, the moon is not made of white cheddar cheese. So we're on line (tf) of the truth table, and the proposition is false.

The truth table for implications can be summarized in words as follows:

An implication is true exactly when the if-part is false or the then-part is true.

This sentence is worth remembering; a large fraction of all mathematical statements are of the if-then form!

IFF

Mathematicians commonly join propositions in one additional way that doesn't arise in ordinary speech. The proposition "P if and only if Q" asserts that P and Q are logically equivalent; that is, either both are true or both are false.

P	Q	P IFF Q
T	**T**	**T**
T	**F**	**F**
F	**T**	**F**
F	**F**	**T**

The following if-and-only-if statement is true for every real number x:

$$x^2 - 4 \geq 0 \quad \text{iff} \quad |x| \geq 2$$

For some values of x, *both* inequalities are true. For other values of x, *neither* inequality is true . In every case, however, the proposition as a whole is true.

1.2.2 Problems

Class Problems

Problem 1.2.
When the mathematician says to his student, "If a function is not continuous, then it is not differentiable," then letting D stand for "differentiable" and C for continuous, the only proper translation of the mathematician's statement would be

$$\text{NOT}(C) \ \text{IMPLIES} \ \text{NOT}(D),$$

or equivalently,

$$D \ \text{IMPLIES} \ C.$$

But when a mother says to her son, "If you don't do your homework, then you can't watch TV," then letting T stand for "watch TV" and H for "do your homework," a reasonable translation of the mother's statement would be

$$\text{NOT}(H) \ \text{IFF} \ \text{NOT}(T),$$

or equivalently,

$$H \ \text{IFF} \ T.$$

Explain why it is reasonable to translate these two IF-THEN statements in different ways into propositional formulas.

Problem 1.3.
Prove by truth table that OR distributes over AND:

$$[P \ \text{OR} \ (Q \ \text{AND} \ R)] \quad \text{is equivalent to} \quad [(P \ \text{OR} \ Q) \ \text{AND} \ (P \ \text{OR} \ R)] \tag{1.1}$$

Homework Problems

Problem 1.4.
Describe a simple recursive procedure which, given a positive integer argument, n, produces a truth table whose rows are all the assignments of truth values to n propositional variables. For example, for $n = 2$, the table might look like:

T	T
T	F
F	T
F	F

Your description can be in English, or a simple program in some familiar language (say Scheme or Java), but if you do write a program, be sure to include some sample output.

1.3 Propositional Logic in Computer Programs

Propositions and logical connectives arise all the time in computer programs. For example, consider the following snippet, which could be either C, C++, or Java:

```
if ( x > 0 || (x <= 0 && y > 100) )
      ⋮
(further instructions)
```

The symbol `||` denotes "or", and the symbol `&&` denotes "and". The *further instructions* are carried out only if the proposition following the word `if` is true. On closer inspection, this big expression is built from two simpler propositions. Let A be the proposition that `x > 0`, and let B be the proposition that `y > 100`. Then we can rewrite the condition this way:

$$A \text{ or } ((\text{not } A) \text{ and } B) \tag{1.2}$$

A truth table reveals that this complicated expression is logically equivalent to

$$A \text{ or } B. \tag{1.3}$$

A	B	A or ((not A) and B)	A or B
T	**T**	**T**	**T**
T	**F**	**T**	**T**
F	**T**	**T**	**T**
F	**F**	**F**	**F**

This means that we can simplify the code snippet without changing the program's behavior:

```
if ( x > 0 || y > 100 )
      ⋮
(further instructions)
```

The equivalence of (1.2) and (1.3) can also be confirmed reasoning by cases:

A is **T**. Then an expression of the form (A or anything) will have truth value **T**. Since both expressions are of this form, both have the same truth value in this case, namely, **T**.

A is **F**. Then (A or P) will have the same truth value as P for any proposition, P. So (1.3) has the same truth value as B. Similarly, (1.2) has the same truth value as ((not **F**) and B), which also has the same value as B. So in this case, both expressions will have the same truth value, namely, the value of B.

Rewriting a logical expression involving many variables in the simplest form is both difficult and important. Simplifying expressions in software might slightly increase the speed of your program. But, more significantly, chip designers face essentially the same challenge. However, instead of minimizing `&&` and `||` symbols in a program, their job is to minimize the number of analogous physical devices on a chip. The payoff is potentially enormous: a chip with fewer devices is smaller, consumes less power, has a lower defect rate, and is cheaper to manufacture.

1.3.1 Cryptic Notation

Programming languages use symbols like && and ! in place of words like "and" and "not". Mathematicians have devised their own cryptic symbols to represent these words, which are summarized in the table below.

English	**Cryptic Notation**
NOT(P)	$\neg P$ (alternatively, $\overline{P}$)
P AND Q	$P \wedge Q$
P OR Q	$P \vee Q$
P IMPLIES Q	$P \longrightarrow Q$
if P then Q	$P \longrightarrow Q$
P IFF Q	$P \longleftrightarrow Q$ (alternatively, P iff Q)

For example, using this notation, "If P and not Q, then R" would be written:

$$(P \wedge \overline{Q}) \longrightarrow R$$

But words such as "OR" and "IMPLIES" generally serve just as well as the cryptic symbols $\wedge$ and $\longrightarrow$, and their meaning is easy to remember. So we'll use the cryptic notation only when it's essential to have a compact formula, and we advise you to do the same.

1.3.2 Logically Equivalent Implications

Do these two sentences say the same thing?

If I am hungry, then I am grumpy.
If I am not grumpy, then I am not hungry.

We can settle the issue by recasting both sentences in terms of propositional logic. Let P be the proposition "I am hungry", and let Q be "I am grumpy". The first sentence says "P implies Q" and the second says "(not Q) implies (not P)". We can compare these two statements in a truth table:

P	Q	P IMPLIES Q	$\overline{Q}$ IMPLIES $\overline{P}$
T	**T**	**T**	**T**
T	**F**	**F**	**F**
F	**T**	**T**	**T**
F	**F**	**T**	**T**

Sure enough, the columns of truth values under these two statements are the same, which precisely means they are equivalent. In general, "(NOT Q) IMPLIES (NOT P)"

is called the *contrapositive* of the implication "P IMPLIES Q." And, as we've just shown, the two are just different ways of saying the same thing.

In contrast, the *converse* of "P IMPLIES Q" is the statement "Q IMPLIES P". In terms of our example, the converse is:

If I am grumpy, then I am hungry.

This sounds like a rather different contention, and a truth table confirms this suspicion:

P	Q	P IMPLIES Q	Q IMPLIES P
T	**T**	**T**	**T**
T	**F**	**F**	**T**
F	**T**	**T**	**F**
F	**F**	**T**	**T**

Thus, an implication *is* logically equivalent to its contrapositive but is *not* equivalent to its converse.

One final relationship: an implication and its converse together are equivalent to an iff statement, specifically, to these two statements together. For example,

If I am grumpy, then I am hungry.
If I am hungry, then I am grumpy.

are equivalent to the single statement:

I am grumpy iff I am hungry.

Once again, we can verify this with a truth table:

P | Q | (P IMPLIES Q) AND (Q IMPLIES P) | Q IFF P
height
T T T T T T T F F F T F F T T F F F F F T T T T

The underlined operators have the same column of truth values, proving that the corresponding formulas are equivalent.

1.3.3 Problems

Class Problems

Problem 1.5.
This problem[2] examines whether the following specifications are *satisfiable*:

1. If the file system is not locked, then
   (a) new messages will be queued.
   (b) new messages will be sent to the messages buffer.

[2]From Rosen, 5th edition, Exercise 1.1.36

(c) the system is functioning normally, and conversely, if the system is functioning normally, then the file system is not locked.

2. If new messages are not queued, then they will be sent to the messages buffer.

3. New messages will not be sent to the message buffer.

(a) Begin by translating the five specifications into propositional formulas using four propositional variables:

$$\begin{aligned} L &::= \text{file system locked,} \\ Q &::= \text{new messages are queued,} \\ B &::= \text{new messages are sent to the message buffer,} \\ N &::= \text{system functioning normally.} \end{aligned}$$

(b) Demonstrate that this set of specifications is satisfiable by describing a single truth assignment for the variables L, Q, B, N and verifying that under this assignment, all the specifications are true.

(c) Argue that the assignment determined in part (b) is the only one that does the job.

Problem 1.6.
Propositional logic comes up in digital circuit design using the convention that **T** corresponds to 1 and **F** to 0. A simple example is a 2-bit *half-adder* circuit. This circuit has 3 binary inputs, a_1, a_0 and b, and 3 binary outputs, c, o_1, o_0. The 2-bit word $a_1 a_0$ gives the binary representation of an integer, k, between 0 and 3. The 3-bit word $cs_1 s_0$ gives the binary representation of $k + b$. The third output bit, c, is called the final *carry bit*.

So if k and b were both 1, then the value of $a_1 a_0$ would be `01` and the value of the output $cs_1 s_0$ would `010`, namely, the 3-bit binary representation of $1 + 1$.

In fact, the final carry bit equals 1 only when all three binary inputs are 1, that is, when $k = 3$ and $b = 1$. In that case, the value of $cs_1 s_0$ is `100`, namely, the binary representation of $3 + 1$.

This 2-bit half-adder could be described by the following formulas:

$$\begin{aligned} c_0 &= b \\ s_0 &= a_0 \text{ XOR } c_0 \\ c_1 &= a_0 \text{ AND } c_0 && \text{the carry into column 1} \\ s_1 &= a_1 \text{ XOR } c_1 \\ c_2 &= a_1 \text{ AND } c_1 && \text{the carry into column 2} \\ c &= c_2. \end{aligned}$$

(a) Generalize the above construction of a 2-bit half-adder to an $n+1$ bit half-adder with inputs $a_n, \ldots, a_1, a_0$ and b for arbitrary $n \geq 0$. That is, give simple formulas for s_i and c_i for $0 \leq i \leq n+1$, where c_i is the carry into column i and $c = c_{n+1}$.

(b) Write similar definitions for the digits and carries in the sum of two $n+1$-bit binary numbers $a_n \ldots a_1 a_0$ and $b_n \ldots b_1 b_0$.

Visualized as digital circuits, the above adders consist of a sequence of single-digit half-adders or adders strung together in series. These circuits mimic ordinary pencil-and-paper addition, where a carry into a column is calculated directly from the carry into the previous column, and the carries have to ripple across all the columns before the carry into the final column is determined. Circuits with this design are called *ripple-carry* adders. Ripple-carry adders are easy to understand and remember and require a nearly minimal number of operations. But the higher-order output bits and the final carry take time proportional to n to reach their final values.

(c) How many of each of the propositional operations does your adder from part (b) use to calculate the sum?

Homework Problems

Problem 1.7.
Considerably faster adder circuits work by computing the values in later columns for both a carry of 0 and a carry of 1, *in parallel*. Then, when the carry from the earlier columns finally arrives, the pre-computed answer can be quickly selected. We'll illustrate this idea by working out the equations for an $n+1$-bit parallel half-adder.

Parallel half-adders are built out of parallel "add1" modules. An $n+1$-bit add1 module takes as input the $n+1$-bit binary representation, $a_n \ldots a_1 a_0$, of an integer, s, and produces as output the binary representation, $c\, p_n \ldots p_1\, p_0$, of $s+1$.

(a) A 1-bit add1 module just has input a_0. Write propositional formulas for its outputs c and p_0.

(b) Explain how to build an $n+1$-bit parallel half-adder from an $n+1$-bit add1 module by writing a propositional formula for the half-adder output, o_i, using only the variables a_i, p_i, and b.

We can build a double-size add1 module with $2(n+1)$ inputs using two single-size add1 modules with $n+1$ inputs. Suppose the inputs of the double-size module are $a_{2n+1}, \ldots, a_1, a_0$ and the outputs are $c, p_{2n+1}, \ldots, p_1, p_0$. The setup is illustrated in Figure 1.2.

Namely, the first single size add1 module handles the first $n+1$ inputs. The inputs to this module are the low-order $n+1$ input bits $a_n, \ldots, a_1, a_0$, and its outputs will serve as the first $n+1$ outputs $p_n, \ldots, p_1, p_0$ of the double-size module. Let $c_{(1)}$ be the remaining carry output from this module.

The inputs to the second single-size module are the higher-order $n+1$ input bits $a_{2n+1}, \ldots, a_{n+2}, a_{n+1}$. Call its first $n+1$ outputs $r_n, \ldots, r_1, r_0$ and let $c_{(2)}$ be its carry.

(c) Write a formula for the carry, c, in terms of $c_{(1)}$ and $c_{(2)}$.

(d) Complete the specification of the double-size module by writing propositional formulas for the remaining outputs, p_i, for $n+1 \leq i \leq 2n+1$. The formula for p_i should only involve the variables a_i, $r_{i-(n+1)}$, and $c_{(1)}$.

(e) Parallel half-adders are exponentially faster than ripple-carry half-adders. Confirm this by determining the largest number of propositional operations required to compute any one output bit of an n-bit add module. (You may assume n is a power of 2.)

1.4 Satisfiability

A proposition is **satisfiable** if some setting of the variables makes the proposition true. For example, P AND $\overline{Q}$ is satisfiable because the expression is true when P is true and Q is false. On the other hand, P AND $\overline{P}$ is not satisfiable because the expression as a whole is false for both settings of P. But determining whether or not a more complicated proposition is satisfiable is not so easy. How about this one?

$$(P \text{ OR } Q \text{ OR } R) \text{ AND } (\overline{P} \text{ OR } \overline{Q}) \text{ AND } (\overline{P} \text{ OR } \overline{R}) \text{ AND } (\overline{R} \text{ OR } \overline{Q})$$

The general problem of deciding whether a proposition is satisfiable is called *SAT*. One approach to SAT is to construct a truth table and check whether or not a **T** ever appears. But this approach is not very efficient; a proposition with n variables has a truth table with 2^n lines, so the effort required to decide about a proposition grows exponentially with the number of variables. For a proposition with just 30 variables, that's already over a billion!

Is there a more *efficient* solution to SAT? In particular, is there some, presumably very ingenious, procedure that determines in a number of steps that grows *polynomially* —like n^2 of n^{14} —instead of exponentially, whether any given proposition is satifiable or not? No one knows. And an awful lot hangs on the answer. An efficient solution to SAT would immediately imply efficient solutions to many, many other important problems involving packing, scheduling, routing, and circuit verification, among other things. This would be wonderful, but there would also be worldwide chaos. Decrypting coded messages would also become an easy task (for most codes). Online financial transactions would be insecure and secret communications could be read by everyone.

Recently there has been exciting progress on *sat-solvers* for practical applications like digital circuit verification. These programs find satisfying assignments with amazing efficiency even for formulas with millions of variables. Unfortunately, it's hard to predict which kind of formulas are amenable to sat-solver meth-

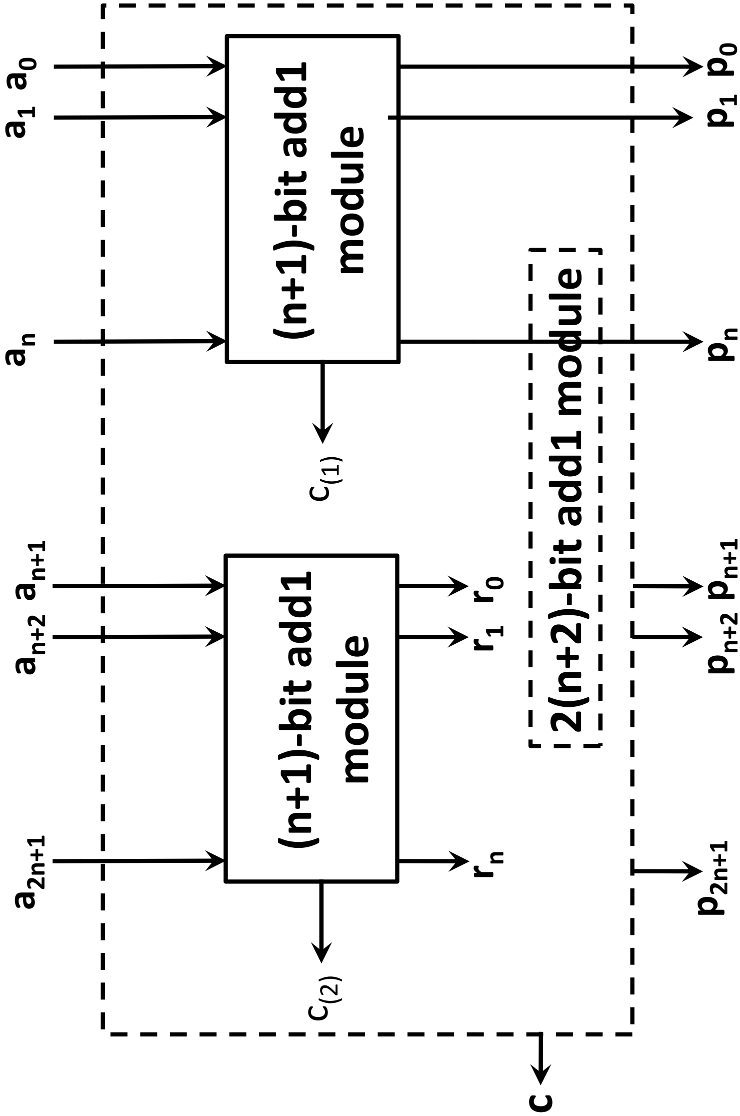


Figure 1.1: Structure of a Double-size Add1 Module.

ods, and for formulas that are NOT satisfiable, sat-solvers generally take exponential time to verify that.

So no one has a good idea how to solve SAT in polynomial time or else to prove that it can't be done —researchers are completely stuck. The problem of determining whether or not SAT has a polynomial time solution is known as the "**P** vs. **NP**" problem. It is the outstanding unanswered question in theoretical computer science. It is also one of the seven Millenium Problems: the Clay Institute will award you $1,000,000 if you solve the **P** vs. **NP** problem.

1.4.1 Problems

Class Problems

Problem 1.8.
This problem[3] examines whether the following specifications are *satisfiable*:

1. If the file system is not locked, then
   (a) new messages will be queued.
   (b) new messages will be sent to the messages buffer.
   (c) the system is functioning normally, and conversely, if the system is functioning normally, then the file system is not locked.
2. If new messages are not queued, then they will be sent to the messages buffer.
3. New messages will not be sent to the message buffer.

(a) Begin by translating the five specifications into propositional formulas using four propositional variables:

$$\begin{aligned} L &::= \text{file system locked}, \\ Q &::= \text{new messages are queued}, \\ B &::= \text{new messages are sent to the message buffer}, \\ N &::= \text{system functioning normally}. \end{aligned}$$

(b) Demonstrate that this set of specifications is satisfiable by describing a single truth assignment for the variables L, Q, B, N and verifying that under this assignment, all the specifications are true.

(c) Argue that the assignment determined in part (b) is the only one that does the job.

[3]From Rosen, 5th edition, Exercise 1.1.36

Problem 1.9.
Propositional logic comes up in digital circuit design using the convention that **T** corresponds to 1 and **F** to 0. A simple example is a 2-bit *half-adder* circuit. This circuit has 3 binary inputs, a_1, a_0 and b, and 3 binary outputs, c, o_1, o_0. The 2-bit word $a_1 a_0$ gives the binary representation of an integer, k, between 0 and 3. The 3-bit word $cs_1 s_0$ gives the binary representation of $k + b$. The third output bit, c, is called the final *carry bit*.

So if k and b were both 1, then the value of $a_1 a_0$ would be `01` and the value of the output $cs_1 s_0$ would `010`, namely, the 3-bit binary representation of $1 + 1$.

In fact, the final carry bit equals 1 only when all three binary inputs are 1, that is, when $k = 3$ and $b = 1$. In that case, the value of $cs_1 s_0$ is `100`, namely, the binary representation of $3 + 1$.

This 2-bit half-adder could be described by the following formulas:

$$\begin{aligned}
c_0 &= b \\
s_0 &= a_0 \text{ XOR } c_0 \\
c_1 &= a_0 \text{ AND } c_0 && \text{the carry into column 1} \\
s_1 &= a_1 \text{ XOR } c_1 \\
c_2 &= a_1 \text{ AND } c_1 && \text{the carry into column 2} \\
c &= c_2.
\end{aligned}$$

(a) Generalize the above construction of a 2-bit half-adder to an $n + 1$ bit half-adder with inputs $a_n, \ldots, a_1, a_0$ and b for arbitrary $n \geq 0$. That is, give simple formulas for s_i and c_i for $0 \leq i \leq n + 1$, where c_i is the carry into column i and $c = c_{n+1}$.

(b) Write similar definitions for the digits and carries in the sum of two $n + 1$-bit binary numbers $a_n \ldots a_1 a_0$ and $b_n \ldots b_1 b_0$.

Visualized as digital circuits, the above adders consist of a sequence of single-digit half-adders or adders strung together in series. These circuits mimic ordinary pencil-and-paper addition, where a carry into a column is calculated directly from the carry into the previous column, and the carries have to ripple across all the columns before the carry into the final column is determined. Circuits with this design are called *ripple-carry* adders. Ripple-carry adders are easy to understand and remember and require a nearly minimal number of operations. But the higher-order output bits and the final carry take time proportional to n to reach their final values.

(c) How many of each of the propositional operations does your adder from part (b) use to calculate the sum?

Problem 1.10. **(a)** A propositional formula is *valid* iff it is equivalent to **T**. Verify by truth table that

$$(P \text{ IMPLIES } Q) \text{ OR } (Q \text{ IMPLIES } P)$$

is valid.

(b) Let P and Q be propositional formulas. Describe a single propositional formula, R, involving P and Q such that R is valid iff P and Q are equivalent.

(c) A propositional formula is *satisfiable* iff there is an assignment of truth values to its variables —an *environment* —which makes it true. Explain why

P is valid iff $\text{NOT}(P)$ is *not* satisfiable.

(d) A set of propositional formulas $P_1, \ldots, P_k$ is *consistent* iff there is an environment in which they are all true. Write a formula, S, so that the set $P_1, \ldots, P_k$ is *not* consistent iff S is valid.

Problem 1.11.

Considerably faster adder circuits work by computing the values in later columns for both a carry of 0 and a carry of 1, *in parallel*. Then, when the carry from the earlier columns finally arrives, the pre-computed answer can be quickly selected. We'll illustrate this idea by working out the equations for an $n+1$-bit parallel half-adder.

Parallel half-adders are built out of parallel "add1" modules. An $n+1$-bit add1 module takes as input the $n+1$-bit binary representation, $a_n \ldots a_1 a_0$, of an integer, s, and produces as output the binary representation, $c\, p_n \ldots p_1\, p_0$, of $s+1$.

(a) A 1-bit add1 module just has input a_0. Write propositional formulas for its outputs c and p_0.

(b) Explain how to build an $n+1$-bit parallel half-adder from an $n+1$-bit add1 module by writing a propositional formula for the half-adder output, o_i, using only the variables a_i, p_i, and b.

We can build a double-size add1 module with $2(n+1)$ inputs using two single-size add1 modules with $n+1$ inputs. Suppose the inputs of the double-size module are $a_{2n+1}, \ldots, a_1, a_0$ and the outputs are $c, p_{2n+1}, \ldots, p_1, p_0$. The setup is illustrated in Figure 1.2.

Namely, the first single size add1 module handles the first $n+1$ inputs. The inputs to this module are the low-order $n+1$ input bits $a_n, \ldots, a_1, a_0$, and its outputs will serve as the first $n+1$ outputs $p_n, \ldots, p_1, p_0$ of the double-size module. Let $c_{(1)}$ be the remaining carry output from this module.

The inputs to the second single-size module are the higher-order $n+1$ input bits $a_{2n+1}, \ldots, a_{n+2}, a_{n+1}$. Call its first $n+1$ outputs $r_n, \ldots, r_1, r_0$ and let $c_{(2)}$ be its carry.

(c) Write a formula for the carry, c, in terms of $c_{(1)}$ and $c_{(2)}$.

(d) Complete the specification of the double-size module by writing propositional formulas for the remaining outputs, p_i, for $n+1 \le i \le 2n+1$. The formula for p_i should only involve the variables a_i, $r_{i-(n+1)}$, and $c_{(1)}$.

(e) Parallel half-adders are exponentially faster than ripple-carry half-adders. Confirm this by determining the largest number of propositional operations required to compute any one output bit of an n-bit add module. (You may assume n is a power of 2.)

1.5 Predicates and Quantifiers

1.5.1 Some More Propositions

A *prime* is an integer greater than one that is not divisible by any integer greater than 1 besides itself, for example, 2, 3, 5, 7, 11, ...

Proposition 1.5.1. *For every nonnegative integer, n, the value of $n^2 + n + 41$ is prime.*

Let's try some numerical experimentation to check this proposition. Let [4]

$$p(n) ::= n^2 + n + 41. \tag{1.4}$$

We begin with $p(0) = 41$ which is prime. $p(1) = 43$ which is prime. $p(2) = 47$ which is prime. $p(3) = 53$ which is prime. ... $p(20) = 461$ which is prime. Hmmm, starts to look like a plausible claim. In fact we can keep checking through $n = 39$ and confirm that $p(39) = 1601$ is prime.

But $p(40) = 40^2 + 40 + 41 = 41 \cdot 41$, which is not prime. So it's not true that the expression is prime *for all* nonnegative integers. The point is that in general you can't check a claim about an infinite set by checking a finite set of its elements, no matter how large the finite set.

By the way, propositions like this about *all* numbers or other things are so common that there is a special notation for it. With this notation, Proposition 1.5.1 would be

$$\forall n \in \mathbb{N}.\ p(n) \text{ is prime.} \tag{1.5}$$

Here the symbol $\forall$ is read "for all". The symbol $\mathbb{N}$ stands for the set of *nonnegative integers*, namely, 0, 1, 2, 3, ... (ask your instructor for the complete list). The symbol "$\in$" is read as "is a member of," or "belongs to," or simply as "is in". The period after the $\mathbb{N}$ is just a separator between phrases.

Here are two even more extreme examples:

Proposition 1.5.2. *$a^4 + b^4 + c^4 = d^4$ has no solution when a, b, c, d are positive integers.*

Euler (pronounced "oiler") conjectured this in 1769. But the proposition was proven false 218 years later by Noam Elkies at a liberal arts school up Mass Ave. The solution he found was $a = 95800, b = 217519, c = 414560, d = 422481$.

[4] The symbol ::= means "equal by definition." It's always ok to simply write "=" instead of ::=, but reminding the reader that an equality holds by definition can be helpful.

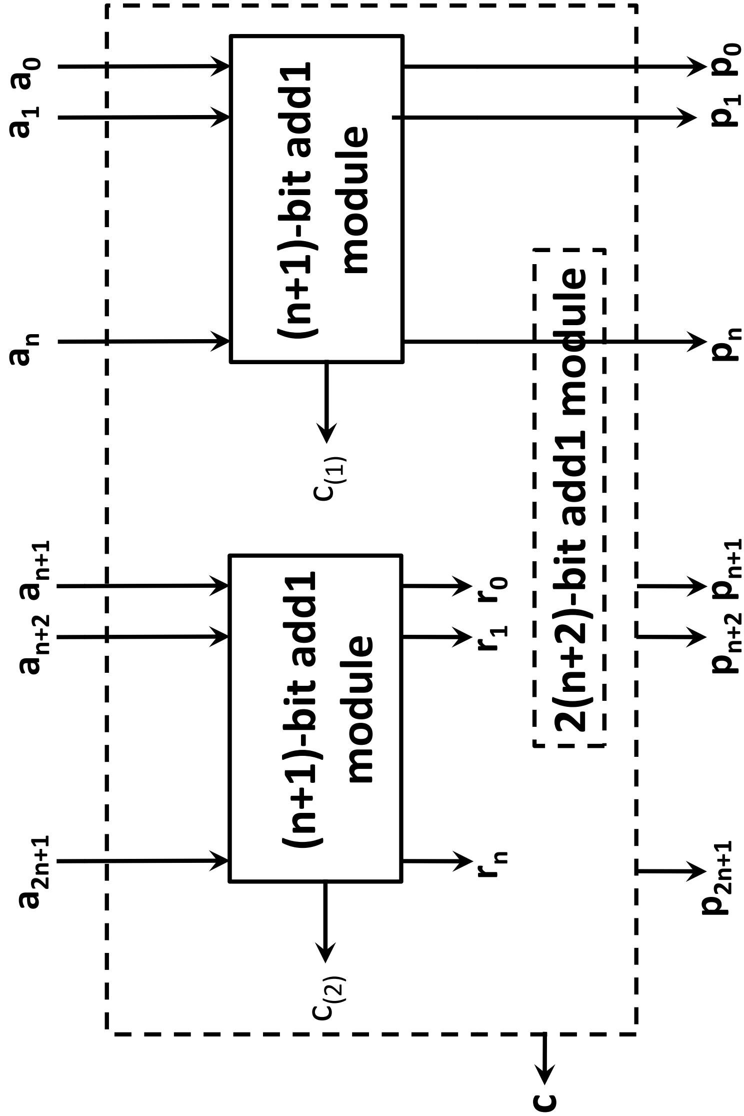


Figure 1.2: Structure of a Double-size Add1 Module.

In logical notation, Proposition 1.5.2 could be written,

$$\forall a \in \mathbb{Z}^+ \, \forall b \in \mathbb{Z}^+ \, \forall c \in \mathbb{Z}^+ \, \forall d \in \mathbb{Z}^+ . \; a^4 + b^4 + c^4 \neq d^4 .$$

Here, $\mathbb{Z}^+$ is a symbol for the positive integers. Strings of $\forall$'s like this are usually abbreviated for easier reading:

$$\forall a, b, c, d \in \mathbb{Z}^+ . \; a^4 + b^4 + c^4 \neq d^4 .$$

Proposition 1.5.3. *$313(x^3 + y^3) = z^3$ has no solution when $x, y, z \in \mathbb{Z}^+$.*

This proposition is also false, but the smallest counterexample has more than 1000 digits!

Proposition 1.5.4. *Every map can be colored with 4 colors so that adjacent[5] regions have different colors.*

This proposition is true and is known as the "*Four-Color Theorem*". However, there have been many incorrect proofs, including one that stood for 10 years in the late 19th century before the mistake was found. An extremely laborious proof was finally found in 1976 by mathematicians Appel and Haken, who used a complex computer program to categorize the four-colorable maps; the program left a few thousand maps uncategorized, and these were checked by hand by Haken and his assistants—including his 15-year-old daughter. There was a lot of debate about whether this was a legitimate proof: the proof was too big to be checked without a computer, and no one could guarantee that the computer calculated correctly, nor did anyone have the energy to recheck the four-colorings of thousands of maps that were done by hand. Within the past decade a mostly intelligible proof of the Four-Color Theorem was found, though a computer is still needed to check colorability of several hundred special maps.[6]

Proposition 1.5.5 (Goldbach)**.** *Every even integer greater than 2 is the sum of two primes.*

No one knows whether this proposition is true or false. It is known as *Goldbach's Conjecture*, and dates back to 1742.

For a computer scientist, some of the most important things to prove are the "correctness" programs and systems —whether a program or system does what it's supposed to. Programs are notoriously buggy, and there's a growing community of researchers and practitioners trying to find ways to prove program correctness. These efforts have been successful enough in the case of CPU chips that they are now routinely used by leading chip manufacturers to prove chip correctness and avoid mistakes like the notorious Intel division bug in the 1990's.

Developing mathematical methods to verify programs and systems remains an active research area. We'll consider some of these methods later in the course.

---

[5]Two regions are adjacent only when they share a boundary segment of positive length. They are not considered to be adjacent if their boundaries meet only at a few points.

[6]See `http://www.math.gatech.edu/~thomas/FC/fourcolor.html`

The story of the Four-Color Proof is told in a well-reviewed popular (non-technical) book: "Four Colors Suffice. How the Map Problem was Solved." *Robin Wilson*. Princeton Univ. Press, 2003, 276pp. ISBN 0-691-11533-8.

1.5.2 Predicates

A *predicate* is a proposition whose truth depends on the value of one or more variables. Most of the propositions above were defined in terms of predicates. For example,

$$\text{“}n \text{ is a perfect square”}$$

is a predicate whose truth depends on the value of n. The predicate is true for $n = 4$ since four is a perfect square, but false for $n = 5$ since five is not a perfect square.

Like other propositions, predicates are often named with a letter. Furthermore, a function-like notation is used to denote a predicate supplied with specific variable values. For example, we might name our earlier predicate P:

$$P(n) ::= \text{“}n \text{ is a perfect square”}$$

Now $P(4)$ is true, and $P(5)$ is false.

This notation for predicates is confusingly similar to ordinary function notation. If P is a predicate, then $P(n)$ is either *true* or *false*, depending on the value of n. On the other hand, if p is an ordinary function, like $n^2 + n$, then $p(n)$ is a *numerical quantity*. **Don't confuse these two!**

1.5.3 Quantifiers

There are a couple of assertions commonly made about a predicate: that it is *sometimes* true and that it is *always* true. For example, the predicate

$$\text{“}x^2 \geq 0\text{”}$$

is always true when x is a real number. On the other hand, the predicate

$$\text{“}5x^2 - 7 = 0\text{”}$$

is only sometimes true; specifically, when $x = \pm\sqrt{7/5}$.

There are several ways to express the notions of "always true" and "sometimes true" in English. The table below gives some general formats on the left and specific examples using those formats on the right. You can expect to see such phrases hundreds of times in mathematical writing!

Always True

For all n, $P(n)$ is true.	For all $x \in \mathbb{R}$, $x^2 \geq 0$.
$P(n)$ is true for every n.	$x^2 \geq 0$ for every $x \in \mathbb{R}$.

Sometimes True

There exists an n such that $P(n)$ is true.	There exists an $x \in \mathbb{R}$ such that $5x^2 - 7 = 0$.
$P(n)$ is true for some n.	$5x^2 - 7 = 0$ for some $x \in \mathbb{R}$.
$P(n)$ is true for at least one n.	$5x^2 - 7 = 0$ for at least one $x \in \mathbb{R}$.

All these sentences quantify how often the predicate is true. Specifically, an assertion that a predicate is always true, is called a *universally quantified* statement. An assertion that a predicate is sometimes true, is called an *existentially quantified* statement.

Sometimes English sentences are unclear about quantification:

"If you can solve any problem we come up with, then you get an A for the course."

The phrase "you can solve any problem we can come up with" could reasonably be interpreted as either a universal or existential statement. It might mean:

"You can solve *every* problem we come up with,"

or maybe

"You can solve *at least one* problem we come up with."

It's worth noticing that however the phrase is meant to be quantified, it appears as part of a larger "if ..., then" statement, which is typical. Quantified statements themselves define predicates or propositions and can be combined with AND, OR, IMPLIES, etc., just like any other statement.

1.5.4 More Cryptic Notation

There are symbols to represent universal and existential quantification, just as there are symbols for "AND" ($\wedge$), "IMPLIES" ($\longrightarrow$), and so forth. In particular, to say that a predicate, $P(x)$, is true for all values of x in some set, D, we write:

$$\forall x \in D.\ P(x) \tag{1.6}$$

The *universal quantifier* symbol $\forall$ is read "for all," so this whole expression (1.6) is read "For all x in D, $P(x)$ is true." Remember that upside-down "A" stands for "**A**ll."

To say that a predicate $P(x)$ is true for at least one value of x in D, we write:

$$\exists x \in D.\ P(x) \tag{1.7}$$

The *existential quantifier* symbol $\exists$, is read "there exists." So expression (1.7) is read, "There exists an x in D such that $P(x)$ is true." Remember that backward "E" stands for "**E**xists."

The symbols $\forall$ and $\exists$ are always followed by a variable —typically with an indication of the set the variable ranges over —and then a predicate, as in the two examples above.

As an example, let Probs be the set of problems we come up with, Solves(x) be the predicate "You can solve problem x", and G be the proposition, "You get an A for the course." Then the two different interpretations of

"If you can solve any problem we come up with, then you get an A for the course."

can be written as follows:

$$(\forall x \in \text{Probs. Solves}(x)) \text{ IMPLIES } G,$$

or maybe

$$(\exists x \in \text{Probs. Solves}(x)) \text{ IMPLIES } G.$$

1.5.5 Mixing Quantifiers

Many mathematical statements involve several quantifiers. For example, *Goldbach's Conjecture* states:

"Every even integer greater than 2 is the sum of two primes."

Let's write this more verbosely to make the use of quantification clearer:

> For every even integer n greater than 2, there exist primes p and q such that $n = p + q$.

Let Evens be the set of even integers greater than 2, and let Primes be the set of primes. Then we can write Goldbach's Conjecture in logic notation as follows:

$$\underbrace{\forall n \in \text{Evens}}_{\substack{\text{for every even} \\ \text{integer } n > 2}} \underbrace{\exists p \in \text{Primes} \; \exists q \in \text{Primes}.}_{\substack{\text{there exist primes} \\ p \text{ and } q \text{ such that}}} \; n = p + q.$$

1.5.6 Order of Quantifiers

Swapping the order of different kinds of quantifiers (existential or universal) usually changes the meaning of a proposition. For example, let's return to one of our initial, confusing statements:

"Every American has a dream."

This sentence is ambiguous because the order of quantifiers is unclear. Let A be the set of Americans, let D be the set of dreams, and define the predicate $H(a, d)$ to be "American a has dream d.". Now the sentence could mean there is a single dream that every American shares:

$$\exists\, d \in D \; \forall a \in A. \; H(a, d)$$

For example, it might be that every American shares the dream of owning their own home.

Or it could mean that every American has a personal dream:

$$\forall a \in A \; \exists\, d \in D. \; H(a, d)$$

For example, some Americans may dream of a peaceful retirement, while others dream of continuing practicing their profession as long as they live, and still others may dream of being so rich they needn't think at all about work.

Swapping quantifiers in Goldbach's Conjecture creates a patently false statement that every even number ≥ 2 is the sum of *the same* two primes:

$$\underbrace{\exists\, p \in \text{Primes}\ \exists\, q \in \text{Primes}}_{\substack{\text{there exist primes}\\ p \text{ and } q \text{ such that}}}\ \underbrace{\forall n \in \text{Evens.}}_{\substack{\text{for every even}\\ \text{integer } n > 2}}\ n = p + q.$$

Variables Over One Domain

When all the variables in a formula are understood to take values from the same nonempty set, D, it's conventional to omit mention of D. For example, instead of $\forall x \in D\ \exists y \in D.\ Q(x, y)$ we'd write $\forall x \exists y.\ Q(x, y)$. The unnamed nonempty set that x and y range over is called the *domain of discourse*, or just plain *domain*, of the formula.

It's easy to arrange for all the variables to range over one domain. For example, Goldbach's Conjecture could be expressed with all variables ranging over the domain $\mathbb{N}$ as

$$\forall n.\ n \in \text{Evens IMPLIES } (\exists\, p \exists\, q.\ p \in \text{Primes} \land q \in \text{Primes} \land n = p + q).$$

1.5.7 Negating Quantifiers

There is a simple relationship between the two kinds of quantifiers. The following two sentences mean the same thing:

> It is not the case that everyone likes to snowboard.
>
> There exists someone who does not like to snowboard.

In terms of logic notation, this follows from a general property of predicate formulas:

$$\text{NOT}\ \forall x.\ P(x) \quad \text{is equivalent to} \quad \exists x.\ \text{NOT}\ P(x).$$

Similarly, these sentences mean the same thing:

> There does not exist anyone who likes skiing over magma.
>
> Everyone dislikes skiing over magma.

We can express the equivalence in logic notation this way:

$$(\text{NOT}\ \exists x.\ P(x)) \text{ IFF } \forall x.\ \text{NOT}\ P(x). \tag{1.8}$$

The general principle is that *moving a "not" across a quantifier changes the kind of quantifier.*

1.6 Validity

A propositional formula is called *valid* when it evaluates to **T** no matter what truth values are assigned to the individual propositional variables. For example, the propositional version of the Distributive Law is that P AND $(Q$ OR $R)$ is equivalent to $(P$ AND $Q)$ OR $(P$ AND $R)$. This is the same as saying that

$$[P \text{ AND } (Q \text{ OR } R)] \text{ IFF } [(P \text{ AND } Q) \text{ OR } (P \text{ AND } R)]$$

is valid.

The same idea extends to predicate formulas, but to be valid, a formula now must evaluate to true no matter what values its variables may take over any unspecified domain, and no matter what interpretation a predicate variable may be given. For example, we already observed that the rule for negating a quantifier is captured by the valid assertion (1.8).

Another useful example of a valid assertion is

$$\exists x \forall y.\ P(x, y) \text{ IMPLIES } \forall y \exists x.\ P(x, y). \tag{1.9}$$

Here's an explanation why this is valid:

> Let D be the domain for the variables and P_0 be some binary predicate[7] on D. We need to show that if
>
> $$\exists x \in D\ \forall y \in D.\ P_0(x, y) \tag{1.10}$$
>
> holds under this interpretation, then so does
>
> $$\forall y \in D\ \exists x \in D.\ P_0(x, y). \tag{1.11}$$
>
> So suppose (1.10) is true. Then by definition of $\exists$, this means that some element $d_0 \in D$ has the property that
>
> $$\forall y \in D. P_0(d_0, y).$$
>
> By definition of $\forall$, this means that
>
> $$P_0(d_0, d)$$
>
> is true for all $d \in D$. So given any $d \in D$, there is an element in D, namely, d_0, such that $P_0(d_0, d)$ is true. But that's exactly what (1.11) means, so we've proved that (1.11) holds under this interpretation, as required.

We hope this is helpful as an explanation, but we don't really want to call it a "proof." The problem is that with something as basic as (1.9), it's hard to see what more elementary axioms are ok to use in proving it. What the explanation

[7] That is, a predicate that depends on two variables.

above did was translate the logical formula (1.9) into English and then appeal to the meaning, in English, of "for all" and "there exists" as justification. So this wasn't a proof, just an explanation that once you understand what (1.9) means, it becomes obvious.

In contrast to (1.9), the formula

$$\forall y \exists x.\ P(x, y) \text{ IMPLIES } \exists x \forall y.\ P(x, y). \tag{1.12}$$

is *not* valid. We can prove this just by describing an interpretation where the hypothesis, $\forall y \exists x.\ P(x, y)$, is true but the conclusion, $\exists x \forall y.\ P(x, y)$, is not true. For example, let the domain be the integers and $P(x, y)$ mean $x > y$. Then the hypothesis would be true because, given a value, n, for y we could choose the value of x to be $n + 1$, for example. But under this interpretation the conclusion asserts that there is an integer that is bigger than all integers, which is certainly false. An interpretation like this which falsifies an assertion is called a *counter model* to the assertion.

1.6.1 Problems

Class Problems

Problem 1.12.
A media tycoon has an idea for an all-news television network called LNN: The Logic News Network. Each segment will begin with a definition of the domain of discourse and a few predicates. The day's happenings can then be communicated concisely in logic notation. For example, a broadcast might begin as follows:

> "THIS IS LNN. The domain of discourse is {Albert, Ben, Claire, David, Emily}. Let $D(x)$ be a predicate that is true if x is deceitful. Let $L(x, y)$ be a predicate that is true if x likes y. Let $G(x, y)$ be a predicate that is true if x gave gifts to y."

Translate the following broadcasted logic notation into (English) statements.

(a)

$$(\neg(D(\text{Ben}) \vee D(\text{David}))) \longrightarrow (L(\text{Albert}, \text{Ben}) \wedge L(\text{Ben}, \text{Albert}))$$

(b)

$$\forall x\ (x = \text{Claire} \wedge \neg L(x, \text{Emily})) \vee (x \neq \text{Claire} \wedge L(x, \text{Emily})) \wedge$$
$$\forall x\ (x = \text{David} \wedge L(x, \text{Claire})) \vee (x \neq \text{David} \wedge \neg L(x, \text{Claire}))$$

(c)

$$\neg D(\text{Claire}) \longrightarrow (G(\text{Albert}, \text{Ben}) \wedge \exists\, x G(\text{Ben}, x))$$

(d)

$$\forall x \exists y \exists z\ (y \neq z) \wedge L(x, y) \wedge \neg L(x, z)$$

(e) How could you express "Everyone except for Claire likes Emily" using just propositional connectives *without* using any quantifiers ($\forall, \exists$)? Can you generalize to explain how *any* logical formula over this domain of discourse can be expressed without quantifiers? How big would the formula in the previous part be if it was expressed this way?

Problem 1.13.
The goal of this problem is to translate some assertions about binary strings into logic notation. The domain of discourse is the set of all finite-length binary strings: λ, 0, 1, 00, 01, 10, 11, 000, 001, ... (Here λ denotes the empty string.) In your translations, you may use all the ordinary logic symbols (including =), variables, and the binary symbols `0`, `1` denoting 0, 1.

A string like $\texttt{01}x\texttt{0}y$ of binary symbols and variables denotes the *concatenation* of the symbols and the binary strings represented by the variables. For example, if the value of x is `011` and the value of y is `1111`, then the value of $\texttt{01}x\texttt{0}y$ is the binary string `0101101111`.

Here are some examples of formulas and their English translations. Names for these predicates are listed in the third column so that you can reuse them in your solutions (as we do in the definition of the predicate NO-1S below).

Meaning	Formula	Name
x is a prefix of y	$\exists z\ (xz = y)$	PREFIX(x, y)
x is a substring of y	$\exists u \exists v\ (uxv = y)$	SUBSTRING(x, y)
x is empty or a string of 0's	NOT(SUBSTRING$(\texttt{1}, x)$)	NO-1S(x)

(a) x consists of three copies of some string.

(b) x is an even-length string of 0's.

(c) x does not contain both a 0 and a 1.

(d) x is the binary representation of $2^k + 1$ for some integer $k \geq 0$.

(e) An elegant, slightly trickier way to define NO-1S(x) is:

$$\text{PREFIX}(x, \texttt{0}x). \qquad (*)$$

Explain why (*) is true only when x is a string of 0's.

Problem 1.14.
For each of the logical formulas, indicate whether or not it is true when the domain of discourse is $\mathbb{N}$, (the nonnegative integers 0, 1, 2, ...), $\mathbb{Z}$ (the integers), $\mathbb{Q}$ (the rationals), $\mathbb{R}$ (the real numbers), and $\mathbb{C}$ (the complex numbers). Add a brief explanation to the few cases that merit one.

$$\begin{array}{rrrcl}
 & \exists x & (x^2 & = & 2) \\
\forall x & \exists y & (x^2 & = & y) \\
\forall y & \exists x & (x^2 & = & y) \\
\forall x \neq 0 & \exists y & (xy & = & 1) \\
\exists x & \exists y & (x + 2y & = & 2) \wedge (2x + 4y = 5)
\end{array}$$

Problem 1.15.
Show that

$$(\forall x \exists y.\ P(x, y)) \longrightarrow \forall z.\ P(z, z)$$

is not valid by describing a counter-model.

Homework Problems

Problem 1.16.
Express each of the following predicates and propositions in formal logic notation. The domain of discourse is the nonnegative integers, $\mathbb{N}$. Moreover, in addition to the propositional operators, variables and quantifiers, you may define predicates using addition, multiplication, and equality symbols, but no *constants* (like 0, 1,...) and no *exponentiation* (like x^y). For example, the proposition "n is an even number" could be written

$$\exists m.\ (m + m = n).$$

(a) n is the sum of two fourth-powers (a fourth-power is k^4 for some integer k).

Since the constant 0 is not allowed to appear explicitly, the predicate "$x = 0$" can't be written directly, but note that it could be expressed in a simple way as:

$$x + x = x.$$

Then the predicate $x > y$ could be expressed

$$\exists w.\ (y + w = x) \wedge (w \neq 0).$$

Note that we've used "$w \neq 0$" in this formula, even though it's technically not allowed. But since "$w \neq 0$" is equivalent to the allowed formula "$\neg(w + w = w)$," we can use "$w \neq 0$" with the understanding that it abbreviates the real thing. And now that we've shown how to express "$x > y$," it's ok to use it too.

(b) $x = 1$.

(c) m is a divisor of n (notation: $m \mid n$)

(d) n is a prime number (hint: use the predicates from the previous parts)

(e) n is a power of 3.

Problem 1.17.
Translate the following sentence into a predicate formula:

> There is a student who has emailed exactly two other people in the class, besides possibly herself.

The domain of discourse should be the set of students in the class; in addition, the only predicates that you may use are

- equality, and
- $E(x, y)$, meaning that "x has sent e-mail to y."

Chapter 2

Patterns of Proof

2.1 The Axiomatic Method

The standard procedure for establishing truth in mathematics was invented by Euclid, a mathematician working in Alexandria, Egypt around 300 BC. His idea was to begin with five *assumptions* about geometry, which seemed undeniable based on direct experience. (For example, "There is a straight line segment between every pair of points.) Propositions like these that are simply accepted as true are called *axioms*.

Starting from these axioms, Euclid established the truth of many additional propositions by providing "proofs". A *proof* is a sequence of logical deductions from axioms and previously-proved statements that concludes with the proposition in question. You probably wrote many proofs in high school geometry class, and you'll see a lot more in this course.

There are several common terms for a proposition that has been proved. The different terms hint at the role of the proposition within a larger body of work.

- Important propositions are called *theorems*.
- A *lemma* is a preliminary proposition useful for proving later propositions.
- A *corollary* is a proposition that follows in just a few logical steps from a theorem.

The definitions are not precise. In fact, sometimes a good lemma turns out to be far more important than the theorem it was originally used to prove.

Euclid's axiom-and-proof approach, now called the *axiomatic method*, is the foundation for mathematics today. In fact, just a handful of axioms, called the axioms Zermelo-Frankel with Choice (*ZFC*), together with a few logical deduction rules, appear to be sufficient to derive essentially all of mathematics. We'll examine these in Chapter 5.

2.1.1 Our Axioms

The ZFC axioms are important in studying and justifying the foundations of mathematics, but for practical purposes, they are much too primitive. Proving theorems in ZFC is a little like writing programs in byte code instead of a full-fledged programming language —by one reckoning, a formal proof in ZFC that $2 + 2 = 4$ requires more than 20,000 steps! So instead of starting with ZFC, we're going to take a *huge* set of axioms as our foundation: we'll accept all familiar facts from high school math!

This will give us a quick launch, but you may find this imprecise specification of the axioms troubling at times. For example, in the midst of a proof, you may find yourself wondering, "Must I prove this little fact or can I take it as an axiom?" Feel free to ask for guidance, but really there is no absolute answer. Just be up front about what you're assuming, and don't try to evade homework and exam problems by declaring everything an axiom!

2.1.2 Logical Deductions

Logical deductions or *inference rules* are used to prove new propositions using previously proved ones.

A fundamental inference rule is *modus ponens*. This rule says that a proof of P together with a proof that P IMPLIES Q is a proof of Q.

Inference rules are sometimes written in a funny notation. For example, *modus ponens* is written:

Rule.

$$\frac{P, \quad P \text{ IMPLIES } Q}{Q}$$

When the statements above the line, called the *antecedents*, are proved, then we can consider the statement below the line, called the *conclusion* or *consequent*, to also be proved.

A key requirement of an inference rule is that it must be *sound*: any assignment of truth values that makes all the antecedents true must also make the consequent true. So if we start off with true axioms and apply sound inference rules, everything we prove will also be true.

There are many other natural, sound inference rules, for example:

Rule.

$$\frac{P \text{ IMPLIES } Q, \quad Q \text{ IMPLIES } R}{P \text{ IMPLIES } R}$$

Rule.

$$\frac{\text{NOT}(P) \text{ IMPLIES NOT}(Q)}{Q \text{ IMPLIES } P}$$

On the other hand,

Rule.

$$\frac{\text{NOT}(P) \text{ IMPLIES NOT}(Q)}{P \text{ IMPLIES } Q}$$

is not sound: if P is assigned **T** and Q is assigned **F**, then the antecedent is true and the consequent is not.

Note that a propositional inference rule is sound precisely when the conjunction (AND) of all its antecedents implies its consequent.

As with axioms, we will not be too formal about the set of legal inference rules. Each step in a proof should be clear and "logical"; in particular, you should state what previously proved facts are used to derive each new conclusion.

2.2 Proof Templates

In principle, a proof can be *any* sequence of logical deductions from axioms and previously proved statements that concludes with the proposition in question. This freedom in constructing a proof can seem overwhelming at first. How do you even *start* a proof?

Here's the good news: many proofs follow one of a handful of standard templates. Each proof has it own details, of course, but these templates at least provide you with an outline to fill in. We'll go through several of these standard patterns, pointing out the basic idea and common pitfalls and giving some examples. Many of these templates fit together; one may give you a top-level outline while others help you at the next level of detail. And we'll show you other, more sophisticated proof techniques later on.

The recipes below are very specific at times, telling you exactly which words to write down on your piece of paper. You're certainly free to say things your own way instead; we're just giving you something you *could* say so that you're never at a complete loss.

2.2.1 Proving an Implication

Propositions of the form "If P, then Q" are called *implications*. This implication is often rephrased as "P IMPLIES Q."

Here are some examples:

- (Quadratic Formula) If $ax^2 + bx + c = 0$ and $a \neq 0$, then
$$x = \left(-b \pm \sqrt{b^2 - 4ac}\right) / 2a.$$
- (Goldbach's Conjecture) If n is an even integer greater than 2, then n is a sum of two primes.
- If $0 \leq x \leq 2$, then $-x^3 + 4x + 1 > 0$.

There are a couple of standard methods for proving an implication.

2.2.2 Method #1

In order to prove that P IMPLIES Q:

1. Write, "Assume P."

2. Show that Q logically follows.

Example

Theorem 2.2.1. *If* $0 \leq x \leq 2$, *then* $-x^3 + 4x + 1 > 0$.

Before we write a proof of this theorem, we have to do some scratchwork to figure out why it is true.

The inequality certainly holds for $x = 0$; then the left side is equal to 1 and $1 > 0$. As x grows, the $4x$ term (which is positive) initially seems to have greater magnitude than $-x^3$ (which is negative). For example, when $x = 1$, we have $4x = 4$, but $-x^3 = -1$ only. In fact, it looks like $-x^3$ doesn't begin to dominate until $x > 2$. So it seems the $-x^3 + 4x$ part should be nonnegative for all x between 0 and 2, which would imply that $-x^3 + 4x + 1$ is positive.

So far, so good. But we still have to replace all those "seems like" phrases with solid, logical arguments. We can get a better handle on the critical $-x^3 + 4x$ part by factoring it, which is not too hard:

$$-x^3 + 4x = x(2 - x)(2 + x)$$

Aha! For x between 0 and 2, all of the terms on the right side are nonnegative. And a product of nonnegative terms is also nonnegative. Let's organize this blizzard of observations into a clean proof.

Proof. Assume $0 \leq x \leq 2$. Then x, $2 - x$, and $2 + x$ are all nonnegative. Therefore, the product of these terms is also nonnegative. Adding 1 to this product gives a positive number, so:

$$x(2 - x)(2 + x) + 1 > 0$$

Multiplying out on the left side proves that

$$-x^3 + 4x + 1 > 0$$

as claimed. ■

There are a couple points here that apply to all proofs:

- You'll often need to do some scratchwork while you're trying to figure out the logical steps of a proof. Your scratchwork can be as disorganized as you like— full of dead-ends, strange diagrams, obscene words, whatever. But keep your scratchwork separate from your final proof, which should be clear and concise.

- Proofs typically begin with the word "Proof" and end with some sort of doohickey like □ or "q.e.d". The only purpose for these conventions is to clarify where proofs begin and end.

2.2.3 Method #2 - Prove the Contrapositive

An implication ("P IMPLIES Q") is logically equivalent to its *contrapositive*

$$\text{NOT}(Q) \text{ IMPLIES NOT}(P)$$

Proving one is as good as proving the other, and proving the contrapositive is sometimes easier than proving the original statement. If so, then you can proceed as follows:

1. Write, "We prove the contrapositive:" and then state the contrapositive.

2. Proceed as in Method #1.

Example

Theorem 2.2.2. *If r is irrational, then $\sqrt{r}$ is also irrational.*

Recall that rational numbers are equal to a ratio of integers and irrational numbers are not. So we must show that if r is *not* a ratio of integers, then $\sqrt{r}$ is also *not* a ratio of integers. That's pretty convoluted! We can eliminate both *not*'s and make the proof straightforward by considering the contrapositive instead.

Proof. We prove the contrapositive: if $\sqrt{r}$ is rational, then r is rational.

Assume that $\sqrt{r}$ is rational. Then there exist integers a and b such that:

$$\sqrt{r} = \frac{a}{b}$$

Squaring both sides gives:

$$r = \frac{a^2}{b^2}$$

Since a^2 and b^2 are integers, r is also rational. ■

2.2.4 Problems

Homework Problems

Problem 2.1.
Show that $\log_7 n$ is either an integer or irrational, where n is a positive integer. Use whatever familiar facts about integers and primes you need, but explicitly state such facts. (This problem will be graded on the clarity and simplicity of your proof. If you can't figure out how to prove it, ask the staff for help and they'll tell you how.)

2.2.5 Proving an "If and Only If"

Many mathematical theorems assert that two statements are logically equivalent; that is, one holds if and only if the other does. Here is an example that has been known for several thousand years:

> Two triangles have the same side lengths if and only if two side lengths and the angle between those sides are the same.

The phrase "if and only if" comes up so often that it is often abbreviated "iff".

2.2.6 Method #1: Prove Each Statement Implies the Other

The statement "P IFF Q" is equivalent to the two statements "P IMPLIES Q" and "Q IMPLIES P". So you can prove an "iff" by proving *two* implications:

1. Write, "We prove P implies Q and vice-versa."
2. Write, "First, we show P implies Q." Do this by one of the methods in Section 2.2.1.
3. Write, "Now, we show Q implies P." Again, do this by one of the methods in Section 2.2.1.

2.2.7 Method #2: Construct a Chain of Iffs

In order to prove that P is true iff Q is true:

1. Write, "We construct a chain of if-and-only-if implications."
2. Prove P is equivalent to a second statement which is equivalent to a third statement and so forth until you reach Q.

This method sometimes requires more ingenuity than the first, but the result can be a short, elegant proof.

Example

The *standard deviation* of a sequence of values $x_1, x_2, \ldots, x_n$ is defined to be:

$$\sqrt{\frac{(x_1-\mu)^2+(x_2-\mu)^2+\cdots+(x_n-\mu)^2}{n}} \tag{2.1}$$

where μ is the *mean* of the values:

$$\mu ::= \frac{x_1+x_2+\cdots+x_n}{n}$$

Theorem 2.2.3. *The standard deviation of a sequence of values $x_1, \ldots, x_n$ is zero iff all the values are equal to the mean.*

For example, the standard deviation of test scores is zero if and only if everyone scored exactly the class average.

Proof. We construct a chain of "iff" implications, starting with the statement that the standard deviation (2.1) is zero:

$$\sqrt{\frac{(x_1-\mu)^2+(x_2-\mu)^2+\cdots+(x_n-\mu)^2}{n}}=0. \tag{2.2}$$

Now since zero is the only number whose square root is zero, equation (2.2) holds iff

$$(x_1-\mu)^2+(x_2-\mu)^2+\cdots+(x_n-\mu)^2=0. \tag{2.3}$$

Now squares of real numbers are always nonnegative, so every term on the left hand side of equation (2.3) is nonnegative. This means that (2.3) holds iff

$$\text{Every term on the left hand side of (2.3) is zero.} \tag{2.4}$$

But a term $(x_i-\mu)^2$ is zero iff $x_i=\mu$, so (2.4) is true iff

Every x_i equals the mean.

■

2.3 Proof by Cases

Breaking a complicated proof into cases and proving each case separately is a useful, common proof strategy. Here's an amusing example.

Let's agree that given any two people, either they have met or not. If every pair of people in a group has met, we'll call the group a *club*. If every pair of people in a group has not met, we'll call it a group of *strangers*.

Theorem. *Every collection of 6 people includes a club of 3 people or a group of 3 strangers.*

Proof. The proof is by case analysis[1]. Let x denote one of the six people. There are two cases:

1. Among 5 other people besides x, at least 3 have met x.
2. Among the 5 other people, at least 3 have not met x.

Now we have to be sure that at least one of these two cases must hold,[2] but that's easy: we've split the 5 people into two groups, those who have shaken hands with x and those who have not, so one the groups must have at least half the people.

Case 1: Suppose that at least 3 people did meet x.

This case splits into two subcases:

[1] Describing your approach at the outset helps orient the reader.

[2] Part of a case analysis argument is showing that you've covered all the cases. Often this is obvious, because the two cases are of the form "P" and "not P". However, the situation above is not stated quite so simply.

Case 1.1: No pair among those people met each other. Then these people are a group of at least 3 strangers. So the Theorem holds in this subcase.

Case 1.2: Some pair among those people have met each other. Then that pair, together with x, form a club of 3 people. So the Theorem holds in this subcase.

This implies that the Theorem holds in Case 1.

Case 2: Suppose that at least 3 people did not meet x.

This case also splits into two subcases:

Case 2.1: Every pair among those people met each other. Then these people are a club of at least 3 people. So the Theorem holds in this subcase.

Case 2.2: Some pair among those people have not met each other. Then that pair, together with x, form a group of at least 3 strangers. So the Theorem holds in this subcase.

This implies that the Theorem also holds in Case 2, and therefore holds in all cases. ■

2.3.1 Problems

Class Problems

Problem 2.2.
If we raise an irrational number to an irrational power, can the result be rational? Show that it can by considering $\sqrt{2}^{\sqrt{2}}$ and arguing by cases.

Homework Problems

Problem 2.3.
For $n = 40$, the value of polynomial $p(n) ::= n^2 + n + 41$ 1.4, is not prime, as noted in Section 1.5. But we could have predicted based on general principles that no nonconstant polynomial, $q(n)$, with integer coefficients can map each nonnegative integer into a prime number. Prove it.

Hint: Let $c ::= q(0)$ be the constant term of q. Consider two cases: c is not prime, and c is prime. In the second case, note that $q(cn)$ is a multiple of c for all $n \in \mathbb{Z}$. You may assume the familiar fact that the magnitude (absolute value) of any nonconstant polynomial, $q(n)$, grows unboundedly as n grows.

2.3.2 Proof by Contradiction

In a *proof by contradiction* or *indirect proof*, you show that if a proposition were false, then some false fact would be true. Since a false fact can't be true, the proposition had better not be false. That is, the proposition really must be true.

Proof by contradiction is *always* a viable approach. However, as the name suggests, indirect proofs can be a little convoluted. So direct proofs are generally preferable as a matter of clarity.

Method: In order to prove a proposition P by contradiction:

1. Write, "We use proof by contradiction."
2. Write, "Suppose P is false."
3. Deduce something known to be false (a logical contradiction).
4. Write, "This is a contradiction. Therefore, P must be true."

Example

Remember that a number is *rational* if it is equal to a ratio of integers. For example, $3.5 = 7/2$ and $0.1111\cdots = 1/9$ are rational numbers. On the other hand, we'll prove by contradiction that $\sqrt{2}$ is irrational.

Theorem 2.3.1. *$\sqrt{2}$ is irrational.*

Proof. We use proof by contradiction. Suppose the claim is false; that is, $\sqrt{2}$ is rational. Then we can write $\sqrt{2}$ as a fraction n/d in *lowest terms*.

Squaring both sides gives $2 = n^2/d^2$ and so $2d^2 = n^2$. This implies that n is a multiple of 2. Therefore n^2 must be a multiple of 4. But since $2d^2 = n^2$, we know $2d^2$ is a multiple of 4 and so d^2 is a multiple of 2. This implies that d is a multiple of 2.

So the numerator and denominator have 2 as a common factor, which contradicts the fact that n/d is in lowest terms. So $\sqrt{2}$ must be irrational. ∎

2.4 Sets

Propositions of the sort we've considered so far are good for reasoning about individual statements, but not so good for reasoning about a collection of objects. Let's first review a couple mathematical tools for grouping objects and then extend our logical language to cope with such collections.

Informally, a *set* is a bunch of objects, which are called the *elements* of the set. The elements of a set can be just about anything: numbers, points in space, or even other sets. The conventional way to write down a set is to list the elements inside curly-braces. For example, here are some sets:

$$\begin{array}{rcll} A & = & \{\text{Alex}, \text{Tippy}, \text{Shells}, \text{Shadow}\} & \text{dead pets} \\ B & = & \{\text{red}, \text{blue}, \text{yellow}\} & \text{primary colors} \\ C & = & \{\{a,b\}, \{a,c\}, \{b,c\}\} & \text{a set of sets} \end{array}$$

This works fine for small finite sets. Other sets might be defined by indicating how to generate a list of them:

$$D = \{1, 2, 4, 8, 16, \dots\} \qquad \text{the powers of 2}$$

The order of elements is not significant, so $\{x, y\}$ and $\{y, x\}$ are the same set written two different ways. Also, any object is, or is not, an element of a given set —there is no notion of an element appearing more than once in a set.[3] So writing $\{x, x\}$ is just indicating the same thing twice, namely, that x is in the set. In particular, $\{x, x\} = \{x\}$.

The expression $e \in S$ asserts that e is an element of set S. For example, $32 \in D$ and blue $\in B$, but Tailspin $\notin A$ —yet.

Sets are simple, flexible, and everywhere. You'll find some set mentioned in nearly every section of this text.

2.4.1 Some Popular Sets

Mathematicians have devised special symbols to represent some common sets.

symbol	**set**	**elements**
$\emptyset$	the empty set	none
$\mathbb{N}$	nonnegative integers	$\{0, 1, 2, 3, \ldots\}$
$\mathbb{Z}$	integers	$\{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$
$\mathbb{Q}$	rational numbers	$\frac{1}{2},\ -\frac{5}{3},\ 16,$ etc.
$\mathbb{R}$	real numbers	$\pi,\ e,\ -9,\ \sqrt{2},$ etc.
$\mathbb{C}$	complex numbers	$i,\ \frac{19}{2},\ \sqrt{2} - 2i,$ etc.

A superscript "$^+$" restricts a set to its positive elements; for example, $\mathbb{R}^+$ denotes the set of positive real numbers. Similarly, $\mathbb{R}^-$ denotes the set of negative reals.

2.4.2 Comparing and Combining Sets

The expression $S \subseteq T$ indicates that set S is a *subset* of set T, which means that every element of S is also an element of T (it could be that $S = T$). For example, $\mathbb{N} \subseteq \mathbb{Z}$ and $\mathbb{Q} \subseteq \mathbb{R}$ (every rational number is a real number), but $\mathbb{C} \not\subseteq \mathbb{Z}$ (not every complex number is an integer).

As a memory trick, notice that the $\subseteq$ points to the smaller set, just like a $\leq$ sign points to the smaller number. Actually, this connection goes a little further: there is a symbol $\subset$ analogous to $<$. Thus, $S \subset T$ means that S is a subset of T, but the two are *not* equal. So $A \subseteq A$, but $A \not\subset A$, for every set A.

There are several ways to combine sets. Let's define a couple of sets for use in examples:

$$X ::= \{1, 2, 3\}$$
$$Y ::= \{2, 3, 4\}$$

- The *union* of sets X and Y (denoted $X \cup Y$) contains all elements appearing in X or Y or both. Thus, $X \cup Y = \{1, 2, 3, 4\}$.

[3] It's not hard to develop a notion of *multisets* in which elements can occur more than once, but multisets are not ordinary sets.

- The *intersection* of X and Y (denoted $X \cap Y$) consists of all elements that appear in *both* X and Y. So $X \cap Y = \{2, 3\}$.
- The *set difference* of X and Y (denoted $X - Y$) consists of all elements that are in X, but not in Y. Therefore, $X - Y = \{1\}$ and $Y - X = \{4\}$.

2.4.3 Complement of a Set

Sometimes we are focused on a particular domain, D. Then for any subset, A, of D, we define $\overline{A}$ to be the set of all elements of D *not* in A. That is, $\overline{A} ::= D - A$. The set $\overline{A}$ is called the *complement* of A.

For example, when the domain we're working with is the real numbers, the complement of the positive real numbers is the set of negative real numbers together with zero. That is,

$$\overline{\mathbb{R}^+} = \mathbb{R}^- \cup \{0\}.$$

It can be helpful to rephrase properties of sets using complements. For example, two sets, A and B, are said to be *disjoint* iff they have no elements in common, that is, $A \cap B = \emptyset$. This is the same as saying that A is a subset of the complement of B, that is, $A \subseteq \overline{B}$.

2.4.4 Power Set

The set of all the subsets of a set, A, is called the *power set*, $\mathcal{P}(A)$, of A. So $B \in \mathcal{P}(A)$ iff $B \subseteq A$. For example, the elements of $\mathcal{P}(\{1, 2\})$ are $\emptyset, \{1\}, \{2\}$ and $\{1, 2\}$.

More generally, if A has n elements, then there are 2^n sets in $\mathcal{P}(A)$. For this reason, some authors use the notation 2^A instead of $\mathcal{P}(A)$.

2.4.5 Set Builder Notation

An important use of predicates is in *set builder notation*. We'll often want to talk about sets that cannot be described very well by listing the elements explicitly or by taking unions, intersections, etc., of easily-described sets. Set builder notation often comes to the rescue. The idea is to define a *set* using a *predicate*; in particular, the set consists of all values that make the predicate true. Here are some examples of set builder notation:

$$\begin{aligned}
A &::= \{n \in \mathbb{N} \mid n \text{ is a prime and } n = 4k + 1 \text{ for some integer } k\} \\
B &::= \left\{x \in \mathbb{R} \mid x^3 - 3x + 1 > 0\right\} \\
C &::= \left\{a + bi \in \mathbb{C} \mid a^2 + 2b^2 \le 1\right\}
\end{aligned}$$

The set A consists of all nonnegative integers n for which the predicate

"n is a prime and $n = 4k + 1$ for some integer k"

is true. Thus, the smallest elements of A are:

$$5, 13, 17, 29, 37, 41, 53, 57, 61, 73, \ldots.$$

Trying to indicate the set A by listing these first few elements wouldn't work very well; even after ten terms, the pattern is not obvious! Similarly, the set B consists of all real numbers x for which the predicate

$$x^3 - 3x + 1 > 0$$

is true. In this case, an explicit description of the set B in terms of intervals would require solving a cubic equation. Finally, set C consists of all complex numbers $a + bi$ such that:

$$a^2 + 2b^2 \leq 1$$

This is an oval-shaped region around the origin in the complex plane.

2.4.6 Proving Set Equalities

Two sets are defined to be equal if they contain the same elements. That is, $X = Y$ means that $z \in X$ if and only if $z \in Y$, for all elements, z. (This is actually the first of the ZFC axioms.) So set equalities can be formulated and proved as "iff" theorems. For example:

Theorem 2.4.1 (*Distributive Law* for Sets)**.** *Let A, B, and C be sets. Then:*

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C) \tag{2.5}$$

Proof. The equality (5.1) is equivalent to the assertion that

$$z \in A \cap (B \cup C) \quad \text{iff} \quad z \in (A \cap B) \cup (A \cap C) \tag{2.6}$$

for all z. Now we'll prove (2.6) by a chain of iff's.

First we need a rule for distributing a propositional AND operation over an OR operation. It's easy to verify by truth-table that

Lemma 2.4.2. *The propositional formulas*

$$P \text{ AND } (Q \text{ OR } R)$$

and

$$(P \text{ AND } Q) \text{ OR } (P \text{ AND } R)$$

are equivalent.

Now we have

$$\begin{aligned}
& z \in A \cap (B \cup C) \\
& \quad \text{iff} \quad (z \in A) \text{ AND } (z \in B \cup C) && \text{(def of } \cap) \\
& \quad \text{iff} \quad (z \in A) \text{ AND } (z \in B \text{ OR } z \in C) && \text{(def of } \cup) \\
& \quad \text{iff} \quad (z \in A \text{ AND } z \in B) \text{ OR } (z \in A \text{ AND } z \in C) && \text{(Lemma 2.4.2)} \\
& \quad \text{iff} \quad (z \in A \cap B) \text{ OR } (z \in A \cap C) && \text{(def of } \cap) \\
& \quad \text{iff} \quad z \in (A \cap B) \cup (A \cap C) && \text{(def of } \cup)
\end{aligned}$$

■

2.4.7 Glossary of Symbols

symbol	meaning
$::=$	is defined to be
$\wedge$	and
$\vee$	or
$\longrightarrow$	implies
$\neg$	not
$\neg P$	not P
$\overline{P}$	not P
$\longleftrightarrow$	iff, equivalent
$\oplus$	xor
$\exists$	exists
$\forall$	for all
$\in$	is a member of, belongs to
$\subseteq$	is a subset of, is contained by
$\subset$	is a proper subset of, is properly contained by
$\cup$	set union
$\cap$	set intersection
$\overline{A}$	complement of the set A
$\mathcal{P}(A)$	powerset of the set A
$\emptyset$	the empty set, $\{\}$
$\mathbb{N}$	nonnegative integers
$\mathbb{Z}$	integers
$\mathbb{Z}^+$	positive integers
$\mathbb{Z}^-$	negative integers
$\mathbb{Q}$	rational numbers
$\mathbb{R}$	real numbers
$\mathbb{C}$	complex numbers

2.4.8 Problems

Homework Problems

Problem 2.4.
Let A, B, and C be sets. Prove that:

$$A \cup B \cup C = (A - B) \cup (B - C) \cup (C - A) \cup (A \cap B \cap C). \tag{2.7}$$

Hint: P OR Q OR R is equivalent to

$$(P \text{ AND } \overline{Q}) \text{ OR } (Q \text{ AND } \overline{R}) \text{ OR } (R \text{ AND } \overline{P}) \text{ OR } (P \text{ AND } Q \text{ AND } R).$$

2.5 *Good* Proofs in Practice

One purpose of a proof is to establish the truth of an assertion with absolute certainty. Mechanically checkable proofs of enormous length or complexity can accomplish this. But humanly intelligible proofs are the only ones that help someone understand the subject. Mathematicians generally agree that important mathematical results can't be fully understood until their proofs are understood. That is why proofs are an important part of the curriculum.

To be understandable and helpful, more is required of a proof than just logical correctness: a good proof must also be clear. Correctness and clarity usually go together; a well-written proof is more likely to be a correct proof, since mistakes are harder to hide.

In practice, the notion of proof is a moving target. Proofs in a professional research journal are generally unintelligible to all but a few experts who know all the terminology and prior results used in the proof. Conversely, proofs in the first weeks of a beginning Math for Computer Science subject would be regarded as tediously long-winded by a professional mathematician. In fact, what might not be acceptable as a good proof in the first weeks of the subject might be perfectly ok in later weeks. But even so, we can offer some general tips on writing good proofs:

State your game plan. A good proof begins by explaining the general line of reasoning, for example, "We use case analysis" or "We argue by contradiction."

Keep a linear flow. Sometimes proofs are written like mathematical mosaics, with juicy tidbits of independent reasoning sprinkled throughout. This is not good. The steps of an argument should follow one another in an intelligible order.

A proof is an essay, not a calculation. Many students initially write proofs the way they compute integrals. The result is a long sequence of expressions without explanation, making it very hard to follow. This is bad. A good proof usually looks like an essay with some equations thrown in. Use complete sentences.

Avoid excessive symbolism. Your reader is probably good at understanding words, but much less skilled at reading arcane mathematical symbols. So use words where you reasonably can.

Revise and simplify. Your readers will be grateful.

Introduce notation thoughtfully. Sometimes an argument can be greatly simplified by introducing a variable, devising a special notation, or defining a new term. But do this sparingly since you're requiring the reader to remember all that new stuff. And remember to actually *define* the meanings of new variables, terms, or notations; don't just start using them!

Structure long proofs. Long programs are usually broken into a hierarchy of smaller procedures. Long proofs are much the same. Facts needed in your proof that are easily stated, but not readily proved are best pulled out and proved in preliminary lemmas. Also, if you are repeating essentially the same argument over and over, try to capture that argument in a general lemma, which you can cite repeatedly instead.

Be wary of the "obvious". When familiar or truly obvious facts are needed in a proof, it's OK to label them as such and to not prove them. But remember that what's obvious to you, may not be —and typically is not —obvious to your reader.

Most especially, don't use phrases like "clearly" or "obviously" in an attempt to bully the reader into accepting something you're having trouble proving. Also, go on the alert whenever you see one of these phrases in someone else's proof.

Finish. At some point in a proof, you'll have established all the essential facts you need. Resist the temptation to quit and leave the reader to draw the "obvious" conclusion. Instead, tie everything together yourself and explain why the original claim follows.

The analogy between good proofs and good programs extends beyond structure. The same rigorous thinking needed for proofs is essential in the design of critical computer systems. When algorithms and protocols only "mostly work" due to reliance on hand-waving arguments, the results can range from problematic to catastrophic. An early example was the Therac 25, a machine that provided radiation therapy to cancer victims, but occasionally killed them with massive overdoses due to a software race condition. A more recent (August 2004) example involved a single faulty command to a computer system used by United and American Airlines that grounded the entire fleet of both companies— and all their passengers!

It is a certainty that we'll all one day be at the mercy of critical computer systems designed by you and your classmates. So we really hope that you'll develop the ability to formulate rock-solid logical arguments that a system actually does what you think it does!

2.5.1 Problems

Class Problems

Problem 2.5.
Identify exactly where the bugs are in each of the following bogus proofs.[4]

(a) Bogus Claim: $1/8 > 1/4$.

[4]From Stueben, Michael and Diane Sandford. *Twenty Years Before the Blackboard*, Mathematical Association of America, ©1998.

Bogus proof.

$$\begin{gathered} 3 > 2 \\ 3\log_{10}(1/2) > 2\log_{10}(1/2) \\ \log_{10}(1/2)^3 > \log_{10}(1/2)^2 \\ (1/2)^3 > (1/2)^2, \end{gathered}$$

and the claim now follows by the rules for multiplying fractions. ■

(b) *Bogus proof*: $1¢ = \$0.01 = (\$0.1)^2 = (10¢)^2 = 100¢ = \$1$. ■

(c) **Bogus Claim**: If a and b are two equal real numbers, then $a = 0$.

Bogus proof.

$$\begin{aligned} a &= b \\ a^2 &= ab \\ a^2 - b^2 &= ab - b^2 \\ (a-b)(a+b) &= (a-b)b \\ a+b &= b \\ a &= 0. \end{aligned}$$

■

Problem 2.6.
It's a fact that the Arithmetic Mean is at least as large the Geometric Mean, namely,

$$\frac{a+b}{2} \geq \sqrt{ab}$$

for all nonnegative real numbers a and b. But there's something objectionable about the following proof of this fact. What's the objection, and how would you fix it?

Bogus proof.

$$\begin{aligned} \frac{a+b}{2} &\overset{?}{\geq} \sqrt{ab}, && \text{so} \\ a+b &\overset{?}{\geq} 2\sqrt{ab}, && \text{so} \\ a^2 + 2ab + b^2 &\overset{?}{\geq} 4ab, && \text{so} \\ a^2 - 2ab + b^2 &\overset{?}{\geq} 0, && \text{so} \\ (a-b)^2 &\geq 0 && \text{which we know is true.} \end{aligned}$$

The last statement is true because $a-b$ is a real number, and the square of a real number is never negative. This proves the claim. ■

Problem 2.7.
Generalize the proof from lecture (reproduced below) that $\sqrt{2}$ is irrational, for example, how about $\sqrt[3]{2}$? Remember that an irrational number is a number that cannot be expressed as a ratio of two integers.

Theorem. *$\sqrt{2}$ is an irrational number.*

Proof. The proof is by contradiction: assume that $\sqrt{2}$ is rational, that is,

$$\sqrt{2} = \frac{n}{d}, \tag{2.8}$$

where n and d are integers. Now consider the smallest such positive integer denominator, d. We will prove in a moment that the numerator, n, and the denominator, d, are both even. This implies that

$$\frac{n/2}{d/2}$$

is a fraction equal to $\sqrt{2}$ with a smaller positive integer denominator, a contradiction.

> *Since the assumption that $\sqrt{2}$ is rational leads to this contradiction, the assumption must be false. That is, $\sqrt{2}$ is indeed irrational.* This italicized comment on the implication of the contradiction normally goes without saying, but since this is the first 6.042 exercise about proof by contradiction, we've said it.

To prove that n and d have 2 as a common factor, we start by squaring both sides of (2.8) and get $2 = n^2/d^2$, so

$$2d^2 = n^2. \tag{2.9}$$

So 2 is a factor of n^2, which is only possible if 2 is in fact a factor of n.

This means that $n = 2k$ for some integer, k, so

$$n^2 = (2k)^2 = 4k^2. \tag{2.10}$$

Combining (2.9) and (2.10) gives $2d^2 = 4k^2$, so

$$d^2 = 2k^2. \tag{2.11}$$

So 2 is a factor of d^2, which again is only possible if 2 is in fact also a factor of d, as claimed. ■

Problem 2.8.
Here is a different proof that $\sqrt{2}$ is irrational, taken from the American Mathematical Monthly, v.116, #1, Jan. 2009, p.69:

Proof. Suppose for the sake of contradiction that $\sqrt{2}$ is rational, and choose the least integer, $q > 0$, such that $\left(\sqrt{2} - 1\right) q$ is a nonnegative integer. Let $q' ::= \left(\sqrt{2} - 1\right) q$. Clearly $0 < q' < q$. But an easy computation shows that $\left(\sqrt{2} - 1\right) q'$ is a nonnegative integer, contradicting the minimality of q. ■

(a) This proof was written for an audience of college teachers, and is a little more concise than desirable at this point in 6.042. Write out a more complete version which includes an explanation of each step.

(b) Now that you have justified the steps in this proof, do you have a preference for one of these proofs over the other? Why? Discuss these questions with your teammates for a few minutes and summarize your team's answers on your whiteboard.

Problem 2.9.
Here is a generalization of Problem 2.7 that you may not have thought of:

Lemma 2.5.1. *Let the coefficients of the polynomial* $a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{m-1} + x^m$ *be integers. Then any real root of the polynomial is either integral or irrational.*

(a) Explain why Lemma 2.5.1 immediately implies that $\sqrt[m]{k}$ is irrational whenever k is not an mth power of some integer.

(b) Collaborate with your tablemates to write a clear, textbook quality proof of Lemma 2.5.1 on your whiteboard. (Besides clarity and correctness, textbook quality requires good English with proper punctuation. When a real textbook writer does this, it usually takes multiple revisions; if you're satisfied with your first draft, you're probably misjudging.) You may find it helpful to appeal to the following:
Lemma 2.5.2. *If a prime, p, is a factor of some power of an integer, then it is a factor of that integer.*

You may assume Lemma 2.5.2 without writing down its proof, but see if you can explain why it is true.

Homework Problems

Problem 2.10.
The fact that that there are irrational numbers a, b such that a^b is rational was proved in Problem 2.2. Unfortunately, that proof was *nonconstructive*: it didn't reveal a specific pair, a, b, with this property. But in fact, it's easy to do this: let $a ::= \sqrt{2}$ and $b ::= 2 \log_2 3$.

We know $\sqrt{2}$ is irrational, and obviously $a^b = 3$. Finish the proof that this a, b pair works, by showing that $2 \log_2 3$ is irrational.

Chapter 3

Induction

3.1 The Well Ordering Principle

Every *nonempty* set of *nonnegative integers* has a *smallest* element.

This statement is known as The *Well Ordering Principle*. Do you believe it? Seems sort of obvious, right? But notice how tight it is: it requires a *nonempty* set —it's false for the empty set which has *no* smallest element because it has no elements at all! And it requires a set of *nonnegative* integers —it's false for the set of *negative* integers and also false for some sets of nonnegative *rationals* —for example, the set of positive rationals. So, the Well Ordering Principle captures something special about the nonnegative integers.

3.1.1 Well Ordering Proofs

While the Well Ordering Principle may seem obvious, it's hard to see offhand why it is useful. But in fact, it provides one of the most important proof rules in discrete mathematics.

In fact, looking back, we took the Well Ordering Principle for granted in proving that $\sqrt{2}$ is irrational. That proof assumed that for any positive integers m and n, the fraction m/n can be written in *lowest terms*, that is, in the form m'/n' where m' and n' are positive integers with no common factors. How do we know this is always possible?

Suppose to the contrary that there were $m, n \in \mathbb{Z}^+$ such that the fraction m/n cannot be written in lowest terms. Now let C be the set of positive integers that are numerators of such fractions. Then $m \in C$, so C is nonempty. Therefore, by Well Ordering, there must be a smallest integer, $m_0 \in C$. So by definition of C, there is an integer $n_0 > 0$ such that

$$\text{the fraction } \frac{m_0}{n_0} \text{ cannot be written in lowest terms.}$$

This means that m_0 and n_0 must have a common factor, $p > 1$. But

$$\frac{m_0/p}{n_0/p} = \frac{m_0}{n_0},$$

so any way of expressing the left hand fraction in lowest terms would also work for m_0/n_0, which implies

$$\text{the fraction } \frac{m_0/p}{n_0/p} \text{ cannot be in written in lowest terms either.}$$

So by definition of C, the numerator, m_0/p, is in C. But $m_0/p < m_0$, which contradicts the fact that m_0 is the smallest element of C.

Since the assumption that C is nonempty leads to a contradiction, it follows that C must be empty. That is, that there are no numerators of fractions that can't be written in lowest terms, and hence there are no such fractions at all.

We've been using the Well Ordering Principle on the sly from early on!

3.1.2 Template for Well Ordering Proofs

More generally, there is a standard way to use Well Ordering to prove that some property, $P(n)$ holds for every nonnegative integer, n. Here is a standard way to organize such a well ordering proof:

To prove that "$P(n)$ is true for all $n \in \mathbb{N}$" using the Well Ordering Principle:

- Define the set, C, of *counterexamples* to P being true. Namely, define[a]

$$C ::= \{n \in \mathbb{N} \mid P(n) \text{ is false}\}.$$

- Assume for proof by contradiction that C is nonempty.
- By the Well Ordering Principle, there will be a smallest element, n, in C.
- Reach a contradiction (somehow) —often by showing how to use n to find another member of C that is smaller than n. (This is the open-ended part of the proof task.)
- Conclude that C must be empty, that is, no counterexamples exist. QED

[a]The notation $\{n \mid P(n)\}$ means "the set of all elements n, for which $P(n)$ is true.

3.1.3 Summing the Integers

Let's use this this template to prove

Theorem.
$$1 + 2 + 3 + \cdots + n = n(n + 1)/2 \tag{3.1}$$
for all nonnegative integers, n.

First, we better address of a couple of ambiguous special cases before they trip us up:

- If $n = 1$, then there is only one term in the summation, and so $1+2+3+\cdots+n$ is just the term 1. Don't be misled by the appearance of 2 and 3 and the suggestion that 1 and n are distinct terms!

- If $n \leq 0$, then there are no terms at all in the summation. By convention, the sum in this case is 0.

So while the dots notation is convenient, you have to watch out for these special cases where the notation is misleading! (In fact, whenever you see the dots, you should be on the lookout to be sure you understand the pattern, watching out for the beginning and the end.)

We could have eliminated the need for guessing by rewriting the left side of (3.1) with *summation notation*:
$$\sum_{i=1}^{n} i \qquad \text{or} \qquad \sum_{1 \leq i \leq n} i.$$

Both of these expressions denote the sum of all values taken by the expression to the right of the sigma as the variable, i, ranges from 1 to n. Both expressions make it clear what (3.1) means when $n = 1$. The second expression makes it clear that when $n = 0$, there are no terms in the sum, though you still have to know the convention that a sum of no numbers equals 0 (the *product* of no numbers is 1, by the way).

OK, back to the proof:

Proof. By contradiction. Assume that the theorem is *false*. Then, some nonnegative integers serve as *counterexamples* to it. Let's collect them in a set:
$$C ::= \left\{ n \in \mathbb{N} \mid 1 + 2 + 3 + \cdots + n \neq \frac{n(n+1)}{2} \right\}.$$

By our assumption that the theorem admits counterexamples, C is a nonempty set of nonnegative integers. So, by the Well Ordering Principle, C has a minimum element, call it c. That is, c is the *smallest counterexample* to the theorem.

Since c is the smallest counterexample, we know that (3.1) is false for $n = c$ but true for all nonnegative integers $n < c$. But (3.1) is true for $n = 0$, so $c > 0$. This means $c-1$ is a nonnegative integer, and since it is less than c, equation (3.1) is true for $c-1$. That is,

$$1+2+3+\cdots+(c-1)=\frac{(c-1)c}{2}.$$

But then, adding c to both sides we get

$$1+2+3+\cdots+(c-1)+c=\frac{(c-1)c}{2}+c=\frac{c^2-c+2c}{2}=\frac{c(c+1)}{2},$$

which means that (3.1) does hold for c, after all! This is a contradiction, and we are done. ■

3.1.4 Factoring into Primes

We've previously taken for granted the *Prime Factorization Theorem* that every integer greater than one has a unique[1] expression as a product of prime numbers. This is another of those familiar mathematical facts which are not really obvious. We'll prove the uniqueness of prime factorization in a later chapter, but well ordering gives an easy proof that every integer greater than one can be expressed as *some* product of primes.

Theorem 3.1.1. *Every natural number can be factored as a product of primes.*

Proof. The proof is by Well Ordering.

Let C be the set of all integers greater than one that cannot be factored as a product of primes. We assume C is not empty and derive a contradiction.

If C is not empty, there is a least element, $n \in C$, by Well Ordering. The n can't be prime, because a prime by itself is considered a (length one) product of primes and no such products are in C.

So n must be a product of two integers a and b where $1 < a, b < n$. Since a and b are smaller than the smallest element in C, we know that $a, b \notin C$. In other words, a can be written as a product of primes $p_1 p_2 \cdots p_k$ and b as a product of primes $q_1 \cdots q_l$. Therefore, $n = p_1 \cdots p_k q_1 \cdots q_l$ can be written as a product of primes, contradicting the claim that $n \in C$. Our assumption that $C \neq \emptyset$ must therefore be false. ■

3.1.5 Problems

Class Problems

Problem 3.1.

The proof below uses the Well Ordering Principle to prove that every amount of postage that can be paid exactly using only 6 cent and 15 cent stamps, is divisible by 3. Let the notation "$j \mid k$" indicate that integer j is a divisor of integer k, and

[1]...unique up to the order in which the prime factors appear

let $S(n)$ mean that exactly n cents postage can be paid using only 6 and 15 cent stamps. Then the proof shows that

$$S(n) \text{ IMPLIES } 3 \mid n, \quad \text{for all nonnegative integers } n. \tag{*}$$

Fill in the missing portions (indicated by "...") of the following proof of (*).

> Let C be the set of *counterexamples* to (*), namely[2]
>
> $$C ::= \{n \mid \ldots\}$$
>
> Assume for the purpose of obtaining a contradiction that C is nonempty. Then by the WOP, there is a smallest number, $m \in C$. This m must be positive because....
>
> But if $S(m)$ holds and m is positive, then $S(m-6)$ or $S(m-15)$ must hold, because....
>
> So suppose $S(m-6)$ holds. Then $3 \mid (m-6)$, because...
>
> But if $3 \mid (m-6)$, then obviously $3 \mid m$, contradicting the fact that m is a counterexample.
>
> Next suppose $S(m-15)$ holds. Then the proof for $m-6$ carries over directly for $m-15$ to yield a contradiction in this case as well. Since we get a contradiction in both cases, we conclude that...
>
> which proves that (*) holds.

Problem 3.2.

The proof below uses the Well Ordering Principle to prove that every amount of postage that can be paid exactly, using only 10 cent and 15 cent stamps, is divisible by 5. Let $S(n)$ mean that exactly n cents postage can be paid using only 10 and 15 cent stamps. Then the proof shows that

$$S(n) \text{ IMPLIES } 5 \mid n, \quad \text{for all nonnegative integers } n. \tag{*}$$

Fill in the missing portions (indicated by "...") of the following proof of (*).

> Let C be the set of *counterexamples* to (*), namely
>
> $C ::= \{n \mid \ldots$ ________________________________ $\}$
>
> Assume for the purpose of obtaining a contradiction that C is nonempty. Then by the WOP, there is a smallest number, $m \in C$. This m must be positive because...

[2]The notation "$\{n \mid \ldots\}$" means "the set of elements, n, such that ...."

But if $S(m)$ holds and m is positive, then $S(m-10)$ or $S(m-15)$ must hold, because....

So suppose $S(m-10)$ holds. Then $5 \mid (m-10)$, because...

But if $5 \mid (m-10)$, then obviously $5 \mid m$, contradicting the fact that m is a counterexample.

Next suppose $S(m-15)$ holds. Then the proof for $m-10$ carries over directly for $m-15$ to yield a contradiction in this case as well. Since we get a contradiction in both cases, we conclude that...

which proves that (*) holds.

Problem 3.3.

Euler's Conjecture in 1769 was that there are no positive integer solutions to the equation

$$a^4 + b^4 + c^4 = d^4.$$

Integer values for a, b, c, d that do satisfy this equation, were first discovered in 1986. So Euler guessed wrong, but it took more two hundred years to prove it.

Now let's consider Lehman's equation, similar to Euler's but with some coefficients:

$$8a^4 + 4b^4 + 2c^4 = d^4 \tag{3.2}$$

Prove that Lehman's equation (3.2) really does not have any positive integer solutions.

Hint: Consider the minimum value of a among all possible solutions to (3.2).

Problem 3.4.

Use the Well Ordering Principle to prove that

$$\sum_{k=0}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}. \tag{3.3}$$

for all nonnegative integers, n.

Homework Problems

Problem 3.5.
Use the Well Ordering Principle to prove that any integer greater than or equal to 8 can be represented as the sum of integer multiples of 3 and 5.

3.2 Induction

Induction is by far the most powerful and commonly-used proof technique in discrete mathematics and computer science. In fact, the use of induction is a defining characteristic of *discrete* —as opposed to *continuous* —mathematics. To understand how it works, suppose there is a professor who brings to class a bottomless bag of assorted miniature candy bars. She offers to share the candy in the following way. First, she lines the students up in order. Next she states two rules:

1. The student at the beginning of the line gets a candy bar.
2. If a student gets a candy bar, then the following student in line also gets a candy bar.

Let's number the students by their order in line, starting the count with 0, as usual in Computer Science. Now we can understand the second rule as a short description of a whole sequence of statements:

- If student 0 gets a candy bar, then student 1 also gets one.
- If student 1 gets a candy bar, then student 2 also gets one.
- If student 2 gets a candy bar, then student 3 also gets one.

$$\vdots$$

Of course this sequence has a more concise mathematical description:

> If student n gets a candy bar, then student $n+1$ gets a candy bar, for all nonnegative integers n.

So suppose you are student 17. By these rules, are you entitled to a miniature candy bar? Well, student 0 gets a candy bar by the first rule. Therefore, by the second rule, student 1 also gets one, which means student 2 gets one, which means student 3 gets one as well, and so on. By 17 applications of the professor's second rule, you get your candy bar! Of course the rules actually guarantee a candy bar to *every* student, no matter how far back in line they may be.

3.2.1 Ordinary Induction

The reasoning that led us to conclude every student gets a candy bar is essentially all there is to induction.

The Principle of Induction.

Let $P(n)$ be a predicate. If

- $P(0)$ is true, and
- $P(n)$ IMPLIES $P(n+1)$ for all nonnegative integers, n,

then

- $P(m)$ is true for all nonnegative integers, m.

Since we're going to consider several useful variants of induction in later sections, we'll refer to the induction method described above as *ordinary induction* when we need to distinguish it. Formulated as a proof rule, this would be

Rule. ***Induction Rule***

$$\frac{P(0), \quad \forall n \in \mathbb{N}\,[P(n) \text{ IMPLIES } P(n+1)]}{\forall m \in \mathbb{N}.\, P(m)}$$

This general induction rule works for the same intuitive reason that all the students get candy bars, and we hope the explanation using candy bars makes it clear why the soundness of the ordinary induction can be taken for granted. In fact, the rule is so obvious that it's hard to see what more basic principle could be used to justify it.[3] What's not so obvious is how much mileage we get by using it.

Using Ordinary Induction

Ordinary induction often works directly in proving that some statement about nonnegative integers holds for all of them. For example, here is the formula for the sum of the nonnegative integer that we already proved (equation (3.1)) using the Well Ordering Principle:

Theorem 3.2.1. *For all $n \in \mathbb{N}$,*

$$1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2} \tag{3.4}$$

This time, let's use the Induction Principle to prove Theorem 3.2.1.

Suppose that we define predicate $P(n)$ to be the equation (3.4). Recast in terms of this predicate, the theorem claims that $P(n)$ is true for all $n \in \mathbb{N}$. This is great, because the induction principle lets us reach precisely that conclusion, provided we establish two simpler facts:

[3] But see section 3.3.

- $P(0)$ is true.
- For all $n \in \mathbb{N}$, $P(n)$ IMPLIES $P(n+1)$.

So now our job is reduced to proving these two statements. The first is true because $P(0)$ asserts that a sum of zero terms is equal to $0(0+1)/2 = 0$, which is true by definition. The second statement is more complicated. But remember the basic plan for proving the validity of any implication: *assume* the statement on the left and then *prove* the statement on the right. In this case, we assume $P(n)$ in order to prove $P(n+1)$, which is the equation

$$1 + 2 + 3 + \cdots + n + (n+1) = \frac{(n+1)(n+2)}{2}. \tag{3.5}$$

These two equations are quite similar; in fact, adding $(n+1)$ to both sides of equation (3.4) and simplifying the right side gives the equation (3.5):

$$\begin{aligned} 1 + 2 + 3 + \cdots + n + (n+1) &= \frac{n(n+1)}{2} + (n+1) \\ &= \frac{(n+2)(n+1)}{2} \end{aligned}$$

Thus, if $P(n)$ is true, then so is $P(n+1)$. This argument is valid for every nonnegative integer n, so this establishes the second fact required by the induction principle. Therefore, the induction principle says that the predicate $P(m)$ is true for all nonnegative integers, m, so the theorem is proved.

A Template for Induction Proofs

The proof of Theorem 3.2.1 was relatively simple, but even the most complicated induction proof follows exactly the same template. There are five components:

1. **State that the proof uses induction.** This immediately conveys the overall structure of the proof, which helps the reader understand your argument.

2. **Define an appropriate predicate $P(n)$.** The eventual conclusion of the induction argument will be that $P(n)$ is true for all nonnegative n. Thus, you should define the predicate $P(n)$ so that your theorem is equivalent to (or follows from) this conclusion. Often the predicate can be lifted straight from the claim, as in the example above. The predicate $P(n)$ is called the *induction hypothesis*. Sometimes the induction hypothesis will involve several variables, in which case you should indicate which variable serves as n.

3. **Prove that $P(0)$ is true.** This is usually easy, as in the example above. This part of the proof is called the *base case* or *basis step*.

4. **Prove that $P(n)$ implies $P(n+1)$ for every nonnegative integer n.** This is called the *inductive step*. The basic plan is always the same: assume that $P(n)$

is true and then use this assumption to prove that $P(n+1)$ is true. These two statements should be fairly similar, but bridging the gap may require some ingenuity. Whatever argument you give must be valid for every nonnegative integer n, since the goal is to prove the implications $P(0) \rightarrow P(1)$, $P(1) \rightarrow P(2)$, $P(2) \rightarrow P(3)$, etc. all at once.

5. **Invoke induction.** Given these facts, the induction principle allows you to conclude that $P(n)$ is true for all nonnegative n. This is the logical capstone to the whole argument, but it is so standard that it's usual not to mention it explicitly,

Explicitly labeling the *base case* and *inductive step* may make your proofs clearer.

A Clean Writeup

The proof of Theorem 3.2.1 given above is perfectly valid; however, it contains a lot of extraneous explanation that you won't usually see in induction proofs. The writeup below is closer to what you might see in print and should be prepared to produce yourself.

Proof. We use induction. The induction hypothesis, $P(n)$, will be equation (3.4).

Base case: $P(0)$ is true, because both sides of equation (3.4) equal zero when $n = 0$.

Inductive step: Assume that $P(n)$ is true, where n is any nonnegative integer. Then

$$\begin{aligned} 1 + 2 + 3 + \cdots + n + (n+1) &= \frac{n(n+1)}{2} + (n+1) && \text{(by induction hypothesis)} \\ &= \frac{(n+1)(n+2)}{2} && \text{(by simple algebra)} \end{aligned}$$

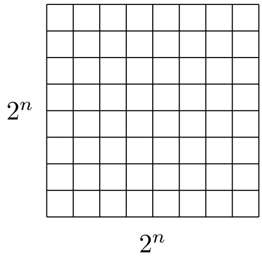
which proves $P(n+1)$.

So it follows by induction that $P(n)$ is true for all nonnegative n. ■

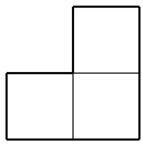
Induction was helpful for *proving the correctness* of this summation formula, but not helpful for *discovering* it in the first place. Tricks and methods for finding such formulas will appear in a later chapter.

Courtyard Tiling

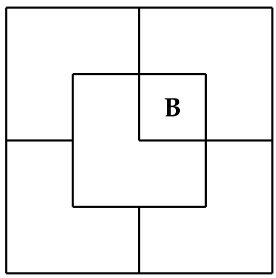
During the development of MIT's famous Stata Center, costs rose further and further over budget, and there were some radical fundraising ideas. One rumored plan was to install a big courtyard with dimensions $2^n \times 2^n$:



One of the central squares would be occupied by a statue of a wealthy potential donor. Let's call him "Bill". (In the special case $n = 0$, the whole courtyard consists of a single central square; otherwise, there are four central squares.) A complication was that the building's unconventional architect, Frank Gehry, was alleged to require that only special L-shaped tiles be used:



A courtyard meeting these constraints exists, at least for $n = 2$:



For larger values of n, is there a way to tile a $2^n \times 2^n$ courtyard with L-shaped tiles and a statue in the center? Let's try to prove that this is so.

Theorem 3.2.2. *For all $n \geq 0$ there exists a tiling of a $2^n \times 2^n$ courtyard with Bill in a central square.*

Proof. (doomed attempt) The proof is by induction. Let $P(n)$ be the proposition that there exists a tiling of a $2^n \times 2^n$ courtyard with Bill in the center.

Base case: $P(0)$ is true because Bill fills the whole courtyard.

Inductive step: Assume that there is a tiling of a $2^n \times 2^n$ courtyard with Bill in the center for some $n \geq 0$. We must prove that there is a way to tile a $2^{n+1} \times 2^{n+1}$ courtyard with Bill in the center .... ■

Now we're in trouble! The ability to tile a smaller courtyard with Bill in the center isn't much help in tiling a larger courtyard with Bill in the center. We haven't figured out how to bridge the gap between $P(n)$ and $P(n + 1)$.

So if we're going to prove Theorem 3.2.2 by induction, we're going to need some *other* induction hypothesis than simply the statement about n that we're trying to prove.

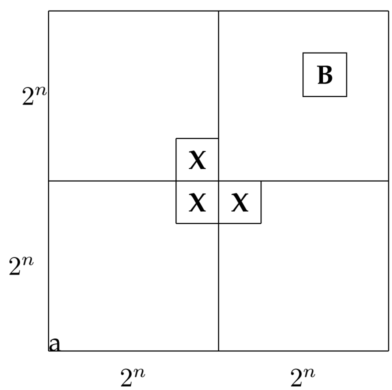
When this happens, your first fallback should be to look for a *stronger* induction hypothesis; that is, one which implies your previous hypothesis. For example, we could make $P(n)$ the proposition that for *every* location of Bill in a $2^n \times 2^n$ courtyard, there exists a tiling of the remainder.

This advice may sound bizarre: "If you can't prove something, try to prove something grander!" But for induction arguments, this makes sense. In the inductive step, where you have to prove $P(n)$ IMPLIES $P(n+1)$, you're in better shape because you can *assume* $P(n)$, which is now a more powerful statement. Let's see how this plays out in the case of courtyard tiling.

Proof. (successful attempt) The proof is by induction. Let $P(n)$ be the proposition that for every location of Bill in a $2^n \times 2^n$ courtyard, there exists a tiling of the remainder.

Base case: $P(0)$ is true because Bill fills the whole courtyard.

Inductive step: Assume that $P(n)$ is true for some $n \geq 0$; that is, for every location of Bill in a $2^n \times 2^n$ courtyard, there exists a tiling of the remainder. Divide the $2^{n+1} \times 2^{n+1}$ courtyard into four quadrants, each $2^n \times 2^n$. One quadrant contains Bill (**B** in the diagram below). Place a temporary Bill (**X** in the diagram) in each of the three central squares lying outside this quadrant:



Now we can tile each of the four quadrants by the induction assumption. Replacing the three temporary Bills with a single L-shaped tile completes the job. This proves that $P(n)$ implies $P(n+1)$ for all $n \geq 0$. The theorem follows as a special case. ■

This proof has two nice properties. First, not only does the argument guarantee that a tiling exists, but also it gives an algorithm for finding such a tiling. Second, we have a stronger result: if Bill wanted a statue on the edge of the courtyard, away from the pigeons, we could accommodate him!

Strengthening the induction hypothesis is often a good move when an induction proof won't go through. But keep in mind that the stronger assertion must actually be *true*; otherwise, there isn't much hope of constructing a valid proof! Sometimes finding just the right induction hypothesis requires trial, error, and insight. For example, mathematicians spent almost twenty years trying to prove or disprove the conjecture that "Every planar graph is 5-choosable"[4]. Then, in 1994, Carsten Thomassen gave an induction proof simple enough to explain on a napkin. The key turned out to be finding an extremely clever induction hypothesis; with that in hand, completing the argument is easy!

A Faulty Induction Proof

False Theorem. *All horses are the same color.*

Notice that no n is mentioned in this assertion, so we're going to have to reformulate it in a way that makes an n explicit. In particular, we'll (falsely) prove that

False Theorem 3.2.3. *In every set of $n \geq 1$ horses, all the horses are the same color.*

This a statement about all integers $n \geq 1$ rather ≥ 0, so it's natural to use a slight variation on induction: prove $P(1)$ in the base case and then prove that $P(n)$ implies $P(n+1)$ for all $n \geq 1$ in the inductive step. This is a perfectly valid variant of induction and is *not* the problem with the proof below.

False proof. The proof is by induction on n. The induction hypothesis, $P(n)$, will be

$$\text{In every set of } n \text{ horses, all are the same color.} \tag{3.6}$$

Base case: ($n = 1$). $P(1)$ is true, because in a set of horses of size 1, there's only one horse, and this horse is definitely the same color as itself.

Inductive step: Assume that $P(n)$ is true for some $n \geq 1$. that is, assume that in every set of n horses, all are the same color. Now consider a set of $n + 1$ horses:

$$h_1,\ h_2,\ \ldots,\ h_n,\ h_{n+1}$$

By our assumption, the first n horses are the same color:

$$\underbrace{h_1,\ h_2,\ \ldots,\ h_n,}_{\text{same color}}\ h_{n+1}$$

Also by our assumption, the last n horses are the same color:

$$h_1,\ \underbrace{h_2,\ \ldots,\ h_n,\ h_{n+1}}_{\text{same color}}$$

[4] 5-choosability is a slight generalization of 5-colorability. Although every planar graph is 4-colorable and therefore 5-colorable, not every planar graph is 4-choosable. If this all sounds like nonsense, don't panic. We'll discuss graphs, planarity, and coloring in a later chapter.

So h_1 is the same color as the remaining horses besides h_{n+1}, and likewise h_{n+1} is the same color as the remaining horses besides h_1. So h_1 and h_{n+1} are the same color. That is, horses $h_1, h_2, \ldots, h_{n+1}$ must all be the same color, and so $P(n+1)$ is true. Thus, $P(n)$ implies $P(n+1)$.

By the principle of induction, $P(n)$ is true for all $n \geq 1$. ■

We've proved something false! Is math broken? Should we all become poets? No, this proof has a mistake.

The error in this argument is in the sentence that begins, "So h_1 and h_{n+1} are the same color." The "..." notation creates the impression that there are some remaining horses besides h_1 and h_{n+1}. However, this is not true when $n = 1$. In that case, the first set is just h_1 and the second is h_2, and there are no remaining horses besides them. So h_1 and h_2 need not be the same color!

This mistake knocks a critical link out of our induction argument. We proved $P(1)$ and we *correctly* proved $P(2) \longrightarrow P(3)$, $P(3) \longrightarrow P(4)$, etc. But we failed to prove $P(1) \longrightarrow P(2)$, and so everything falls apart: we can not conclude that $P(2)$, $P(3)$, etc., are true. And, of course, these propositions are all false; there are horses of a different color.

Students sometimes claim that the mistake in the proof is because $P(n)$ is false for $n \geq 2$, and the proof assumes something false, namely, $P(n)$, in order to prove $P(n+1)$. You should think about how to explain to such a student why this claim would get no credit on a Math for Computer Science exam.

Induction versus Well Ordering

The Induction Axiom looks nothing like the Well Ordering Principle, but these two proof methods are closely related. In fact, as the examples above suggest, we can take any Well Ordering proof and reformat it into an Induction proof. Conversely, it's equally easy to take any Induction proof and reformat it into a Well Ordering proof.

So what's the difference? Well, sometimes induction proofs are clearer because they resemble recursive procedures that reduce handling an input of size $n+1$ to handling one of size n. On the other hand, Well Ordering proofs sometimes seem more natural, and also come out slightly shorter. The choice of method is really a matter of style—but style does matter.

3.2.2 Problems

Class Problems

Problem 3.6.
Use induction to prove that

$$1^3 + 2^3 + \cdots + n^3 = \left(\frac{n(n+1)}{2}\right)^2. \tag{3.7}$$

for all $n \geq 1$.

Remember to formally

1. Declare proof by induction.
2. Identify the induction hypothesis $P(n)$.
3. Establish the base case.
4. Prove that $P(n) \Rightarrow P(n+1)$.
5. Conclude that $P(n)$ holds for all $n \geq 1$.

as in the five part template.

Problem 3.7.
Prove by induction on n that

$$1 + r + r^2 + \cdots + r^n = \frac{r^{n+1} - 1}{r - 1} \tag{3.8}$$

for all $n \in \mathbb{N}$ and numbers $r \neq 1$.

Problem 3.8.
Prove by induction:

$$1 + \frac{1}{4} + \frac{1}{9} + \cdots + \frac{1}{n^2} < 2 - \frac{1}{n}, \tag{3.9}$$

for all $n > 1$.

Problem 3.9. **(a)** Prove by induction that a $2^n \times 2^n$ courtyard with a 1×1 statue of Bill in *a corner* can be covered with L-shaped tiles. (Do not assume or reprove the (stronger) result of Theorem 3.2.2 that Bill can be placed anywhere. The point of this problem is to show a different induction hypothesis that works.)

(b) Use the result of part (a) to prove the original claim that there is a tiling with Bill in the middle.

Problem 3.10.
Find the flaw in the following bogus proof that $a^n = 1$ for all nonnegative integers n, whenever a is a nonzero real number.

Bogus proof. The proof is by induction on n, with hypothesis

$$P(n) ::= \forall k \leq n.\, a^k = 1,$$

where k is a nonnegative integer valued variable.

Base Case: $P(0)$ is equivalent to $a^0 = 1$, which is true by definition of a^0. (By convention, this holds even if $a = 0$.)

Inductive Step: By induction hypothesis, $a^k = 1$ for all $k \in \mathbb{N}$ such that $k \leq n$. But then

$$a^{n+1} = \frac{a^n \cdot a^n}{a^{n-1}} = \frac{1 \cdot 1}{1} = 1,$$

which implies that $P(n+1)$ holds. It follows by induction that $P(n)$ holds for all $n \in \mathbb{N}$, and in particular, $a^n = 1$ holds for all $n \in \mathbb{N}$.

■

Problem 3.11.
We've proved in two different ways that

$$1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}$$

But now we're going to prove a *contradictory* theorem!

False Theorem. *For all $n \geq 0$,*

$$2 + 3 + 4 + \cdots + n = \frac{n(n+1)}{2}$$

Proof. We use induction. Let $P(n)$ be the proposition that $2 + 3 + 4 + \cdots + n = n(n+1)/2$.

Base case: $P(0)$ is true, since both sides of the equation are equal to zero. (Recall that a sum with no terms is zero.)

Inductive step: Now we must show that $P(n)$ implies $P(n+1)$ for all $n \geq 0$. So suppose that $P(n)$ is true; that is, $2 + 3 + 4 + \cdots + n = n(n+1)/2$. Then we can reason as follows:

$$\begin{aligned} 2 + 3 + 4 + \cdots + n + (n+1) &= [2 + 3 + 4 + \cdots + n] + (n+1) \\ &= \frac{n(n+1)}{2} + (n+1) \\ &= \frac{(n+1)(n+2)}{2} \end{aligned}$$

Above, we group some terms, use the assumption $P(n)$, and then simplify. This shows that $P(n)$ implies $P(n+1)$. By the principle of induction, $P(n)$ is true for all $n \in \mathbb{N}$. ■

Where exactly is the error in this proof?

Homework Problems

Problem 3.12.

Claim 3.2.4. *If a collection of positive integers (not necessarily distinct) has sum $n \geq 1$, then the collection has product at most $3^{n/3}$.*

For example, the collection 2, 2, 3, 4, 4, 7 has the sum:

$$2+2+3+4+4+7 \quad = \quad 22$$

On the other hand, the product is:

$$\begin{aligned} 2 \cdot 2 \cdot 3 \cdot 4 \cdot 4 \cdot 7 &= 1344 \\ &\leq 3^{22/3} \\ &\approx 3154.2 \end{aligned}$$

(a) Use strong induction to prove that $n \leq 3^{n/3}$ for every integer $n \geq 0$.

(b) Prove the claim using induction or strong induction. (You may find it easier to use induction on the *number of positive integers in the collection* rather than induction on the sum n.)

Problem 3.13.

For any binary string, α, let $\text{num}(\alpha)$ be the nonnegative integer it represents in binary notation. For example, $\text{num}(\texttt{10}) = 2$, and $\text{num}(\texttt{0101}) = 5$.

An $n+1$-*bit adder* adds two $n+1$-bit binary numbers. More precisely, an $n+1$-bit adder takes two length $n+1$ binary strings

$$\alpha_n ::= a_n \ldots a_1 a_0,$$
$$\beta_n ::= b_n \ldots b_1 b_0,$$

and a binary digit, c_0, as inputs, and produces a length $n+1$ binary string

$$\sigma_n ::= s_n \ldots s_1 s_0,$$

and a binary digit, c_{n+1}, as outputs, and satisfies the specification:

$$\text{num}(\alpha_n) + \text{num}(\beta_n) + c_0 = 2^{n+1} c_{n+1} + \text{num}(\sigma_n). \tag{3.10}$$

There is a straighforward way to implement an $n+1$-bit adder as a digital circuit: an $n+1$-bit *ripple-carry circuit* has $1 + 2(n+1)$ binary inputs

$$a_n, \ldots, a_1, a_0, b_n, \ldots, b_1, b_0, c_0,$$

and $n+2$ binary outputs,

$$c_{n+1}, s_n, \dots, s_1, s_0.$$

As in Problem 1.9, the ripple-carry circuit is specified by the following formulas:

$$s_i ::= a_i \text{ XOR } b_i \text{ XOR } c_i \tag{3.11}$$

$$c_{i+1} ::= (a_i \text{ AND } b_i) \text{ OR } (a_i \text{ AND } c_i) \text{ OR } (b_i \text{ AND } c_i),. \tag{3.12}$$

for $0 \le i \le n$.

(a) Verify that definitions (3.19) and (3.20) imply that

$$a_n + b_n + c_n = 2c_{n+1} + s_n. \tag{3.13}$$

for all $n \in \mathbb{N}$.

(b) Prove by induction on n that an $n+1$-bit ripple-carry circuit really is an $n+1$-bit adder, that is, its outputs satisfy (3.18).

Hint: You may assume that, by definition of binary representation of integers,

$$\text{num}(\alpha_{n+1}) = a_{n+1}2^{n+1} + \text{num}(\alpha_n). \tag{3.14}$$

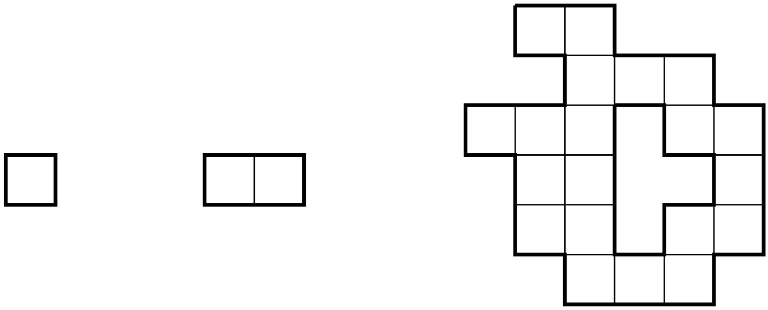
Problem 3.14.
The 6.042 mascot, Theory Hippotamus, made a startling discovery while playing with his prized collection of unit squares over the weekend. Here is what happened.

First, Theory Hippotamus put his favorite unit square down on the floor as in Figure 3.2 (a). He noted that the length of the periphery of the resulting shape was 4, an even number. Next, he put a second unit square down next to the first so that the two squares shared an edge as in Figure 3.2 (b). He noticed that the length of the periphery of the resulting shape was now 6, which is also an even number. (The periphery of each shape in the figure is indicated by a thicker line.) Theory Hippotamus continued to place squares so that each new square shared an edge with at least one previously-placed square and no squares overlapped. Eventually, he arrived at the shape in Figure 3.2 (c). He realized that the length of the periphery of this shape was 36, which is again an even number.

Our plucky porcine pal is perplexed by this peculiar pattern. Use induction on the number of squares to prove that the length of the periphery is always even, no matter how many squares Theory Hippotamus places or how he arranges them.

3.2.3 Strong Induction

A useful variant of induction is called *strong induction*. Strong Induction and Ordinary Induction are used for exactly the same thing: proving that a predicate $P(n)$ is true for all $n \in \mathbb{N}$.



(a) (b) (c)

Figure 3.1: Some shapes that Theory Hippotamus created.

Principle of Strong Induction. Let $P(n)$ be a predicate. If

- $P(0)$ is true, and
- for all $n \in \mathbb{N}$, $P(0)$, $P(1)$, $\ldots$, $P(n)$ *together* imply $P(n+1)$,

then $P(n)$ is true for all $n \in \mathbb{N}$.

Rule. ***Strong Induction Rule***

$$\frac{P(0), \quad \forall n \in \mathbb{N}[(\forall m \leq n.\, P(m)) \text{ IMPLIES } P(n+1)]}{\forall n \in \mathbb{N}.\, P(n)}$$

The only change from the ordinary induction principle is that strong induction allows you to assume more stuff in the inductive step of your proof! In an ordinary induction argument, you assume that $P(n)$ is true and try to prove that $P(n+1)$ is also true. In a strong induction argument, you may assume that $P(0)$, $P(1)$, $\ldots$, and $P(n)$ are *all* true when you go to prove $P(n+1)$. These extra assumptions can only make your job easier.

Products of Primes

As a first example, we'll use strong induction to re-prove Theorem 3.1.1 which we previously proved using Well Ordering.

Lemma 3.2.5. *Every integer greater than 1 is a product of primes.*

Proof. We will prove Lemma 3.2.5 by strong induction, letting the induction hypothesis, $P(n)$, be

n is a product of primes.

So Lemma 3.2.5 will follow if we prove that $P(n)$ holds for all $n \geq 2$.

Base Case: ($n = 2$) $P(2)$ is true because 2 is prime, and so it is a length one product of primes by convention.

Inductive step: Suppose that $n \geq 2$ and that i is a product of primes for every integer i where $2 \leq i < n + 1$. We must show that $P(n + 1)$ holds, namely, that $n + 1$ is also a product of primes. We argue by cases:

If $n + 1$ is itself prime, then it is a length one product of primes by convention, so $P(n + 1)$ holds in this case.

Otherwise, $n + 1$ is not prime, which by definition means $n + 1 = km$ for some integers k, m such that $2 \leq k, m < n + 1$. Now by strong induction hypothesis, we know that k is a product of primes. Likewise, m is a product of primes. it follows immediately that $km = n$ is also a product of primes. Therefore, $P(n + 1)$ holds in this case as well.

So $P(n + 1)$ holds in any case, which completes the proof by strong induction that $P(n)$ holds for all nonnegative integers, n.

■

Making Change

The country Inductia, whose unit of currency is the Strong, has coins worth 3Sg (3 Strongs) and 5Sg. Although the Inductians have some trouble making small change like 4Sg or 7Sg, it turns out that they can collect coins to make change for any number that is at least 8 Strongs.

Strong induction makes this easy to prove for $n+1 \geq 11$, because then $(n+1) - 3 \geq 8$, so by strong induction the Inductians can make change for exactly $(n+1) - 3$ Strongs, and then they can add a 3Sg coin to get $(n + 1)$Sg. So the only thing to do is check that they can make change for all the amounts from 8 to 10Sg, which is not too hard to do.

Here's a detailed writeup using the official format:

Proof. We prove by strong induction that the Inductians can make change for any amount of at least 8Sg. The induction hypothesis, $P(n)$ will be:

There is a collection of coins whose value is $n + 8$ Strongs.

Base case: $P(0)$ is vacuously true.

Inductive step: We assume $P(i)$ holds for all $i \leq n$, and prove that $P(n + 1)$ holds. We argue by cases:

Case ($n + 1 < 8$): $P(n + 1)$ is vacuously true in this case.

Base case: $P(0)$ is true because a 3Sg coin together with 5Sgcoin makes 8Sg.

Inductive step: We assume $P(m)$ holds for all $m \leq n$, and prove that $P(n + 1)$ holds. We argue by cases:

Case ($n+1=1$): We have to make $(n+1)+8 = 9$Sg. We can do this using three 3Sg coins.

Case ($n+1=2$): We have to make $(n+1)+8 = 10$Sg. Use two 5Sg coins.

Case ($n+1 \geq 3$): Then $0 \leq n-2 \leq n$, so by the strong induction hypothesis, the Inductians can make change for $n-2$ Strong. Now by adding a 3Sg coin, they can make change for $(n+1)$Sg.

So in any case, $P(n+1)$ is true, and we conclude by strong induction that for all $n = 0, 1, \ldots,$ the Inductians can make change for $n+8$ Strong. That is, they can make change for any number of eight or more Strong.

■

The Stacking Game

Here is another exciting game that's surely about to sweep the nation :-) !

You begin with a stack of n boxes. Then you make a sequence of moves. In each move, you divide one stack of boxes into two nonempty stacks. The game ends when you have n stacks, each containing a single box. You earn points for each move; in particular, if you divide one stack of height $a+b$ into two stacks with heights a and b, then you score ab points for that move. Your overall score is the sum of the points that you earn for each move. What strategy should you use to maximize your total score?

As an example, suppose that we begin with a stack of $n = 10$ boxes. Then the game might proceed as follows:

Stack Heights										**Score**
$\underline{10}$										
5	$\underline{5}$									25 points
$\underline{5}$	3	2								6
$\underline{4}$	3	2	1							4
2	$\underline{3}$	2	1	2						4
$\underline{2}$	2	2	1	2	1					2
1	$\underline{2}$	2	1	2	1	1				1
1	1	$\underline{2}$	1	2	1	1	1			1
1	1	1	1	$\underline{2}$	1	1	1	1		1
1	1	1	1	1	1	1	1	1	1	1
						Total Score			=	45 points

On each line, the underlined stack is divided in the next step. Can you find a better strategy?

Analyzing the Game

Let's use strong induction to analyze the unstacking game. We'll prove that your score is determined entirely by the number of boxes —your strategy is irrelevant!

Theorem 3.2.6. *Every way of unstacking n blocks gives a score of $n(n-1)/2$ points.*

There are a couple technical points to notice in the proof:

- The template for a strong induction proof is exactly the same as for ordinary induction.
- As with ordinary induction, we have some freedom to adjust indices. In this case, we prove $P(1)$ in the base case and prove that $P(1), \ldots, P(n)$ imply $P(n+1)$ for all $n \geq 1$ in the inductive step.

Proof. The proof is by strong induction. Let $P(n)$ be the proposition that every way of unstacking n blocks gives a score of $n(n-1)/2$.

Base case: If $n = 1$, then there is only one block. No moves are possible, and so the total score for the game is $1(1-1)/2 = 0$. Therefore, $P(1)$ is true.

Inductive step: Now we must show that $P(1), \ldots, P(n)$ imply $P(n+1)$ for all $n \geq 1$. So assume that $P(1), \ldots, P(n)$ are all true and that we have a stack of $n+1$ blocks. The first move must split this stack into substacks with positive sizes a and b where $a + b = n + 1$ and $0 < a, b \leq n$. Now the total score for the game is the sum of points for this first move plus points obtained by unstacking the two resulting substacks:

$$
\begin{aligned}
\text{total score} &= (\text{score for 1st move}) \\
&\quad + (\text{score for unstacking } a \text{ blocks}) \\
&\quad + (\text{score for unstacking } b \text{ blocks}) \\
&= ab + \frac{a(a-1)}{2} + \frac{b(b-1)}{2} && \text{by } P(a) \text{ and } P(b) \\
&= \frac{(a+b)^2 - (a+b)}{2} = \frac{(a+b)((a+b)-1)}{2} \\
&= \frac{(n+1)n}{2}
\end{aligned}
$$

This shows that $P(1), P(2), \ldots, P(n)$ imply $P(n+1)$.

Therefore, the claim is true by strong induction. ■

3.2.4 Strong Induction versus Induction

Is strong induction really "stronger" than ordinary induction? You can assume a lot more when proving the induction step, so it may seem that strong induction is much more powerful, but it's not. Strong induction may make it easier to prove a proposition, but any proof by strong induction can be reformatted to prove the same thing by ordinary induction (using a slightly more complicated induction hypothesis). Again, the choice of method is a matter of style.

When you're doing a proof by strong induction, you should say so: it will help your reader to know that $P(n+1)$ may not follow directly from just $P(n)$.

3.3 Induction versus Well Ordering

3.3.1 Problems

Practice Problems

Problem 3.15.
Find all possible (nonzero) amounts of postage that can be paid exactly using 3 and 5 cent stamps. Use induction to prove that your answer is correct.

Hint: Let $S(n)$ mean that exactly n cents of postage can be paid using only 3 and 5 cent stamps. Prove that the following proposition is true as part of your solution.

$$\forall n.\ (n \geq 8) \text{ IMPLIES } S(n).$$

Class Problems

Problem 3.16.
Use induction to prove that

$$1^3 + 2^3 + \cdots + n^3 = \left(\frac{n(n+1)}{2}\right)^2. \tag{3.15}$$

for all $n \geq 1$.

Remember to formally

1. Declare proof by induction.
2. Identify the induction hypothesis $P(n)$.
3. Establish the base case.
4. Prove that $P(n) \Rightarrow P(n+1)$.
5. Conclude that $P(n)$ holds for all $n \geq 1$.

as in the five part template.

Problem 3.17.
Prove by induction on n that

$$1 + r + r^2 + \cdots + r^n = \frac{r^{n+1} - 1}{r - 1} \tag{3.16}$$

for all $n \in \mathbb{N}$ and numbers $r \neq 1$.

Problem 3.18.
Prove by induction:

$$1 + \frac{1}{4} + \frac{1}{9} + \cdots + \frac{1}{n^2} < 2 - \frac{1}{n}, \tag{3.17}$$

for all $n > 1$.

Problem 3.19. **(a)** Prove by induction that a $2^n \times 2^n$ courtyard with a 1×1 statue of Bill in *a corner* can be covered with L-shaped tiles. (Do not assume or reprove the (stronger) result of Theorem 3.2.2 that Bill can be placed anywhere. The point of this problem is to show a different induction hypothesis that works.)

(b) Use the result of part (a) to prove the original claim that there is a tiling with Bill in the middle.

Problem 3.20.
Find the flaw in the following bogus proof that $a^n = 1$ for all nonnegative integers n, whenever a is a nonzero real number.

Bogus proof. The proof is by induction on n, with hypothesis

$$P(n) ::= \forall k \leq n.\, a^k = 1,$$

where k is a nonnegative integer valued variable.

Base Case: $P(0)$ is equivalent to $a^0 = 1$, which is true by definition of a^0. (By convention, this holds even if $a = 0$.)

Inductive Step: By induction hypothesis, $a^k = 1$ for all $k \in \mathbb{N}$ such that $k \leq n$. But then

$$a^{n+1} = \frac{a^n \cdot a^n}{a^{n-1}} = \frac{1 \cdot 1}{1} = 1,$$

which implies that $P(n+1)$ holds. It follows by induction that $P(n)$ holds for all $n \in \mathbb{N}$, and in particular, $a^n = 1$ holds for all $n \in \mathbb{N}$.

■

Problem 3.21.
We've proved in two different ways that

$$1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}$$

But now we're going to prove a *contradictory* theorem!

False Theorem. *For all $n \geq 0$,*

$$2 + 3 + 4 + \cdots + n = \frac{n(n+1)}{2}$$

Proof. We use induction. Let $P(n)$ be the proposition that $2 + 3 + 4 + \cdots + n = n(n + 1)/2$.
Base case: $P(0)$ is true, since both sides of the equation are equal to zero. (Recall that a sum with no terms is zero.)
Inductive step: Now we must show that $P(n)$ implies $P(n + 1)$ for all $n \geq 0$. So suppose that $P(n)$ is true; that is, $2 + 3 + 4 + \cdots + n = n(n + 1)/2$. Then we can reason as follows:

$$\begin{aligned} 2 + 3 + 4 + \cdots + n + (n + 1) &= [2 + 3 + 4 + \cdots + n] + (n + 1) \\ &= \frac{n(n+1)}{2} + (n + 1) \\ &= \frac{(n+1)(n+2)}{2} \end{aligned}$$

Above, we group some terms, use the assumption $P(n)$, and then simplify. This shows that $P(n)$ implies $P(n + 1)$. By the principle of induction, $P(n)$ is true for all $n \in \mathbb{N}$. ∎

Where exactly is the error in this proof?

Problem 3.22.
Define the *potential*, $p(S)$, of a stack of blocks, S, to be $k(k - 1)/2$ where k is the number of blocks in S. Define the potential, $p(A)$, of a set of stacks, A, to be the sum of the potentials of the stacks in A.

Generalize Theorem 3.2.6 about scores in the stacking game to show that for any set of stacks, A, if a sequence of moves starting with A leads to another set of stacks, B, then $p(A) \geq p(B)$, and the score for this sequence of moves is $p(A) - p(B)$.

Hint: Try induction on the number of moves to get from A to B.

Homework Problems

Problem 3.23.

Claim 3.3.1. *If a collection of positive integers (not necessarily distinct) has sum $n \geq 1$, then the collection has product at most $3^{n/3}$.*

For example, the collection 2, 2, 3, 4, 4, 7 has the sum:

$$2 + 2 + 3 + 4 + 4 + 7 \quad = \quad 22$$

On the other hand, the product is:

$$\begin{aligned} 2 \cdot 2 \cdot 3 \cdot 4 \cdot 4 \cdot 7 &= 1344 \\ &\leq 3^{22/3} \\ &\approx 3154.2 \end{aligned}$$

(a) Use strong induction to prove that $n \leq 3^{n/3}$ for every integer $n \geq 0$.

(b) Prove the claim using induction or strong induction. (You may find it easier to use induction on the *number of positive integers in the collection* rather than induction on the sum n.)

Problem 3.24.
For any binary string, α, let $\text{num}(\alpha)$ be the nonnegative integer it represents in binary notation. For example, $\text{num}(\texttt{10}) = 2$, and $\text{num}(\texttt{0101}) = 5$.

An $n+1$-*bit adder* adds two $n+1$-bit binary numbers. More precisely, an $n+1$-bit adder takes two length $n+1$ binary strings

$$\alpha_n ::= a_n \ldots a_1 a_0,$$
$$\beta_n ::= b_n \ldots b_1 b_0,$$

and a binary digit, c_0, as inputs, and produces a length $n+1$ binary string

$$\sigma_n ::= s_n \ldots s_1 s_0,$$

and a binary digit, c_{n+1}, as outputs, and satisfies the specification:

$$\text{num}(\alpha_n) + \text{num}(\beta_n) + c_0 = 2^{n+1} c_{n+1} + \text{num}(\sigma_n). \tag{3.18}$$

There is a straighforward way to implement an $n+1$-bit adder as a digital circuit: an $n+1$-bit *ripple-carry circuit* has $1 + 2(n+1)$ binary inputs

$$a_n, \ldots, a_1, a_0, b_n, \ldots, b_1, b_0, c_0,$$

and $n+2$ binary outputs,

$$c_{n+1}, s_n, \ldots, s_1, s_0.$$

As in Problem 1.9, the ripple-carry circuit is specified by the following formulas:

$$s_i ::= a_i \ \text{XOR} \ b_i \ \text{XOR} \ c_i \tag{3.19}$$
$$c_{i+1} ::= (a_i \ \text{AND} \ b_i) \ \text{OR} \ (a_i \ \text{AND} \ c_i) \ \text{OR} \ (b_i \ \text{AND} \ c_i),. \tag{3.20}$$

for $0 \leq i \leq n$.

(a) Verify that definitions (3.19) and (3.20) imply that

$$a_n + b_n + c_n = 2c_{n+1} + s_n. \tag{3.21}$$

for all $n \in \mathbb{N}$.

(b) Prove by induction on n that an $n+1$-bit ripple-carry circuit really is an $n+1$-bit adder, that is, its outputs satisfy (3.18).

Hint: You may assume that, by definition of binary representation of integers,

$$\text{num}\left(\alpha_{n+1}\right) = a_{n+1}2^{n+1} + \text{num}\left(\alpha_n\right). \tag{3.22}$$

Problem 3.25.
The 6.042 mascot, Theory Hippotamus, made a startling discovery while playing with his prized collection of unit squares over the weekend. Here is what happened.

First, Theory Hippotamus put his favorite unit square down on the floor as in Figure 3.2 (a). He noted that the length of the periphery of the resulting shape was 4, an even number. Next, he put a second unit square down next to the first so that the two squares shared an edge as in Figure 3.2 (b). He noticed that the length of the periphery of the resulting shape was now 6, which is also an even number. (The periphery of each shape in the figure is indicated by a thicker line.) Theory Hippotamus continued to place squares so that each new square shared an edge with at least one previously-placed square and no squares overlapped. Eventually, he arrived at the shape in Figure 3.2 (c). He realized that the length of the periphery of this shape was 36, which is again an even number.

Our plucky porcine pal is perplexed by this peculiar pattern. Use induction on the number of squares to prove that the length of the periphery is always even, no matter how many squares Theory Hippotamus places or how he arranges them.

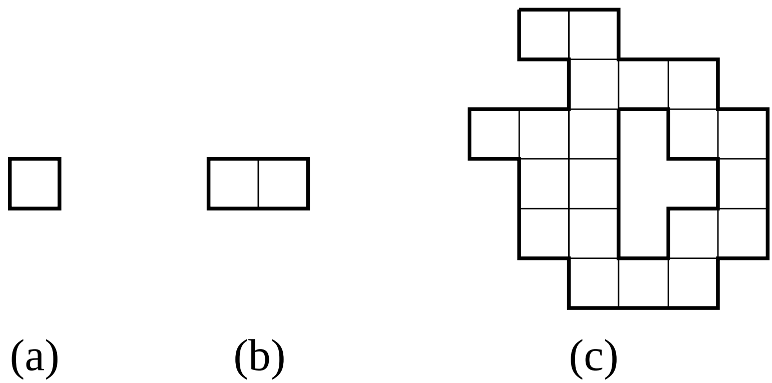


Figure 3.2: Some shapes that Theory Hippotamus created.

Problem 3.26.
Find all possible (nonzero) amounts of postage that can be paid exactly using 3 and 7 cent stamps. Use induction to prove that your answer is correct.

Hint: Let $S(n)$ mean that exactly n cents of postage can be paid using only 3 and 7 cent stamps. Prove that the following proposition is true as part of your solution.

$$\forall n.\ (n \geq 12) \text{ IMPLIES } S(n).$$

Problem 3.27.
A group of $n \geq 1$ people can be divided into teams, each containing either 4 or 7 people. What are all the possible values of n? Use induction to prove that your answer is correct.

Problem 3.28.
The following Lemma is true, but the *proof* given for it below is defective. Pinpoint *exactly* where the proof first makes an unjustified step and explain why it is unjustified.

Lemma 3.3.2. *For any prime p and positive integers $n, x_1, x_2, \ldots, x_n$, if $p \mid x_1 x_2 \ldots x_n$, then $p \mid x_i$ for some $1 \leq i \leq n$.*

False proof. Proof by strong induction on n. The induction hypothesis, $P(n)$, is that Lemma holds for n.

Base case $n = 1$: When $n = 1$, we have $p \mid x_1$, therefore we can let $i = 1$ and conclude $p \mid x_i$.

Induction step: Now assuming the claim holds for all $k \leq n$, we must prove it for $n + 1$.

So suppose $p \mid x_1 x_2 \ldots x_{n+1}$. Let $y_n = x_n x_{n+1}$, so $x_1 x_2 \ldots x_{n+1} = x_1 x_2 \ldots x_{n-1} y_n$. Since the righthand side of this equality is a product of n terms, we have by induction that p divides one of them. If $p \mid x_i$ for some $i < n$, then we have the desired i. Otherwise $p \mid y_n$. But since y_n is a product of the two terms x_n, x_{n+1}, we have by strong induction that p divides one of them. So in this case $p \mid x_i$ for $i = n$ or $i = n + 1$. ■

Chapter 4

Number Theory

Number theory is the study of the integers. *Why* anyone would want to study the integers is not immediately obvious. First of all, what's to know? There's 0, there's 1, 2, 3, and so on, and, oh yeah, -1, -2, ... Which one don't you understand? Second, what practical value is there in it? The mathematician G. H. Hardy expressed pleasure in its impracticality when he wrote:

> [Number theorists] may be justified in rejoicing that there is one science, at any rate, and that their own, whose very remoteness from ordinary human activities should keep it gentle and clean.

Hardy was specially concerned that number theory not be used in warfare; he was a pacifist. You may applaud his sentiments, but he got it wrong: Number Theory underlies modern cryptography, which is what makes secure online communication possible. Secure communication is of course crucial in war —which may leave poor Hardy spinning in his grave. It's also central to online commerce. Every time you buy a book from Amazon, check your grades on WebSIS, or use a PayPal account, you are relying on number theoretic algorithms.

4.1 Divisibility

Since we'll be focussing on properties of the integers, we'll adopt the default convention in this chapter that *variables range over integers*, $\mathbb{Z}$.

The nature of number theory emerges as soon as we consider the *divides* relation

$$a \text{ divides } b \quad \text{iff} \quad ak = b \text{ for some } k.$$

The notation, $a \mid b$, is an abbreviation for "a divides b." If $a \mid b$, then we also say that b is a *multiple* of a. A consequence of this definition is that every number divides zero.

This seems simple enough, but let's play with this definition. The Pythagoreans, an ancient sect of mathematical mystics, said that a number is *perfect* if it equals the sum of its positive integral divisors, excluding itself. For example, $6 = 1+2+3$ and $28 = 1+2+4+7+14$ are perfect numbers. On the other hand, 10 is not perfect because $1+2+5 = 8$, and 12 is not perfect because $1+2+3+4+6 = 16$. Euclid characterized all the *even* perfect numbers around 300 BC. But is there an *odd* perfect number? More than two thousand years later, we still don't know! All numbers up to about 10^{300} have been ruled out, but no one has proved that there isn't an odd perfect number waiting just over the horizon.

So a half-page into number theory, we've strayed past the outer limits of human knowledge! This is pretty typical; number theory is full of questions that are easy to pose, but incredibly difficult to answer. Interestingly, we'll see that computer scientists have found ways to turn some of these difficulties to their advantage.

Don't Panic —we're going to stick to some relatively benign parts of number theory. These super-hard unsolved problems rarely get put on exams :-) .

4.1.1 Facts About Divisibility

The lemma below states some basic facts about divisibility that are *not* difficult to prove:

Lemma 4.1.1. *The following statements about divisibility hold.*

1. *If $a \mid b$, then $a \mid bc$ for all c.*

2. *If $a \mid b$ and $b \mid c$, then $a \mid c$.*

3. *If $a \mid b$ and $a \mid c$, then $a \mid sb + tc$ for all s and t.*

4. *For all $c \neq 0$, $a \mid b$ if and only if $ca \mid cb$.*

Proof. We'll prove only part 2.; the other proofs are similar.

Proof of 2.: Since $a \mid b$, there exists an integer k_1 such that $ak_1 = b$. Since $b \mid c$, there exists an integer k_2 such that $bk_2 = c$. Substituting ak_1 for b in the second equation gives $(ak_1)k_2 = c$. So $a(k_1k_2) = c$, which implies that $a \mid c$.

■

A number $p > 1$ with no positive divisors other than 1 and itself is called a *prime*. Every other number greater than 1 is called *composite*. For example, 2, 3, 5, 7, 11, and 13 are all prime, but 4, 6, 8, and 9 are composite. Because of its special properties, the number 1 is considered to be neither prime nor composite.

Famous Conjectures in Number Theory

Fermat's Last Theorem There are no positive integers x, y, and z such that

$$x^n + y^n = z^n$$

for some integer $n > 2$. In a book he was reading around 1630, Fermat claimed to have a proof but not enough space in the margin to write it down. Wiles finally gave a proof of the theorem in 1994, after seven years of working in secrecy and isolation in his attic. His proof did not fit in any margin.

Goldbach Conjecture Every even integer greater than two equal to the sum of two primes. For example, $4 = 2 + 2$, $6 = 3 + 3$, $8 = 3 + 5$, etc. The conjecture holds for all numbers up to 10^{16}. In 1939 Schnirelman proved that every even number can be written as the sum of not more than 300,000 primes, which was a start. Today, we know that every even number is the sum of at most 6 primes.

Twin Prime Conjecture There are infinitely many primes p such that $p + 2$ is also a prime. In 1966 Chen showed that there are infinitely many primes p such that $p + 2$ is the product of at most two primes. So the conjecture is known to be *almost* true!

Primality Testing There is an efficient way to determine whether a number is prime. A naive search for factors of an integer n takes a number of steps proportional to $\sqrt{n}$, which is exponential in the *size* of n in decimal or binary notation. All known procedures for prime checking blew up like this on various inputs. Finally in 2002, an amazingly simple, new method was discovered by Agrawal, Kayal, and Saxena, which showed that prime testing only required a polynomial number of steps. Their paper began with a quote from Gauss emphasizing the importance and antiquity of the problem even in his time— two centuries ago. So prime testing is definitely not in the category of infeasible problems requiring an exponentially growing number of steps in bad cases.

Factoring Given the product of two large primes $n = pq$, there is no efficient way to recover the primes p and q. The best known algorithm is the "number field sieve", which runs in time proportional to:

$$e^{1.9(\ln n)^{1/3}(\ln \ln n)^{2/3}}$$

This is infeasible when n has 300 digits or more.

4.1.2 When Divisibility Goes Bad

As you learned in elementary school, if one number does *not* evenly divide another, you get a "quotient" and a "remainder" left over. More precisely:

Theorem 4.1.2 (Division Theorem)**.** [1] *Let n and d be integers such that $d > 0$. Then there exists a unique pair of integers q and r, such that*

$$n = q \cdot d + r \text{ AND } 0 \leq r < d. \tag{4.1}$$

The number q is called the *quotient* and the number r is called the *remainder* of n divided by d. We use the notation $\mathrm{qcnt}(n, d)$ for the quotient and $\mathrm{rem}(n, d)$ for the remainder.

For example, $\mathrm{qcnt}(2716, 10) = 271$ and $\mathrm{rem}(2716, 10) = 6$, since $2716 = 271 \cdot 10 + 6$. Similarly, $\mathrm{rem}(-11, 7) = 3$, since $-11 = (-2) \cdot 7 + 3$. There is a remainder operator built into many programming languages. For example, the expression "32 % 5" evaluates to 2 in Java, C, and C++. However, all these languages treat negative numbers strangely.

We'll take this familiar Division Theorem for granted without proof.

4.1.3 Die Hard

We've previously looked at the Die Hard water jug problem with jugs of sizes 3 and 5, and 3 and 9. A little number theory lets us solve all these silly water jug questions at once. In particular, it will be easy to figure out exactly which amounts of water can be measured out using jugs with capacities a and b.

Finding an Invariant Property

Suppose that we have water jugs with capacities a and b with $b \geq a$. The state of the system is described below with a pair of numbers (x, y), where x is the amount of water in the jug with capacity a and y is the amount in the jug with capacity b. Let's carry out sample operations and see what happens, assuming the b-jug is big enough:

$$\begin{aligned}
(0,0) &\to (a,0) && \text{fill first jug}\\
&\to (0,a) && \text{pour first into second}\\
&\to (a,a) && \text{fill first jug}\\
&\to (2a-b,b) && \text{pour first into second (assuming } 2a \geq b)\\
&\to (2a-b,0) && \text{empty second jug}\\
&\to (0,2a-b) && \text{pour first into second}\\
&\to (a,2a-b) && \text{fill first}\\
&\to (3a-2b,b) && \text{pour first into second (assuming } 3a \geq 2b)
\end{aligned}$$

[1] This theorem is often called the "Division Algorithm," even though it is not what we would call an algorithm.

What leaps out is that at every step, the amount of water in each jug is of the form

$$s \cdot a + t \cdot b \tag{4.2}$$

for some integers s and t. An expression of the form (4.2) is called an *integer linear combination* of a and b, but in this chapter we'll just call it a *linear combination*, since we're only talking integers. So we're suggesting:

Lemma 4.1.3. *Suppose that we have water jugs with capacities a and b. Then the amount of water in each jug is always a linear combination of a and b.*

Lemma 4.1.3 is easy to prove by induction on the number of pourings.

Proof. The induction hypothesis, $P(n)$, is the proposition that after n steps, the amount of water in each jug is a linear combination of a and b.
Base case: $(n = 0)$. $P(0)$ is true, because both jugs are initially empty, and $0 \cdot a + 0 \cdot b = 0$.
Inductive step. We assume by induction hypothesis that after n steps the amount of water in each jug is a linear combination of a and b. There are two cases:

- If we fill a jug from the fountain or empty a jug into the fountain, then that jug is empty or full. The amount in the other jug remains a linear combination of a and b. So $P(n + 1)$ holds.

- Otherwise, we pour water from one jug to another until one is empty or the other is full. By our assumption, the amount in each jug is a linear combination of a and b before we begin pouring:

$$j_1 = s_1 \cdot a + t_1 \cdot b$$
$$j_2 = s_2 \cdot a + t_2 \cdot b$$

  After pouring, one jug is either empty (contains 0 gallons) or full (contains a or b gallons). Thus, the other jug contains either $j_1 + j_2$ gallons, $j_1 + j_2 - a$, or $j_1 + j_2 - b$ gallons, all of which are linear combinations of a and b. So $P(n+1)$ holds in this case as well.

So in any case, $P(n + 1)$ follows, completing the proof by induction. ■

This theorem has an important corollary:

Corollary 4.1.4. *Bruce dies.*

Proof. In Die Hard 6, Bruce has water jugs with capacities 3 and 6 and must form 4 gallons of water. However, the amount in each jug is always of the form $3s + 6t$ by Lemma 4.1.3. This is always a multiple of 3 by Lemma 4.1.1.3, so he cannot measure out 4 gallons. ■

But Lemma 4.1.3 isn't very satisfying. We've just managed to recast a pretty understandable question about water jugs into a complicated question about linear combinations. This might not seem like progress. Fortunately, linear combinations are closely related to something more familiar, namely greatest common divisors, and these will help us solve the water jug problem.

4.2 The Greatest Common Divisor

We've already examined the Euclidean Algorithm for computing $\gcd(a, b)$, the greatest common divisor of a and b. This quantity turns out to be a very valuable piece of information about the relationship between a and b. We'll be making arguments about greatest common divisors all the time.

4.2.1 Linear Combinations and the GCD

The theorem below relates the greatest common divisor to linear combinations. This theorem is *very* useful; take the time to understand it and then remember it!

Theorem 4.2.1. *The greatest common divisor of a and b is equal to the smallest positive linear combination of a and b.*

For example, the greatest common divisor of 52 and 44 is 4. And, sure enough, 4 is a linear combination of 52 and 44:

$$6 \cdot 52 + (-7) \cdot 44 = 4$$

Furthermore, no linear combination of 52 and 44 is equal to a smaller positive integer.

Proof. By the Well Ordering Principle, there is a smallest positive linear combination of a and b; call it m. We'll prove that $m = \gcd(a, b)$ by showing both $\gcd(a, b) \leq m$ and $m \leq \gcd(a, b)$.

First, we show that $\gcd(a, b) \leq m$. Now any common divisor of a and b —that is, any c such that $c \mid a$ and $c \mid b$ —will divide both sa and tb, and therefore also divides $sa + tb$. The $\gcd(a, b)$ is by definition a common divisor of a and b, so

$$\gcd(a, b) \mid sa + tb \tag{4.3}$$

every s and t. In particular, $\gcd(a, b) \mid m$, which implies that $\gcd(a, b) \leq m$.

Now, we show that $m \leq \gcd(a, b)$. We do this by showing that $m \mid a$. A symmetric argument shows that $m \mid b$, which means that m is a common divisor of a and b. Thus, m must be less than or equal to the *greatest* common divisor of a and b.

All that remains is to show that $m \mid a$. By the Division Algorithm, there exists a quotient q and remainder r such that:

$$a = q \cdot m + r \qquad \text{(where } 0 \leq r < m\text{)}$$

Recall that $m = sa + tb$ for some integers s and t. Substituting in for m gives:

$$\begin{aligned} a &= q \cdot (sa + tb) + r, \qquad \text{so} \\ r &= (1 - qs)a + (-qt)b. \end{aligned}$$

We've just expressed r as a linear combination of a and b. However, m is the *smallest* positive linear combination and $0 \leq r < m$. The only possibility is that the remainder r is not positive; that is, $r = 0$. This implies $m \mid a$. ■

Corollary 4.2.2. *An integer is linear combination of a and b iff it is a multiple of* $\gcd(a, b)$.

Proof. By (4.3), every linear combination of a and b is a multiple of $\gcd(a, b)$. Conversely, since $\gcd(a, b)$ is a linear combination of a and b, every multiple of $\gcd(a, b)$ is as well. ∎

Now we can restate the water jugs lemma in terms of the greatest common divisor:

Corollary 4.2.3. *Suppose that we have water jugs with capacities a and b. Then the amount of water in each jug is always a multiple of* $\gcd(a, b)$.

For example, there is no way to form 2 gallons using 1247 and 899 gallon jugs, because 2 is not a multiple of $\gcd(1247, 899) = 29$.

4.2.2 Properties of the Greatest Common Divisor

We'll often make use of some basic $\gcd$ facts:

Lemma 4.2.4. *The following statements about the greatest common divisor hold:*

1. *Every common divisor of a and b divides* $\gcd(a, b)$.
2. $\gcd(ka, kb) = k \cdot \gcd(a, b)$ *for all* $k > 0$.
3. *If* $\gcd(a, b) = 1$ *and* $\gcd(a, c) = 1$, *then* $\gcd(a, bc) = 1$.
4. *If* $a \mid bc$ *and* $\gcd(a, b) = 1$, *then* $a \mid c$.
5. $\gcd(a, b) = \gcd(b, \operatorname{rem}(a, b))$.

Here's the trick to proving these statements: translate the $\gcd$ world to the linear combination world using Theorem 4.2.1, argue about linear combinations, and then translate back using Theorem 4.2.1 again.

Proof. We prove only parts 3. and 4.

Proof of 3. The assumptions together with Theorem 4.2.1 imply that there exist integers s, t, u, and v such that:

$$sa + tb = 1$$
$$ua + vc = 1$$

Multiplying these two equations gives:

$$(sa + tb)(ua + vc) = 1$$

The left side can be rewritten as $a \cdot (asu + btu + csv) + bc(tv)$. This is a linear combination of a and bc that is equal to 1, so $\gcd(a, bc) = 1$ by Theorem 4.2.1.

Proof of 4. Theorem 4.2.1 says that $\gcd(ac, bc)$ is equal to a linear combination of ac and bc. Now $a \mid ac$ trivially and $a \mid bc$ by assumption. Therefore, a divides *every* linear combination of ac and bc. In particular, a divides $\gcd(ac, bc) = c \cdot \gcd(a, b) = c \cdot 1 = c$. The first equality uses part 2. of this lemma, and the second uses the assumption that $\gcd(a, b) = 1$. ∎

Lemma 4.2.4.5 is the preserved invariant from Lemma 8.3.1 that we used to prove partial correctness of the Euclidean Algorithm.

Now let's see if it's possible to make 3 gallons using 21 and 26-gallon jugs. Using Euclid's algorithm:

$$\gcd(26, 21) = \gcd(21, 5) = \gcd(5, 1) = 1.$$

Now 3 is a multiple of 1, so we can't *rule out* the possibility that 3 gallons can be formed. On the other hand, we don't know it can be done.

4.2.3 One Solution for All Water Jug Problems

Can Bruce form 3 gallons using 21 and 26-gallon jugs? This question is not so easy to answer without some number theory.

Corollary 4.2.2 says that 3 can be written as a linear combination of 21 and 26, since 3 is a multiple of $\gcd(21, 26) = 1$. In other words, there exist integers s and t such that:

$$3 = s \cdot 21 + t \cdot 26$$

We don't know what the coefficients s and t are, but we do know that they exist.

Now the coefficient s could be either positive or negative. However, we can readily transform this linear combination into an equivalent linear combination

$$3 = s' \cdot 21 + t' \cdot 26 \tag{4.4}$$

where the coefficient s' is positive. The trick is to notice that if we increase s by 26 in the original equation and decrease t by 21, then the value of the expression $s \cdot 21 + t \cdot 26$ is unchanged overall. Thus, by repeatedly increasing the value of s (by 26 at a time) and decreasing the value of t (by 21 at a time), we get a linear combination $s' \cdot 21 + t' \cdot 26 = 3$ where the coefficient s' is positive. Notice that then t' must be negative; otherwise, this expression would be much greater than 3.

Now here's how to form 3 gallons using jugs with capacities 21 and 26:

Repeat s' times:

1. Fill the 21-gallon jug.

2. Pour all the water in the 21-gallon jug into the 26-gallon jug. Whenever the 26-gallon jug becomes full, empty it out.

At the end of this process, we must have have emptied the 26-gallon jug exactly $|t'|$ times. Here's why: we've taken $s' \cdot 21$ gallons of water from the fountain, and we've poured out some multiple of 26 gallons. If we emptied fewer than $|t'|$ times, then by (4.4), the big jug would be left with at least $3 + 26$ gallons, which is more than it can hold; if we emptied it more times, the big jug would be left containing at most $3 - 26$ gallons, which is nonsense. But once we have emptied the 26-gallon jug exactly $|t'|$ times, equation (4.4) implies that there are exactly 3 gallons left.

Remarkably, we don't even need to know the coefficients s' and t' in order to use this strategy! Instead of repeating the outer loop s' times, we could just repeat *until we obtain 3 gallons*, since that must happen eventually. Of course, we have to keep track of the amounts in the two jugs so we know when we're done. Here's the solution that approach gives:

$$
\begin{array}{ccccccccc}
(0,0) & \xrightarrow{\text{fill 21}} & (21,0) & \xrightarrow{\text{pour 21 into 26}} & (0,21) & & & & \\
 & \xrightarrow{\text{fill 21}} & (21,21) & \xrightarrow{\text{pour 21 into 26}} & (16,26) & \xrightarrow{\text{empty 26}} & (16,0) & \xrightarrow{\text{pour 21 into 26}} & (0,16) \\
 & \xrightarrow{\text{fill 21}} & (21,16) & \xrightarrow{\text{pour 21 into 26}} & (11,26) & \xrightarrow{\text{empty 26}} & (11,0) & \xrightarrow{\text{pour 21 into 26}} & (0,11) \\
 & \xrightarrow{\text{fill 21}} & (21,11) & \xrightarrow{\text{pour 21 into 26}} & (6,26) & \xrightarrow{\text{empty 26}} & (6,0) & \xrightarrow{\text{pour 21 into 26}} & (0,6) \\
 & \xrightarrow{\text{fill 21}} & (21,6) & \xrightarrow{\text{pour 21 into 26}} & (1,26) & \xrightarrow{\text{empty 26}} & (1,0) & \xrightarrow{\text{pour 21 into 26}} & (0,1) \\
 & \xrightarrow{\text{fill 21}} & (21,1) & \xrightarrow{\text{pour 21 into 26}} & (0,22) & & & & \\
 & \xrightarrow{\text{fill 21}} & (21,22) & \xrightarrow{\text{pour 21 into 26}} & (17,26) & \xrightarrow{\text{empty 26}} & (17,0) & \xrightarrow{\text{pour 21 into 26}} & (0,17) \\
 & \xrightarrow{\text{fill 21}} & (21,17) & \xrightarrow{\text{pour 21 into 26}} & (12,26) & \xrightarrow{\text{empty 26}} & (12,0) & \xrightarrow{\text{pour 21 into 26}} & (0,12) \\
 & \xrightarrow{\text{fill 21}} & (21,12) & \xrightarrow{\text{pour 21 into 26}} & (7,26) & \xrightarrow{\text{empty 26}} & (7,0) & \xrightarrow{\text{pour 21 into 26}} & (0,7) \\
 & \xrightarrow{\text{fill 21}} & (21,7) & \xrightarrow{\text{pour 21 into 26}} & (2,26) & \xrightarrow{\text{empty 26}} & (2,0) & \xrightarrow{\text{pour 21 into 26}} & (0,2) \\
 & \xrightarrow{\text{fill 21}} & (21,2) & \xrightarrow{\text{pour 21 into 26}} & (0,23) & & & & \\
 & \xrightarrow{\text{fill 21}} & (21,23) & \xrightarrow{\text{pour 21 into 26}} & (18,26) & \xrightarrow{\text{empty 26}} & (18,0) & \xrightarrow{\text{pour 21 into 26}} & (0,18) \\
 & \xrightarrow{\text{fill 21}} & (21,18) & \xrightarrow{\text{pour 21 into 26}} & (13,26) & \xrightarrow{\text{empty 26}} & (13,0) & \xrightarrow{\text{pour 21 into 26}} & (0,13) \\
 & \xrightarrow{\text{fill 21}} & (21,13) & \xrightarrow{\text{pour 21 into 26}} & (8,26) & \xrightarrow{\text{empty 26}} & (8,0) & \xrightarrow{\text{pour 21 into 26}} & (0,8) \\
 & \xrightarrow{\text{fill 21}} & (21,8) & \xrightarrow{\text{pour 21 into 26}} & (3,26) & \xrightarrow{\text{empty 26}} & (3,0) & \xrightarrow{\text{pour 21 into 26}} & (0,3)
\end{array}
$$

The same approach works regardless of the jug capacities and even regardless the amount we're trying to produce! Simply repeat these two steps until the desired amount of water is obtained:

1. Fill the smaller jug.
2. Pour all the water in the smaller jug into the larger jug. Whenever the larger jug becomes full, empty it out.

By the same reasoning as before, this method eventually generates every multiple of the greatest common divisor of the jug capacities —all the quantities we can possibly produce. No ingenuity is needed at all!

4.2.4 The Pulverizer

We saw that no matter which pair of integers a and b we are given, there is always a pair of integer coefficients s and t such that

$$\gcd(a, b) = sa + tb.$$

The previous subsection gives a roundabout and not very efficient method of finding such coefficients s and t. In Chapter **??** we defined and verified the "Extended

Euclidean GCD algorithm," which is a much more efficient way to find these coefficients. In this section we finally explain where the obscure procedure in Chapter **??** came from by describing it in a way that dates to sixth-century India, where it was called *kuttak*, which means "The Pulverizer."

Suppose we use Euclid's Algorithm to compute the GCD of 259 and 70, for example:

$$\begin{aligned} \gcd(259, 70) &= \gcd(70, 49) && \text{since } \operatorname{rem}(259, 70) = 49 \\ &= \gcd(49, 21) && \text{since } \operatorname{rem}(70, 49) = 21 \\ &= \gcd(21, 7) && \text{since } \operatorname{rem}(49, 21) = 7 \\ &= \gcd(7, 0) && \text{since } \operatorname{rem}(21, 7) = 0 \\ &= 7. \end{aligned}$$

The Pulverizer goes through the same steps, but requires some extra bookkeeping along the way: as we compute $\gcd(a, b)$, we keep track of how to write each of the remainders (49, 21, and 7, in the example) as a linear combination of a and b (this is worthwhile, because our objective is to write the last nonzero remainder, which is the GCD, as such a linear combination). For our example, here is this extra bookkeeping:

x	y	$(\operatorname{rem}(x, y))$		$x - q \cdot y$
259	70	49	$=$	$259 - 3 \cdot 70$
70	49	21	$=$	$70 - 1 \cdot 49$
			$=$	$70 - 1 \cdot (259 - 3 \cdot 70)$
			$=$	$-1 \cdot 259 + 4 \cdot 70$
49	21	7	$=$	$49 - 2 \cdot 21$
			$=$	$(259 - 3 \cdot 70) - 2 \cdot (-1 \cdot 259 + 4 \cdot 70)$
			$=$	$\boxed{3 \cdot 259 - 11 \cdot 70}$
21	7	0		

We began by initializing two variables, $x = a$ and $y = b$. In the first two columns above, we carried out Euclid's algorithm. At each step, we computed $\operatorname{rem}(x, y)$, which can be written in the form $x - q \cdot y$. (Remember that the Division Algorithm says $x = q \cdot y + r$, where r is the remainder. We get $r = x - q \cdot y$ by rearranging terms.) Then we replaced x and y in this equation with equivalent linear combinations of a and b, which we already had computed. After simplifying, we were left with a linear combination of a and b that was equal to the remainder as desired. The final solution is boxed.

4.2.5 Problems

Class Problems

Problem 4.1.
A number is *perfect* if it is equal to the sum of its positive divisors, other than itself.

For example, 6 is perfect, because $6 = 1 + 2 + 3$. Similarly, 28 is perfect, because $28 = 1+2+4+7+14$. Explain why $2^{k-1}(2^k - 1)$ is perfect when $2^k - 1$ is prime.[2]

Problem 4.2. **(a)** Use the Pulverizer to find integers x, y such that

$$x \cdot 50 + y \cdot 21 = \gcd(50, 21).$$

(b) Now find integers x', y' with $y' > 0$ such that

$$x' \cdot 50 + y' \cdot 21 = \gcd(50, 21)$$

Problem 4.3.
For nonzero integers, a, b, prove the following properties of divisibility and GCD'S. (You may use the fact that $\gcd(a, b)$ is an integer linear combination of a and b. You may *not* appeal to uniqueness of prime factorization because the properties below are needed to *prove* unique factorization.)

(a) Every common divisor of a and b divides $\gcd(a, b)$.

(b) If $a \mid bc$ and $\gcd(a, b) = 1$, then $a \mid c$.

(c) If $p \mid ab$ for some prime, p, then $p \mid a$ or $p \mid b$.

(d) Let m be the smallest integer linear combination of a and b that is positive. Show that $m = \gcd(a, b)$.

4.3 The Fundamental Theorem of Arithmetic

We now have almost enough tools to prove something that you probably already know.

Theorem 4.3.1 (Fundamental Theorem of Arithmetic)**.** *Every positive integer n can be written in a unique way as a product of primes:*

$$n = p_1 \cdot p_2 \cdots p_j \qquad (p_1 \leq p_2 \leq \cdots \leq p_j)$$

Notice that the theorem would be false if 1 were considered a prime; for example, 15 could be written as $3 \cdot 5$ or $1 \cdot 3 \cdot 5$ or $1^2 \cdot 3 \cdot 5$. Also, we're relying on a standard convention: the product of an empty set of numbers is defined to be 1, much as the

[2]Euclid proved this 2300 years ago. About 250 years ago, Euler proved the converse: *every* even perfect number is of this form (for a simple proof see `http://primes.utm.edu/notes/proofs/EvenPerfect.html`). As is typical in number theory, apparently simple results lie at the brink of the unknown. For example, it is not known if there are an infinite number of even perfect numbers or any odd perfect numbers at all.

sum of an empty set of numbers is defined to be 0. Without this convention, the theorem would be false for $n = 1$.

There is a certain wonder in the Fundamental Theorem, even if you've known it since you were in a crib. Primes show up erratically in the sequence of integers. In fact, their distribution seems almost random:

$$2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, \dots$$

Basic questions about this sequence have stumped humanity for centuries. And yet we know that every natural number can be built up from primes in *exactly one way*. These quirky numbers are the building blocks for the integers. The Fundamental Theorem is not hard to prove, but we'll need a couple of preliminary facts.

Lemma 4.3.2. *If p is a prime and $p \mid ab$, then $p \mid a$ or $p \mid b$.*

Proof. The greatest common divisor of a and p must be either 1 or p, since these are the only positive divisors of p. If $\gcd(a, p) = p$, then the claim holds, because a is a multiple of p. Otherwise, $\gcd(a, p) = 1$ and so $p \mid b$ by Lemma 4.2.4.4. ■

A routine induction argument extends this statement to:

Lemma 4.3.3. *Let p be a prime. If $p \mid a_1 a_2 \cdots a_n$, then p divides some a_i.*

Now we're ready to prove the Fundamental Theorem of Arithmetic.

Proof. Theorem 3.1.1 showed, using the Well Ordering Principle, that every positive integer can be expressed as a product of primes. So we just have to prove this expression is unique. We will use Well Ordering to prove this too.

The proof is by contradiction: assume, contrary to the claim, that there exist positive integers that can be written as products of primes in more than one way. By the Well Ordering Principle, there is a smallest integer with this property. Call this integer n, and let

$$\begin{aligned} n &= p_1 \cdot p_2 \cdots p_j \\ &= q_1 \cdot q_2 \cdots q_k \end{aligned}$$

be two of the (possibly many) ways to write n as a product of primes. Then $p_1 \mid n$ and so $p_1 \mid q_1 q_2 \cdots q_k$. Lemma 4.3.3 implies that p_1 divides one of the primes q_i. But since q_i is a prime, it must be that $p_1 = q_i$. Deleting p_1 from the first product and q_i from the second, we find that n/p_1 is a positive integer *smaller* than n that can also be written as a product of primes in two distinct ways. But this contradicts the definition of n as the smallest such positive integer. ■

4.3.1 Problems

Class Problems

Problem 4.4. **(a)** Let $m = 2^9 5^{24} 11^7 17^{12}$ and $n = 2^3 7^{22} 11^{211} 13^1 17^9 19^2$. What is the $\gcd(m, n)$? What is the *least common multiple*, $\text{lcm}(m, n)$, of m and n? Verify that

$$\gcd(m, n) \cdot \text{lcm}(m, n) = mn. \tag{4.5}$$

The Prime Number Theorem

Let $\pi(x)$ denote the number of primes less than or equal to x. For example, $\pi(10) = 4$ because 2, 3, 5, and 7 are the primes less than or equal to 10. Primes are very irregularly distributed, so the growth of π is similarly erratic. However, the Prime Number Theorem gives an approximate answer:

$$\lim_{x \to \infty} \frac{\pi(x)}{x / \ln x} = 1$$

Thus, primes gradually taper off. As a rule of thumb, about 1 integer out of every $\ln x$ in the vicinity of x is a prime.

The Prime Number Theorem was conjectured by Legendre in 1798 and proved a century later by de la Vallee Poussin and Hadamard in 1896. However, after his death, a notebook of Gauss was found to contain the same conjecture, which he apparently made in 1791 at age 15. (You sort of have to feel sorry for all the otherwise "great" mathematicians who had the misfortune of being contemporaries of Gauss.)

In late 2004 a billboard appeared in various locations around the country:

$$\left\{ \begin{array}{c} \text{first 10-digit prime found} \\ \text{in consecutive digits of } e \end{array} \right\} \textbf{. com}$$

Substituting the correct number for the expression in curly-braces produced the URL for a Google employment page. The idea was that Google was interested in hiring the sort of people that could and would solve such a problem.

How hard is this problem? Would you have to look through thousands or millions or billions of digits of e to find a 10-digit prime? The rule of thumb derived from the Prime Number Theorem says that among 10-digit numbers, about 1 in

$$\ln 10^{10} \approx 23$$

is prime. This suggests that the problem isn't really so hard! Sure enough, the first 10-digit prime in consecutive digits of e appears quite early:

$$\begin{aligned} e = &2.718281828459045235360287471352662497757247093699959574966 \\ &967627724076630353547594571382178525166427\mathbf{7427466391}9320030 \\ &599218174135966290435729003342952605956307381323286279434 \ldots \end{aligned}$$

(b) Describe in general how to find the $\gcd(m, n)$ and $\operatorname{lcm}(m, n)$ from the prime factorizations of m and n. Conclude that equation (4.5) holds for all positive integers m, n.

4.4 Alan Turing

The man pictured above is Alan Turing, the most important figure in the history of computer science. For decades, his fascinating life story was shrouded by government secrecy, societal taboo, and even his own deceptions.

At age 24, Turing wrote a paper entitled *On Computable Numbers, with an Application to the Entscheidungsproblem*. The crux of the paper was an elegant way to model a computer in mathematical terms. This was a breakthrough, because it allowed the tools of mathematics to be brought to bear on questions of computation. For example, with his model in hand, Turing immediately proved that there exist problems that no computer can solve— no matter how ingenious the programmer. Turing's paper is all the more remarkable because he wrote it in 1936, a full decade before any electronic computer actually existed.

The word "Entscheidungsproblem" in the title refers to one of the 28 mathematical problems posed by David Hilbert in 1900 as challenges to mathematicians of the 20th century. Turing knocked that one off in the same paper. And perhaps you've heard of the "Church-Turing thesis"? Same paper. So Turing was obviously a brilliant guy who generated lots of amazing ideas. But this lecture is about one of Turing's less-amazing ideas. It involved codes. It involved number theory. And it was sort of stupid.

Let's look back to the fall of 1937. Nazi Germany was rearming under Adolf Hitler, world-shattering war looked imminent, and— like us— Alan Turing was pondering the usefulness of number theory. He foresaw that preserving military secrets would be vital in the coming conflict and proposed a way *to encrypt communications using number theory*. This is an idea that has ricocheted up to our own time. Today, number theory is the basis for numerous public-key cryptosystems, digital signature schemes, cryptographic hash functions, and electronic payment systems. Furthermore, military funding agencies are among the biggest investors in cryptographic research. Sorry Hardy!

Soon after devising his code, Turing disappeared from public view, and half a century would pass before the world learned the full story of where he'd gone and what he did there. We'll come back to Turing's life in a little while; for now, let's investigate the code Turing left behind. The details are uncertain, since he never formally published the idea, so we'll consider a couple of possibilities.

4.4.1 Turing's Code (Version 1.0)

The first challenge is to translate a text message into an integer so we can perform mathematical operations on it. This step is not intended to make a message harder to read, so the details are not too important. Here is one approach: replace each letter of the message with two digits ($A = 01$, $B = 02$, $C = 03$, etc.) and string all the digits together to form one huge number. For example, the message "victory" could be translated this way:

$$\begin{array}{cccccccc} & \text{"v} & \text{i} & \text{c} & \text{t} & \text{o} & \text{r} & \text{y"} \\ \rightarrow & 22 & 09 & 03 & 20 & 15 & 18 & 25 \end{array}$$

Turing's code requires the message to be a prime number, so we may need to pad the result with a few more digits to make a prime. In this case, appending the digits 13 gives the number 2209032015182513, which is prime.

Now here is how the encryption process works. In the description below, m is the unencoded message (which we want to keep secret), m^* is the encrypted message (which the Nazis may intercept), and k is the key.

Beforehand The sender and receiver agree on a secret key, which is a large prime k.

Encryption The sender encrypts the message m by computing:

$$m^* = m \cdot k$$

Decryption The receiver decrypts m^* by computing:

$$\frac{m^*}{k} = \frac{m \cdot k}{k} = m$$

For example, suppose that the secret key is the prime number $k = 22801763489$ and the message m is "victory". Then the encrypted message is:

$$\begin{aligned} m^* &= m \cdot k \\ &= 2209032015182513 \cdot 22801763489 \\ &= 50369825549820718594667857 \end{aligned}$$

There are a couple of questions that one might naturally ask about Turing's code.

1. How can the sender and receiver ensure that m and k are prime numbers, as required?

   The general problem of determining whether a large number is prime or composite has been studied for centuries, and reasonably good primality tests were known even in Turing's time. In 2002, Manindra Agrawal, Neeraj Kayal, and Nitin Saxena announced a primality test that is guaranteed to work on a number n in about $(\log n)^{12}$ steps, that is, a number of steps bounded by a twelfth degree polynomial in the length (in bits) of the input, n. This definitively places primality testing way below the problems of exponential difficulty. Amazingly, the description of their breakthrough algorithm was only thirteen lines long!

   Of course, a twelfth degree polynomial grows pretty fast, so the Agrawal, *et al.* procedure is of no practical use. Still, good ideas have a way of breeding more good ideas, so there's certainly hope further improvements will lead to a procedure that is useful in practice. But the truth is, there's no practical need to improve it, since very efficient *probabilistic* procedures for prime-testing have been known since the early 1970's. These procedures have some probability of giving a wrong answer, but their probability of being wrong is so tiny that relying on their answers is the best bet you'll ever make.

2. Is Turing's code secure?

   The Nazis see only the encrypted message $m^* = m \cdot k$, so recovering the original message m requires factoring m^*. Despite immense efforts, no really efficient factoring algorithm has ever been found. It appears to be a fundamentally difficult problem, though a breakthrough someday is not impossible. In effect, Turing's code puts to practical use his discovery that there are limits to the power of computation. Thus, provided m and k are sufficiently large, the Nazis seem to be out of luck!

This all sounds promising, but there is a major flaw in Turing's code.

4.4.2 Breaking Turing's Code

Let's consider what happens when the sender transmits a *second* message using Turing's code and the same key. This gives the Nazis two encrypted messages to look at:

$$m_1^* = m_1 \cdot k \qquad \text{and} \qquad m_2^* = m_2 \cdot k$$

The greatest common divisor of the two encrypted messages, m_1^* and m_2^*, is the secret key k. And, as we've seen, the GCD of two numbers can be computed very efficiently. So after the second message is sent, the Nazis can recover the secret key and read *every* message!

It is difficult to believe a mathematician as brilliant as Turing could overlook such a glaring problem. One possible explanation is that he had a slightly different system in mind, one based on *modular* arithmetic.

4.5 Modular Arithmetic

On page 1 of his masterpiece on number theory, *Disquisitiones Arithmeticae*, Gauss introduced the notion of "congruence". Now, Gauss is another guy who managed to cough up a half-decent idea every now and then, so let's take a look at this one. Gauss said that a is *congruent* to b *modulo* n iff $n \mid (a - b)$. This is written

$$a \equiv b \pmod{n}.$$

For example:

$$29 \equiv 15 \pmod{7} \quad \text{because } 7 \mid (29 - 15).$$

There is a close connection between congruences and remainders:

Lemma 4.5.1 (Congruences and Remainders)**.**

$$a \equiv b \pmod{n} \quad \textit{iff} \quad \operatorname{rem}(a, n) = \operatorname{rem}(b, n).$$

Proof. By the Division Theorem, there exist unique pairs of integers q_1, r_1 and q_2, r_2 such that:

$$\begin{aligned} a &= q_1 n + r_1 && \text{where } 0 \leq r_1 < n, \\ b &= q_2 n + r_2 && \text{where } 0 \leq r_2 < n. \end{aligned}$$

Subtracting the second equation from the first gives:

$$a - b = (q_1 - q_2)n + (r_1 - r_2) \qquad \text{where } -n < r_1 - r_2 < n.$$

Now $a \equiv b \pmod{n}$ if and only if n divides the left side. This is true if and only if n divides the right side, which holds if and only if $r_1 - r_2$ is a multiple of n. Given the bounds on $r_1 - r_2$, this happens precisely when $r_1 = r_2$, that is, when $\operatorname{rem}(a, n) = \operatorname{rem}(b, n)$. ■

So we can also see that

$$29 \equiv 15 \pmod{7} \quad \text{because } \operatorname{rem}(29, 7) = 1 = \operatorname{rem}(15, 7).$$

This formulation explains why the congruence relation has properties like an equality relation. Notice that even though (mod 7) appears over on the right side the $\equiv$ symbol, it isn't any more strongly associated with the 15 than with the 29. It would really be clearer to write $29 \equiv_{\text{mod } 7} 15$ for example, but the notation with the modulus at the end is firmly entrenched and we'll stick to it.

We'll make frequent use of the following immediate Corollary of Lemma 4.5.1:

Corollary 4.5.2.

$$a \equiv \text{rem}(a, n) \pmod{n}$$

Still another way to think about congruence modulo n is that it *defines a partition of the integers into n sets so that congruent numbers are all in the same set*. For example, suppose that we're working modulo 3. Then we can partition the integers into 3 sets as follows:

$$\begin{array}{cccccccccc} \{ & \dots, & -6, & -3, & 0, & 3, & 6, & 9, & \dots & \} \\ \{ & \dots, & -5, & -2, & 1, & 4, & 7, & 10, & \dots & \} \\ \{ & \dots, & -4, & -1, & 2, & 5, & 8, & 11, & \dots & \} \end{array}$$

according to whether their remainders on division by 3 are 0, 1, or 2. The upshot is that when arithmetic is done modulo n there are really only n different kinds of numbers to worry about, because there are only n possible remainders. In this sense, modular arithmetic is a simplification of ordinary arithmetic and thus is a good reasoning tool.

There are many useful facts about congruences, some of which are listed in the lemma below. The overall theme is that *congruences work a lot like equations*, though there are a couple of exceptions.

Lemma 4.5.3 (Facts About Congruences)**.** *The following hold for $n \geq 1$:*

1. $a \equiv a \pmod{n}$
2. $a \equiv b \pmod{n}$ *implies* $b \equiv a \pmod{n}$
3. $a \equiv b \pmod{n}$ *and* $b \equiv c \pmod{n}$ *implies* $a \equiv c \pmod{n}$
4. $a \equiv b \pmod{n}$ *implies* $a + c \equiv b + c \pmod{n}$
5. $a \equiv b \pmod{n}$ *implies* $ac \equiv bc \pmod{n}$
6. $a \equiv b \pmod{n}$ *and* $c \equiv d \pmod{n}$ *imply* $a + c \equiv b + d \pmod{n}$
7. $a \equiv b \pmod{n}$ *and* $c \equiv d \pmod{n}$ *imply* $ac \equiv bd \pmod{n}$

Proof. Parts 1.–3. follow immediately from Lemma 4.5.1. Part 4. follows immediately from the definition that $a \equiv b \pmod{n}$ iff $n \mid (a - b)$. Likewise, part 5. follows because if $n \mid (a - b)$ then it divides $(a - b)c = ac - bc$. To prove part 6., assume

$$a \equiv b \pmod{n} \tag{4.6}$$

and

$$c \equiv d \pmod{n}. \tag{4.7}$$

Then

$$\begin{aligned}
a + c &\equiv b + c \pmod{n} && \text{(by part 4. and (4.6)),}\\
c + b &\equiv d + b \pmod{n} && \text{(by part 4. and (4.7)), so}\\
b + c &\equiv b + d \pmod{n} && \text{and therefore}\\
a + c &\equiv b + d \pmod{n} && \text{(by part 3.)}
\end{aligned}$$

Part 7. has a similar proof. ■

4.5.1 Turing's Code (Version 2.0)

In 1940 France had fallen before Hitler's army, and Britain alone stood against the Nazis in western Europe. British resistance depended on a steady flow of supplies brought across the north Atlantic from the United States by convoys of ships. These convoys were engaged in a cat-and-mouse game with German "U-boats" —submarines —which prowled the Atlantic, trying to sink supply ships and starve Britain into submission. The outcome of this struggle pivoted on a balance of information: could the Germans locate convoys better than the Allies could locate U-boats or vice versa?

Germany lost.

But a critical reason behind Germany's loss was made public only in 1974: Germany's naval code, *Enigma*, had been broken by the Polish Cipher Bureau (see `http://en.wikipedia.org/wiki/Polish_Cipher_Bureau`) and the secret had been turned over to the British a few weeks before the Nazi invasion of Poland in 1939. Throughout much of the war, the Allies were able to route convoys around German submarines by listening in to German communications. The British government didn't explain *how* Enigma was broken until 1996. When it was finally released (by the US), the story revealed that Alan Turing had joined the secret British codebreaking effort at Bletchley Park in 1939, where he became the lead developer of methods for rapid, bulk decryption of German Enigma messages. Turing's Enigma deciphering was an invaluable contribution to the Allied victory over Hitler.

Governments are always tight-lipped about cryptography, but the half-century of official silence about Turing's role in breaking Enigma and saving Britain may be related to some disturbing events after the war.

Let's consider an alternative interpretation of Turing's code. Perhaps we had the basic idea right (multiply the message by the key), but erred in using *conventional* arithmetic instead of *modular* arithmetic. Maybe this is what Turing meant:

Beforehand The sender and receiver agree on a large prime p, which may be made public. (This will be the modulus for all our arithmetic.) They also agree on a secret key $k \in \{1, 2, \ldots, p-1\}$.

Encryption The message m can be any integer in the set $\{0, 1, 2, \ldots, p-1\}$; in particular, the message is no longer required to be a prime. The sender encrypts the message m to produce m^* by computing:

$$m^* = \text{rem}(mk, p) \tag{4.8}$$

Decryption (Uh-oh.)

The decryption step is a problem. We might hope to decrypt in the same way as before: by dividing the encrypted message m^* by the key k. The difficulty is that m^* is the *remainder* when mk is divided by p. So dividing m^* by k might not even give us an integer!

This decoding difficulty can be overcome with a better understanding of arithmetic modulo a prime.

4.5.2 Problems

Class Problems

Problem 4.5.
The following properties of equivalence mod n follow directly from its definition and simple properties of divisibility. See if you can prove them without looking up the proofs in the text.

(a) If $a \equiv b \pmod{n}$, then $ac \equiv bc \pmod{n}$.

(b) If $a \equiv b \pmod{n}$ and $b \equiv c \pmod{n}$, then $a \equiv c \pmod{n}$.

(c) If $a \equiv b \pmod{n}$ and $c \equiv d \pmod{n}$, then $ac \equiv bd \pmod{n}$.

(d) $\text{rem}(a, n) \equiv a \pmod{n}$.

Problem 4.6. **(a)** Why is a number written in decimal evenly divisible by 9 if and only if the sum of its digits is a multiple of 9? *Hint:* $10 \equiv 1 \pmod{9}$.

(b) Take a big number, such as 37273761261. Sum the digits, where every other one is negated:

$$3 + (-7) + 2 + (-7) + 3 + (-7) + 6 + (-1) + 2 + (-6) + 1 = -11$$

Explain why the original number is a multiple of 11 if and only if this sum is a multiple of 11.

Problem 4.7.
At one time, the Guinness Book of World Records reported that the "greatest human calculator" was a guy who could compute 13th roots of 100-digit numbers that were powers of 13. What a curious choice of tasks ....

(a) Prove that
$$d^{13} \equiv d \pmod{10} \tag{4.9}$$
for $0 \leq d < 10$.

(b) Now prove that
$$n^{13} \equiv n \pmod{10} \tag{4.10}$$
for all n.

4.6 Arithmetic with a Prime Modulus

4.6.1 Multiplicative Inverses

The *multiplicative inverse* of a number x is another number x^{-1} such that:

$$x \cdot x^{-1} = 1$$

Generally, multiplicative inverses exist over the real numbers. For example, the multiplicative inverse of 3 is $1/3$ since:

$$3 \cdot \frac{1}{3} = 1$$

The sole exception is that 0 does not have an inverse.

On the other hand, inverses generally do not exist over the integers. For example, 7 can not be multiplied by another integer to give 1.

Surprisingly, multiplicative inverses do exist when we're working *modulo a prime number*. For example, if we're working modulo 5, then 3 is a multiplicative inverse of 7, since:

$$7 \cdot 3 \equiv 1 \pmod{5}$$

(All numbers congruent to 3 modulo 5 are also multiplicative inverses of 7; for example, $7 \cdot 8 \equiv 1 \pmod{5}$ as well.) The only exception is that numbers congruent to 0 modulo 5 (that is, the multiples of 5) do not have inverses, much as 0 does not have an inverse over the real numbers. Let's prove this.

Lemma 4.6.1. *If p is prime and k is not a multiple of p, then k has a multiplicative inverse.*

Proof. Since p is prime, it has only two divisors: 1 and p. And since k is not a multiple of p, we must have $\gcd(p, k) = 1$. Therefore, there is a linear combination of p and k equal to 1:

$$sp + tk = 1$$

Rearranging terms gives:

$$sp = 1 - tk$$

This implies that $p \mid (1 - tk)$ by the definition of divisibility, and therefore $tk \equiv 1 \pmod{p}$ by the definition of congruence. Thus, t is a multiplicative inverse of k. ∎

Multiplicative inverses are the key to decryption in Turing's code. Specifically, we can recover the original message by multiplying the encoded message by the *inverse* of the key:

$$\begin{aligned}
m^* \cdot k^{-1} &= \text{rem}(mk, p) \cdot k^{-1} && \text{(the def. (4.8) of } m^*\text{)} \\
&\equiv (mk)k^{-1} \pmod{p} && \text{(by Cor. 4.5.2)} \\
&\equiv m \pmod{p}.
\end{aligned}$$

This shows that $m^* k^{-1}$ is congruent to the original message m. Since m was in the range $0, 1, \ldots, p-1$, we can recover it exactly by taking a remainder:

$$m = \text{rem}(m^* k^{-1}, p)$$

So now we can decrypt!

4.6.2 Cancellation

Another sense in which real numbers are nice is that one can cancel multiplicative terms. In other words, if we know that $m_1 k = m_2 k$, then we can cancel the k's and conclude that $m_1 = m_2$, provided $k \neq 0$. In general, cancellation is *not* valid in modular arithmetic. For example,

$$2 \cdot 3 \equiv 4 \cdot 3 \pmod{6},$$

cancelling the 3's leads to the *false* conclusion that $2 \equiv 4 \pmod{6}$. The fact that multiplicative terms can not be cancelled is the most significant sense in which congruences differ from ordinary equations. However, this difference goes away if we're working modulo a *prime*; then cancellation is valid.

Lemma 4.6.2. *Suppose p is a prime and k is not a multiple of p. Then*

$$ak \equiv bk \pmod{p} \quad \text{IMPLIES} \quad a \equiv b \pmod{p}.$$

Proof. Multiply both sides of the congruence by k^{-1}. ■

We can use this lemma to get a bit more insight into how Turing's code works. In particular, the encryption operation in Turing's code *permutes the set of possible messages*. This is stated more precisely in the following corollary.

Corollary 4.6.3. *Suppose p is a prime and k is not a multiple of p. Then the sequence:*

$$\text{rem}((1 \cdot k), p), \quad \text{rem}((2 \cdot k), p), \quad \ldots, \quad \text{rem}(((p-1) \cdot k), p)$$

is a permutation[3] *of the sequence:*

$$1, \quad 2, \quad \ldots, \quad (p-1).$$

Proof. The sequence of remainders contains $p-1$ numbers. Since $i \cdot k$ is not divisible by p for $i = 1, \ldots p-1$, all these remainders are in the range 1 to $p-1$ by the definition of remainder. Furthermore, the remainders are all different: no two numbers in the range 1 to $p-1$ are congruent modulo p, and by Lemma 4.6.2, $i \cdot k \equiv j \cdot k \pmod{p}$ if and only if $i \equiv j \pmod{p}$. Thus, the sequence of remainders must contain *all* of the numbers from 1 to $p-1$ in some order. ■

For example, suppose $p = 5$ and $k = 3$. Then the sequence:

$$\underbrace{\text{rem}((1 \cdot 3), 5)}_{=3}, \quad \underbrace{\text{rem}((2 \cdot 3), 5)}_{=1}, \quad \underbrace{\text{rem}((3 \cdot 3), 5)}_{=4}, \quad \underbrace{\text{rem}((4 \cdot 3), 5)}_{=2}$$

is a permutation of 1, 2, 3, 4. As long as the Nazis don't know the secret key k, they don't know how the set of possible messages are permuted by the process of encryption and thus can't read encoded messages.

4.6.3 Fermat's Little Theorem

A remaining challenge in using Turing's code is that decryption requires the inverse of the secret key k. An effective way to calculate k^{-1} follows from the proof of Lemma 4.6.1, namely

$$k^{-1} = \text{rem}(t, p)$$

where s, t are coefficients such that $sp + tk = 1$. Notice that t is easy to find using the Pulverizer.

An alternative approach, about equally efficient and probably more memorable, is to rely on Fermat's Little Theorem, which is much easier than his famous Last Theorem.

Theorem 4.6.4 (Fermat's Little Theorem)**.** *Suppose p is a prime and k is not a multiple of p. Then:*

$$k^{p-1} \equiv 1 \pmod{p}$$

[3] A *permutation* of a sequence of elements is a sequence with the same elements (including repeats) possibly in a different order. More formally, if

$$\vec{e} ::= e_1, e_2, \ldots, e_n$$

is a length n sequence, and $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ is a bijection, then

$$e_{\pi(1)}, e_{\pi(2)}, \ldots, e_{\pi(n)},$$

is a *permutation* of $\vec{e}$.

Proof. We reason as follows:

$$
\begin{aligned}
(p-1)! &::= 1 \cdot 2 \cdots (p-1) \\
&= \text{rem}(k, p) \cdot \text{rem}(2k, p) \cdots \text{rem}((p-1)k, p) && \text{(by Cor 4.6.3)} \\
&\equiv k \cdot 2k \cdots (p-1)k \pmod{p} && \text{(by Cor 4.5.2)} \\
&\equiv (p-1)! \cdot k^{p-1} \pmod{p} && \text{(rearranging terms)}
\end{aligned}
$$

Now $(p-1)!$ is not a multiple of p because the prime factorizations of $1, 2, \ldots, (p-1)$ contain only primes smaller than p. So by Lemma 4.6.2, we can cancel $(p-1)!$ from the first and last expressions, which proves the claim. ■

Here is how we can find inverses using Fermat's Theorem. Suppose p is a prime and k is not a multiple of p. Then, by Fermat's Theorem, we know that:

$$k^{p-2} \cdot k \equiv 1 \pmod{p}$$

Therefore, k^{p-2} must be a multiplicative inverse of k. For example, suppose that we want the multiplicative inverse of 6 modulo 17. Then we need to compute $\text{rem}(6^{15}, 17)$, which we can do by successive squaring. All the congruences below hold modulo 17.

$$
\begin{aligned}
6^2 &\equiv 36 \equiv 2 \\
6^4 &\equiv (6^2)^2 \equiv 2^2 \equiv 4 \\
6^8 &\equiv (6^4)^2 \equiv 4^2 \equiv 16 \\
6^{15} &\equiv 6^8 \cdot 6^4 \cdot 6^2 \cdot 6 \equiv 16 \cdot 4 \cdot 2 \cdot 6 \equiv 3
\end{aligned}
$$

Therefore, $\text{rem}(6^{15}, 17) = 3$. Sure enough, 3 is the multiplicative inverse of 6 modulo 17, since:

$$3 \cdot 6 \equiv 1 \pmod{17}$$

In general, if we were working modulo a prime p, finding a multiplicative inverse by trying every value between 1 and $p-1$ would require about p operations. However, the approach above requires only about $\log p$ operations, which is far better when p is large.

4.6.4 Breaking Turing's Code —Again

The Germans didn't bother to encrypt their weather reports with the highly-secure Enigma system. After all, so what if the Allies learned that there was rain off the south coast of Iceland? But, amazingly, this practice provided the British with a critical edge in the Atlantic naval battle during 1941.

The problem was that some of those weather reports had originally been transmitted using Enigma from U-boats out in the Atlantic. Thus, the British obtained both unencrypted reports and the same reports encrypted with Enigma. By comparing the two, the British were able to determine which key the Germans were using that day and could read all other Enigma-encoded traffic. Today, this would be called a *known-plaintext attack*.

Let's see how a known-plaintext attack would work against Turing's code. Suppose that the Nazis know both m and m^* where:

$$m^* \equiv mk \pmod{p}$$

Now they can compute:

$$\begin{aligned}
m^{p-2} \cdot m^* &= m^{p-2} \cdot \text{rem}(mk, p) && \text{(def. (4.8) of } m^*\text{)} \\
&\equiv m^{p-2} \cdot mk \pmod{p} && \text{(by Cor 4.5.2)} \\
&\equiv m^{p-1} \cdot k \pmod{p} \\
&\equiv k \pmod{p} && \text{(Fermat's Theorem)}
\end{aligned}$$

Now the Nazis have the secret key k and can decrypt any message!

This is a huge vulnerability, so Turing's code has no practical value. Fortunately, Turing got better at cryptography after devising this code; his subsequent deciphering of Enigma messages surely saved thousands of lives, if not the whole of Britain.

4.6.5 Turing Postscript

A few years after the war, Turing's home was robbed. Detectives soon determined that a former homosexual lover of Turing's had conspired in the robbery. So they arrested him —that is, they arrested Alan Turing —because homosexuality was a British crime punishable by up to two years in prison at that time. Turing was sentenced to a hormonal "treatment" for his homosexuality: he was given estrogen injections. He began to develop breasts.

Three years later, Alan Turing, the founder of computer science, was dead. His mother explained what happened in a biography of her own son. Despite her repeated warnings, Turing carried out chemistry experiments in his own home. Apparently, her worst fear was realized: by working with potassium cyanide while eating an apple, he poisoned himself.

However, Turing remained a puzzle to the very end. His mother was a devoutly religious woman who considered suicide a sin. And, other biographers have pointed out, Turing had previously discussed committing suicide by eating a poisoned apple. Evidently, Alan Turing, who founded computer science and saved his country, took his own life in the end, and in just such a way that his mother could believe it was an accident.

Turing's last project before he disappeared from public view in 1939 involved the construction of an elaborate mechanical device to test a mathematical conjecture called the Riemann Hypothesis. This conjecture first appeared in a sketchy paper by Berhard Riemann in 1859 and is now one of the most famous unsolved problem in mathematics.

The Riemann Hypothesis

The formula for the sum of an infinite geometric series says:

$$1 + x + x^2 + x^3 + \cdots = \frac{1}{1-x}$$

Substituting $x = \frac{1}{2^s}$, $x = \frac{1}{3^s}$, $x = \frac{1}{5^s}$, and so on for each prime number gives a sequence of equations:

$$\begin{aligned}
1 + \frac{1}{2^s} + \frac{1}{2^{2s}} + \frac{1}{2^{3s}} + \cdots &= \frac{1}{1 - 1/2^s} \\
1 + \frac{1}{3^s} + \frac{1}{3^{2s}} + \frac{1}{3^{3s}} + \cdots &= \frac{1}{1 - 1/3^s} \\
1 + \frac{1}{5^s} + \frac{1}{5^{2s}} + \frac{1}{5^{3s}} + \cdots &= \frac{1}{1 - 1/5^s} \\
&\text{etc.}
\end{aligned}$$

Multiplying together all the left sides and all the right sides gives:

$$\sum_{n=1}^{\infty} \frac{1}{n^s} = \prod_{p \in \text{primes}} \left(\frac{1}{1 - 1/p^s} \right)$$

The sum on the left is obtained by multiplying out all the infinite series and applying the Fundamental Theorem of Arithmetic. For example, the term $1/300^s$ in the sum is obtained by multiplying $1/2^{2s}$ from the first equation by $1/3^s$ in the second and $1/5^{2s}$ in the third. Riemann noted that every prime appears in the expression on the right. So he proposed to learn about the primes by studying the equivalent, but simpler expression on the left. In particular, he regarded s as a complex number and the left side as a function, $\zeta(s)$. Riemann found that the distribution of primes is related to values of s for which $\zeta(s) = 0$, which led to his famous conjecture:

Definition 4.6.5. *The Riemann Hypothesis*: Every nontrivial zero of the zeta function $\zeta(s)$ lies on the line $s = 1/2 + ci$ in the complex plane.

A proof would immediately imply, among other things, a strong form of the Prime Number Theorem.

Researchers continue to work intensely to settle this conjecture, as they have for over a century. It is another of the Millenium Problems whose solver will earn $1,000,000 from the Clay Institute.

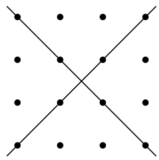
4.6.6 Problems

Class Problems

Problem 4.8.
Two nonparallel lines in the real plane intersect at a point. Algebraically, this means that the equations

$$\begin{aligned} y &= m_1 x + b_1 \\ y &= m_2 x + b_2 \end{aligned}$$

have a unique solution (x, y), provided $m_1 \neq m_2$. This statement would be false if we restricted x and y to the integers, since the two lines could cross at a noninteger point:



However, an analogous statement holds if we work over the integers *modulo a prime*, p. Find a solution to the congruences

$$\begin{aligned} y &\equiv m_1 x + b_1 \pmod{p} \\ y &\equiv m_2 x + b_2 \pmod{p} \end{aligned}$$

when $m_1 \not\equiv m_2 \pmod{p}$. Express your solution in the form $x \equiv ? \pmod{p}$ and $y \equiv ? \pmod{p}$ where the ?'s denote expressions involving m_1, m_2, b_1, and b_2. You may find it helpful to solve the original equations over the reals first.

Problem 4.9.
Let $S_k = 1^k + 2^k + \ldots + (p-1)^k$, where p is an odd prime and k is a positive multiple of $p - 1$. Use Fermat's theorem to prove that $S_k \equiv -1 \pmod{p}$.

Homework Problems

Problem 4.10. (a) Use the Pulverizer to find the inverse of 13 modulo 23 in the range $\{1, \ldots, 22\}$.

(b) Use Fermat's theorem to find the inverse of 13 modulo 23 in the range $\{1, \ldots, 22\}$.

4.7 Arithmetic with an Arbitrary Modulus

Turing's code did not work as he hoped. However, his essential idea— using number theory as the basis for cryptography— succeeded spectacularly in the decades after his death.

In 1977, Ronald Rivest, Adi Shamir, and Leonard Adleman at MIT proposed a highly secure cryptosystem (called **RSA**) based on number theory. Despite decades of attack, no significant weakness has been found. Moreover, RSA has a major advantage over traditional codes: the sender and receiver of an encrypted message need not meet beforehand to agree on a secret key. Rather, the receiver has both a *secret key*, which she guards closely, and a *public key*, which she distributes as widely as possible. The sender then encrypts his message using her widely-distributed public key. Then she decrypts the received message using her closely-held private key. The use of such a *public key cryptography* system allows you and Amazon, for example, to engage in a secure transaction without meeting up beforehand in a dark alley to exchange a key.

Interestingly, RSA does not operate modulo a prime, as Turing's scheme may have, but rather modulo the product of *two* large primes. Thus, we'll need to know a bit about how arithmetic works modulo a composite number in order to understand RSA. Arithmetic modulo an arbitrary positive integer is really only a little more painful than working modulo a prime —though you may think this is like the doctor saying, "This is only going to hurt a little," before he jams a big needle in your arm.

4.7.1 Relative Primality and Phi

First, we need a new definition. Integers a and b are *relatively prime* iff $\gcd(a, b) = 1$. For example, 8 and 15 are relatively prime, since $\gcd(8, 15) = 1$. Note that, except for multiples of p, every integer is relatively prime to a prime number p.

We'll also need a certain function that is defined using relative primality. Let n be a positive integer. Then $\phi(n)$ denotes the number of integers in $\{1, 2, \ldots, n-1\}$ that are relatively prime to n. For example, $\phi(7) = 6$, since 1, 2, 3, 4, 5, and 6 are all relatively prime to 7. Similarly, $\phi(12) = 4$, since only 1, 5, 7, and 11 are relatively prime to 12. If you know the prime factorization of n, then computing $\phi(n)$ is a piece of cake, thanks to the following theorem. The function ϕ is known as *Euler's ϕ function*; it's also called Euler's *totient* function.

Theorem 4.7.1. *The function ϕ obeys the following relationships:*

(a) If a and b are relatively prime, then $\phi(ab) = \phi(a)\phi(b)$.

(b) If p is a prime, then $\phi(p^k) = p^k - p^{k-1}$ for $k \geq 1$.

Here's an example of using Theorem 4.7.1 to compute $\phi(300)$:

$$\begin{aligned}
\phi(300) &= \phi(2^2 \cdot 3 \cdot 5^2) \\
&= \phi(2^2) \cdot \phi(3) \cdot \phi(5^2) && \text{(by Theorem 4.7.1.(a))} \\
&= (2^2 - 2^1)(3^1 - 3^0)(5^2 - 5^1) && \text{(by Theorem 4.7.1.(b))} \\
&= 80.
\end{aligned}$$

The proof of Theorem 4.7.1.(a) requires a few more properties of modular arithmetic worked out in the next section (see Problem 4.15). We'll also give another a proof in a few weeks based on rules for counting things.

To prove Theorem 4.7.1.(b), notice that every pth number among the p^k numbers in the interval from 0 to $p^k - 1$ is divisible by p, and only these are divisible by p. So $1/p$th of these numbers are divisible by p and the remaining ones are not. That is,

$$\phi(p^k) = p^k - (1/p)p^k = p^k - p^{k-1}.$$

4.7.2 Generalizing to an Arbitrary Modulus

Let's generalize what we know about arithmetic modulo a prime. Now, instead of working modulo a prime p, we'll work modulo an arbitrary positive integer n. The basic theme is that arithmetic modulo n may be complicated, but the integers *relatively prime* to n remain fairly well-behaved. For example, the proof of Lemma 4.6.1 of an inverse for k modulo p extends to an inverse for k relatively prime to n:

Lemma 4.7.2. *Let n be a positive integer. If k is* relatively prime *to n, then there exists an integer k^{-1} such that:*

$$k \cdot k^{-1} \equiv 1 \pmod{n}$$

As a consequence of this lemma, we can cancel a multiplicative term from both sides of a congruence if that term is relatively prime to the modulus:

Corollary 4.7.3. *Suppose n is a positive integer and k is relatively prime to n. If*

$$ak \equiv bk \pmod{n}$$

then

$$a \equiv b \pmod{n}$$

This holds because we can multiply both sides of the first congruence by k^{-1} and simplify to obtain the second.

4.7.3 Euler's Theorem

RSA essentially relies on Euler's Theorem, a generalization of Fermat's Theorem to an arbitrary modulus. The proof is much like the proof of Fermat's Theorem, except that we focus on integers relatively prime to the modulus. Let's start with a lemma.

Lemma 4.7.4. *Suppose n is a positive integer and k is relatively prime to n. Let $k_1, \ldots, k_r$ denote all the integers relatively prime to n in the range 1 to $n - 1$. Then the sequence:*

$$\text{rem}(k_1 \cdot k, n), \quad \text{rem}(k_2 \cdot k, n), \quad \text{rem}(k_3 \cdot k, n), \quad \ldots \quad, \text{rem}(k_r \cdot k, n)$$

is a permutation of the sequence:

$$k_1, \quad k_2, \quad \ldots \quad, k_r.$$

Proof. We will show that the remainders in the first sequence are all distinct and are equal to some member of the sequence of k_j's. Since the two sequences have the same length, the first must be a permutation of the second.

First, we show that the remainders in the first sequence are all distinct. Suppose that $\operatorname{rem}(k_i k, n) = \operatorname{rem}(k_j k, n)$. This is equivalent to $k_i k \equiv k_j k \pmod{n}$, which implies $k_i \equiv k_j \pmod{n}$ by Corollary 4.7.3. This, in turn, means that $k_i = k_j$ since both are between 1 and $n-1$. Thus, none of the remainder terms in the first sequence is equal to any other remainder term.

Next, we show that each remainder in the first sequence equals one of the k_i. By assumption, $\gcd(k_i, n) = 1$ and $\gcd(k, n) = 1$, which means that

$$\begin{aligned} \gcd(n, \operatorname{rem}(k_i k, n)) &= \gcd(k_i k, n) && \text{(by Lemma 4.2.4.5)} \\ &= 1 && \text{(by Lemma 4.2.4.3).} \end{aligned}$$

Now $\operatorname{rem}(k_i k, n)$ is in the range from 0 to $n-1$ by the definition of remainder, but since it is relatively prime to n, it is actually in the range 0 to $n-1$. The k_j's are defined to be the set of all such integers, so $\operatorname{rem}(k_i k, n)$ must equal some k_j. ■

We can now prove Euler's Theorem:

Theorem 4.7.5 (Euler's Theorem)**.** *Suppose n is a positive integer and k is relatively prime to n. Then*

$$k^{\phi(n)} \equiv 1 \pmod{n}$$

Proof. Let $k_1, \ldots, k_r$ denote all integers relatively prime to n such that $0 \leq k_i < n$. Then $r = \phi(n)$, by the definition of the function ϕ. Now we can reason as follows:

$$\begin{aligned} & k_1 \cdot k_2 \cdots k_r \\ &\quad = \operatorname{rem}(k_1 \cdot k, n) \cdot \operatorname{rem}(k_2 \cdot k, n) \cdots \operatorname{rem}(k_r \cdot k, n) && \text{(by Lemma 4.7.4)} \\ &\quad \equiv (k_1 \cdot k) \cdot (k_2 \cdot k) \cdot \cdots (k_r \cdot k) \pmod{n} && \text{(by Cor 4.5.2)} \\ &\quad \equiv (k_1 \cdot k_2 \cdots k_r) \cdot k^r \pmod{n} && \text{(rearranging terms)} \end{aligned}$$

Lemma 4.2.4.3. implies that $k_1 \cdot k_2 \cdots k_r$ is prime relative to n. So by Corollary 4.7.3, we can cancel this product from the first and last expressions. This proves the claim. ■

We can find multiplicative inverses using Euler's theorem as we did with Fermat's theorem: if k is relatively prime to n, then $k^{\phi(n)-1}$ is a multiplicative inverse of k modulo n. However, this approach requires computing $\phi(n)$. Unfortunately, finding $\phi(n)$ is about as hard as factoring n, and factoring is hard in general. However, when we know how to factor n, we can use Theorem 4.7.1 to compute $\phi(n)$ efficiently. Then computing $k^{\phi(n)-1}$ to find inverses is a competitive alternative to the Pulverizer.

4.7.4 RSA

Here, then, is the *RSA public key encryption scheme*:

RSA Public Key Encryption

RSA Public Key Encryption

Beforehand The receiver creates a public key and a secret key as follows.

1. Generate two distinct primes, p and q.
2. Let $n = pq$.
3. Select an integer e such that $\gcd(e, (p-1)(q-1)) = 1$.
   The *public key* is the pair (e, n). This should be distributed widely.
4. Compute d such that $de \equiv 1 \pmod{(p-1)(q-1)}$.
   The *secret key* is the pair (d, n). This should be kept hidden!

Encoding The sender encrypts message m to produce m' using the public key:

$$m' = \text{rem}(m^e, n).$$

Decoding The receiver decrypts message m' back to message m using the secret key:

$$m = \text{rem}((m')^d, n).$$

We'll explain why this way of Decoding works in Problem 4.14.

The explanation of why this way of Decoding works is worked out in Problem 4.14.

4.7.5 Problems

Practice Problems

Problem 4.11. **(a)** Prove that 22^{12001} has a multiplicative inverse modulo 175.

(b) What is the value of $\phi(175)$, where ϕ is Euler's function?

(c) What is the remainder of 22^{12001} divided by 175?

Problem 4.12. **(a)** Use the Pulverizer to find integers s, t such that

$$40s + 7t = \gcd(40, 7).$$

Show your work.

(b) Adjust your answer to part (a) to find an inverse modulo 40 of 7 in the range $\{1, \ldots, 39\}$.

Class Problems

Problem 4.13.
Let's try out RSA! There is a complete description of the algorithm at the bottom of the page. You'll probably need extra paper. **Check your work carefully!**

(a) As a team, go through the **beforehand** steps.

- Choose primes p and q to be relatively small, say in the range 10-40. In practice, p and q might contain several hundred digits, but small numbers are easier to handle with pencil and paper.
- Try $e = 3, 5, 7, \ldots$ until you find something that works. Use Euclid's algorithm to compute the gcd.
- Find d (using the Pulverizer —see appendix for a reminder on how the Pulverizer works —or Euler's Theorem).

When you're done, put your public key on the board. This lets another team send you a message.

(b) Now send an encrypted message to another team using their public key. Select your message m from the codebook below:

- 2 = Greetings and salutations!
- 3 = Yo, wassup?
- 4 = You guys are slow!
- 5 = All your base are belong to us.
- 6 = Someone on *our* team thinks someone on *your* team is kinda cute.
- 7 = You *are* the weakest link. Goodbye.

(c) Decrypt the message sent to you and verify that you received what the other team sent!

Problem 4.14.
A critical fact about RSA is, of course, that decrypting an encrypted message always gives back the original message! That is, that $\text{rem}((m^d)^e, pq) = m$. This will follow from something slightly more general:

Lemma 4.7.6. *Let n be a product of distinct primes and $a \equiv 1 \pmod{\phi(n)}$ for some nonnegative integer, a. Then*

$$m^a \equiv m \pmod{n}. \tag{4.11}$$

(a) Explain why Lemma 4.7.6 implies that k and k^5 have the same last digit. For example:

$$\underline{2}^5 = 3\underline{2} \qquad\qquad 7\underline{9}^5 = 307705639\underline{9}$$

Hint: What is $\phi(10)$?

(b) Explain why Lemma 4.7.6 implies that the original message, m, equals $\text{rem}((m^e)^d, pq)$.

(c) Prove that if p is prime, then

$$m^a \equiv m \pmod{p} \tag{4.12}$$

for all nonnegative integers $a \equiv 1 \pmod{p-1}$.

(d) Prove that if n is a product of distinct primes, and $a \equiv b \pmod{p}$ for all prime factors, p, of n, then $a \equiv b \pmod{n}$.

(e) Combine the previous parts to complete the proof of Lemma 4.7.6.

Homework Problems

Problem 4.15.
Suppose m, n are relatively prime. In the problem you will prove the key property of Euler's function that $\phi(mn) = \phi(m)\phi(n)$.

(a) Prove that for any a, b, there is an x such that

$$x \equiv a \pmod{m}, \tag{4.13}$$
$$x \equiv b \pmod{n}. \tag{4.14}$$

Hint: Congruence (4.13) holds iff

$$x = jm + a. \tag{4.15}$$

for some j. So there is such an x only if

$$jm + a \equiv b \pmod{n}. \tag{4.16}$$

Solve (4.16) for j.

(b) Prove that there is an x satisfying the congruences (4.13) and (4.14) such that $0 \le x < mn$.

(c) Prove that the x satisfying part (b) is unique.

(d) For an integer k, let k^* be the integers between 1 and $k-1$ that are relatively prime to k. Conclude from part (c) that the function

$$f : (mn)^* \to m^* \times n^*$$

defined by

$$f(x) ::= (\text{rem}(x, m), \text{rem}(x, n))$$

is a bijection.

(e) Conclude from the preceding parts of this problem that

$$\phi(mn) = \phi(m)\phi(n).$$

Exam Problems

Part II

Mathematical Data Types

Chapter 5

Sets and Relations

5.1 The Logic of Sets

5.1.1 Russell's Paradox

Reasoning naively about sets turns out to be risky. In fact, one of the earliest attempts to come up with precise axioms for sets by a late nineteenth century logican named Gotlob *Frege* was shot down by a three line argument known as *Russell's Paradox*:[1] This was an astonishing blow to efforts to provide an axiomatic foundation for mathematics.

> Let S be a variable ranging over all sets, and define
>
> $$W ::= \{S \mid S \notin S\}.$$
>
> So by definition,
>
> $$S \in W \text{ iff } S \notin S,$$
>
> for every set S. In particular, we can let S be W, and obtain the contradictory result that
>
> $$W \in W \text{ iff } W \notin W.$$

A way out of the paradox was clear to Russell and others at the time: *it's unjustified to assume that W is a set*. So the step in the proof where we let S be W has no justification, because S ranges over sets, and W may not be a set. In fact, the paradox implies that W had better not be a set!

[1] Bertrand *Russell* was a mathematician/logician at Cambridge University at the turn of the Twentieth Century. He reported that when he felt too old to do mathematics, he began to study and write about philosophy, and when he was no longer smart enough to do philosophy, he began writing about politics. He was jailed as a conscientious objector during World War I. For his extensive philosophical and political writing, he won a Nobel Prize for Literature.

But denying that W is a set means we must *reject* the very natural axiom that every mathematically well-defined collection of elements is actually a set. So the problem faced by Frege, Russell and their colleagues was how to specify *which* well-defined collections are sets. Russell and his fellow Cambridge University colleague Whitehead immediately went to work on this problem. They spent a dozen years developing a huge new axiom system in an even huger monograph called *Principia Mathematica*.

symbol	**set**	**elements**
$\emptyset$	the empty set	none
$\mathbb{N}$	nonnegative integers	$\{0, 1, 2, 3, \ldots\}$
$\mathbb{Z}$	integers	$\{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$
$\mathbb{Q}$	rational numbers	$\frac{1}{2},\ -\frac{5}{3},\ 16,$ etc.
$\mathbb{R}$	real numbers	$\pi,\ e,\ -9,\ \sqrt{2},$ etc.
$\mathbb{C}$	complex numbers	$i,\ \frac{19}{2},\ \sqrt{2} - 2i,$ etc.

A superscript "$^+$" restricts a set to its positive elements; for example, $\mathbb{R}^+$ denotes the set of positive real numbers. Similarly, $\mathbb{R}^-$ denotes the set of negative reals.

5.1.2 Comparing and Combining Sets

The expression $S \subseteq T$ indicates that set S is a *subset* of set T, which means that every element of S is also an element of T (it could be that $S = T$). For example, $\mathbb{N} \subseteq \mathbb{Z}$ and $\mathbb{Q} \subseteq \mathbb{R}$ (every rational number is a real number), but $\mathbb{C} \not\subseteq \mathbb{Z}$ (not every complex number is an integer).

As a memory trick, notice that the $\subseteq$ points to the smaller set, just like a $\leq$ sign points to the smaller number. Actually, this connection goes a little further: there is a symbol $\subset$ analogous to $<$. Thus, $S \subset T$ means that S is a subset of T, but the two are *not* equal. So $A \subseteq A$, but $A \not\subset A$, for every set A.

5.1.3 The ZFC Axioms for Sets

It's generally agreed that, using some simple logical deduction rules, essentially all of mathematics can be derived from some axioms about sets called the Axioms of Zermelo-Frankel Set Theory with Choice (ZFC).

We're *not* going to be working with these axioms in this course, but we thought you might like to see them –and while you're at it, get some practice reading quantified formulas:

Extensionality. Two sets are equal if they have the same members. In formal logical notation, this would be stated as:

$$(\forall z.\ (z \in x \text{ IFF } z \in y)) \text{ IMPLIES } x = y.$$

Pairing. For any two sets x and y, there is a set, $\{x, y\}$, with x and y as its only elements:

$$\forall x, y.\ \exists u.\ \forall z.\ [z \in u \text{ IFF } (z = x \text{ OR } z = y)]$$

Union. The union, u, of a collection, z, of sets is also a set:

$$\forall z.\, \exists u \forall x.\, (\exists y.\, x \in y \text{ AND } y \in z) \text{ IFF } x \in u.$$

Infinity. There is an infinite set. Specifically, there is a nonempty set, x, such that for any set $y \in x$, the set $\{y\}$ is also a member of x.

Power Set. All the subsets of a set form another set:

$$\forall x.\, \exists p.\, \forall u.\, u \subseteq x \text{ IFF } u \in p.$$

Replacement. Suppose a formula, ϕ, of set theory defines the graph of a function, that is,

$$\forall x, y, z.\, [\phi(x, y) \text{ AND } \phi(x, z)] \text{ IMPLIES } y = z.$$

Then the image of any set, s, under that function is also a set, t. Namely,

$$\forall s\, \exists t\, \forall y.\, [\exists x.\, \phi(x, y) \text{ IFF } y \in t].$$

This is an oval-shaped region around the origin in the complex plane.

5.1.4 Proving Set Equalities

Two sets are defined to be equal if they contain the same elements. That is, $X = Y$ means that $z \in X$ if and only if $z \in Y$, for all elements, z. (This is actually the first of the ZFC axioms.) So set equalities can be formulated and proved as "iff" theorems. For example:

Theorem 5.1.1 (*Distributive Law* for Sets)**.** *Let A, B, and C be sets. Then:*

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C) \tag{5.1}$$

Foundation. There cannot be an infinite sequence

$$\cdots \in x_n \in \cdots \in x_1 \in x_0$$

of sets each of which is a member of the previous one. This is equivalent to saying every nonempty set has a "member-minimal" element. Namely, define

$$\text{member-minimal}(m, x) ::= [m \in x \text{ AND } \forall y \in x.\, y \notin m].$$

Then the Foundation axiom is

$$\forall x.\, x \neq \emptyset \text{ IMPLIES } \exists m.\, \text{member-minimal}(m, x).$$

are equivalent.

Given a set, s, whose members are nonempty sets no two of which have any element in common, then there is a set, c, consisting of exactly one element from each set in s.

5.1.5 Avoiding Russell's Paradox

These modern ZFC axioms for set theory are much simpler than the system Russell and Whitehead first came up with to avoid paradox. In fact, the ZFC axioms are as simple and intuitive as Frege's original axioms, with one technical addition: the Foundation axiom. Foundation captures the intuitive idea that sets must be built up from "simpler" sets in certain standard ways. And in particular, Foundation implies that no set is ever a member of itself. So the modern resolution of Russell's paradox goes as follows: since $S \not\in S$ for all sets S, it follows that W, defined above, contains every set. This means W can't be a set —or it would be a member of itself.

5.1.6 Does All This Really Work?

So this is where mainstream mathematics stands today: there is a handful of ZFC axioms from which virtually everything else in mathematics can be logically derived. This sounds like a rosy situation, but there are several dark clouds, suggesting that the essence of truth in mathematics is not completely resolved.

- The ZFC axioms weren't etched in stone by God. Instead, they were mostly made up by some guy named Zermelo. Probably some days he forgot his house keys.

  So maybe Zermelo, just like Frege, didn't get his axioms right and will be shot down by some successor to Russell who will use Zermelo's axioms to prove a proposition P and its negation NOT P. Then math would be broken. This sounds crazy, but after all, it has happened before.

  In fact, while there is broad agreement that the ZFC axioms are capable of proving all of standard mathematics, the axioms have some further consequences that sound paradoxical. For example, the Banach-Tarski Theorem says that, as a consequence of the Axiom of Choice, a solid ball can be divided into six pieces and then the pieces can be rigidly rearranged to give *two* solid balls, each the same size as the original!

- Georg *Cantor* was a contemporary of Frege and Russell who first developed the theory of infinite sizes (because he thought he needed it in his study of Fourier series). Cantor raised the question whether there is a set whose size is strictly between the "smallest[2]" infinite set, $\mathbb{N}$, and $\mathcal{P}(\mathbb{N})$; he guessed not:

  Cantor's *Continuum Hypothesis*: There is no set, A, such that $\mathcal{P}(\mathbb{N})$ is strictly bigger than A and A is strictly bigger than $\mathbb{N}$.

[2]See Problem 5.2

The Continuum Hypothesis remains an open problem a century later. Its difficulty arises from one of the deepest results in modern Set Theory — discovered in part by Gödel in the 1930's and Paul Cohen in the 1960's — namely, the ZFC axioms are not sufficient to settle the Continuum Hypothesis: there are two collections of sets, each obeying the laws of ZFC, and in one collection the Continuum Hypothesis is true, and in the other it is false. So settling the Continuum Hypothesis requires a new understanding of what Sets should be to arrive at persuasive new axioms that extend ZFC and are strong enough to determine the truth of the Continuum Hypothesis one way or the other.

- But even if we use more or different axioms about sets, there are some unavoidable problems. In the 1930's, Gödel proved that, assuming that an axiom system like ZFC is consistent —meaning you can't prove both P and NOT(P) for any proposition, P —then the very proposition that the system is consistent (which is not too hard to express as a logical formula) cannot be proved in the system. In other words, no consistent system is strong enough to verify itself.

5.2 Sequences

Sets provide one way to group a collection of objects. Another way is in a *sequence*, which is a list of objects called *terms* or *components*. Short sequences are commonly described by listing the elements between parentheses; for example, (a, b, c) is a sequence with three terms.

While both sets and sequences perform a gathering role, there are several differences.

- The elements of a set are required to be distinct, but terms in a sequence can be the same. Thus, (a, b, a) is a valid sequence of length three, but $\{a, b, a\}$ is a set with two elements —not three.

- The terms in a sequence have a specified order, but the elements of a set do not. For example, (a, b, c) and (a, c, b) are different sequences, but $\{a, b, c\}$ and $\{a, c, b\}$ are the same set.

- Texts differ on notation for the *empty sequence*; we use λ for the empty sequence.

The product operation is one link between sets and sequences. A *product of sets*, $S_1 \times S_2 \times \cdots \times S_n$, is a new set consisting of all sequences where the first component is drawn from S_1, the second from S_2, and so forth. For example, $\mathbb{N} \times \{a, b\}$ is the set of all pairs whose first element is a nonnegative integer and whose second element is an a or a b:

$$\mathbb{N} \times \{a, b\} = \{(0, a), (0, b), (1, a), (1, b), (2, a), (2, b), \ldots\}$$

A product of n copies of a set S is denoted S^n. For example, $\{0,1\}^3$ is the set of all 3-bit sequences:

$$\{0,1\}^3 = \{(0,0,0),(0,0,1),(0,1,0),(0,1,1),(1,0,0),(1,0,1),(1,1,0),(1,1,1)\}$$

5.3 Functions

A *function* assigns an element of one set, called the *domain*, to elements of another set, called the *codomain*. The notation

$$f : A \to B$$

indicates that f is a function with domain, A, and codomain, B. The familiar notation "$f(a) = b$" indicates that f assigns the element $b \in B$ to a. Here b would be called the *value* of f at *argument* a.

Functions are often defined by formulas as in:

$$f_1(x) ::= \frac{1}{x^2}$$

where x is a real-valued variable, or

$$f_2(y,z) ::= y\texttt{10}yz$$

where y and z range over binary strings, or

$$f_3(x,n) ::= \text{ the pair } (n,x)$$

where n ranges over the nonnegative integers.

A function with a finite domain could be specified by a table that shows the value of the function at each element of the domain. For example, a function $f_4(P,Q)$ where P and Q are propositional variables is specified by:

P	Q	$f_4(P,Q)$
T	**T**	**T**
T	**F**	**F**
F	**T**	**T**
F	**F**	**T**

Notice that f_4 could also have been described by a formula:

$$f_4(P,Q) ::= [P \text{ IMPLIES } Q].$$

A function might also be defined by a procedure for computing its value at any element of its domain, or by some other kind of specification. For example, define

$f_5(y)$ to be the length of a left to right search of the bits in the binary string y until a `1` appears, so

$$\begin{aligned} f_5(0010) &= 3, \\ f_5(100) &= 1, \\ f_5(0000) &\text{ is } \text{undefined}. \end{aligned}$$

Notice that f_5 does not assign a value to any string of just `0`'s. This illustrates an important fact about functions: they need not assign a value to every element in the domain. In fact this came up in our first example $f_1(x) = 1/x^2$, which does not assign a value to 0. So in general, functions may be *partial functions*, meaning that there may be domain elements for which the function is not defined. If a function is defined on every element of its domain, it is called a *total function*.

It's often useful to find the set of values a function takes when applied to the elements in *a set* of arguments. So if $f : A \to B$, and S is a subset of A, we define $f(S)$ to be the set of all the values that f takes when it is applied to elements of S. That is,

$$f(S) ::= \{b \in B \mid f(s) = b \text{ for some } s \in S\}.$$

For example, if we let $[r, s]$ denote the interval from r to s on the real line, then $f_1([1, 2]) = [1/4, 1]$.

For another example, let's take the "search for a `1`" function, f_5. If we let X be the set of binary words which start with an even number of `0`'s followed by a `1`, then $f_5(X)$ would be the odd nonnegative integers.

Applying f to a set, S, of arguments is referred to as "applying f pointwise to S", and the set $f(S)$ is referred to as the *image* of S under f.[3] The set of values that arise from applying f to all possible arguments is called the *range* of f. That is,

$$\text{range}(f) ::= f(\text{domain}(f)).$$

Some authors refer to the codomain as the range of a function, but they shouldn't. The distinction between the range and codomain will be important in Sections 5.4.4 and 5.5 when we relate sizes of sets to properties of functions between them.

5.3.1 Function Composition

Doing things step by step is a universal idea. Taking a walk is a literal example, but so is cooking from a recipe, executing a computer program, evaluating a formula, and recovering from substance abuse.

Abstractly, taking a step amounts to applying a function, and going step by step corresponds to applying functions one after the other. This is captured by the operation of *composing* functions. Composing the functions f and g means that

[3]There is a picky distinction between the function f which applies to elements of A and the function which applies f pointwise to subsets of A, because the domain of f is A, while the domain of pointwise-f is $\mathcal{P}(A)$. It is usually clear from context whether f or pointwise-f is meant, so there is no harm in overloading the symbol f in this way.

first f applied is to some argument, x, to produce $f(x)$, and then g is applied to that result to produce $g(f(x))$.

Definition 5.3.1. For functions $f : A \to B$ and $g : B \to C$, the *composition*, $g \circ f$, of g with f is defined to be the function $h : A \to C$ defined by the rule:

$$h(x) ::= (g \circ f)(x) ::= g(f(x)),$$

for all $x \in A$.

Function composition is familiar as a basic concept from elementary calculus, and it plays an equally basic role in discrete mathematics.

5.4 Relations

Relations are another fundamental mathematical data type. Equality and "less-than" are very familiar examples of mathematical relations. These are called *binary relations* because they apply to a pair (a, b) of objects; the equality relation holds for the pair when $a = b$, and less-than holds when a and b are real numbers and $a < b$.

In this section we'll define some basic vocabulary and properties of binary relations.

5.4.1 Binary Relations and Functions

Binary relations are far more general than equality or less-than. Here's the official definition:

Definition 5.4.1. A *binary relation*, R, consists of a set, A, called the *domain* of R, a set, B, called the *codomain* of R, and a subset of $A \times B$ called the *graph of* R.

Notice that Definition 5.4.1 is exactly the same as the definition in Section 5.3 of a *function*, except that it doesn't require the functional condition that, for each domain element, a, there is *at most* one pair in the graph whose first coordinate is a. So a function is a special case of a binary relation.

A relation whose domain is A and codomain is B is said to be "between A and B", or "from A to B." When the domain and codomain are the same set, A, we simply say the relation is "on A." It's common to use infix notation "$a \ R \ b$" to mean that the pair (a, b) is in the graph of R.

For example, we can define an "in-charge of" relation, T, for MIT in Spring '10 to have domain equal to the set, F, of names of the faculty and codomain equal to all the set, N, of subject numbers in the current catalogue. The graph of T contains precisely the pairs of the form

$$(\langle \text{instructor-name} \rangle, \langle \text{subject-num} \rangle)$$

such that the faculty member named $\langle$instructor-name$\rangle$ is in charge of the subject with number $\langle$subject-num$\rangle$ in Spring '10. So graph (T) contains pairs like

```
(A. R. Meyer,  6.042),
(A. R. Meyer,  18.062),
(A. R. Meyer,  6.844),
(T. Leighton,  6.042),
(T. Leighton,  18.062),
(G, Freeman,   6.011),
(G, Freeman,   6.UAT),
(G. Freeman,   6.881)
(G. Freeman,   6.882)
(G. Freeman,   6.UAT)
(T. Eng,       6.UAT)
(J. Guttag,    6.00)
      ⋮
```

This is a surprisingly complicated relation: Meyer is in charge of subjects with three numbers. Leighton is also in charge of subjects with two of these three numbers —because the same subject, Mathematics for Computer Science, has two numbers: 6.042 and 18.062, and Meyer and Leighton are co-in-charge of the subject. Freeman is in-charge of even more subjects numbers (around 20), since as Department Education Officer, he is in charge of whole blocks of special subject numbers. Some subjects, like 6.844 and 6.00 have only one person in-charge. Some faculty, like Guttag, are in charge of only one subject number, and no one else is co-in-charge of his subject, 6.00.

Some subjects in the codomain, N, do not appear in the list —that is, they are not an element of any of the pairs in the graph of T; these are the Fall term only subjects. Similarly, there are faculty in the domain, F, who do not appear in the list because all their in-charge subjects are Fall term only.

5.4.2 Relational Images

The idea of the image of a set under a function extends directly to relations.

Definition 5.4.2. The *image* of a set, Y, under a relation, R, written $R(Y)$, is the set of elements of the codomain, B, of R that are related to some element in Y, namely,

$$R(Y) ::= \{b \in B \mid yRb \text{ for some } y \in Y\}.$$

For example, to find the subject numbers that Meyer is in charge of in Spring '09, we can look for all the pairs of the form

$$(\text{A. Meyer}, \langle\text{subject-number}\rangle)$$

in the graph of the teaching relation, T, and then just list the right hand sides of these pairs. These righthand sides are exactly the image $T(\text{A. Meyer})$, which happens to be $\{6.042, 18.062, 6.844\}$. Similarly, to find the subject numbers that either Freeman or Eng are in charge of, we can collect all the pairs in T of the form

$$(\text{G. Freeman}, \langle\text{subject-number}\rangle)$$

or

$$(\text{T. Eng}, \langle\text{subject-number}\rangle);$$

and list their right hand sides. These right hand sides are exactly the image $T(\{\text{G. Freeman}, \text{T. Eng}\}$. So the partial list of pairs in T given above implies that

$$\{\texttt{6.011, 6.881, 6.882, 6.UAT}\} \subseteq T(\{\text{G. Freeman}, \text{T. Eng}\}\,.$$

Finally, since the domain, F, is the set of all in-charge faculty, $T(F)$ is exactly the set of *all* Spring '09 subjects being taught.

5.4.3 Inverse Relations and Images

Definition 5.4.3. The *inverse*, R^{-1} of a relation $R : A \to B$ is the relation from B to A defined by the rule

$$bR^{-1}a \text{ IFF } a\ R\ B.$$

The image of a set under the relation, R^{-1}, is called the *inverse image* of the set. That is, the inverse image of a set, X, under the relation, R, is $R^{-1}X$.

Continuing with the in-charge example above, we can find the faculty in charge of 6.UAT in Spring '10 can be found by taking the pairs of the form

$$(\langle\text{instructor-name}\rangle\,, 6.UAT)$$

in the graph of the teaching relation, T, and then just listing the left hand sides of these pairs; these turn out to be just Eng and Freeman. These left hand sides are exactly the inverse image of $\{6.\text{UAT}\}$ under T.

Now let D be the set of introductory course 6 subject numbers. These are the subject numbers that start with *6.0* . Now we can likewise find out all the instructors who were in-charge of introductory course 6 subjects in Spring '09, by taking all the pairs of the form $(\langle\text{instructor-name}\rangle\,, 6.0\ldots)$ and list the left hand sides of these pairs. These left hand sides are exactly the inverse image of of D under T. From the part of the graph of T shown above, we can see that

$$\{\text{Meyer, Leighton, Freeman, Guttag}\} \subseteq T^{-1}(D).$$

That is, Meyer, Leighton, Freeman, and Guttag were among the instructors in charge of introductory subjects in Spring '10. Finally, the inverse image under T of the set, N, of all subject numbers is the set of all instructors who were in charge of a Spring '09 subject.

It gets interesting when we write composite expressions mixing images, inverse images and set operations. For example, $T(T^{-1}(D))$ is the set of Spring '09 subjects that have an instructor in charge who also is in in charge of an introductory subject. So $T(T^{-1}(D)) - D$ are the advanced subjects with someone in-charge who is also in-charge of an introductory subject. Similarly, $T^{-1}(D) \cap T^{-1}(N - D)$ is the set of faculty in charge of both an introductory *and* an advanced subject in Spring '09.

5.4.4 Surjective and Injective Relations

There are a few properties of relations that will be useful when we take up the topic of counting because they imply certain relations between the *sizes* of domains and codomains. We say a binary relation $R : A \to B$ is:

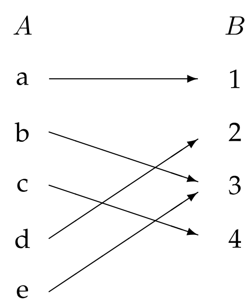
- *total* when every element of A is assigned to some element of B; more concisely, R is total iff $A = RB$.
- *surjective* when every element of B is mapped to *at least once*; more concisely, R is surjective iff $R(A) = B$.
- *total* when every element of A is assigned to some element of B; more concisely, R is total iff $A = R^{-1}(B)$.
- *injective* if every element of B is mapped to *at most once*, and
- *bijective* if R is total, surjective, and injective *function*.[4]

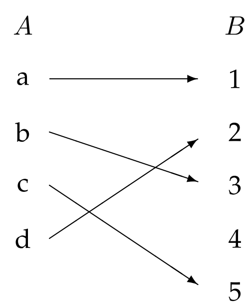
Note that this definition of R being total agrees with the definition in Section 5.3 when R is a function.

If R is a binary relation from A to B, we define $R(A)$ to to be the *range* of R. So a relation is surjective iff its range equals its codomain. Again, in the case that R is a function, these definitions of "range" and "total" agree with the definitions in Section 5.3.

5.4.5 Relation Diagrams

We can explain all these properties of a relation $R : A \to B$ in terms of a diagram where all the elements of the domain, A, appear in one column (a very long one if A is infinite) and all the elements of the codomain, B, appear in another column, and we draw an arrow from a point a in the first column to a point b in the second column when a is related to b by R. For example, here are diagrams for two functions:





[4]These words "surjective," "injective," and "bijective" are not very memorable. Some authors use the possibly more memorable phrases *onto* for surjective, *one-to-one* for injective, and *exact correspondence* for bijective.

Here is what the definitions say about such pictures:

- "R is a function" means that every point in the domain column, A, has *at most one arrow out of it*.
- "R is total" means that *every* point in the A column has *at least one arrow out of it*. So if R is a function, being total really means every point in the A column has *exactly one arrow out of it*.
- "R is surjective" means that *every* point in the codomain column, B, has *at least one arrow into it*.
- "R is injective" means that every point in the codomain column, B, has *at most one arrow into it*.
- "R is bijective" means that *every* point in the A column has exactly one arrow out of it, and *every* point in the B column has exactly one arrow into it.

So in the diagrams above, the relation on the left is a total, surjective function (every element in the A column has exactly one arrow out, and every element in the B column has at least one arrow in), but not injective (element 3 has two arrows going into it). The relation on the right is a total, injective function (every element in the A column has exactly one arrow out, and every element in the B column has at most one arrow in), but not surjective (element 4 has no arrow going into it).

Notice that the arrows in a diagram for R precisely correspond to the pairs in the graph of R. But graph(R) does not determine by itself whether R is total or surjective; we also need to know what the domain is to determine if R is total, and we need to know the codomain to tell if it's surjective.

Example 5.4.4. The function defined by the formula $1/x^2$ is total if its domain is $\mathbb{R}^+$ but partial if its domain is some set of real numbers including 0. It is bijective if its domain and codomain are both $\mathbb{R}^+$, but neither injective nor surjective if its domain and codomain are both $\mathbb{R}$.

5.5 Cardinality

5.5.1 Mappings and Cardinality

The relational properties in Section 5.4 are useful in figuring out the relative sizes of domains and codomains.

If A is a finite set, we let $|A|$ be the number of elements in A. A finite set may have no elements (the empty set), or one element, or two elements,… or any nonnegative integer number of elements.

Now suppose $R : A \to B$ is a function. Then every arrow in the diagram for R comes from exactly one element of A, so the number of arrows is at most the number of elements in A. That is, if R is a function, then

$$|A| \geq \#\text{arrows}.$$

Similarly, if R is surjective, then every element of B has an arrow into it, so there must be at least as many arrows in the diagram as the size of B. That is,

$$\#\text{arrows} \geq |B|.$$

Combining these inequalities implies that if R is a surjective function, then $|A| \geq |B|$. In short, if we write $A \operatorname{surj} B$ to mean that there is a surjective function from A to B, then we've just proved a lemma: if $A \operatorname{surj} B$, then $|A| \geq |B|$. The following definition and lemma lists this statement and three similar rules relating domain and codomain size to relational properties.

Definition 5.5.1. Let A, B be (not necessarily finite) sets. Then

1. $A \operatorname{surj} B$ iff there is a surjective *function* from A to B.
2. $A \operatorname{inj} B$ iff there is a total injective *relation* from A to B.
3. $A \operatorname{bij} B$ iff there is a bijection from A to B.
4. $A \operatorname{strict} B$ iff $A \operatorname{surj} B$, but not $B \operatorname{surj} A$.

Lemma 5.5.2. *[Mapping Rules] Let A and B be finite sets.*

1. *If $A \operatorname{surj} B$, then $|A| \geq |B|$.*
2. *If $A \operatorname{inj} B$, then $|A| \leq |B|$.*
3. *If $R \operatorname{bij} B$, then $|A| = |B|$.*
4. *If $R \operatorname{strict} B$, then $|A| > |B|$.*

Mapping rule 2. can be explained by the same kind of "arrow reasoning" we used for rule 1. Rules 3. and 4. are immediate consequences of these first two mapping rules.

5.5.2 The sizes of infinite sets

Mapping Rule 1 has a converse: if the size of a finite set, A, is greater than or equal to the size of another finite set, B, then it's always possible to define a surjective function from A to B. In fact, the surjection can be a total function. To see how this works, suppose for example that

$$\begin{aligned} A &= \{a_0, a_1, a_2, a_3, a_4, a_5\} \\ B &= \{b_0, b_1, b_2, b_3\}. \end{aligned}$$

Then define a total function $f : A \to B$ by the rules

$$f(a_0) ::= b_0,\ f(a_1) ::= b_1,\ f(a_2) ::= b_2,\ f(a_3) = f(a_4) = f(a_5) ::= b_3.$$

In fact, if A and B are finite sets of the same size, then we could also define a bijection from A to B by this method.

In short, we have figured out if A and B are finite sets, then $|A| \geq |B|$ *if and only if* $A \operatorname{surj} B$, and similar iff's hold for all the other Mapping Rules:

Lemma 5.5.3. *For* finite *sets, A, B,*

$$\begin{aligned}
|A| \geq |B| \quad & \textit{iff} \quad A \text{ surj } B, \\
|A| \leq |B| \quad & \textit{iff} \quad A \text{ inj } B, \\
|A| = |B| \quad & \textit{iff} \quad A \text{ bij } B, \\
|A| > |B| \quad & \textit{iff} \quad A \text{ strict } B.
\end{aligned}$$

This lemma suggests a way to generalize size comparisons to infinite sets, namely, we can think of the relation surj as an "*at least as big as*" relation between sets, even if they are infinite. Similarly, the relation bij can be regarded as a "*same size*" relation between (possibly infinite) sets, and strict can be thought of as a "*strictly bigger* than" relation between sets.

Warning: We haven't, and won't, define what the "size" of an infinite is. The definition of infinite "sizes" is cumbersome and technical, and we can get by just fine without it. All we need are the "as big as" and "same size" relations, surj and bij, between sets.

But there's something else to watch out for. We've referred to surj as an "as big as" relation and bij as a "same size" relation on sets. Of course most of the "as big as" and "same size" properties of surj and bij on finite sets do carry over to infinite sets, but *some important ones don't* —as we're about to show. So you have to be careful: don't assume that surj has any particular "as big as" property on *infinite* sets until it's been proved.

Let's begin with some familiar properties of the "as big as" and "same size" relations on finite sets that do carry over exactly to infinite sets:

Lemma 5.5.4. *For any sets, A, B, C,*

1. *A surj B and B surj C, implies A surj C.*

2. *A bij B and B bij C, implies A bij C.*

3. *A bij B implies B bij A.*

Lemma 5.5.4.1 and 5.5.4.2 follow immediately from the fact that compositions of surjections are surjections, and likewise for bijections, and Lemma 5.5.4.3 follows from the fact that the inverse of a bijection is a bijection. We'll leave a proof of these facts to Problem 5.1.

Another familiar property of finite sets carries over to infinite sets, but this time it's not so obvious:

Theorem 5.5.5 (Schröder-Bernstein)**.** *For any sets A, B, if A surj B and B surj A, then A bij B.*

That is, the Schröder-Bernstein Theorem says that if A is at least as big as B and conversely, B is at least as big as A, then A is the same size as B. Phrased this way, you might be tempted to take this theorem for granted, but that would be a mistake. For infinite sets A and B, the Schröder-Bernstein Theorem is actually pretty technical. Just because there is a surjective function $f : A \to B$ —which need not be a bijection —and a surjective function $g : B \to A$ —which also need not be a bijection —it's not at all clear that there must be a bijection $e : A \to B$. The idea is to construct e from parts of both f and g. We'll leave the actual construction to Problem 5.6.

5.5.3 Infinity is different

A basic property of finite sets that does *not* carry over to infinite sets is that adding something new makes a set bigger. That is, if A is a finite set and $b \notin A$, then $|A \cup \{b\}| = |A| + 1$, and so A and $A \cup \{b\}$ are not the same size. But if A is infinite, then these two sets *are* the same size!

Lemma 5.5.6. *Let A be a set and $b \notin A$. Then A is infinite iff A bij $A \cup \{b\}$.*

Proof. Since A is *not* the same size as $A \cup \{b\}$ when A is finite, we only have to show that $A \cup \{b\}$ *is* the same size as A when A is infinite.

That is, we have to find a bijection between $A \cup \{b\}$ and A when A is infinite. Here's how: since A is infinite, it certainly has at least one element; call it a_0. But since A is infinite, it has at least two elements, and one of them must not be equal to a_0; call this new element a_1. But since A is infinite, it has at least three elements, one of which must not equal a_0 or a_1; call this new element a_2. Continuing in the way, we conclude that there is an infinite sequence $a_0, a_1, a_2, \dots, a_n, \dots$ of different elements of A. Now it's easy to define a bijection $e : A \cup \{b\} \to A$:

$$\begin{aligned} e(b) &::= a_0, \\ e(a_n) &::= a_{n+1} && \text{for } n \in \mathbb{N}, \\ e(a) &::= a && \text{for } a \in A - \{b, a_0, a_1, \dots\}. \end{aligned}$$

■

A set, C, is *countable* iff its elements can be listed in order, that is, the distinct elements is A are precisely

$$c_0, c_1, \dots, c_n, \dots.$$

This means that if we defined a function, f, on the nonnegative integers by the rule that $f(i) ::= c_i$, then f would be a bijection from $\mathbb{N}$ to C. More formally,

Definition 5.5.7. A set, C, is *countably infinite* iff $\mathbb{N}$ bij C. A set is *countable* iff it is finite or countably infinite.

A small modification[5] of the proof of Lemma 5.5.6 shows that countably infinite sets are the "smallest" infinite sets, namely, if A is a countably infinite set, then A surj $\mathbb{N}$.

Since adding one new element to an infinite set doesn't change its size, it's obvious that neither will adding any *finite* number of elements. It's a common

[5]See Problem 5.2

mistake to think that this proves that you can throw in countably infinitely many new elements. But just because it's ok to do something any finite number of times doesn't make it OK to do an infinite number of times. For example, starting from 3, you can add 1 any finite number of times and the result will be some integer greater than or equal to 3. But if you add add 1 a countably infinite number of times, you don't get an integer at all.

It turns out you really can add a countably infinite number of new elements to a countable set and still wind up with just a countably infinite set, but another argument is needed to prove this:

Lemma 5.5.8. *If A and B are countable sets, then so is $A \cup B$.*

Proof. Suppose the list of distinct elements of A is $a_0, a_1, \dots$ and the list of B is $b_0, b_1, \dots$. Then a list of all the elements in $A \cup B$ is just

$$a_0, b_0, a_1, b_1, \dots a_n, b_n, \dots. \tag{5.2}$$

Of course this list will contain duplicates if A and B have elements in common, but then deleting all but the first occurrences of each element in list (5.2) leaves a list of all the distinct elements of A and B. ■

5.5.4 Power sets are strictly bigger

It turns out that the ideas behind Russell's Paradox, which caused so much trouble for the early efforts to formulate Set Theory, also lead to a correct and astonishing fact discovered by Georg Cantor in the late nineteenth century: infinite sets are *not all the same size*.

In particular,

Theorem 5.5.9. *For any set, A, the power set, $\mathcal{P}(A)$, is strictly bigger than A.*

Proof. First of all, $\mathcal{P}(A)$ is as big as A: for example, the partial function $f : \mathcal{P}(A) \to A$, where $f(\{a\}) ::= a$ for $a \in A$ and f is only defined on one-element sets, is a surjection.

To show that $\mathcal{P}(A)$ is strictly bigger than A, we have to show that if g is a function from A to $\mathcal{P}(A)$, then g is not a surjection. So, mimicking Russell's Paradox, define

$$A_g ::= \{a \in A \mid a \notin g(a)\}.$$

Now A_g is a well-defined subset of A, which means it is a member of $\mathcal{P}(A)$. But A_g can't be in the range of g, because if it were, we would have

$$A_g = g(a_0)$$

for some $a_0 \in A$, so by definition of A_g,

$$a \in g(a_0) \quad \text{iff} \quad a \in A_g \quad \text{iff} \quad a \notin g(a)$$

for all $a \in A$. Now letting $a = a_0$ yields the contradiction

$$a_0 \in g(a_0) \quad \text{iff} \quad a_0 \notin g(a_0).$$

So g is not a surjection, because there is an element in the power set of A, namely the set A_g, that is not in the range of g. ■

Larger Infinities

There are lots of different sizes of infinite sets. For example, starting with the infinite set, $\mathbb{N}$, of nonnegative integers, we can build the infinite sequence of sets

$$\mathbb{N},\ \mathcal{P}(\mathbb{N}),\ \mathcal{P}(\mathcal{P}(\mathbb{N})),\ \mathcal{P}(\mathcal{P}(\mathcal{P}(\mathbb{N}))),\ \ldots.$$

By Theorem 5.5.9, each of these sets is strictly bigger than all the preceding ones. But that's not all: the union of all the sets in the sequence is strictly bigger than each set in the sequence (see Problem 5.7). In this way you can keep going, building still bigger infinities.

So there is an endless variety of different size infinities.

5.6 Infinities in Computer Science

We've run into a lot of computer science students who wonder why they should care about infinite sets. They point out that any data set in a computer memory is limited by the size of memory, and there is a finite limit on the possible size of computer memory for the simple reason that the universe is (or at least appears to be) finite.

The problem with this argument is that universe-size bounds on data items are so big and uncertain (the universe seems to be getting bigger all the time), that it's simply not helpful to make use of such bounds. For example, by this argument the physical sciences shouldn't assume that measurements might yield arbitrary real numbers, because there can only be a finite number of finite measurements in a universe with a finite lifetime. What do you think scientific theories would look like without using the infinite set of real numbers?

Similarly, in computer science it simply isn't plausible that writing a program to add nonnegative integers with up to as many digits as, say, the stars in the sky (billions of galaxies each with billions of stars), would be any different than writing a program that would add any two integers no matter how many digits they had.

That's why basic programming data types like integers or strings, for example, can be defined without imposing any bound on the sizes of data items. Each datum of type `string` has only a finite number of letters, but there are an infinite number of data items of type `string`. When we then consider string procedures of type `string-->string`, not only are there an infinite number of such procedures, but each procedure generally behaves differently on different inputs, so that a single `string-->string` procedure may embody an infinite number of behaviors. In short, an educated computer scientist can't get around having to cope with infinite sets.

On the other hand, the more exotic theory of different size infinities and continuum hypotheses rarely comes up in mainstream mathematics, and it hardly comes up at all in computer science, where the focus is mainly on finite sets, and occasionallly on countable sets. In practice, only logicians and set theorists have to worry about collections that are too big to be sets. In fact, at the end of the 19th century, the general mathematical community doubted the relevance of what they called "Cantor's paradise" of unfamiliar sets of arbitrary infinite size. So if the romance of really big infinities doesn't appeal to you, be assured that not knowing about them won't lower your professional abilities as a computer scientist.

Yet the idea behind Russell's paradox and Cantor's proof embodies the simplest form of what is known as a "diagonal argument." Diagonal arguments are used to prove many fundamental results about the limitations of computation, such as the undecidability of the Halting Problem for programs (see Problem 5.8) and the inherent, unavoidable, inefficiency (exponential time or worse) of procedures for other computational problems. So computer scientists do need to study diagonal arguments in order to understand the logical limits of computation.

5.6.1 Problems

Class Problems

Problem 5.1.
Define a *surjection relation*, surj, on sets by the rule

Definition. $A \ \mathrm{surj}\ B$ iff there is a surjective **function** from A to B.

Define the *injection relation*, inj, on sets by the rule

Definition. $A \ \mathrm{inj}\ B$ iff there is a total injective *relation* from A to B.

(a) Prove that if $A \ \mathrm{surj}\ B$ and $B \ \mathrm{surj}\ C$, then $A \ \mathrm{surj}\ C$.

(b) Explain why $A \ \mathrm{surj}\ B$ iff $B \ \mathrm{inj}\ A$.

(c) Conclude from (a) and (b) that if $A \ \mathrm{inj}\ B$ and $B \ \mathrm{inj}\ C$, then $A \ \mathrm{inj}\ C$.

Problem 5.2. (a) Several students felt the proof of Lemma 5.5.6 was worrisome, if not circular. What do you think?

(b) Use the proof of Lemma 5.5.6 to show that if A is an infinite set, then there is surjective function from A to $\mathbb{N}$, that is, every infinite set is "as big as" the set of nonnegative integers.

Problem 5.3.
Let $R : A \to B$ be a binary relation. Use an arrow counting argument to prove the following generalization of the Mapping Rule:

Lemma. *If R is a function, and $X \subseteq A$, then*

$$|X| \geq |XR| .$$

Problem 5.4.
Let $A = \{a_0, a_1, \ldots, a_{n-1}\}$ be a set of size n, and $B = \{b_0, b_1, \ldots, b_{m-1}\}$ a set of size m. Prove that $|A \times B| = mn$ by defining a simple bijection from $A \times B$ to the nonnegative integers from 0 to $mn - 1$.

Problem 5.5.
The rational numbers fill in all the spaces between the integers, so a first thought is that there must be more of them than the integers, but it's not true. In this problem you'll show that there are the same number of nonnegative rational as nonnegative integers. In short, the nonnegative rationals are countable.

(a) Describe a bijection between all the integers, $\mathbb{Z}$, and the nonnegative integers, $\mathbb{N}$.

(b) Define a bijection between the nonnegative integers and the set, $\mathbb{N} \times \mathbb{N}$, of all the ordered pairs of nonnegative integers:

$$\begin{array}{l}
(0,0),(0,1),(0,2),(0,3),(0,4),\ldots \\
(1,0),(1,1),(1,2),(1,3),(1,4),\ldots \\
(2,0),(2,1),(2,2),(2,3),(2,4),\ldots \\
(3.0),(3,1),(3,2),(3,3),(3,4),\ldots \\
\quad\quad\vdots
\end{array}$$

(c) Conclude that $\mathbb{N}$ is the same size as the set, $\mathbb{Q}$, of all nonnegative rational numbers.

Problem 5.6.
Suppose sets A and B have no elements in common, and

- A is as small as B because there is a total injective function $f : A \to B$, and
- B is as small as A because there is a total injective function $g : B \to A$.

Picturing the diagrams for f and g, there is *exactly one* arrow *out* of each element —a left-to-right f-arrow if the element in A and a right-to-left g-arrow if the element in B. This is because f and g are total functions. Also, there is *at most one* arrow *into* any element, because f and g are injections.

So starting at any element, there is a unique, and unending path of arrows going forwards. There is also a unique path of arrows going backwards, which might be unending, or might end at an element that has no arrow into it. These paths are completely separate: if two ran into each other, there would be two arrows into the element where they ran together.

This divides all the elements into separate paths of four kinds:

i. paths that are infinite in both directions,

ii. paths that are infinite going forwards starting from some element of A.

iii. paths that are infinite going forwards starting from some element of B.

iv. paths that are unending but finite.

(a) What do the paths of the last type (iv) look like?

(b) Show that for each type of path, either

- the f-arrows define a bijection between the A and B elements on the path, or
- the g-arrows define a bijection between B and A elements on the path, or
- both sets of arrows define bijections.

For which kinds of paths do both sets of arrows define bijections?

(c) Explain how to piece these bijections together to prove that A and B are the same size.

Problem 5.7.
There are lots of different sizes of infinite sets. For example, starting with the infinite set, $\mathbb{N}$, of nonnegative integers, we can build the infinite sequence of sets

$$\mathbb{N},\ \mathcal{P}(\mathbb{N}),\ \mathcal{P}(\mathcal{P}(\mathbb{N})),\ \mathcal{P}(\mathcal{P}(\mathcal{P}(\mathbb{N}))),\ \ldots.$$

By Theorem 5.5.9 from the Notes, each of these sets is *strictly bigger*[6] than all the preceding ones. But that's not all: if we let U be the union of the sequence of sets above, then U is strictly bigger than every set in the sequence! Prove this:

Lemma. *Let $\mathcal{P}^n(\mathbb{N})$ be the nth set in the sequence, and*

$$U ::= \bigcup_{n=0}^{\infty} \mathcal{P}^n(\mathbb{N}).$$

Then

[6] Reminder: set A is *strictly bigger* than set B just means that A surj B, but NOT(B surj A).

1. *U surj $\mathcal{P}^n(\mathbb{N})$ for every $n \in \mathbb{N}$, but*

2. *there is no $n \in \mathbb{N}$ for which $\mathcal{P}^n(\mathbb{N})$ surj U.*

Now of course, we could take $U, \mathcal{P}(U), \mathcal{P}(\mathcal{P}(U)), \ldots$ and can keep on indefinitely building still bigger infinities.

Problem 5.8.
Let's refer to a programming procedure (written in your favorite programming language —C++, or Java, or Python, ...) as a *string procedure* when it is applicable to data of type `string` and only returns values of type `boolean`. When a string procedure, P, applied to a `string`, s, returns **True**, we'll say that P *recognizes* s. If $\mathcal{R}$ is the set of strings that P recognizes, we'll call P a *recognizer* for $\mathcal{R}$.

(a) Describe how a recognizer would work for the set of strings containing only lower case Roman letter —`a,b,...,z` —such that each letter occurs twice in a row. For example, `aaccaabbzz`, is such a string, but `abb`, `00bb`, `AAbb`, and `a` are not. (Even better, actually write a recognizer procedure in your favorite programming language).

A set of `string`s is called *recognizable* if there is a recognizer procedure for it.

When you actually program a procedure, you have to type the program text into a computer system. This means that every procedure is described by some `string` of typed characters. If a `string`, s, is actually the typed description of some string procedure, let's refer to that procedure as P_s. You can think of P_s as the result of compiling s.[7]

In fact, it will be helpful to associate every string, s, with a procedure, P_s; we can do this by defining P_s to be some fixed string procedure —it doesn't matter which one —whenever s is not the typed description of an actual procedure that can be applied to `string` s. The result of this is that we have now defined a total function, f, mapping every `string`, s, to the set, $f(s)$, of `string`s recognized by P_s. That is we have a total function,

$$f : \texttt{string} \to \mathcal{P}(\texttt{string}). \tag{5.3}$$

(b) Explain why the actual range of f is the set of all recognizable sets of strings.

This is exactly the set up we need to apply the reasoning behind Russell's Paradox to define a set that is not in the range of f, that is, a set of strings, $\mathcal{N}$, that is *not* recognizable.

(c) Let

$$\mathcal{N} ::= \{s \in \texttt{string} \mid s \notin f(s)\}.$$

[7] The string, s, and the procedure, P_s, have to be distinguished to avoid a type error: you can't apply a string to string. For example, let s be the string that you wrote as your program to answer part (a). Applying s to a string argument, say `oorrmm`, should throw a type exception; what you need to do is apply the procedure P_s to `oorrmm`. This should result in a returned value **True**, since `oorrmm` consists of three pairs of lowercase roman letters

Prove that $\mathcal{N}$ is not recognizable.

Hint: Similar to Russell's paradox or the proof of Theorem 5.5.9.

(d) Discuss what the conclusion of part (c) implies about the possibility of writing "program analyzers" that take programs as inputs and analyze their behavior.

Problem 5.9.
Though it was a serious challenge for set theorists to overcome Russells' Paradox, the idea behind the paradox led to some important (and correct :-)) results in Logic and Computer Science.

To show how the idea applies, let's recall the formulas from Problem 1.13 that made assertions about binary strings. For example, one of the formulas in that problem was

$$\text{NOT}[\exists y\,\exists z.s = y1z] \qquad \textbf{(all-0s)}$$

This formula defines a property of a binary string, s, namely that s has no occurrence of a 1. In other words, s is a string of (zero or more) 0's. So we can say that this formula *describes* the set of strings of 0's.

More generally, when G is any formula that defines a string property, let ok-strings(G) be the set of all the strings that have this property. A set of binary strings that equals ok-strings(G) for some G is called a *describable* set of strings. So, for example, the set of all strings of 0's is describable because it equals ok-strings(**all-0s**).

Now let's shift gears for a moment and think about the fact that formula **all-0s** appears above. This happens because instructions for formatting the formula were generated by a computer text processor (in 6.042, we use the LaTeX text processing system), and then an image suitable for printing or display was constructed according to these instructions. Since everybody knows that data is stored in computer memory as binary strings, this means there must have been some binary string in computer memory —call it $t_{\textbf{all-0s}}$ —that enabled a computer to display formula **all-0s** once $t_{\textbf{all-0s}}$ was retrieved from memory.

In fact, it's not hard to find ways to represent *any* formula, G, by a corresponding binary word, t_G, that would allow a computer to reconstruct G from t_G. We needn't be concerned with how this reconstruction process works; all that matters for our purposes is that every formula, G, has a representation as binary string, t_G.

Now let

$$V ::= \{t_G \mid G \text{ defines a property of strings and } t_G \notin \text{ok-strings}(G)\}.$$

Use reasoning similar to Russell's paradox to show that V is not describable.

Homework Problems

Problem 5.10.
Let $f : A \to B$ and $g : B \to C$ be functions and $h : A \to C$ be their composition, namely, $h(a) ::= g(f(a))$ for all $a \in A$.

(a) Prove that if f and g are surjections, then so is h.

(b) Prove that if f and g are bijections, then so is h.

(c) If f is a bijection, then define $f' : B \to A$ so that

$$f'(b) ::= \text{ the unique } a \in A \text{ such that } f(a) = b.$$

Prove that f' is a bijection. (The function f' is called the *inverse* of f. The notation f^{-1} is often used for the inverse of f.)

Problem 5.11.
In this problem you will prove a fact that may surprise you —or make you even more convinced that set theory is nonsense: the half-open unit interval is actually the *same size* as the nonnegative quadrant of the real plane![8] Namely, there is a bijection from $(0, 1]$ to $[0, \infty)^2$.

(a) Describe a bijection from $(0, 1]$ to $[0, \infty)$.

Hint: $1/x$ almost works.

(b) An infinite sequence of the decimal digits $\{\texttt{0}, \texttt{1}, \ldots, \texttt{9}\}$ will be called *long* if it has infinitely many occurrences of some digit other than 0. Let L be the set of all such long sequences. Describe a bijection from L to the half-open real interval $(0, 1]$.

Hint: Put a decimal point at the beginning of the sequence.

(c) Describe a surjective function from L to L^2 that involves alternating digits from two long sequences. a *Hint:* The surjection need not be total.

(d) Prove the following lemma and use it to conclude that there is a bijection from L^2 to $(0, 1]^2$.

Lemma 5.6.1. *Let A and B be nonempty sets. If there is a bijection from A to B, then there is also a bijection from $A \times A$ to $B \times B$.*

(e) Conclude from the previous parts that there is a surjection from $(0, 1]$ and $(0, 1]^2$. Then appeal to the Schröder-Bernstein Theorem to show that there is actually a bijection from $(0, 1]$ and $(0, 1]^2$.

(f) Complete the proof that there is a bijection from $(0, 1]$ to $[0, \infty)^2$.

Problem 5.12.
Let $[\mathbb{N} \to \{1, 2, 3\}]$ be the set of all sequences containing only the numbers 1, 2, and

[8] The half open unit interval, $(0, 1]$, is $\{r \in \mathbb{R} \mid 0 < r \leq 1\}$. Similarly, $[0, \infty) ::= \{r \in \mathbb{R} \mid r \geq 0\}$.

3, for example,

$$\begin{aligned}&(1, 1, 1, 1...),\\&(2, 2, 2, 2...),\\&(3, 2, 1, 3...).\end{aligned}$$

For any sequence, s, let $s[m]$ be its mth element.

Prove that $[\mathbb{N} \to \{1, 2, 3\}]$ is uncountable.

Hint: Suppose there was a list

$$\mathcal{L} = \text{sequence}_0, \text{sequence}_1, \text{sequence}_2, \ldots$$

of sequences in $[\mathbb{N} \to \{1, 2, 3\}]$ and show that there is a "diagonal" sequence diag $\in [\mathbb{N} \to \{1, 2, 3\}]$ that does not appear in the list. Namely,

$$\text{diag} ::= r(\text{sequence}_0[0]), r(\text{sequence}_1[1]), r(\text{sequence}_2[2]), \ldots,$$

where $r : \{1, 2, 3\} \to \{1, 2, 3\}$ is some function such that $r(i) \neq i$ for $i = 1, 2, 3$.

Problem 5.13.

For any sets, A, and B, let $[A \to B]$ be the set of total functions from A to B. Prove that if A is not empty and B has more than one element, then NOT$(A \text{ surj } [A \to B])$.

Hint: Suppose there is a function, σ, that maps each element $a \in A$ to a function $\sigma_a : A \to B$. Pick any two elements of B; call them 0 and 1. Then define

$$\text{diag}(a) ::= \begin{cases} 0 \text{ if } \sigma_a(a) = 1, \\ 1 \text{ otherwise.} \end{cases}$$

5.7 Glossary of Symbols

symbol	meaning
$\in$	is a member of
$\subseteq$	is a subset of
$\subset$	is a proper subset of
$\cup$	set union
$\cap$	set intersection
$\overline{A}$	complement of a set, A
$\mathcal{P}(A)$	powerset of a set, A
$\emptyset$	the empty set, $\{\}$
$\mathbb{N}$	nonnegative integers
$\mathbb{Z}$	integers
$\mathbb{Z}^+$	positive integers
$\mathbb{Z}^-$	negative integers
$\mathbb{Q}$	rational numbers
$\mathbb{R}$	real numbers
$\mathbb{C}$	complex numbers
λ	the empty string/list

Chapter 6

Partial Orders and Equivalence Relations

Partial orders are a kind of binary relation that come up a lot. The familiar $\leq$ order on numbers is a partial order, but so is the containment relation on sets and the divisibility relation on integers.

Partial orders have particular importance in computer science because they capture key concepts used, for example, in solving task scheduling problems, analyzing concurrency control, and proving program termination.

6.1 Axioms for Partial Orders

The prerequisite structure among MIT subjects provides a nice illustration of partial orders. Here is a table indicating some of the prerequisites of subjects in the the Course 6 program of Spring '07:

Direct Prerequisites	Subject
18.01	6.042
18.01	18.02
18.01	18.03
8.01	8.02
6.001	6.034
6.042	6.046
18.03, 8.02	6.002
6.001, 6.002	6.004
6.001, 6.002	6.003
6.004	6.033
6.033	6.857
6.046	6.840

Since 18.01 is a direct prerequisite for 6.042, a student must take 18.01 before 6.042. Also, 6.042 is a direct prerequisite for 6.046, so in fact, a student has to take *both* 18.01 and 6.042 before taking 6.046. So 18.01 is also really a prerequisite for 6.046, though an implicit or indirect one; we'll indicate this by writing

$$18.01 \to 6.046.$$

This prerequisite relation has a basic property known as *transitivity*: if subject a is an indirect prerequisite of subject b, and b is an indirect prerequisite of subject c, then a is also an indirect prerequisite of c.

In this table, a longest sequence of prerequisites is

$$18.01 \to 18.03 \to 6.002 \to 6.004 \to 6.033 \to 6.857$$

so a student would need at least six terms to work through this sequence of subjects. But it would take a lot longer to complete a Course 6 major if the direct prerequisites led to a situation[1] where two subjects turned out to be prerequisites of *each other*! So another crucial property of the prerequisite relation is that if $a \to b$, then it is not the case that $b \to a$. This property is called *asymmetry*.

Another basic example of a partial order is the subset relation, $\subseteq$, on sets. In fact, we'll see that every partial order can be represented by the subset relation.

Definition 6.1.1. A binary relation, R, on a set A is:

- *transitive* iff $[a\ R\ b \text{ and } b\ R\ c]$ IMPLIES $a\ R\ c$ for every $a, b, c \in A$,
- *asymmetric* iff $a\ R\ b$ IMPLIES NOT$(b\ R\ a)$ for all $a, b \in A$,
- a *strict partial order* iff it is transitive and asymmetric.

So the prerequisite relation, $\to$, on subjects in the MIT catalogue is a strict partial order. More familiar examples of strict partial orders are the relation, $<$, on real numbers, and the proper subset relation, $\subset$, on sets.

The subset relation, $\subseteq$, on sets and $\leq$ relation on numbers are examples of *reflexive* relations in which each element is related to itself. Reflexive partial orders are called *weak* partial orders. Since asymmetry is incompatible with reflexivity, the asymmetry property in weak partial orders is relaxed so it applies only to two different elements. This relaxation of the asymmetry is called antisymmetry:

Definition 6.1.2. A binary relation, R, on a set A, is

- *reflexive* iff $a\ R\ a$ for all $a \in A$,
- *antisymmetric* iff $a\ R\ b$ IMPLIES NOT$(b\ R\ a)$ for all $a \neq b \in A$,
- a *weak partial order* iff it is transitive, reflexive and antisymmetric.

[1] MIT's Committee on Curricula has the responsibility of watching out for such bugs that might creep into departmental requirements.

Some authors define partial orders to be what we call weak partial orders, but we'll use the phrase "partial order" to mean either a weak or strict one.

For weak partial orders in general, we often write an ordering-style symbol like $\preceq$ or $\sqsubseteq$ instead of a letter symbol like R. (General relations are usually denoted by a letter like R instead of a cryptic squiggly symbol, so $\preceq$ is kind of like the musical performer/composer Prince, who redefined the spelling of his name to be his own squiggly symbol. A few years ago he gave up and went back to the spelling "Prince.") Likewise, we generally use $\prec$ or $\sqsubset$ to indicate a strict partial order.

Two more examples of partial orders are worth mentioning:

Example 6.1.3. Let A be some family of sets and define $a\ R\ b$ iff $a \supset b$. Then R is a strict partial order.

For integers, m, n we write $m \mid n$ to mean that m *divides* n, namely, there is an integer, k, such that $n = km$.

Example 6.1.4. The divides relation is a weak partial order on the nonnegative integers.

6.2 Representing Partial Orders by Set Containment

Axioms can be a great way to abstract and reason about important properties of objects, but it helps to have a clear picture of the things that satisfy the axioms. We'll show that every partial order can be pictured as a collection of sets related by containment. That is, every partial order has the "same shape" as such a collection. The technical word for "same shape" is "isomorphic."

Definition 6.2.1. A binary relation, R, on a set, A, is *isomorphic* to a relation, S, on a set D iff there is a relation-preserving bijection from A to D. That is, there is bijection $f : A \to D$, such that for all $a, a' \in A$,

$$a\ R\ a' \quad \text{iff} \quad f(a)\ S\ f(a').$$

Theorem 6.2.2. *Every weak partial order, $\preceq$, is isomorphic to the subset relation, on a collection of sets.*

To picture a partial order, $\preceq$, on a set, A, as a collection of sets, we simply represent each element A by the set of elements that are $\preceq$ to that element, that is,

$$a \longleftrightarrow \{b \in A \mid b \preceq a\}.$$

For example, if $\preceq$ is the divisibility relation on the set of integers, $\{1, 3, 4, 6, 8, 12\}$, then we represent each of these integers by the set of integers in A that divides it.

So

$$\begin{aligned}
1 &\longleftrightarrow \{1\} \\
3 &\longleftrightarrow \{1,3\} \\
4 &\longleftrightarrow \{1,4\} \\
6 &\longleftrightarrow \{1,3,6\} \\
8 &\longleftrightarrow \{1,4,8\} \\
12 &\longleftrightarrow \{1,3,4,6,12\}
\end{aligned}$$

So, the fact that $3 \mid 12$ corresponds to the fact that $\{1,3\} \subseteq \{1,3,4,6,12\}$.

In this way we have completely captured the weak partial order $\preceq$ by the subset relation on the corresponding sets. Formally, we have

Lemma 6.2.3. *Let $\preceq$ be a weak partial order on a set, A. Then $\preceq$ is isomorphic to the subset relation on the collection of inverse images of elements $a \in A$ under the $\preceq$ relation.*

We leave the proof to Problem 6.3. Essentially the same construction shows that strict partial orders can be represented by set under the proper subset relation, $\subset$.

6.2.1 Problems

Class Problems

Problem 6.1.

Direct Prerequisites	Subject
18.01	6.042
18.01	18.02
18.01	18.03
8.01	8.02
8.01	6.01
6.042	6.046
18.02, 18.03, 8.02, 6.01	6.02
6.01, 6.042	6.006
6.01	6.034
6.02	6.004

(a) For the above table of MIT subject prerequisites, draw a diagram showing the subject numbers with a line going down to every subject from each of its (direct) prerequisites.

(b) Give an example of a collection of sets partially ordered by the proper subset relation, $\subset$, that is isomorphic to ("same shape as") the prerequisite relation among MIT subjects from part (a).

(c) Explain why the empty relation is a strict partial order and describe a collection of sets partially ordered by the proper subset relation that is isomorphic to the empty relation on five elements —that is, the relation under which none of the five elements is related to anything.

(d) Describe a *simple* collection of sets partially ordered by the proper subset relation that is isomorphic to the "properly contains" relation, $\supset$, on $\mathcal{P}\{1, 2, 3, 4\}$.

Problem 6.2.
Consider the proper subset partial order, $\subset$, on the power set $\mathcal{P}\{1, 2, \dots, 6\}$.

(a) What is the size of a maximal chain in this partial order? Describe one.

(b) Describe the largest antichain you can find in this partial order.

(c) What are the maximal and minimal elements? Are they maximum and minimum?

(d) Answer the previous part for the $\subset$ partial order on the set $\mathcal{P}\{1, 2, \dots, 6\} - \emptyset$.

Homework Problems

Problem 6.3.
This problem asks for a proof of Lemma 6.2.3 showing that every weak partial order can be represented by (is isomorphic to) a collection of sets partially ordered under set inclusion ($\subseteq$). Namely,

Lemma. *Let $\preceq$ be a weak partial order on a set, A. For any element $a \in A$, let*

$$\begin{aligned} L(a) &::= \{b \in A \mid b \preceq a\}, \\ \mathcal{L} &::= \{L(a) \mid a \in A\}. \end{aligned}$$

Then the function $L : A \to \mathcal{L}$ is an isomorphism from the $\preceq$ relation on A, to the subset relation on $\mathcal{L}$.

(a) Prove that the function $\mathrm{L} : A \to \mathcal{L}$ is a bijection.

(b) Complete the proof by showing that

$$a \preceq b \quad \text{iff} \quad \mathrm{L}(a) \subseteq \mathrm{L}(b) \tag{6.1}$$

for all $a, b \in A$.

6.3 Total Orders

The familiar order relations on numbers have an important additional property: given two different numbers, one will be bigger than the other. Partial orders with this property are said to be *total*[2] *orders*.

Definition 6.3.1. Let R be a binary relation on a set, A, and let a, b be elements of A. Then a and b are *comparable* with respect to R iff $[a \ R \ b \ \text{ OR } \ b \ R \ a]$. A partial order for which every two different elements are comparable is called a *total order*.

So $<$ and $\leq$ are total orders on $\mathbb{R}$. On the other hand, the subset relation is *not* total, since, for example, any two different finite sets of the same size will be incomparable under $\subseteq$. The prerequisite relation on Course 6 required subjects is also not total because, for example, neither 8.01 nor 6.001 is a prerequisite of the other.

6.3.1 Problems

Practice Problems

Problem 6.4.
For each of the binary relations below, state whether it is a strict partial order, a weak partial order, or neither. If it is not a partial order, indicate which of the axioms for partial order it violates. If it is a partial order, state whether it is a total order and identify its maximal and minimal elements, if any.

(a) The superset relation, $\supseteq$ on the power set $\mathcal{P}\{1, 2, 3, 4, 5\}$.

(b) The relation between any two nonnegative integers, a, b that the remainder of a divided by 8 equals the remainder of b divided by 8.

(c) The relation between propositional formulas, G, H, that G IMPLIES H is valid.

(d) The relation 'beats' on Rock, Paper and Scissor (for those who don't know the game Rock, Paper, Scissors, Rock beats Scissors, Scissors beats Paper and Paper beats Rock).

(e) The empty relation on the set of real numbers.

(f) The identity relation on the set of integers.

(g) The divisibility relation on the integers, $\mathbb{Z}$.

Class Problems

Problem 6.5. (a) Verify that the divisibility relation on the set of nonnegative integers is a weak partial order.

(b) What about the divisibility relation on the set of integers?

[2]"Total" is an overloaded term when talking about partial orders: being a total order is a much stronger condition than being a partial order that is a total relation. For example, any weak partial order such as $\subseteq$ is a total relation.

Problem 6.6.
Consider the nonnegative numbers partially ordered by divisibility.

(a) Show that this partial order has a unique minimal element.

(b) Show that this partial order has a unique maximal element.

(c) Prove that this partial order has an infinite chain.

(d) An *antichain* in a partial order is a set of elements such that any two elements in the set are incomparable. Prove that this partial order has an infinite antichain. *Hint:* The primes.

(e) What are the minimal elements of divisibility on the integers greater than 1? What are the maximal elements?

Problem 6.7.
How many binary relations are there on the set $\{0, 1\}$?

How many are there that are transitive?, ... asymmetric?, ... reflexive?, ... irreflexive?, ... strict partial orders?, ... weak partial orders?

Hint: There are easier ways to find these numbers than listing all the relations and checking which properties each one has.

Problem 6.8.
A binary relation, R, on a set, A, is *irreflexive* iff $\text{NOT}(a\ R\ a)$ for all $a \in A$. Prove that if a binary relation on a set is transitive and irreflexive, then it is strict partial order.

Problem 6.9.
Prove that if R is a partial order, then so is R^{-1}

Homework Problems

Problem 6.10.
Let R and S be binary relations on the same set, A.

Definition 6.3.2. The *composition*, $S \circ R$, of R and S is the binary relation on A defined by the rule:[3]

$$a\ (S \circ R)\ c \quad \text{iff} \quad \exists b\, [a\ R\ b \text{ AND } b\ S\ c].$$

[3]Note the reversal in the order of R and S. This is so that relational composition generalizes function composition, Composing the functions f and g means that f is applied first, and then g is applied to the result. That is, the value of the composition of f and g applied to an argument, x, is $g(f(x))$. To reflect this, the notation $g \circ f$ is commonly used for the composition of f and g. Some texts do define $g \circ f$ the other way around.

Suppose both R and S are transitive. Which of the following new relations must also be transitive? For each part, justify your answer with a brief argument if the new relation is transitive and a counterexample if it is not.

(a) R^{-1}

(b) $R \cap S$

(c) $R \circ R$

(d) $R \circ S$

Exam Problems

Problem 6.11.

(a) For each row in the following table, indicate whether the binary relation, R, on the set, A, is a weak partial order or a total order by filling in the appropriate entries with either Y = YES or N = NO. In addition, list the minimal and maximal elements for each relation.

A	**a R b**	**weak partial order**	**total order**	**minimal(s)**	**maximal(s)**
$\mathbb{R} - \mathbb{R}^+$	$a \mid b$				
$\mathcal{P}(\{1,2,3\})$	$a \subseteq b$				
$\mathbb{N} \cup \{i\}$	$a > b$				

(b) What is the longest *chain* on the subset relation, $\subseteq$, on $P(\{1,2,3\})$? (If there is more than one, provide ONE of them.)

(c) What is the longest *antichain* on the subset relation, $\subseteq$, on $P(\{1,2,3\})$? (If there is more than one, provide *one* of them.)

6.4 Product Orders

Taking the product of two relations is a useful way to construct new relations from old ones.

Definition 6.4.1. The product, $R_1 \times R_2$, of relations R_1 and R_2 is defined to be the relation with

$$\begin{aligned} \text{domain}(R_1 \times R_2) &::= \text{domain}(R_1) \times \text{domain}(R_2), \\ \text{codomain}(R_1 \times R_2) &::= \text{codomain}(R_1) \times \text{codomain}(R_2), \\ (a_1, a_2)(R_1 \times R_2)(b_1, b_2) &\quad \text{iff} \quad [a_1 \, R_1 \, b_1 \text{ and } a_2 \, R_2 \, b_2]. \end{aligned}$$

Example 6.4.2. Define a relation, Y, on age-height pairs of being younger *and* shorter. This is the relation on the set of pairs (y, h) where y is a nonnegative integer ≤ 2400 which we interpret as an age in months, and h is a nonnegative integer ≤ 120 describing height in inches. We define Y by the rule

$$(y_1, h_1) \, Y \, (y_2, h_2) \quad \text{iff} \quad y_1 \leq y_2 \text{ AND } h_1 \leq h_2.$$

That is, Y is the product of the $\leq$-relation on ages and the $\leq$-relation on heights.

It follows directly from the definitions that products preserve the properties of transitivity, reflexivity, irreflexivity, and antisymmetry, as shown in Problem 6.12. That is, if R_1 and R_2 both have one of these properties, then so does $R_1 \times R_2$. This implies that if R_1 and R_2 are both partial orders, then so is $R_1 \times R_2$.

On the other hand, the property of being a total order is not preserved. For example, the age-height relation Y is the product of two total orders, but it is not total: the age 240 months, height 68 inches pair, (240,68), and the pair (228,72) are incomparable under Y.

6.4.1 Problems

Class Problems

Problem 6.12.
Let R_1, R_2 be binary relations on the same set, A. A relational property is preserved under product, if $R_1 \times R_2$ has the property whenever both R_1 and R_2 have the property.

(a) Verify that each of the following properties are preserved under product.

1. reflexivity,
2. antisymmetry,
3. transitivity.

(b) Verify that if *either* of R_1 or R_2 is irreflexive, then so is $R_1 \times R_2$.

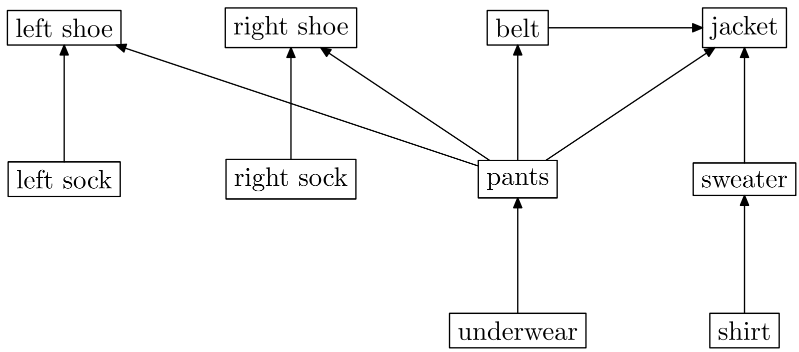
Note that it now follows immediately that if if R_1 and R_2 are partial orders and at least one of them is strict, then $R_1 \times R_2$ is a strict partial order.

6.5 Scheduling

6.5.1 Scheduling with Constraints

Scheduling problems are a common source of partial orders: there is a set, A, of tasks and a set of constraints specifying that starting a certain task depends on other tasks being completed beforehand. We can picture the constraints by drawing labelled boxes corresponding to different tasks, with an arrow from one box to another if the first box corresponds to a task that must be completed before starting the second one.

Example 6.5.1. Here is a drawing describing the order in which you could put on clothes. The tasks are the clothes to be put on, and the arrows indicate what should be put on directly before what.



When we have a partial order of tasks to be performed, it can be useful to have an order in which to perform all the tasks, one at a time, while respecting the dependency constraints. This amounts to finding a total order that is consistent with the partial order. This task of finding a total ordering that is consistent with a partial order is known as *topological sorting*.

Definition 6.5.2. A *topological sort* of a partial order, $\prec$, on a set, A, is a total ordering, $\sqsubset$, on A such that

$$a \prec b \text{ IMPLIES } a \sqsubset b.$$

For example,

$$\begin{aligned} \text{shirt} \sqsubset \text{sweater} \sqsubset \text{underwear} \sqsubset \text{leftsock} \sqsubset \text{rightsock} \sqsubset \text{pants} \\ \sqsubset \text{leftshoe} \sqsubset \text{rightshoe} \sqsubset \text{belt} \sqsubset \text{jacket}, \end{aligned}$$

is one topological sort of the partial order of dressing tasks given by Example 6.5.1; there are several other possible sorts as well.

Topological sorts for partial orders on finite sets are easy to construct by starting from *minimal* elements:

Definition 6.5.3. Let $\preceq$ be a partial order on a set, A. An element $a_0 \in A$ is *minimum* iff it is $\preceq$ every other element of A, that is, $a_0 \preceq b$ for all $b \neq a_0$.

The element a_0 is *minimal* iff no other element is $\preceq a_0$, that is, $\text{NOT}(b \preceq a_0)$ for all $b \neq a_0$.

There are corresponding definitions for *maximum* and *maximal*. Alternatively, a maximum(al) element for a relation, R, could be defined to be as a minimum(al) element for R^{-1}.

In a total order, minimum and minimal elements are the same thing. But a partial order may have no minimum element but lots of minimal elements. There are four minimal elements in the clothes example: leftsock, rightsock, underwear, and shirt.

To construct a total ordering for getting dressed, we pick one of these minimal elements, say shirt. Next we pick a minimal element among the remaining ones. For example, once we have removed shirt, sweater becomes minimal. We continue in this way removing successive minimal elements until all elements have been picked. The sequence of elements in the order they were picked will be a topological sort. This is how the topological sort above for getting dressed was constructed.

So our construction shows:

Theorem 6.5.4. *Every partial order on a finite set has a topological sort.*

There are many other ways of constructing topological sorts. For example, instead of starting "from the bottom" with minimal elements, we could build a total starting *anywhere* and simply keep putting additional elements into the total order wherever they will fit. In fact, the domain of the partial order need not even be finite: we won't prove it, but *all* partial orders, even infinite ones, have topological sorts.

6.5.2 Parallel Task Scheduling

For a partial order of task dependencies, topological sorting provides a way to execute tasks one after another while respecting the dependencies. But what if we have the ability to execute more than one task at the same time? For example, say tasks are programs, the partial order indicates data dependence, and we have a parallel machine with lots of processors instead of a sequential machine with only one. How should we schedule the tasks? Our goal should be to minimize the total *time* to complete all the tasks. For simplicity, let's say all the tasks take one unit of time, and all the processors are identical.

So, given a finite partially ordered set of tasks, how long does it take to do them all, in an optimal parallel schedule? We can also use partial order concepts to analyze this problem.

In the clothes example, we could do all the minimal elements first (leftsock, rightsock, underwear, shirt), remove them and repeat. We'd need lots of hands, or maybe dressing servants. We can do pants and sweater next, and then leftshoe, rightshoe, and belt, and finally jacket.

In general, a *schedule* for performing tasks specifies which tasks to do at successive steps. Every task, a, has be scheduled at some step, and all the tasks that have to be completed before task a must be scheduled for an earlier step.

Definition 6.5.5. A *parallel schedule* for a strict partial order, $\prec$, on a set, A, is a partition[4] of A into sets $A_0, A_1, \ldots$, such that for all $a, b \in A$, $k \in \mathbb{N}$,

$$[a \in A_k \text{ AND } b \prec a] \quad \text{IMPLIES} \quad b \in A_j \text{ for some } j < k.$$

The set A_k is called the set of elements *scheduled at step k*, and the *length* of the schedule is the number of sets A_k in the partition. The maximum number of elements scheduled at any step is called the *number of processors* required by the schedule.

So the schedule we chose above for clothes has four steps

$$\begin{aligned} A_0 &= \{\text{leftsock, rightsock, underwear, shirt}\}, \\ A_1 &= \{\text{pants, sweater}\}, \\ A_2 &= \{\text{leftshoe, rightshoe, belt}\}, \\ A_3 &= \{\text{jacket}\}. \end{aligned}$$

and requires four processors (to complete the first step).

Because you have to put on your underwear before your pants, your pants before your belt, and your belt before your jacket, at least four steps are needed in *every* schedule for getting dressed —if we used fewer than four steps, two of these tasks would have to be scheduled at the same time. A set of tasks that must be done in sequence like this is called a *chain*.

Definition 6.5.6. A *chain* in a partial order is a set of elements such that any two different elements in the set are comparable. A chain is said to *end at* an its maximum element.

In general, the earliest step at which a task, a, can ever be scheduled must be at least as large as any chain that ends at a. A *largest* chain ending at a is called a *critical path* to a, and the size of the critical path is called the *depth* of a. So in any possible parallel schedule, it takes at least depth (a) steps to complete task a.

There is a very simple schedule that completes every task in this minimum number of steps. Just use a "greedy" strategy of performing tasks as soon as possible. Namely, schedule all the elements of depth k at step k. That's how we found the schedule for getting dressed given above.

Theorem 6.5.7. *Let $\prec$ be a strict partial order on a set, A. A minimum length schedule for $\prec$ consists of the sets $A_0, A_1, \ldots$, where*

$$A_k ::= \{a \mid \text{depth}(a) = k\}.$$

[4] Partitioning a set, A, means "cutting it up" into non-overlapping, nonempty pieces. The pieces are called the blocks of the partition. More precisely, a *partition* of A is a set $\mathcal{B}$ whose elements are nonempty subsets of A such that

- if $B, B' \in \mathcal{B}$ are different sets, then $B \cap B' = \emptyset$, and
- $\bigcup_{B \in \mathcal{B}} B = A$.

We'll leave to Problem 6.19 the proof that the sets A_k are a parallel schedule according to Definition 6.5.5.

The minimum number of steps needed to schedule a partial order, $\prec$, is called the *parallel time* required by $\prec$, and a largest possible chain in $\prec$ is called a *critical path* for $\prec$. So we can summarize the story above by this way: with an unlimited number of processors, the minimum parallel time to complete all tasks is simply the size of a critical path:

Corollary 6.5.8. *Parallel time = length of critical path.*

6.6 Dilworth's Lemma

Definition 6.6.1. An *antichain* in a partial order is a set of elements such that any two elements in the set are incomparable.

For example, it's easy to verify that each set A_k is an antichain (see Problem 6.19). So our conclusions about scheduling also tell us something about antichains.

Corollary 6.6.2. *If the largest chain in a partial order on a set, A, is of size t, then A can be partitioned into t antichains.*

Proof. Let the antichains be the sets $A_2, A_2, \ldots, A_t$. ■

Corollary 6.6.2 implies a famous result[5] about partially ordered sets:

Lemma 6.6.3 (Dilworth)**.** *For all $t > 0$, every partially ordered set with n elements must have either a chain of size greater than t or an antichain of size at least n/t.*

Proof. Suppose the largest chain is of size $\leq t$. Then by Corollary 6.6.2, the n elements can be partitioned into at most t antichains. Let ℓ be the size of the largest antichain. Since every element belongs to exactly one antichain, and there are at most t antichains, there can't be more than ℓt elements, namely, $\ell t \geq n$. So there is an antichain with at least $\ell \geq n/t$ elements. ■

Corollary 6.6.4. *Every partially ordered set with n elements has a chain of size greater than $\sqrt{n}$ or an antichain of size at least $\sqrt{n}$.*

Proof. Set $t = \sqrt{n}$ in Lemma 6.6.3. ■

Example 6.6.5. In the dressing partially ordered set, $n = 10$.

Try $t = 3$. There is a chain of size 4.

Try $t = 4$. There is no chain of size 5, but there is an antichain of size $4 \geq 10/4$.

[5] Lemma 6.6.3 also follows from a more general result known as Dilworth's Theorem which we will not discuss.

Example 6.6.6. Suppose we have a class of 101 students. Then using the product partial order, Y, from Example 6.4.2, we can apply Dilworth's Lemma to conclude that there is a chain of 11 students who get taller as they get older, or an antichain of 11 students who get taller as they get younger, which makes for an amusing in-class demo.

6.6.1 Problems

Practice Problems

Problem 6.13.
What is the size of the longest chain that is guaranteed to exist in any partially ordered set of n elements? What about the largest antichain?

Problem 6.14.
Describe a sequence consisting of the integers from 1 to 10,000 in some order so that there is no increasing or decreasing subsequence of size 101.

Problem 6.15.
What is the smallest number of partially ordered tasks for which there can be more than one minimum time schedule? Explain.

Class Problems

Problem 6.16.
The table below lists some prerequisite information for some subjects in the MIT Computer Science program (in 2006). This defines an indirect prerequisite relation, $\prec$, that is a strict partial order on these subjects.

$$\begin{array}{rr}
18.01 \rightarrow 6.042 & 18.01 \rightarrow 18.02 \\
18.01 \rightarrow 18.03 & 6.046 \rightarrow 6.840 \\
8.01 \rightarrow 8.02 & 6.001 \rightarrow 6.034 \\
6.042 \rightarrow 6.046 & 18.03, 8.02 \rightarrow 6.002 \\
6.001, 6.002 \rightarrow 6.003 & 6.001, 6.002 \rightarrow 6.004 \\
6.004 \rightarrow 6.033 & 6.033 \rightarrow 6.857
\end{array}$$

(a) Explain why exactly six terms are required to finish all these subjects, if you can take as many subjects as you want per term. Using a *greedy* subject selection strategy, you should take as many subjects as possible each term. Exhibit your complete class schedule each term using a greedy strategy.

(b) In the second term of the greedy schedule, you took five subjects including 18.03. Identify a set of five subjects not including 18.03 such that it would be possible to take them in any one term (using some nongreedy schedule). Can you figure out how many such sets there are?

(c) Exhibit a schedule for taking all the courses —but only one per term.

(d) Suppose that you want to take all of the subjects, but can handle only two per term. Exactly how many terms are required to graduate? Explain why.

(e) What if you could take three subjects per term?

Problem 6.17.
A pair of 6.042 TAs, Liz and Oscar, have decided to devote some of their spare time this term to establishing dominion over the entire galaxy. Recognizing this as an ambitious project, they worked out the following table of tasks on the back of Oscar's copy of the lecture notes.

1. **Devise a logo** and cool imperial theme music - 8 days.

2. **Build a fleet** of Hyperwarp Stardestroyers out of eating paraphernalia swiped from Lobdell - 18 days.

3. **Seize control** of the United Nations - 9 days, after task #1.

4. **Get shots** for Liz's cat, Tailspin - 11 days, after task #1.

5. **Open a Starbucks chain** for the army to get their caffeine - 10 days, after task #3.

6. **Train an army** of elite interstellar warriors by dragging people to see *The Phantom Menace* dozens of times - 4 days, after tasks #3, #4, and #5.

7. **Launch the fleet** of Stardestroyers, crush all sentient alien species, and establish a Galactic Empire - 6 days, after tasks #2 and #6.

8. **Defeat Microsoft** - 8 days, after tasks #2 and #6.

We picture this information in Figure 6.1 below by drawing a point for each task, and labelling it with the name and weight of the task. An edge between two points indicates that the task for the higher point must be completed before beginning the task for the lower one.

(a) Give some valid order in which the tasks might be completed.

Liz and Oscar want to complete all these tasks in the shortest possible time. However, they have agreed on some constraining work rules.

- Only one person can be assigned to a particular task; they can not work together on a single task.

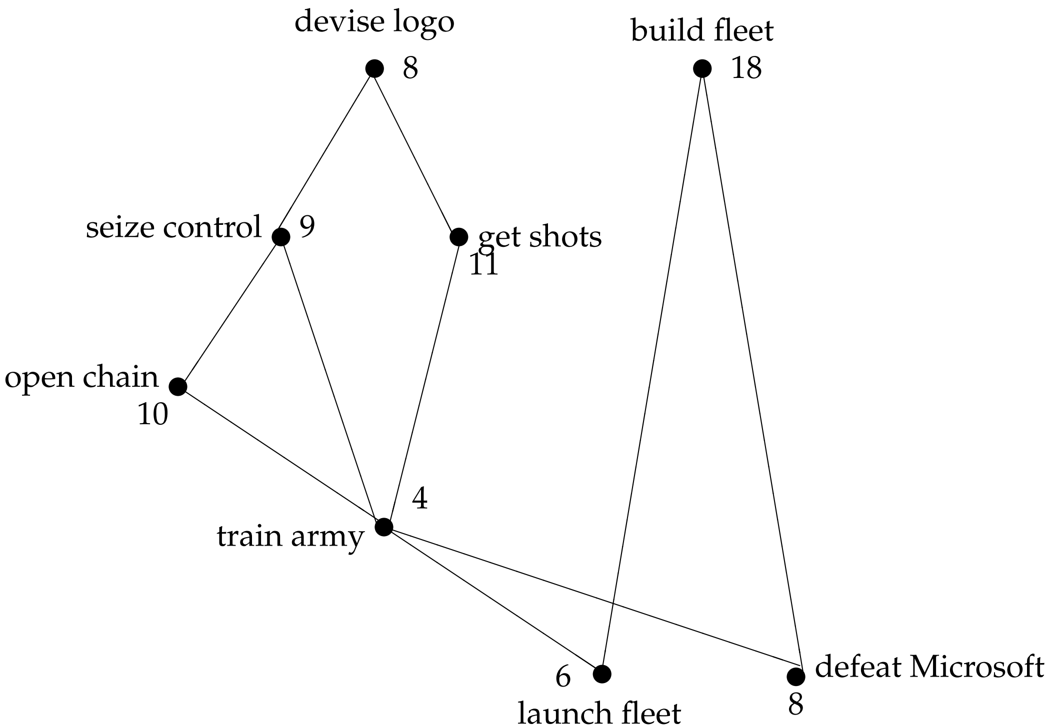


Figure 6.1: Graph representing the task precedence constraints.

- Once a person is assigned to a task, that person must work exclusively on the assignment until it is completed. So, for example, Liz cannot work on building a fleet for a few days, run to get shots for Tailspin, and then return to building the fleet.

(b) Liz and Oscar want to know how long conquering the galaxy will take. Oscar suggests dividing the total number of days of work by the number of workers, which is two. What lower bound on the time to conquer the galaxy does this give, and why might the actual time required be greater?

(c) Liz proposes a different method for determining the duration of their project. He suggests looking at the duration of the "critical path", the most time-consuming sequence of tasks such that each depends on the one before. What lower bound does this give, and why might it also be too low?

(d) What is the minimum number of days that Liz and Oscar need to conquer the galaxy? No proof is required.

Problem 6.18. **(a)** What are the maxim*al* and minim*al* elements, if any, of the power set $\mathcal{P}(\{1, \ldots, n\})$, where n is a positive integer, under the *empty relation*?

(b) What are the maxim*al* and minim*al* elements, if any, of the set, $\mathbb{N}$, of all non-negative integers under divisibility? Is there a minim*um* or maxim*um* element?

(c) What are the minimal and maximal elements, if any, of the set of integers greater than 1 under divisibility?

(d) Describe a partially ordered set that has no minimal or maximal elements.

(e) Describe a partially ordered set that has a *unique minimal* element, but no minimum element. *Hint:* It will have to be infinite.

Homework Problems

Problem 6.19.
Let $\prec$ be a partial order on a set, A, and let

$$A_k ::= \{a \mid \text{depth}\,(a) = k\}$$

where $k \in \mathbb{N}$.

(a) Prove that $A_0, A_1, \ldots$ is a parallel schedule for $\prec$ according to Definition 6.5.5.

(b) Prove that A_k is an antichain.

Problem 6.20.
Let S be a sequence of n different numbers. A *subsequence* of S is a sequence that can be obtained by deleting elements of S.

For example, if

$$S = (6, 4, 7, 9, 1, 2, 5, 3, 8)$$

Then 647 and 7253 are both subsequences of S (for readability, we have dropped the parentheses and commas in sequences, so 647 abbreviates $(6, 4, 7)$, for example).

An *increasing subsequence* of S is a subsequence of whose successive elements get larger. For example, 1238 is an increasing subsequence of S. Decreasing subsequences are defined similarly; 641 is a decreasing subsequence of S.

(a) List all the maximum length increasing subsequences of S, and all the maximum length decreasing subsequences.

Now let A be the *set* of numbers in S. (So $A = \{1, 2, 3, \ldots, 9\}$ for the example above.) There are two straightforward ways to totally order A. The first is to order its elements numerically, that is, to order A with the $<$ relation. The second is to order the elements by which comes first in S; call this order $<_S$. So for the example above, we would have

$$6 <_S 4 <_S 7 <_S 9 <_S 1 <_S 2 <_S 5 <_S 3 <_S 8$$

Next, define the partial order $\prec$ on A defined by the rule

$$a \prec a' \quad ::= \quad a < a' \text{ and } a <_S a'.$$

(It's not hard to prove that $\prec$ is strict partial order, but you may assume it.)

(b) Draw a diagram of the partial order, $\prec$, on A. What are the maximal elements,. . . the minimal elements?

(c) Explain the connection between increasing and decreasing subsequences of S, and chains and anti-chains under $\prec$.

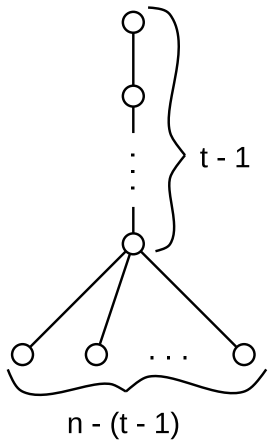
(d) Prove that every sequence, S, of length n has an increasing subsequence of length greater than $\sqrt{n}$ or a decreasing subsequence of length at least $\sqrt{n}$.

(e) (Optional, tricky) Devise an efficient procedure for finding the longest increasing and the longest decreasing subsequence in any given sequence of integers. (There is a nice one.)

Problem 6.21.
We want to schedule n partially ordered tasks.

(a) Explain why any schedule that requires only p processors must take time at least $\lceil n/p \rceil$.

(b) Let $D_{n,t}$ be the strict partial order with n elements that consists of a chain of $t - 1$ elements, with the bottom element in the chain being a prerequisite of all the remaining elements as in the following figure:



What is the minimum time schedule for $D_{n,t}$? Explain why it is unique. How many processors does it require?

(c) Write a simple formula, $M(n,t,p)$, for the minimum time of a p-processor schedule to complete $D_{n,t}$.

(d) Show that *every* partial order with n vertices and maximum chain size, t, has a p-processor schedule that runs in time $M(n,t,p)$.

Hint: Induction on t.

Chapter 7

Directed graphs

7.1 Digraphs

A *directed graph* (*digraph* for short) is formally the same as a binary relation, R, on a set, A —that is, a relation whose domain and codomain are the same set, A. But we describe digraphs as though they were diagrams, with elements of A pictured as points on the plane and arrows drawn between related points. The elements of A are referred to as the *vertices* of the digraph, and the pairs $(a, b) \in \text{graph}(R)$ are *directed edges*. Writing $a \to b$ is a more suggestive alternative for the pair (a, b). Directed edges are also called *arrows*.

For example, the divisibility relation on $\{1, 2, \ldots, 12\}$ is could be pictured by the digraph:

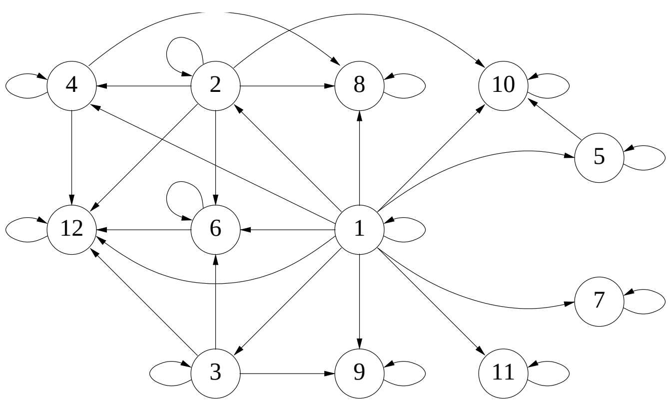


Figure 7.1: The Digraph for Divisibility on $\{1, 2, \ldots, 12\}$.

7.1.1 Paths in Digraphs

Picturing digraphs with points and arrows makes it natural to talk about following a *path* of successive edges through the graph. For example, in the digraph of Figure 7.1, a path might start at vertex 1, successively follow the edges from vertex 1 to vertex 2, from 2 to 4, from 4 to 12, and then from 12 to 12 twice (or as many times as you like). We can represent the path with the sequence of sucessive vertices it went through, in this case:

$$1, 2, 4, 12, 12, 12.$$

So a path is just a sequence of vertices, with consecutive vertices on the path connected by directed edges. Here is a formal definition:

Definition 7.1.1. A *path in a digraph* is a sequence of vertices $a_0, \ldots, a_k$ with $k \geq 0$ such that $a_i \to a_{i+1}$ is an edge of the digraph for $i = 0, 1, \ldots, k-1$. The path is said to *start* at a_0, to *end* at a_k, and the *length* of the path is defined to be k. The path is *simple* iff all the a_i's are different, that is, if $i \neq j$, then $a_i \neq a_j$.

Note that a single vertex counts as length zero path that begins and ends at itself.

It's pretty natural to talk about the edges in a path, but technically, paths only have points, not edges. So to instead, we'll say a path *traverses* an edge $a \to b$ when a and b are consecutive vertices in the path.

For any digraph, R, we can define some new relations on vertices based on paths, namely, the *path relation*, R^*, and the *positive-length path relation*, R^+:

$$a \ R^* \ b ::= \text{there is a path in } R \text{ from } a \text{ to } b,$$

$$a \ R^+ \ b ::= \text{there is a positive length path in } R \text{ from } a \text{ to } b.$$

By the definition of path, both R^* and R^+ are transitive. Since edges count as length one paths, the edges of R^+ include all the edges of R. The edges of R^* in turn include all the edges of R^+ and, in addition include an edge (self-loop) from each vertex to itself. The self-loops get included in R^* because of the a length zero paths in R. So R^* is reflexive. [1]

7.2 Picturing Relational Properties

Many of the relational properties we've discussed have natural descriptions in terms of paths. For example:

Reflexivity: All vertices have self-loops (a *self-loop* at a vertex is an arrow going from the vertex back to itself).

Irreflexivity: No vertices have self-loops.

Antisymmetry: At most one (directed) edge between different vertices.

[1] In many texts, R^+ is called the *transitive closure* and R^* is called the *reflexive transitive closure* of R.

Asymmetry: No self-loops and at most one (directed) edge between different vertices.

Transitivity: Short-circuits—for any path through the graph, there is an arrow from the first vertex to the last vertex on the path.

Symmetry: A binary relation R is *symmetric* iff aRb implies bRa for all a, b in the domain of R. That is, if there is an edge from a to b, there is also one in the reverse direction.

7.3 Composition of Relations

There is a simple way to extend composition of functions to composition of relations, and this gives another way to talk about paths in digraphs.

Let $R : B \to C$ and $S : A \to B$ be relations. Then the composition of R with S is the binary relation $(R \circ S) : A \to C$ defined by the rule

$$a\ (R \circ S)\ c ::= \ \exists b \in B. (b\ R\ c) \text{ AND } (a\ S\ b).$$

This agrees with the Definition 5.3.1 of composition in the special case when R and S are functions.

Now when R is a digraph, it makes sense to compose R with itself. Then if we let R^n denote the composition of R with itself n times, it's easy to check that R^n is the length-n path relation:

$$a\ R^n\ b \quad \text{iff} \quad \text{there is a length } n \text{ path in } R \text{ from } a \text{ to } b.$$

This even works for $n = 0$, if we adopt the convention that R^0 is the identity relation Id_A on the set, A, of vertices. That is, $(a\ \text{Id}_A\ b)$ iff $a = b$.

7.4 Directed Acyclic Graphs

Definition 7.4.1. A *cycle* in a digraph is defined by a path that begins and ends at the same vertex. This includes the cycle of length zero that begins and ends at the vertex. A *directed acyclic graph (DAG)* is a directed graph with no *positive* length cycles.

A *simple cycle* in a digraph is a cycle whose vertices are distinct except for the beginning and end vertices.

DAG's can be an economical way to represent partial orders. For example, in Section 6.1 the *direct prerequisite* relation between MIT subjects was used to determine the partial order of indirect prerequisites on subjects. This indirect prerequisite partial order is precisely the positive length path relation of the direct prerequisites.

Lemma 7.4.2. *If D is a DAG, then D^+ is a strict partial order.*

Proof. We know that D^+ is transitive. Also, a positive length path from a vertex to itself would be a cycle, so there are no such paths. This means D^+ is irreflexive, which implies it is a strict partial order (see problem 6.8). ∎

It's easy to check that conversely, the graph of any strict partial order is a DAG.

The divisibility partial order can also be more economically represented by the path relation in a DAG. A DAG whose *path* relation is divisibility on $\{1, 2, \ldots, 12\}$ is shown in Figure 7.2; the arrowheads are omitted in the Figure, and edges are understood to point upwards.

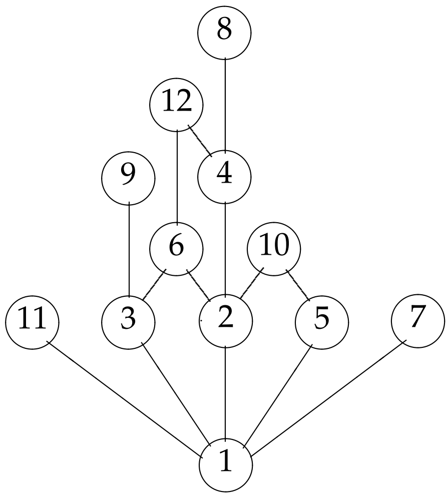


Figure 7.2: A DAG whose Path Relation is Divisibility on $\{1, 2, \ldots, 12\}$.

If we're using a DAG to represent a partial order —so all we care about is the the path relation of the DAG —we could replace the DAG with any other DAG with the same path relation. This raises the question of finding a DAG with the same path relation but the *smallest* number of edges. This DAG turns out to be unique and easy to find (see Problem 7.2).

7.4.1 Problems

Practice Problems

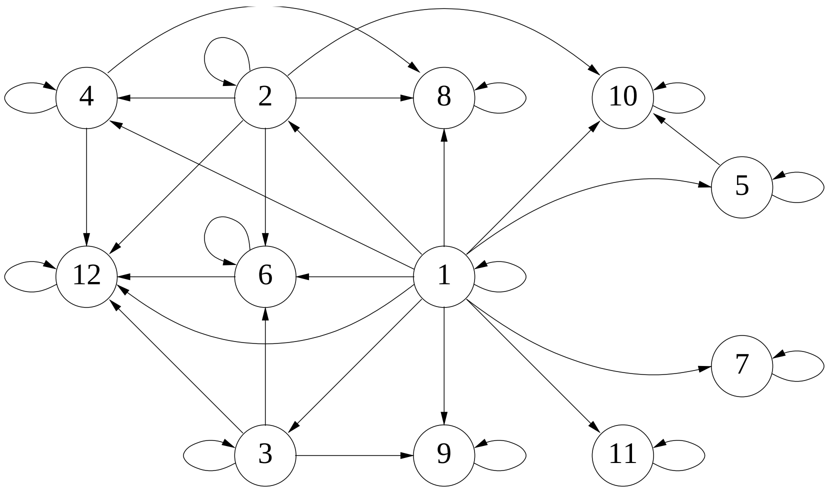
Problem 7.1.
Why is every strict partial order a DAG?

Class Problems

Problem 7.2.
If a and b are distinct nodes of a digraph, then a is said to *cover* b if there is an edge

from a to b and every path from a to b traverses this edge. If a covers b, the edge from a to b is called a *covering edge*.

(a) What are the covering edges in the following DAG?



(b) Let covering (D) be the subgraph of D consisting of only the covering edges. Suppose D is a finite DAG. Explain why covering (D) has the same positive path relation as D.

Hint: Consider *longest* paths between a pair of vertices.

(c) Show that if two *DAG*'s have the same positive path relation, then they have the same set of covering edges.

(d) Conclude that covering (D) is the *unique* DAG with the smallest number of edges among all digraphs with the same positive path relation as D.

The following examples show that the above results don't work in general for digraphs with cycles.

(e) Describe two graphs with vertices $\{1, 2\}$ which have the same set of covering edges, but not the same positive path relation (*Hint:* Self-loops.)

(f) (i) The *complete digraph* without self-loops on vertices $1, 2, 3$ has edges between every two distinct vertices. What are its covering edges?

(ii) What are the covering edges of the graph with vertices $1, 2, 3$ and edges $1 \to 2, 2 \to 3, 3 \to 1$?

(iii) What about their positive path relations?

Homework Problems

Problem 7.3.

Let R be a binary relation on a set A. Then R^n denotes the composition of R with itself n times. Let G_R be the digraph associated with R. That is, A is the set of vertices ofG_R and R is the set of directed edges. Let $R^{(n)}$ denote the length n path relation G_R, that is,

$$a\ R^{(n)}\ b ::= \text{there is a length } n \text{ path from } a \text{ to } b \text{ in } G_R.$$

Prove that

$$R^n = R^{(n)} \tag{7.1}$$

for all $n \in \mathbb{N}$.

Problem 7.4. **(a)** Prove that if R is a relation on a finite set, A, then

$$a\ (R \cup I_A)^n\ b \quad \text{iff} \quad \text{there is a path in } R \text{ of length length} \leq n \text{ from } a \text{ to } b.$$

(b) Conclude that if A is a finite set, then

$$R^* = (R \cup I_A)^{|A|-1}. \tag{7.2}$$

Chapter 8

State Machines

State machines are an abstract model of step-by-step processes, and accordingly, they come up in many areas of computer science. You may already have seen them in a digital logic course, a compiler course, or a probability course.

8.1 Basic definitions

A state machine is really nothing more than a binary relation on a set, except that the elements of the set are called "states," the relation is called the *transition relation*, and a pair (p, q) in the graph of the transition relation is called a *transition*. The transition from state p to state q will be written $p \longrightarrow q$. The transition relation is also called the *state graph* of the machine. A state machine also comes equipped with a designated *start state*.

State machines used in digital logic and compilers usually have only a finite number of states, but machines that model continuing computations typically have an infinite number of states. In many applications, the states, and/or the transitions have labels indicating input or output values, costs, capacities, or probabilities, but for our purposes, unlabelled states and transitions are all we need.[1]

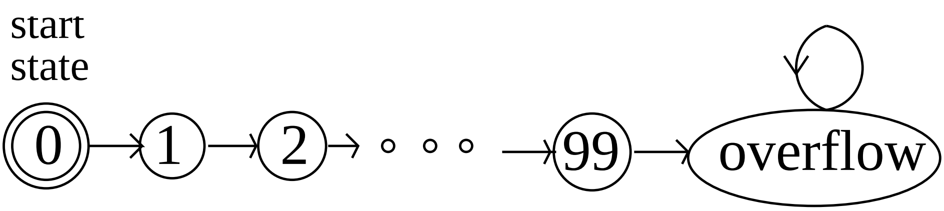


Figure 8.1: *State transitions for the 99-bounded counter.*

[1]We do name states, as in Figure 8.1, so we can talk about them, but the names aren't part of the state machine.

Example 8.1.1. A bounded counter, which counts from 0 to 99 and overflows at 100. The transitions are pictured in Figure 8.1, with start state zero. This machine isn't much use once it overflows, since it has no way to get out of its overflow state.

Example 8.1.2. An unbounded counter is similar, but has an infinite state set. This is harder to draw :-) .

Example 8.1.3. In the movie *Die Hard 3: With a Vengeance*, the characters played by Samuel L. Jackson and Bruce Willis have to disarm a bomb planted by the diabolical Simon Gruber:

Simon: On the fountain, there should be 2 jugs, do you see them? A 5-gallon and a 3-gallon. Fill one of the jugs with exactly 4 gallons of water and place it on the scale and the timer will stop. You must be precise; one ounce more or less will result in detonation. If you're still alive in 5 minutes, we'll speak.

Bruce: Wait, wait a second. I don't get it. Do you get it?

Samuel: No.

Bruce: Get the jugs. Obviously, we can't fill the 3-gallon jug with 4 gallons of water.

Samuel: Obviously.

Bruce: All right. I know, here we go. We fill the 3-gallon jug exactly to the top, right?

Samuel: Uh-huh.

Bruce: Okay, now we pour this 3 gallons into the 5-gallon jug, giving us exactly 3 gallons in the 5-gallon jug, right?

Samuel: Right, then what?

Bruce: All right. We take the 3-gallon jug and fill it a third of the way...

Samuel: No! He said, "Be precise." Exactly 4 gallons.

Bruce: Sh - -. Every cop within 50 miles is running his a - - off and I'm out here playing kids games in the park.

Samuel: Hey, you want to focus on the problem at hand?

Fortunately, they find a solution in the nick of time. We'll let the reader work out how.

The *Die Hard* series is getting tired, so we propose a final *Die Hard Once and For All*. Here Simon's brother returns to avenge him, and he poses the same challenge, but with the 5 gallon jug replaced by a 9 gallon one.

We can model jug-filling scenarios with a state machine. In the scenario with a 3 and a 5 gallon water jug, the states will be pairs, (b, l) of real numbers such that

$0 \leq b \leq 5, 0 \leq l \leq 3$. We let b and l be arbitrary real numbers. (We can prove that the values of b and l will only be nonnegative integers, but we won't assume this.) The start state is $(0, 0)$, since both jugs start empty.

Since the amount of water in the jug must be known exactly, we will only consider moves in which a jug gets completely filled or completely emptied. There are several kinds of transitions:

1. Fill the little jug: $(b, l) \longrightarrow (b, 3)$ for $l < 3$.
2. Fill the big jug: $(b, l) \longrightarrow (5, l)$ for $b < 5$.
3. Empty the little jug: $(b, l) \longrightarrow (b, 0)$ for $l > 0$.
4. Empty the big jug: $(b, l) \longrightarrow (0, l)$ for $b > 0$.
5. Pour from the little jug into the big jug: for $l > 0$,

$$(b, l) \longrightarrow \begin{cases} (b + l, 0) & \text{if } b + l \leq 5, \\ (5, l - (5 - b)) & \text{otherwise.} \end{cases}$$

6. Pour from big jug into little jug: for $b > 0$,

$$(b, l) \longrightarrow \begin{cases} (0, b + l) & \text{if } b + l \leq 3, \\ (b - (3 - l), 3) & \text{otherwise.} \end{cases}$$

Note that in contrast to the 99-counter state machine, there is more than one possible transition out of states in the Die Hard machine. Machines like the 99-counter with at most one transition out of each state are called *deterministic*. The Die Hard machine is *nondeterministic* because some states have transitions to several different states.

Quick exercise: Which states of the Die Hard 3 machine have direct transitions to exactly two states?

8.2 Reachability and Preserved Invariants

The Die Hard 3 machine models every possible way of pouring water among the jugs according to the rules. Die Hard properties that we want to verify can now be expressed and proved using the state machine model. For example, Bruce's character will disarm the bomb if he can get to some state of the form $(4, l)$.

A (possibly infinite) path through the state graph beginning at the start state corresponds to a possible system behavior; such a path is called an *execution* of the state machine. A state is called *reachable* if it appears in some execution. The bomb in Die Hard 3 gets disarmed successfully because the state (4,3) is reachable.

A useful approach in analyzing state machine is to identify properties of states that are preserved by transitions.

Definition 8.2.1. A *preserved invariant* of a state machine is a predicate, P, on states, such that whenever $P(q)$ is true of a state, q, and $q \longrightarrow r$ for some state, r, then $P(r)$ holds.

The Invariant Principle

If a preserved invariant of a state machine is true for the start state, then it is true for all reachable states.

The Invariant Principle is nothing more than the Induction Principle reformulated in a convenient form for state machines. Showing that a predicate is true in the start state is the base case of the induction, and showing that a predicate is a preserved invariant is the inductive step.[2]

8.2.1 Die Hard Once and For All

Now back to Die Hard Once and For All. This time there is a 9 gallon jug instead of the 5 gallon jug. We can model this with a state machine whose states and transitions are specified the same way as for the Die Hard 3 machine, with all occurrences of "5" replaced by "9."

Now reaching any state of the form $(4, l)$ is impossible. We prove this using the Invariant Principle. Namely, we define the preserved invariant predicate, $P(b, l)$, to be that b and l are nonnegative integer multiples of 3. So P obviously holds for the state state $(0, 0)$.

To prove that P is a preserved invariant, we assume $P(b, l)$ holds for some state (b, l) and show that if $(b, l) \longrightarrow (b', l')$, then $P(b', l')$. The proof divides into cases, according to which transition rule is used. For example, suppose the transition followed from the "fill the little jug" rule. This means $(b, l) \longrightarrow (b, 3)$. But $P(b, l)$ implies that b is an integer multiple of 3, and of course 3 is an integer multiple of 3, so P still holds for the new state $(b, 3)$. Another example is when the transition rule used is "pour from big jug into little jug" for the subcase that $b + l > 3$. Then state is $(b, l) \longrightarrow (b - (3 - l), 3)$. But since b and l are integer multiples of 3, so is $b - (3 - l)$. So in this case too, P holds after the transition.

We won't bother to crank out the remaining cases, which can all be checked just as easily. Now by the Invariant Principle, we conclude that every reachable

[2]Preserved invariants are commonly just called "invariants" in the literature on program correctness, but we decided to throw in the extra adjective to avoid confusion with other definitions. For example, another subject at MIT uses "invariant" to mean "predicate true of all reachable states." Let's call this definition "invariant-2." Now invariant-2 seems like a reasonable definition, since unreachable states by definition don't matter, and all we want to show is that a desired property is invariant-2. But this confuses the *objective* of demonstrating that a property is invariant-2 with the *method* for showing that it is. After all, if we already knew that a property was invariant-2, we'd have no need for an Invariant Principle to demonstrate it.

state satisifies P. But since no state of the form $(4, l)$ satisifies P, we have proved rigorously that Bruce dies once and for all!

By the way, notice that the state (1,0), which satisfies $\text{NOT}(P)$, has a transition to (0,0), which satisfies P. So it's wrong to assume that the complement of a preserved invariant is also a preserved invariant.

8.2.2 A Robot on a Grid

There is a robot. It walks around on a grid, and at every step it moves diagonally in a way that changes its position by one unit up or down *and* one unit left or right. The robot starts at position $(0, 0)$. Can the robot reach position $(1, 0)$?

To get some intuition, we can simulate some robot moves. For example, starting at (0,0) the robot could move northeast to (1,1), then southeast to (2,0), then southwest to (1,-1), then southwest again to (0,-2).

Let's model the problem as a state machine and then find a suitable invariant. A state will be a pair of integers corresponding to the coordinates of the robot's position. State (i, j) has transitions to four different states: $(i \pm 1, j \pm 1)$.

The problem is now to choose an appropriate preserved invariant, P, that is true for the start state $(0, 0)$ and false for $(1, 0)$. The Invariant Theorem then will imply that the robot can never reach $(1, 0)$. A direct attempt for a preserved invariant is the predicate $P(q)$ that $q \neq (1, 0)$.

Unfortunately, this is not going to work. Consider the state $(2, 1)$. Clearly $P(2, 1)$ holds because $(2, 1) \neq (1, 0)$. And of course $P(1, 0)$ does not hold. But $(2, 1) \longrightarrow (1, 0)$, so this choice of P will not yield a preserved invariant.

We need a stronger predicate. Looking at our example execution you might be able to guess a proper one, namely, that the sum of the coordinates is even! If we can prove that this is a preserved invariant, then we have proven that the robot never reaches $(1, 0)$ —because the sum $1 + 0$ of its coordinates is odd, while the sum $0 + 0$ of the coordinates of the start state is even.

Theorem 8.2.2. *The sum of the robot's coordinates is always even.*

Proof. The proof uses the Invariant Principle.

Let $P(i, j)$ be the predicate that $i + j$ is even.

First, we must show that the predicate holds for the start state $(0, 0)$. Clearly, $P(0, 0)$ is true because $0 + 0$ is even.

Next, we must show that P is a preserved invariant. That is, we must show that for each transition $(i, j) \longrightarrow (i', j')$, if $i + j$ is even, then $i' + j'$ is even. But $i' = i \pm 1$ and $j' = j \pm 1$ by definition of the transitions. Therefore, $i' + j'$ is equal to $i + j$ or $i + j \pm 2$, all of which are even. ■

Corollary 8.2.3. *The robot cannot reach* $(1, 0)$.

Robert W. Floyd

The Invariant Principle was formulated by Robert Floyd at Carnegie Tech[a] in 1967. Floyd was already famous for work on formal grammars which transformed the field of programming language parsing; that was how he got to be a professor even though he never got a Ph.D. (He was admitted to a PhD program as a teenage prodigy, but flunked out and never went back.)

In that same year, Albert R. Meyer was appointed Assistant Professor in the Carnegie Tech Computer Science Department where he first met Floyd. Floyd and Meyer were the only theoreticians in the department, and they were both delighted to talk about their shared interests. After just a few conversations, Floyd's new junior colleague decided that Floyd was the smartest person he had ever met.

Naturally, one of the first things Floyd wanted to tell Meyer about was his new, as yet unpublished, Invariant Principle. Floyd explained the result to Meyer, and Meyer wondered (privately) how someone as brilliant as Floyd could be excited by such a trivial observation. Floyd had to show Meyer a bunch of examples before Meyer understood Floyd's excitement —not at the truth of the utterly obvious Invariant Principle, but rather at the insight that such a simple theorem could be so widely and easily applied in verifying programs.

Floyd left for Stanford the following year. He won the Turing award —the "Nobel prize" of computer science— in the late 1970's, in recognition both of his work on grammars and on the foundations of program verification. He remained at Stanford from 1968 until his death in September, 2001. You can learn more about Floyd's life and work by reading the eulogy written by his closest colleague, Don Knuth.

[a] The following year, Carnegie Tech was renamed Carnegie-Mellon Univ.

8.3 Sequential algorithm examples

8.3.1 Proving Correctness

Robert Floyd, who pioneered modern approaches to program verification, distinguished two aspects of state machine or process correctness:

1. The property that the final results, if any, of the process satisfy system requirements. This is called *partial correctness*.

   You might suppose that if a result was only partially correct, then it might also be partially incorrect, but that's not what he meant. The word "partial" comes from viewing a process that might not terminate as computing a *partial function*. So partial correctness means that when there is a result, it is correct, but the process might not always produce a result, perhaps because it gets stuck in a loop.

2. The property that the process always finishes, or is guaranteed to produce some legitimate final output. This is called *termination*.

Partial correctness can commonly be proved using the Invariant Principle. Termination can commonly be proved using the Well Ordering Principle. We'll illustrate Floyd's ideas by verifying the Euclidean Greatest Common Divisor (GCD) Algorithm.

8.3.2 The Euclidean Algorithm

The *Euclidean algorithm* is a three-thousand-year-old procedure to compute the greatest common divisor, $\gcd(a, b)$ of integers a and b. We can represent this algorithm as a state machine. A state will be a pair of integers (x, y) which we can think of as integer registers in a register program. The state transitions are defined by the rule

$$(x, y) \longrightarrow (y, \text{remainder}(x, y))$$

for $y \neq 0$. The algorithm terminates when no further transition is possible, namely when $y = 0$. The final answer is in x.

We want to prove:

1. Starting from the state with $x = a$ and $y = b > 0$, if we ever finish, then we have the right answer. That is, at termination, $x = \gcd(a, b)$. This is a *partial correctness* claim.

2. We do actually finish. This is a process *termination* claim.

Partial Correctness of GCD First let's prove that if GCD gives an answer, it is a correct answer. Specifically, let $d ::= \gcd(a, b)$. We want to prove that *if* the procedure finishes in a state (x, y), then $x = d$.

Proof. Define the state predicate

$$P(x, y) ::= \quad [\gcd(x, y) = d \text{ and } (x > 0 \text{ or } y > 0)].$$

P holds for the start state (a, b), by definition of d and the requirement that b is positive. Also, the preserved invariance of P follows immediately from

Lemma 8.3.1. *For all $m, n \in \mathbb{N}$ such that $n \neq 0$,*

$$\gcd(m, n) = \gcd(n, \text{remainder}(m, n)). \tag{8.1}$$

Lemma 8.3.1 is easy to prove: let q be the quotient and r be the remainder of m divided by n. Then $m = qn + r$ by definition. So any factor of both r and n will be a factor of m, and similarly any factor of both m and n will be a factor of r. So r, n

and m, n have the same common factors and therefore the same gcd. Now by the Invariant Principle, P holds for all reachable states.

Since the only rule for termination is that $y = 0$, it follows that if (x, y) is a terminal state, then $y = 0$. If this terminal state is reachable, then the preserved invariant holds for (x, y). This implies that $\gcd(x, 0) = d$ and that $x > 0$. We conclude that $x = \gcd(x, 0) = d$. ■

Termination of GCD Now we turn to the second property, that the procedure must terminate. To prove this, notice that y gets strictly smaller after any one transition. That's because the value of y after the transition is the remainder of x divided by y, and this remainder is smaller than y by definition. But the value of y is always a nonnegative integer, so by the Well Ordering Principle, it reaches a minimum value among all its values at reachable states. But there can't be a transition from a state where y has its minimum value, because the transition would decrease y still further. So the reachable state where y has its minimum value is a state at which no further step is possible, that is, at which the procedure terminates.

Note that this argument does not prove that the minimum value of y is zero, only that the minimum value occurs at termination. But we already noted that the only rule for termination is that $y = 0$, so it follows that the minimum value of y must indeed be zero.

We've already observed that a preserved invariant is really just an induction hypothesis. As with induction, finding the right hypothesis is usually the hard part. We repeat:

> **Given the right preserved invariant, it can be easy to verify a program even if you don't understand it.**

We expect that the Extended Euclidean Algorithm presented above illustrates this point.

8.4 Derived Variables

The preceding termination proofs involved finding a nonnegative integer-valued measure to assign to states. We might call this measure the "size" of the state. We then showed that the size of a state decreased with every state transition. By the Well Ordering Principle, the size can't decrease indefinitely, so when a minimum size state is reached, there can't be any transitions possible: the process has terminated.

More generally, the technique of assigning values to states —not necessarily nonnegative integers and not necessarily decreasing under transitions— is often useful in the analysis of algorithms. *Potential functions* play a similar role in physics. In the context of computational processes, such value assignments for states are called *derived variables*.

For example, for the Die Hard machines we could have introduced a derived variable, $f : \text{states} \to \mathbb{R}$, for the amount of water in both buckets, by setting

$f((a,b)) ::= a+b$. Similarly, in the robot problem, the position of the robot along the x-axis would be given by the derived variable x-coord, where $x\text{-coord}((i,j)) ::= i$.

We can formulate our general termination method as follows:

Definition 8.4.1. Let $\prec$ be a strict partial order on a set, A. A derived variable f : states $\to A$ is *strictly decreasing* iff

$$q \longrightarrow q' \text{ implies } f(q') \prec f(q).$$

We confirmed termination of the GCD and Extended GCD procedures by finding derived variables, y and $\texttt{Y}$, respectively, that were nonnegative integer-valued and strictly decreasing. We can summarize this approach to proving termination as follows:

Theorem 8.4.2. *If f is a strictly decreasing $\mathbb{N}$-valued derived variable of a state machine, then the length of any execution starting at state q is at most $f(q)$.*

Of course we could prove Theorem 8.4.2 by induction on the value of $f(q)$, but think about what it says: "If you start counting down at some nonnegative integer $f(q)$, then you can't count down more than $f(q)$ times." Put this way, it's obvious.

8.4.1 Weakly Decreasing Variables

In addition being strictly decreasing, it will be useful to have derived variables with some other, related properties.

Definition 8.4.3. Let $\preceq$ be a weak partial order on a set, A. A derived variable $f : Q \to A$ is *weakly decreasing* iff

$$q \longrightarrow q' \text{ implies } f(q') \preceq f(q).$$

Strictly increasing and *weakly increasing* derived variables are defined similarly.[3]

8.4.2 Problems

Homework Problems

Problem 8.1.

You are given two buckets, A and B, a water hose, a receptacle, and a drain. The buckets and receptacle are initially empty. The buckets are labeled with their respectively capacities, positive integers a and b. The receptacle can be used to store an unlimited amount of water, but has no measurement markings. Excess water can be dumped into the drain. Among the possible moves are:

1. fill a bucket from the hose,

[3] Weakly increasing variables are often also called *nondecreasing*. We will avoid this terminology to prevent confusion between nondecreasing variables and variables with the much weaker property of *not* being a decreasing variable.

2. pour from the receptacle to a bucket until the bucket is full or the receptacle is empty, whichever happens first,
3. empty a bucket to the drain,
4. empty a bucket to the receptacle,
5. pour from A to B until either A is empty or B is full, whichever happens first,
6. pour from B to A until either B is empty or A is full, whichever happens first.

(a) Model this scenario with a state machine. (What are the states? How does a state change in response to a move?)

(b) Prove that we can put $k \in \mathbb{N}$ gallons of water into the receptacle using the above operations if and only if $\gcd(a, b) \mid k$. *Hint:* Use the fact that if a, b are positive integers then there exist integers s, t such that $\gcd(a, b) = sa + tb$ (see Notes **??**).

Problem 8.2.
Here is a *very, very fun* game. We start with two distinct, positive integers written on a blackboard. Call them a and b. You and I now take turns. (I'll let you decide who goes first.) On each player's turn, he or she must write a new positive integer on the board that is the difference of two numbers that are already there. If a player can not play, then he or she loses.

For example, suppose that 12 and 15 are on the board initially. Your first play must be 3, which is $15 - 12$. Then I might play 9, which is $12 - 3$. Then you might play 6, which is $15 - 9$. Then I can not play, so I lose.

(a) Show that every number on the board at the end of the game is a multiple of $\gcd(a, b)$.

(b) Show that every positive multiple of $\gcd(a, b)$ up to $\max(a, b)$ is on the board at the end of the game.

(c) Describe a strategy that lets you win this game every time.

Problem 8.3.
In the late 1960s, the military junta that ousted the government of the small republic of Nerdia completely outlawed built-in multiplication operations, and also forbade division by any number other than 3. Fortunately, a young dissident found a way to help the population multiply any two nonnegative integers without risking persecution by the junta. The procedure he taught people is:

procedure *multiply*(x, y: nonnegative integers)
$r := x$;
$s := y$;
$a := 0$;
while $s \neq 0$ **do**
  if $3 \mid s$ **then**
    $r := r + r + r$;
    $s := s/3$;
  else if $3 \mid (s-1)$ **then**
    $a := a + r$;
    $r := r + r + r$;
    $s := (s-1)/3$;
  else
    $a := a + r + r$;
    $r := r + r + r$;
    $s := (s-2)/3$;
return a;

We can model the algorithm as a state machine whose states are triples of non-negative integers (r, s, a). The initial state is $(x, y, 0)$. The transitions are given by the rule that for $s > 0$:

$$(r, s, a) \rightarrow \begin{cases} (3r, s/3, a) & \text{if } 3 \mid s \\ (3r, (s-1)/3, a+r) & \text{if } 3 \mid (s-1) \\ (3r, (s-2)/3, a+2r) & \text{otherwise.} \end{cases}$$

(a) List the sequence of steps that appears in the execution of the algorithm for inputs $x = 5$ and $y = 10$.

(b) Use the Invariant Method to prove that the algorithm is partially correct—that is, if $s = 0$, then $a = xy$.

(c) Prove that the algorithm terminates after at most $1 + \log_3 y$ executions of the body of the do statement.

Problem 8.4.
A robot named Wall-E wanders around a two-dimensional grid. He starts out at $(0, 0)$ and is allowed to take four different types of step:

1. $(+2, -1)$
2. $(+1, -2)$
3. $(+1, +1)$

4. $(-3, 0)$

Thus, for example, Wall-E might walk as follows. The types of his steps are listed above the arrows.

$$(0,0) \xrightarrow{1} (2,-1) \xrightarrow{3} (3,0) \xrightarrow{2} (4,-2) \xrightarrow{4} (1,-2) \rightarrow \ldots$$

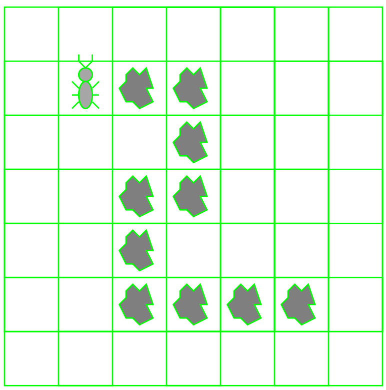
Wall-E's true love, the fashionable and high-powered robot, Eve, awaits at $(0, 2)$.

(a) Describe a state machine model of this problem.

(b) Will Wall-E ever find his true love? Either find a path from Wall-E to Eve or use the Invariant Principle to prove that no such path exists.

Problem 8.5.

A hungry ant is placed on an unbounded grid. Each square of the grid either contains a crumb or is empty. The squares containing crumbs form a path in which, except at the ends, every crumb is adjacent to exactly two other crumbs. The ant is placed at one end of the path and on a square containing a crumb. For example, the figure below shows a situation in which the ant faces North, and there is a trail of food leading approximately Southeast. The ant has already eaten the crumb upon which it was initially placed.



The ant can only smell food directly in front of it. The ant can only remember a small number of things, and what it remembers after any move only depends on what it remembered and smelled immediately before the move. Based on smell and memory, the ant may choose to move forward one square, or it may turn right or left. It eats a crumb when it lands on it.

The above scenario can be nicely modelled as a state machine in which each state is a pair consisting of the "ant's memory" and "everything else" —for example, information about where things are on the grid. Work out the details of such a model state machine; design the ant-memory part of the state machine so the ant will eat all the crumbs on *any* finite path at which it starts and then signal when it is done. Be sure to clearly describe the possible states, transitions, and inputs and outputs (if any) in your model. Briefly explain why your ant will eat all the crumbs.

Note that the last transition is a self-loop; the ant signals done for eternity. One could also add another end state so that the ant signals done only once.

Problem 8.6.
Suppose that you have a regular deck of cards arranged as follows, from top to bottom:

$$A\heartsuit\ 2\heartsuit \ldots K\heartsuit\ A\spadesuit\ 2\spadesuit \ldots K\spadesuit\ A\clubsuit\ 2\clubsuit \ldots K\clubsuit\ A\diamondsuit\ 2\diamondsuit \ldots K\diamondsuit$$

Only two operations on the deck are allowed: *inshuffling* and *outshuffling*. In both, you begin by cutting the deck exactly in half, taking the top half into your right hand and the bottom into your left. Then you shuffle the two halves together so that the cards are perfectly interlaced; that is, the shuffled deck consists of one card from the left, one from the right, one from the left, one from the right, etc. The top card in the shuffled deck comes from the right hand in an outshuffle and from the left hand in an inshuffle.

(a) Model this problem as a state machine.

(b) Use the Invariant Principle to prove that you can not make the entire first half of the deck black through a sequence of inshuffles and outshuffles.

Note: Discovering a suitable invariant can be difficult! The standard approach is to identify a bunch of reachable states and then look for a pattern, some feature that they all share.[4]

Problem 8.7.
The following procedure can be applied to any digraph, G:

1. Delete an edge that is traversed by a directed cycle.
2. Delete edge $u \to v$ if there is a directed path from vertex u to vertex v that does not traverse $u \to v$.
3. Add edge $u \to v$ if there is no directed path in either direction between vertex u and vertex v.

Repeat these operations until none of them are applicable.

This procedure can be modeled as a state machine. The start state is G, and the states are all possible digraphs with the same vertices as G.

[4] If this does not work, consider twitching and drooling until someone takes the problem away.

(a) Let G be the graph with vertices $\{1, 2, 3, 4\}$ and edges

$$\{1 \to 2, 2 \to 3, 3 \to 4, 3 \to 2, 1 \to 4\}$$

What are the possible final states reachable from G?

A *line graph* is a graph whose edges can all be traversed by a directed simple path. All the final graphs in part (a) are line graphs.

(b) Prove that if the procedure terminates with a digraph, H, then H is a line graph with the same vertices as G.

Hint: Show that if H is *not* a line graph, then some operation must be applicable.

(c) Prove that being a DAG is a preserved invariant of the procedure.

(d) Prove that if G is a DAG and the procedure terminates, then the path relation of the final line graph is a topological sort of G.

Hint: Verify that the predicate

$$P(u, v) ::= \text{there is a directed path from } u \text{ to } v$$

is a preserved invariant of the procedure, for any two vertices u, v of a DAG.

(e) Prove that if G is finite, then the procedure terminates.

Hint: Let s be the number of simple cycles, e be the number of edges, and p be the number of pairs of vertices with a directed path (in either direction) between them. Note that $p \leq n^2$ where n is the number of vertices of G. Find coefficients a, b, c such that $as + bp + e + c$ is a strictly decreasing, $\mathbb{N}$-valued variable.

Class Problems

Problem 8.8.

In this problem you will establish a basic property of a puzzle toy called the *Fifteen Puzzle* using the method of invariants. The Fifteen Puzzle consists of sliding square tiles numbered $1, \ldots, 15$ held in a 4×4 frame with one empty square. Any tile adjacent to the empty square can slide into it.

The standard initial position is

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

We would like to reach the target position (known in my youth as "the impossible" — ARM):

15	14	13	12
11	10	9	8
7	6	5	4
3	2	1	

A state machine model of the puzzle has states consisting of a 4×4 matrix with 16 entries consisting of the integers $1, \ldots, 15$ as well as one "empty" entry—like each of the two arrays above.

The state transitions correspond to exchanging the empty square and an adjacent numbered tile. For example, an empty at position $(2, 2)$ can exchange position with tile above it, namely, at position $(1, 2)$:

$$\begin{array}{|c|c|c|c|} \hline n_1 & n_2 & n_3 & n_4 \\ \hline n_5 & & n_6 & n_7 \\ \hline n_8 & n_9 & n_{10} & n_{11} \\ \hline n_{12} & n_{13} & n_{14} & n_{15} \\ \hline \end{array} \longrightarrow \begin{array}{|c|c|c|c|} \hline n_1 & & n_3 & n_4 \\ \hline n_5 & n_2 & n_6 & n_7 \\ \hline n_8 & n_9 & n_{10} & n_{11} \\ \hline n_{12} & n_{13} & n_{14} & n_{15} \\ \hline \end{array}$$

We will use the invariant method to prove that there is no way to reach the target state starting from the initial state.

We begin by noting that a state can also be represented as a pair consisting of two things:

1. a list of the numbers $1, \ldots, 15$ in the order in which they appear—reading rows left-to-right from the top row down, ignoring the empty square, and

2. the coordinates of the empty square—where the upper left square has coordinates $(1, 1)$, the lower right $(4, 4)$.

(a) Write out the "list" representation of the start state and the "impossible" state.

Let L be a list of the numbers $1, \ldots, 15$ in some order. A pair of integers is an *out-of-order pair* in L when the first element of the pair both comes *earlier* in the list and *is larger*, than the second element of the pair. For example, the list $1, 2, 4, 5, 3$ has two out-of-order pairs: (4,3) and (5,3). The increasing list $1, 2 \ldots n$ has no out-of-order pairs.

Let a state, S, be a pair $(L, (i, j))$ described above. We define the *parity* of S to be the mod 2 sum of the number, $p(L)$, of out-of-order pairs in L and the row-number of the empty square, that is the parity of S is $p(L) + i \pmod 2$.

(b) Verify that the parity of the start state and the target state are different.

(c) Show that the parity of a state is preserved under transitions. Conclude that "the impossible" is impossible to reach.

By the way, if two states have the same parity, then in fact there *is* a way to get from one to the other. If you like puzzles, you'll enjoy working this out on your own.

Problem 8.9.
The most straightforward way to compute the bth power of a number, a, is to multiply a by itself b times. This of course requires $b - 1$ multiplications. There is another way to do it using considerably fewer multiplications. This algorithm is called *fast exponentiation*:

Given inputs $a \in \mathbb{R}, b \in \mathbb{N}$, initialize registers x, y, z to $a, 1, b$ respectively, and repeat the following sequence of steps until termination:

- if $z = 0$ **return** y and terminate
- $r :=$ remainder$(z, 2)$
- $z :=$ quotient$(z, 2)$
- if $r = 1$, then $y := xy$
- $x := x^2$

We claim this algorithm always terminates and leaves $y = a^b$.

(a) Model this algorithm with a state machine, carefully defining the states and transitions.

(b) Verify that the predicate $P((x, y, z)) ::= [yx^z = a^b]$ is a preserved invariant.

(c) Prove that the algorithm is partially correct: if it halts, it does so with $y = a^b$.

(d) Prove that the algorithm terminates.

(e) In fact, prove that it requires at most $2 \lceil \log_2(b + 1) \rceil$ multiplications for the Fast Exponentiation algorithm to compute a^b for $b > 1$.

Problem 8.10.
A robot moves on the two-dimensional integer grid. It starts out at $(0, 0)$, and is allowed to move in any of these four ways:

1. (+2,-1) Right 2, down 1
2. (-2,+1) Left 2, up 1
3. (+1,+3)
4. (-1,-3)

Prove that this robot can never reach (1,1).

Problem 8.11.
The Massachusetts Turnpike Authority is concerned about the integrity of the new Zakim bridge. Their consulting architect has warned that the bridge may collapse if more than 1000 cars are on it at the same time. The Authority has also been warned by their traffic consultants that the rate of accidents from cars speeding across bridges has been increasing.

Both to lighten traffic and to discourage speeding, the Authority has decided to make the bridge *one-way* and to put tolls at *both* ends of the bridge (don't laugh, this is Massachusetts). So cars will pay tolls both on entering and exiting the bridge, but the tolls will be different. In particular, a car will pay \$3 to enter onto the bridge and will pay \$2 to exit. To be sure that there are never too many cars on the bridge, the Authority will let a car onto the bridge only if the difference between the amount of money currently at the entry toll booth minus the amount at the exit toll booth is strictly less than a certain threshold amount of $\$T_0$.

The consultants have decided to model this scenario with a state machine whose states are triples of natural numbers, (A, B, C), where

- A is an amount of money at the entry booth,
- B is an amount of money at the exit booth, and
- C is a number of cars on the bridge.

Any state with $C > 1000$ is called a *collapsed* state, which the Authority dearly hopes to avoid. There will be no transition out of a collapsed state.

Since the toll booth collectors may need to start off with some amount of money in order to make change, and there may also be some number of "official" cars already on the bridge when it is opened to the public, the consultants must be ready to analyze the system started at *any* uncollapsed state. So let A_0 be the initial number of dollars at the entrance toll booth, B_0 the initial number of dollars at the exit toll booth, and $C_0 \leq 1000$ the number of official cars on the bridge when it is opened. You should assume that even official cars pay tolls on exiting or entering the bridge after the bridge is opened.

(a) Give a mathematical model of the Authority's system for letting cars on and off the bridge by specifying a transition relation between states of the form (A, B, C) above.

(b) Characterize each of the following derived variables

$$A, B, A+B, A-B, 3C-A, 2A-3B, B+3C, 2A-3B-6C, 2A-2B-3C$$

as one of the following

constant	C
strictly increasing	SI
strictly decreasing	SD
weakly increasing but not constant	WI
weakly decreasing but not constant	WD
none of the above	N

and briefly explain your reasoning.

The Authority has asked their engineering consultants to determine T and to verify that this policy will keep the number of cars from exceeding 1000.

The consultants reason that if C_0 is the number of official cars on the bridge when it is opened, then an additional $1000 - C_0$ cars can be allowed on the bridge. So as long as $A - B$ has not increased by $3(1000 - C_0)$, there shouldn't more than 1000 cars on the bridge. So they recommend defining

$$T_0 ::= 3(1000 - C_0) + (A_0 - B_0), \tag{8.2}$$

where A_0 is the initial number of dollars at the entrance toll booth, B_0 is the initial number of dollars at the exit toll booth.

(c) Use the results of part (b) to define a simple predicate, P, on states of the transition system which is satisfied by the start state, that is $P(A_0, B_0, C_0)$ holds, is not satisfied by any collapsed state, and is a preserved invariant of the system. Explain why your P has these properties.

(d) A clever MIT intern working for the Turnpike Authority agrees that the Turnpike's bridge management policy will be *safe*: the bridge will not collapse. But she warns her boss that the policy will lead to *deadlock*— a situation where traffic can't move on the bridge even though the bridge has not collapsed.

Explain more precisely in terms of system transitions what the intern means, and briefly, but clearly, justify her claim.

Problem 8.12.
Start with 102 coins on a table, 98 showing heads and 4 showing tails. There are two ways to change the coins:

(i) flip over any ten coins, or

(ii) let n be the number of heads showing. Place $n+1$ additional coins, all showing tails, on the table.

For example, you might begin by flipping nine heads and one tail, yielding 90 heads and 12 tails, then add 91 tails, yielding 90 heads and 103 tails.

(a) Model this situation as a state machine, carefully defining the set of states, the start state, and the possible state transitions.

(b) Explain how to reach a state with exactly one tail showing.

(c) Define the following derived variables:

C	::=	the number of coins on the table,	H	::=	the number of heads,
T	::=	the number of tails,	C_2	::=	$\text{remainder}(C/2)$,
H_2	::=	$\text{remainder}(H/2)$,	T_2	::=	$\text{remainder}(T/2)$.

Which of these variables is

1. strictly increasing
2. weakly increasing
3. strictly decreasing
4. weakly decreasing
5. constant

(d) Prove that it is not possible to reach a state in which there is exactly one head showing.

Problem 8.13.
In some terms when 6.042 is not taught in a TEAL room, students sit in a square arrangement during recitations. An outbreak of beaver flu sometimes infects students in recitation; beaver flu is a rare variant of bird flu that lasts forever, with symptoms including a yearning for more quizzes and the thrill of late night problem set sessions.

Here is an example of a 6×6 recitation arrangement with the locations of infected students marked with an asterisk.

$*$				$*$	
	$*$				
		$*$	$*$		
		$*$			
			$*$		$*$

Outbreaks of infection spread rapidly step by step. A student is infected after a step if either

- the student was infected at the previous step (since beaver flu lasts forever), or
- the student was adjacent to *at least two* already-infected students at the previous step.

Here *adjacent* means the students' individual squares share an edge (front, back, left or right, but *not* diagonal). Thus, each student is adjacent to 2, 3 or 4 others.

In the example, the infection spreads as shown below.

$*$				$*$	
	$*$				
		$*$	$*$		
		$*$			
			$*$		$*$

$\Rightarrow$

$*$	$*$			$*$	
$*$	$*$	$*$			
	$*$	$*$	$*$		
		$*$			
		$*$	$*$		
		$*$	$*$	$*$	$*$

$\Rightarrow$

$*$	$*$	$*$		$*$	
$*$	$*$	$*$	$*$		
$*$	$*$	$*$	$*$		
	$*$	$*$	$*$		
		$*$	$*$	$*$	
		$*$	$*$	$*$	$*$

In this example, over the next few time-steps, all the students in class become infected.

Theorem. *If fewer than n students among those in an $n \times n$ arrangment are initially infected in a flu outbreak, then there will be at least one student who never gets infected in this outbreak, even if students attend all the lectures.*

Prove this theorem.

Hint: Think of the state of an outbreak as an $n \times n$ square above, with asterisks indicating infection. The rules for the spread of infection then define the transitions of a state machine. Try to derive a weakly decreasing state variable that leads to a proof of this theorem.

8.5 The Alternating Bit Protocol

The Alternating Bit Protocol is a well-known two-process communication protocol that achieves reliable FIFO communication over unreliable channels. The unreliable channels may lose or duplicate messages, but are assumed not to reorder them. We'll use the Invariant Method to verify that the Protocol

The Protocol allows a **Sender** process to send a sequence of messages from a message alphabet, M, to a **Receiver** process. It works as follows.

Sender repeatedly sends the rightmost message in its **outgoing-queue** of messages, tagged with a **tagbit** that is initially 1. When **Receiver** receives this tagged message, it sets its **ackbit** to be the message tag 1, and adds the message to the lefthand end of its **received-msgs** list. Then as an acknowledgement, **Receiver** sends back **ackbit** 1 repeatedly. When **Sender** gets this acknowledgement bit, it deletes the rightmost outgoing message in its queue, sets its **tagbit** to 0, and begins sending the new rightmost outgoing message, tagged with **tagbit**.

Receiver, having already accepted the message tagged with **ackbit** 1, ignores subsequent messages with tag 1, and waits until it sees the first message with tag 0; it adds this message to the lefthand side of its **received-msgs** list, sets **ackbit** to 0 and acknowledges repeatedly with with **ackbit** 0. **Sender** now waits till it gets acknowledgement bit 0, then goes on to send the next outgoing message with tag 1. In this way, it alternates use of the tags 1 and 0 for successive messages.

We claim that this causes **Sender** to receive *suffix* original **outgoing-msgs** queue. That is, at any stage in the process when the the **outgoing-msgs**

(The fact that **Sender** actually outputs the entire outgoing queuee is a *liveness* claim —liveness properties are a generalization of termination properties. We'll ignore this issue for now.)

We formalize the description above as a state whose states consist of:

outgoing-msgs, a finite sequence of M, whose initial value is called **all-msgs**
tagbit $\in \{0, 1\}$, initially 1

received-msgs, a finite sequence of M, initially empty
ackbit$(\in \{0, 1\}$, initially 0

msg-channel, a finite sequence of $M \times \{0, 1\}$, initially empty,
ack-channel, a finite sequence of $\{0, 1\}$, initially empty

The transitions are:

SEND:

(a) **action**: **send-msg**(m, b)
precondition: $m =$ **rightend**(**outgoing-msgs**) AND $b =$ **tagbit**
effect: add (m, b) to the lefthand end of **msg-channel**, any number ≥ 0 of times

(b) **action**: **send-ack**(b)
precondition: $b =$ **ackbit**
effect: add b to the righthand end of **ack-channel**, any number ≥ 0 of times

RECEIVE: (a) **action**: **receive-msg**(m, b)
precondition: $(m, b) =$ **rightend**(**msg-channel**)
effect: remove **rightend** of **msg-channel**;
if $b \neq$ **ackbit**, then [add m to the lefthand end of **receive-msgs**; **ackbit** $:= b$.]

(b) **action**: **receive-ack**(b)
precondition: $b =$ **leftend**(**ack-channel**())
effect: remove **leftend** of **ack-channel**.
if $b =$ **tagbit**, then [remove **rightend** of **outgoing-msgs** (if nonempty); **tagbit** $:= \overline{\textbf{tagbit}}$]

Our goal is to show that when **tagbit** $\neq$ **ackbit**, then

$$\textbf{outgoing-queue} \cdot \textbf{received-msgs} = \textbf{all-msgs}. \tag{8.3}$$

This requires three auxiliary invariants. For the first of these, we need a definition.

Let **tag-sequence** be the sequence consisting of bits in **ack-channel**, in right-to-left order, followed by **tagbit**, followed by the tag components of the elements of **msg-channel**, in left-to-right order, followed by **ackbit**.

Property 2: tag-sequence consists of one of the following:

1. All 0's.
2. All 1's.
3. A positive number of 0's followed by a positive number of 1's.
4. A positive number of 1's followed by a positive number of 0's.

What is being ruled out by these four cases is the situation where the sequence contains more than one switch of tag value.

The fact that Property 2 is an invariant can be proved easily by induction. We also need:

Property 3: If $(m, \textbf{tag})$ is in **msg-channel** then $m =$ **rightend**(**outgoing-queue**).

Proof. (That Property 3 is an invariant)

By induction, using Property 2.

Base: Obvious, since no message is in the channel initially.

Inductive step: It is easy to see that the property is preserved by $\textbf{send}_{m,b}$, which adds new messages to $\textbf{channel}_{1,2}$. The only other case that could cause a problem is $\textbf{receive}(b)_{2,1}$, which could cause $\textbf{tag}_1$ to change when there is another message already in $\textbf{channel}_{1,2}$ with the same tag. But this can't happen, by Property 2 applied before the step – since the incoming tag g must be equal to $\textbf{tag}_1$ in this case, all the tags in **tag-sequence** must be the same. ∎

Finally, we need that the following counterpart to (8.3): when **tagbit** $=$ **ackbit**, then

$$\mathbf{lefttail}(\mathbf{outgoing\text{-}queue}) \cdot \mathbf{received\text{-}msgs} = \mathbf{all\text{-}msgs}, \tag{8.4}$$

where **lefttail**(**outgoing-queue**) all but the rightmost message, if any, in **outgoing-queue**.

Property 4, part 2, easily implies the goal Property 1. It also implies that $\mathbf{work\text{-}buf}_2$ is always nonempty when $\mathbf{receive}(b)_{2,1}$ occurs with equal tags; therefore, the parenthetical check in the code always works out to be true.

Proof. (That Property 4 is an invariant)

By induction. Base: In an initial state, the tags are unequal, $\mathbf{work\text{-}buf}_1 = \mathbf{buf}_1$ and $\mathbf{buf}_2$ is empty. This suffices to show part 1. part 2 is vacuous.

Inductive step: When a **send** occurs, the tags and buffers are unchanged, so the truth of the invariants must be preserved. It remains to consider **receive** events.

$\mathbf{receive}(m, b)_{1,2}$:

If $b = \mathbf{tag}_2$, nothing happens, so the invariants are preserved. So suppose that $b \neq \mathbf{tag}_2$. Then Property 2 implies that $b = \mathbf{tag}_1$, and then Property 3 implies that m is the first message on $\mathbf{work\text{-}buf}_1$. The effect of the transition is to change $\mathbf{tag}_2$ to make it equal to $\mathbf{tag}_1$, and to replicate the first element of $\mathbf{work\text{-}buf}22_1$ at the end of $\mathbf{buf}_2$.

The inductive hypothesis implies that, before the step, $\mathbf{buf}_2 \cdot \mathbf{work\text{-}buf}_1 = \mathbf{buf}_1$. The changes caused by the step imply that, after the step, $\mathbf{tag}_1 = \mathbf{tag}_2$, $\mathbf{work\text{-}buf}_1$ and $\mathbf{buf}_2$ are nonempty, $\mathbf{head}(\mathbf{work\text{-}buf}_1) = \mathbf{last}(\mathbf{buf}_2)$, and $\mathbf{buf}_2 \cdot \mathbf{tail}(\mathbf{work\text{-}buf}_1) = \mathbf{buf}_1$. This is as needed.

$\mathbf{receive}(b)_{2,1}$:

The argument is similar to the one for $\mathbf{receive}(m, b)_{1,2}$. If $b \neq \mathbf{tag}_1$, nothing happens so the invariants are preserved. So suppose that $b = \mathbf{tag}_1$. Then Property 2 implies that $b = \mathbf{tag}_2$, and the step changes $\mathbf{tag}_1$ to make it unequal to $\mathbf{tag}44_2$. The step also removes the first element of $\mathbf{work\text{-}buf}_1$. The inductive hypothesis implies that, before the step, $\mathbf{work\text{-}buf}_1$ and $\mathbf{buf}_2$ are nonempty, $\mathbf{head}(\mathbf{work\text{-}buf}_1) = \mathbf{last}(\mathbf{buf}_2)$, and $\mathbf{buf}_2 \cdot \mathbf{tail}(\mathbf{work\text{-}buf}_1) = \mathbf{buf}_1$. The changes caused by the step imply that, after the step, $\mathbf{tag}_1 \neq \mathbf{tag}_2$ and $\mathbf{buf}_2 \cdot \mathbf{work\text{-}buf}_1 = \mathbf{buf}_1$. This is as needed. ∎

8.5.1 Applications

Not surprisingly, a stable matching procedure is used by at least one large dating agency. But although "boy-girl-marriage" terminology is traditional and makes some of the definitions easier to remember (we hope without offending anyone), solutions to the Stable Marriage Problem are widely useful.

The Mating Ritual was first announced in a paper by D. Gale and L.S. Shapley in 1962, but ten years before the Gale-Shapley paper was appeared, and unknown by them, the Ritual was being used to assign residents to hospitals by the National

Resident Matching Program (NRMP). The NRMP has, since the turn of the twentieth century, assigned each year's pool of medical school graduates to hospital residencies (formerly called "internships") with hospitals and graduates playing the roles of boys and girls. (In this case there may be multiple boys married to one girl, but there's an easy way to use the Ritual in this situation (see Problem 9.12). Before the Ritual was adopted, there were chronic disruptions and awkward countermeasures taken to preserve assignments of graduates to residencies. The Ritual resolved these problems so successfully, that it was used essentially without change at least through 1989.[5]

MIT Math Prof. Tom Leighton, who regularly teaches 6.042 and also founded the internet infrastructure company, Akamai, reports another application. Akamai uses a variation of the Gale-Shapley procedure to assign web traffic to servers. In the early days, Akamai used other combinatorial optimization algorithms that got to be too slow as the number of servers and traffic increased. Akamai switched to Gale-Shapley since it is fast and can be run in a distributed manner. In this case, the web traffic corresponds to the boys and the web servers to the girls. The servers have preferences based on latency and packet loss; the traffic has preferences based on the cost of bandwidth.

8.5.2 Problems

Practice Problems

Problem 8.14.
Four Students want separate assignments to four VI-A Companies. Here are their preference rankings:

Student	Companies
Albert:	HP, Bellcore, AT&T, Draper
Rich:	AT&T, Bellcore, Draper, HP
Megumi:	HP, Draper, AT&T, Bellcore
Justin:	Draper, AT&T, Bellcore, HP

Company	Students
AT&T:	Justin, Albert, Megumi, Rich
Bellcore:	Megumi, Rich, Albert, Justin
HP:	Justin, Megumi, Albert, Rich
Draper:	Rich, Justin, Megumi, Albert

(a) Use the Mating Ritual to find *two* stable assignments of Students to Companies.

(b) Describe a simple procedure to determine whether any given stable marriage problem has a unique solution, that is, only one possible stable matching.

[5] Much more about the Stable Marriage Problem can be found in the very readable mathematical monograph by Dan Gusfield and Robert W. Irving, The Stable Marriage Problem: Structure and Algorithms, MIT Press, Cambridge, Massachusetts, 1989, 240 pp.

Problem 8.15.
Suppose that Harry is one of the boys and Alice is one of the girls in the *Mating Ritual*. Which of the properties below are preserved invariants? Why?

a. Alice is the only girl on Harry's list.

b. There is a girl who does not have any boys serenading her.

c. If Alice is not on Harry's list, then Alice has a suitor that she prefers to Harry.

d. Alice is crossed off Harry's list and Harry prefers Alice to anyone he is serenading.

e. If Alice is on Harry's list, then she prefers to Harry to any suitor she has.

Class Problems

Problem 8.16.
A preserved invariant of the Mating ritual is:

> For every girl, G, and every boy, B, if G is crossed off B's list, then G has a favorite suitor and she prefers him over B.

Use the invariant to prove that the Mating Algorithm produces stable marriages. (Don't look up the proof in the Notes or slides.)

Problem 8.17.
Consider a stable marriage problem with 4 boys and 4 girls and the following partial information about their preferences:

B1:	G1	G2	–	–
B2:	G2	G1	–	–
B3:	–	–	G4	G3
B4:	–	–	G3	G4
G1:	B2	B1	–	–
G2:	B1	B2	–	–
G3:	–	–	B3	B4
G4:	–	–	B4	B3

(a) Verify that

$$(B1, G1), (B2, G2), (B3, G3), (B4, G4)$$

will be a stable matching whatever the unspecified preferences may be.

(b) Explain why the stable matching above is neither boy-optimal nor boy-pessimal and so will not be an outcome of the Mating Ritual.

(c) Describe how to define a set of marriage preferences among n boys and n girls which have at least $2^{n/2}$ stable assignments.

Hint: Arrange the boys into a list of $n/2$ pairs, and likewise arrange the girls into a list of $n/2$ pairs of girls. Choose preferences so that the kth pair of boys ranks the kth pair of girls just below the previous pairs of girls, and likewise for the kth pair of girls. Within the kth pairs, make sure each boy's first choice girl in the pair prefers the other boy in the pair.

Homework Problems

Problem 8.18.
The most famous application of stable matching was in assigning graduating medical students to hospital residencies. Each hospital has a preference ranking of students and each student has a preference order of hospitals, but unlike the setup in the notes where there are an equal number of boys and girls and monogamous marriages, hospitals generally have differing numbers of available residencies, and the total number of residencies may not equal the number of graduating students. Modify the definition of stable matching so it applies in this situation, and explain how to modify the Mating Ritual so it yields stable assignments of students to residencies. No proof is required.

Problem 8.19.
Give an example of a stable matching between 3 boys and 3 girls where no person gets their first choice. Briefly explain why your matching is stable.

Problem 8.20.
In a stable matching between n boys and girls produced by the Mating Ritual, call a person *lucky* if they are matched up with one of their $\lceil n/2 \rceil$ top choices. We will prove:

Theorem. *There must be at least one lucky person.*

To prove this, define the following derived variables for the Mating Ritual:

$q(B) = j$, where j is the rank of the girl that boy B is courting. That is to say, boy B is always courting the jth girl on his list.

$r(G)$ is the number of boys that girl G has rejected.

(a) Let

$$S ::= \sum_{B \in \text{Boys}} q(B) - \sum_{G \in \text{Girls}} r(G). \tag{8.5}$$

Show that S remains the same from one day to the next in the Mating Ritual.

(b) Prove the Theorem above. (You may assume for simplicity that n is even.)

Hint: A girl is sure to be lucky if she has rejected half the boys.

8.6 Reasoning About While Programs

Real programs and programming languages are often huge and complicated, making them hard to model and even harder to reason about. Still, making programs "reasonable" is a crucial aspect of software engineering. In this section we'll illustrate what it means to have a clean mathematical model of a simple programming language and reasoning principles that go with it —if only real programming languages allowed for such simple, accurate modeling.

8.6.1 While Programs

The programs we'll study are called "***while*** *programs*." We can define them as a recursive data type:

Definition 8.6.1.
base cases:

- $x := e$ is a **while** program, called an *assignment statement*, where x is a variable and e is an expression.
- ***Done*** is a **while** program.

constructor cases: If C and D are **while** programs, and T is a test, then the following are also **while** programs:

- C;D —called the *sequencing* of C and D,
- **if** T **then** C **else** D —called a *conditional* with *test*, T, and *branches*, C and D,
- **while** T **do** C **od** —called a *while loop* with *test*, T, and *body*, C.

For simplicity we'll stick to **while** programs operating on integers. So by expressions we'll mean any of the familiar integer valued expressions involving integer constants and operations such as addition, multiplication, exponentiation, quotient or remainder. As *tests*, we'll allow propositional formulas built from basic formulas of the form $e \leq f$ where e and f are expressions. For example, here is the Euclidean algorithm for $\gcd(a, b)$ expressed as a **while** program.

$\mathtt{x} := a$**;**
$\mathtt{y} := b$**;**
while $\mathtt{y} \neq 0$ **do**
    $\mathtt{t} := \mathtt{y}$**;**
    $\mathtt{y} := \mathrm{rem}(\mathtt{x}, \mathtt{y})$**;**
    $\mathtt{x} := \mathtt{t}$ **od**

8.6.2 The While Program State Machine

A **while** program acts as a pure command: it is run solely for its side effects on stored data and it doesn't return a value. The data consists of integers stored as the values of variables, namely environments:

Definition 8.6.2. An *environment* is a total function from variables to integers. Let Env be set of all environments.

So if ρ is an environment and x is a variable, then $\rho(x)$ is an integer. More generally, the environment determines the integer value of each expression, e, and the truth value of each test, T. We can think of an expression, e as defining a function $[\![e]\!] : \text{Env} \to \mathbb{Z}$, and refer to this function, $[\![e]\!]$ as the *meaning* of e, and likewise for tests.

It's standard in programming language theory to write $[\![e]\!]\rho$ as shorthand for $[\![e]\!](\rho)$, that is, applying the *meaning*, $[\![e]\!]$, of e to ρ. For example, if $\rho(\mathtt{x}) = 4$, and $\rho(\mathtt{y}) = -2$, then

$$[\![\mathtt{x}^2 + \mathtt{y} - \mathtt{3}]\!]\rho = \rho(\mathtt{x})^2 + \rho(y) - 3 = 11. \tag{8.6}$$

Executing a program causes a succession of changes to the environment[6] which may continue until the program halts. Actually the only command which immediately alters the environment is an assignment command. Namely, the effect of the command

$$x := e$$

on an environment is that the value assigned to the variable x is changed to the value of e in the original environment. We can say this precisely and concisely using the following notation: $f[a \leftarrow b]$ is a function that is the same as the function, f, except that when applied to element a its value is b. Namely,

Definition 8.6.3. If $f : A \to B$ is a function and a, b are arbitrary elements, define

$$f[a \leftarrow b]$$

to be the function g such that

$$g(u) = \begin{cases} b & \text{if } u = a. \\ f(u) & \text{otherwise.} \end{cases}$$

Now we can specify the step-by-step execution of a **while** program as a state machine, where the states of the machine consist of a **while** program paired with an environment. The transitions of this state machine are defined recursively on the definition of **while** programs.

Definition 8.6.4. The transitions $\langle C, \rho \rangle \longrightarrow \langle D, \rho' \rangle$ of the **while** program state machine are defined as follows:

[6] More sophisticated programming models distinguish the environment from a *store* which is affected by commands, but this distinction is unnecessary for our purposes.

base cases:

$$\langle x := e,\, \rho\rangle \longrightarrow \langle \textbf{Done},\, \rho[x \leftarrow [\![e]\!]\rho]\rangle$$

constructor cases: If C and D are **while** programs, and T is a test, then:

- if $\langle C,\, \rho\rangle \longrightarrow \langle C',\, \rho'\rangle$, then

  $$\langle C;D,\, \rho\rangle \longrightarrow \langle C';D,\, \rho'\rangle\,.$$

  Also,

  $$\langle \textbf{Done};D,\, \rho\rangle \longrightarrow \langle D,\, \rho\rangle\,.$$

- if $[\![T]\!]\rho = \mathbf{T}$, then

  $$\langle \textbf{if } T \textbf{ then } C \textbf{ else } D,\, \rho\rangle \longrightarrow \langle C,\, \rho\rangle\,,$$

  or if $[\![T]\!]\rho = \mathbf{F}$, then

  $$\langle \textbf{if } T \textbf{ then } C \textbf{ else } D,\, \rho\rangle \longrightarrow \langle D,\, \rho\rangle\,.$$

- if $[\![T]\!]\rho = \mathbf{T}$, then

  $$\langle \textbf{while } T \textbf{ do } C \textbf{ od},\, \rho\rangle \longrightarrow \langle C;\textbf{while } T \textbf{ do } C \textbf{ od},\, \rho\rangle$$

  or if $[\![T]\!]\rho = \mathbf{F}$, then

  $$\langle \textbf{while } T \textbf{ do } C \textbf{ od},\, \rho\rangle \longrightarrow \langle \textbf{Done},\, \rho\rangle\,.$$

Now **while** programs are probably going to be the simplest kind of programs you will ever see, but being condescending about them would be a mistake. It turns that *every function on nonnegative integers that can be computed by any program* on any machine whatsoever can also be computed by **while** programs (maybe more slowly). We can't take the time to explain how such a sweeping claim can be justified, but you can find out by taking a course in computability theory such as 6.045 or 6.840.

8.6.3 Denotational Semantics

The net effect of starting a **while** program in some environment is reflected in the final environment when the program halts. So we can think of a **while** program, C, aas defining a function, $[\![C]\!] : \text{Env} \to \text{Env}$, from initial environments to environments at halting. The function $[\![C]\!]$ is called the *meaning* of C.

$[\![C]\!]$ of a **while** program, C to be a partial function from Env to Env mapping an initial environment to the final halting environment.

We'll need one bit of notation first. For any function $f : S \to S$, let $f^{(n)}$ be the composition of f with itself n times where $n \in \mathbb{N}$. Namely,

$$f^{(0)} ::= \mathrm{Id}_S$$
$$f^{(n+1)} ::= f \circ f^{(n)},$$

where "$\circ$" denotes functional composition.

The recursive definition of the meaning of a program follows the definition of the **while** program recursive data type.

Definition 8.6.5. base cases:

- $[\![x := e]\!]$ is the function from Env to Env defined by the rule:

$$[\![x := e]\!]\rho ::= \rho[x \leftarrow [\![e]\!]\rho].$$

- $$[\![\textbf{Done}]\!] ::= \mathrm{Id}_{\mathrm{Env}}$$

  where $\mathrm{Id}_{\mathrm{Env}}$ is the identity function on Env. In other words, $[\![\textbf{Done}]\!]\rho ::= \rho$.

constructor cases: If C and D are **while** programs, and T is a test, then:

- $$[\![C;D]\!] ::= [\![D]\!] \circ [\![C]\!]$$

  That is,

$$[\![C;D]\!]\rho ::= [\![D]\!]([\![C]\!]\rho).$$

- $$[\![\textbf{if } T \textbf{ then } C \textbf{ else } D]\!]\rho ::= \begin{cases} [\![C]\!]\rho & \text{if } [\![T]\!]\rho = \mathbf{T} \\ [\![D]\!]\rho & \text{if } [\![T]\!]\rho = \mathbf{F}. \end{cases}$$

- $$[\![\textbf{while } T \textbf{ do } C \textbf{ od}]\!]\rho ::= [\![C]\!]^{(n)}\rho$$

  where n is the least nonnegative integer such that $[\![T]\!]([\![C]\!]^{(n)}\rho) = \mathbf{F}$. (If there is no such n, then $[\![\textbf{while } T \textbf{ do } C \textbf{ od}]\!]\rho$ is undefined.)

We can use the denotational semantics of **while** programs to reason about **while** programs using structural induction on programs, and this is often much simpler than reasoning about them using induction on the number of steps in an execution. This is OK as long as the denotational semantics accurately captures the state machine behavior. In particular, using the notation $\longrightarrow^*$ for the transitive closure of the transition relation:

Theorem 8.6.6.

$$\langle C,\, \rho\rangle \longrightarrow^* \langle \textbf{\textit{Done}},\, \rho'\rangle \quad \textit{iff} \quad [\![C]\!]\rho = \rho'$$

Theorem 8.6.6 can be proved easily by induction; it appear in Problem8.21.

8.6.4 Problems

Homework Problems

Problem 8.21.
Prove

Theorem 8.6.7.

$$\langle C,\, \rho \rangle \longrightarrow^* \langle \textbf{Done},\, \rho' \rangle \quad \textit{iff} \quad [\![C]\!]\rho = \rho'$$

Hint: Prove the left to right direction by induction on the number of steps C needs to halt starting in environment ρ. Prove the right to left direction by structural induction on the definition of **while** programs. Both proofs follow almost mechanically from the definitions.

8.6.5 Logic of Programs

A typical program specification describes the kind of inputs and environments the program should handle, and then describes what should result from an execution. The specification of the inputs or initial environment is called the *precondition* for program execution, and the prescription of what the result of execution should be is called the *postcondition*. So if P is a logical formula expressing the precondition for a program, C, and likewise Q expresses the postcondition, the specification requires that

> If P holds when C is started, then Q will hold if and when C halts.

We'll express this requirement as a formula

$$P\ \{C\}\ Q$$

called a *partial correctness assertion*.

For example, if E is **while** program above for the Euclidean algorithm, then the partial correctness of E can be expressed as

$$(a, b \in \mathbb{N} \text{ AND } \mathtt{x} \neq 0)\ \{C\}\ (\mathtt{x} = \gcd(a, b))\,. \tag{8.7}$$

More precisely, notice that just as the value of an expression in an environment is an integer, the value of a logical formula in an environment is a truth value. For example, if $\rho(\mathtt{x}) = 4$, and $\rho(\mathtt{y}) = -2$, then by (8.6), $[\![\mathtt{x}^2 + \mathtt{y} - 3]\!]\rho = 11$, so

$$[\![\exists \mathtt{z}.\, \mathtt{z} > 4 \text{ AND } \mathtt{x}^2 + \mathtt{y} - 3 = \mathtt{z}]\!]\rho = \mathbf{T},$$
$$[\![\exists \mathtt{z}.\, \mathtt{z} > 13 \text{ AND } \mathtt{x}^2 + \mathtt{y} - 3 = \mathtt{z}]\!]\rho = \mathbf{F}.$$

Definition 8.6.8. For logical formulas P and Q, and **while** program, C, the partial correctness assertion

$$P\ \{C\}\ Q$$

is true proving that for all environments, rho, if $[\![P]\!]\rho$ is true, and $\langle C,\, \rho \rangle \longrightarrow^* \langle \textbf{Done},\, \rho' \rangle$ for some ρ', then $[\![Q]\!]\rho'$ is true.

In the 1970′, Univ. Dublin formulated a set of inference rules for proving partial correctness formulas. These rules are known as *Hoare Logic*.

The first rule captures the fact that strengthening the preconditions and weakening the postconditions makes a partial correctness specification easier to satisfy:

$$\frac{P \text{ IMPLIES } R, \quad R\ \{C\}\ S, \quad S \text{ IMPLIES } Q}{P\ \{C\}\ Q}$$

The rest of the logical rules follow the recursive definition of **while** programs. There are axioms for the base case commands:

$$P(x)\ \{x := e\}\ P(e)$$
$$P\ \{\textbf{Done}\}\ P,$$

and proof rules for the constructor cases:

- $$\frac{P\ \{C\}\ Q \text{ AND } Q\ \{D\}\ R}{P\ \{C;D\}\ R}$$
- $$\frac{P \text{ AND } T\ \{C\}\ Q}{P \text{ AND } T\ \{\textbf{if } T \textbf{ then } C \textbf{ else } D\}\ Q \text{ AND } T}$$
- $$\frac{P \text{ AND } T\ \{C\}\ P}{P\ \{\textbf{while } T \textbf{ do } C \textbf{ od}\}\ P \text{ AND NOT}(T)}$$

Example 8.6.9. **Proof of partial correctness** (8.7) **for the Euclidean algorithm.**

TBA - Brief discussion of "relative completeness".

Chapter 9

Simple Graphs

Graphs in which edges are *not* directed are called *simple graphs*. They come up in all sorts of applications, including scheduling, optimization, communications, and the design and analysis of algorithms. Two Stanford students even used graph theory to become multibillionaires!

But we'll start with an application designed to get your attention: we are going to make a professional inquiry into sexual behavior. Namely, we'll look at some data about who, on average, has more opposite-gender partners, men or women.

Sexual demographics have been the subject of many studies. In one of the largest, researchers from the University of Chicago interviewed a random sample of 2500 people over several years to try to get an answer to this question. Their study, published in 1994, and entitled *The Social Organization of Sexuality* found that on average men have 74% more opposite-gender partners than women.

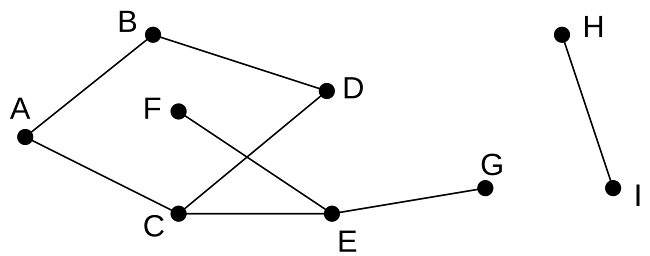
Other studies have found that the disparity is even larger. In particular, ABC News claimed that the average man has 20 partners over his lifetime, and the average woman has 6, for a percentage disparity of 233%. The ABC News study, aired on Primetime Live in 2004, purported to be one of the most scientific ever done, with only a 2.5% margin of error. It was called "American Sex Survey: A peek between the sheets," —which raises some question about the seriousness of their reporting.

Yet again, in August, 2007, the N.Y. Times reported on a study by the National Center for Health Statistics of the U.S. government showing that men had seven partners while women had four. Anyway, whose numbers do you think are more accurate, the University of Chicago, ABC News, or the National Center? —don't answer; this is a setup question like "When did you stop beating your wife?" Using a little graph theory, we'll explain why none of these findings can be anywhere near the truth.

9.1 Degrees & Isomorphism

9.1.1 Definition of Simple Graph

Informally, a graph is a bunch of dots with lines connecting some of them. Here is an example:



For many mathematical purposes, we don't really care how the points and lines are laid out —only which points are connected by lines. The definition of *simple graphs* aims to capture just this connection data.

Definition 9.1.1. A *simple graph*, G, consists of a nonempty set, V, called the *vertices* of G, and a collection, E, of two-element subsets of V. The members of E are called the *edges* of G.

The vertices correspond to the dots in the picture, and the edges correspond to the lines. For example, the connection data given in the diagram above can also be given by listing the vertices and edges according to the official definition of simple graph:

$$V = \{A, B, C, D, E, F, G, H, I\}$$
$$E = \{\{A, B\}, \{A, C\}, \{B, D\}, \{C, D\}, \{C, E\}, \{E, F\}, \{E, G\}, \{H, I\}\}.$$

It will be helpful to use the notation A—B for the edge $\{A, B\}$. Note that A—B and B—A are different descriptions of the same edge, since sets are unordered.

So the definition of simple graphs is the same as for directed graphs, except that instead of a directed edge $v \to w$ which starts at vertex v and ends at vertex w, a simple graph only has an undirected edge, v—w, that connects v and w.

Definition 9.1.2. Two vertices in a simple graph are said to be *adjacent* if they are joined by an edge, and an edge is said to be *incident* to the vertices it joins. The number of edges incident to a vertex is called the *degree* of the vertex; equivalently, the degree of a vertex is equals the number of vertices adjacent to it.

For example, in the simple graph above, A is adjacent to B and B is adjacent to D, and the edge A—C is incident to vertices A and C. Vertex H has degree 1, D has degree 2, and E has degree 3.

Graph Synonyms

A synonym for "vertices" is "*nodes*," and we'll use these words interchangeably. Simple graphs are sometimes called *networks*, edges are sometimes called *arcs*. We mention this as a "heads up" in case you look at other graph theory literature; we won't use these words.

Some technical consequences of Definition 9.1.1 are worth noting right from the start:

1. Simple graphs do *not* have self-loops ($\{a, a\}$ is not an undirected edge because an undirected edge is defined to be a set of *two* vertices.)

2. There is at most one edge between two vertices of a simple graph.

3. Simple graphs have at least one vertex, though they might not have any edges.

There's no harm in relaxing these conditions, and some authors do, but we don't need self-loops, multiple edges between the same two vertices, or graphs with no vertices, and it's simpler not to have them around.

For the rest of this Chapter we'll only be considering simple graphs, so we'll just call them "graphs" from now on.

9.1.2 Sex in America

Let's model the question of heterosexual partners in graph theoretic terms. To do this, we'll let G be the graph whose vertices, V, are all the people in America. Then we split V into two separate subsets: M, which contains all the males, and F, which contains all the females.[1] We'll put an edge between a male and a female iff they have been sexual partners. This graph is pictured in Figure 9.1 with males on the left and females on the right.

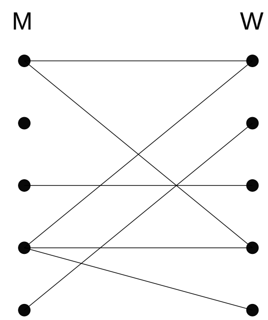


Figure 9.1: The sex partners graph

[1]For simplicity, we'll ignore the possibility of someone being both, or neither, a man and a woman.

Actually, this is a pretty hard graph to figure out, let alone draw. The graph is *enormous*: the US population is about 300 million, so $|V| \approx 300M$. Of these, approximately 50.8% are female and 49.2% are male, so $|M| \approx 147.6M$, and $|F| \approx 152.4M$. And we don't even have trustworthy estimates of how many edges there are, let alone exactly which couples are adjacent. But it turns out that we don't need to know any of this —we just need to figure out the relationship between the average number of partners per male and partners per female. To do this, we note that every edge is incident to exactly one M vertex (remember, we're only considering male-female relationships); so the sum of the degrees of the M vertices equals the number of edges. For the same reason, the sum of the degrees of the F vertices equals the number of edges. So these sums are equal:

$$\sum_{x \in M} \deg(x) = \sum_{y \in F} \deg(y).$$

Now suppose we divide both sides of this equation by the product of the sizes of the two sets, $|M| \cdot |F|$:

$$\left(\frac{\sum_{x \in M} \deg(x)}{|M|} \right) \cdot \frac{1}{|F|} = \left(\frac{\sum_{y \in F} \deg(y)}{|F|} \right) \cdot \frac{1}{|M|}$$

The terms above in parentheses are the *average degree of an M vertex* and the *average degree of a F* vertex. So we know:

$$\text{Avg. deg in } M = \frac{|F|}{|M|} \cdot \text{Avg. deg in } F$$

In other words, we've proved that the average number of female partners of males in the population compared to the average number of males per female is *determined solely by the relative number of males and females in the population*.

Now the Census Bureau reports that there are slightly more females than males in America; in particular $|F| / |M|$ is about 1.035. So we know that on average, males have 3.5% more opposite-gender partners than females, and this tells us nothing about any sex's promiscuity or selectivity. Rather, it just has to do with the relative number of males and females. Collectively, males and females have the same number of opposite gender partners, since it takes one of each set for every partnership, but there are fewer males, so they have a higher ratio. This means that the University of Chicago, ABC, and the Federal government studies are way off. After a huge effort, they gave a totally wrong answer.

There's no definite explanation for why such surveys are consistently wrong. One hypothesis is that males exaggerate their number of partners —or maybe females downplay theirs —but these explanations are speculative. Interestingly, the principal author of the National Center for Health Statistics study reported that she knew the results had to be wrong, but that was the data collected, and her job was to report it.

The same underlying issue has led to serious misinterpretations of other survey data. For example, a couple of years ago, the Boston Globe ran a story on a survey of the study habits of students on Boston area campuses. Their survey showed that on average, minority students tended to study with non-minority students more than the other way around. They went on at great length to explain why this "remarkable phenomenon" might be true. But it's not remarkable at all —using our graph theory formulation, we can see that all it says is that there are fewer minority students than non-minority students, which is, of course what "minority" means.

9.1.3 Handshaking Lemma

The previous argument hinged on the connection between a sum of degrees and the number edges. There is a simple connection between these in any graph:

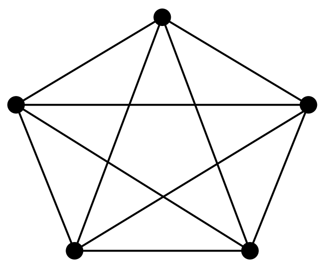
Lemma 9.1.3. *The sum of the degrees of the vertices in a graph equals twice the number of edges.*

Proof. Every edge contributes two to the sum of the degrees, one for each of its endpoints. ■

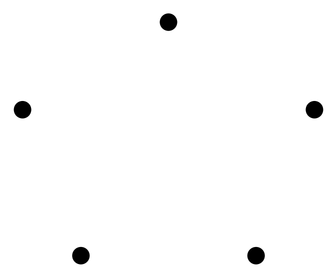
Lemma 9.1.3 is sometimes called the *Handshake Lemma*: if we total up the number of people each person at a party shakes hands with, the total will be twice the number of handshakes that occurred.

9.1.4 Some Common Graphs

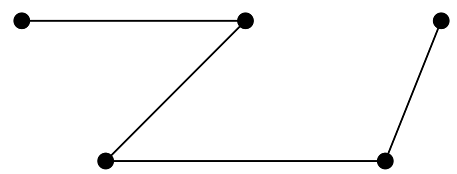
Some graphs come up so frequently that they have names. The *complete graph* on n vertices, also called K_n, has an edge between every two vertices. Here is K_5:



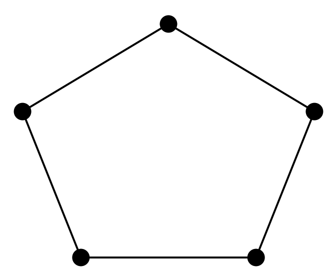
The *empty graph* has no edges at all. Here is the empty graph on 5 vertices:



Another 5 vertex graph is L_4, the *line graph* of length four:

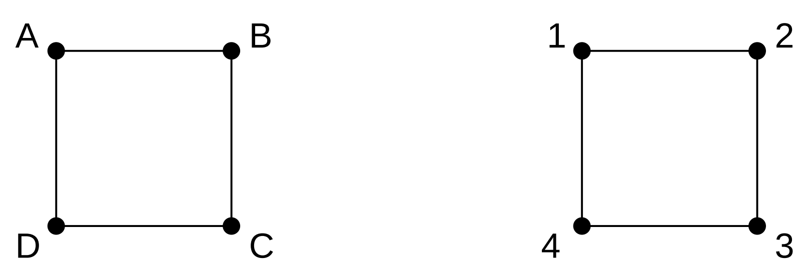


And here is C_5, a *simple cycle* with 5 vertices:



9.1.5 Isomorphism

Two graphs that look the same might actually be different in a formal sense. For example, the two graphs below are both simple cycles with 4 vertices:



But one graph has vertex set $\{A, B, C, D\}$ while the other has vertex set $\{1, 2, 3, 4\}$. If so, then the graphs are different mathematical objects, strictly speaking. But this is a frustrating distinction; the graphs *look the same*!

Fortunately, we can neatly capture the idea of "looks the same" by adapting Definition 6.2.1 of isomorphism of digraphs to handle simple graphs.

Definition 9.1.4. If G_1 is a graph with vertices, V_1, and edges, E_1, and likewise for G_2, then G_1 is *isomorphic* to G_2 iff there exists a **bijection**, $f : V_1 \to V_2$, such that for every pair of vertices $u, v \in V_1$:

$$u\text{—}v \in E_1 \quad \text{iff} \quad f(u)\text{—}f(v) \in E_2.$$

The function f is called an *isomorphism* between G_1 and G_2.

For example, here is an isomorphism between vertices in the two graphs above:

A corresponds to 1	B corresponds to 2
D corresponds to 4	C corresponds to 3.

You can check that there is an edge between two vertices in the graph on the left if and only if there is an edge between the two corresponding vertices in the graph on the right.

Two isomorphic graphs may be drawn very differently. For example, here are two different ways of drawing C_5:

Isomorphism preserves the connection properties of a graph, abstracting out what the vertices are called, what they are made out of, or where they appear in a drawing of the graph. More precisely, a property of a graph is said to be *preserved under isomorphism* if whenever G has that property, every graph isomorphic to G also has that property. For example, since an isomorphism is a bijection between sets of vertices, isomorphic graphs must have the same number of vertices. What's more, if f is a graph isomorphism that maps a vertex, v, of one graph to the vertex, $f(v)$, of an isomorphic graph, then by definition of isomorphism, every vertex adjacent to v in the first graph will be mapped by f to a vertex adjacent to $f(v)$ in the isomorphic graph. That is, v and $f(v)$ will have the same degree. So if one graph has a vertex of degree 4 and another does not, then they can't be isomorphic.

In fact, they can't be isomorphic if the number of degree 4 vertices in each of the graphs is not the same.

Looking for preserved properties can make it easy to determine that two graphs are not isomorphic, or to actually find an isomorphism between them, if there is one. In practice, it's frequently easy to decide whether two graphs are isomorphic. However, no one has yet found a *general* procedure for determining whether two graphs are isomorphic that is *guaranteed* to run much faster than an exhaustive (and exhausting) search through all possible bijections between their vertices.

Having an efficient procedure to detect isomorphic graphs would, for example, make it easy to search for a particular molecule in a database given the molecular bonds. On the other hand, knowing there is no such efficient procedure would also be valuable: secure protocols for encryption and remote authentication can be built on the hypothesis that graph isomorphism is computationally exhausting.

9.1.6 Problems

Class Problems

Problem 9.1. **(a)** Prove that in every graph, there are an even number of vertices of odd degree.

Hint: The Handshaking Lemma 9.1.3.

(b) Conclude that at a party where some people shake hands, the number of people who shake hands an odd number of times is an even number.

(c) Call a sequence of two or more different people at the party a *handshake sequence* if, except for the last person, each person in the sequence has shaken hands with the next person in the sequence.

Suppose George was at the party and has shaken hands with an odd number of people. Explain why, starting with George, there must be a handshake sequence ending with a different person who has shaken an odd number of hands.

Hint: Just look at the people at the ends of handshake sequences that start with George.

Problem 9.2.
For each of the following pairs of graphs, either define an isomorphism between them, or prove that there is none. (We write ab as shorthand for a—b.)

(a)

$$G_1 \text{ with } V_1 = \{1,2,3,4,5,6\}\,,\ E_1 = \{12,23,34,14,15,35,45\}$$
$$G_2 \text{ with } V_2 = \{1,2,3,4,5,6\}\,,\ E_2 = \{12,23,34,45,51,24,25\}$$

(b)

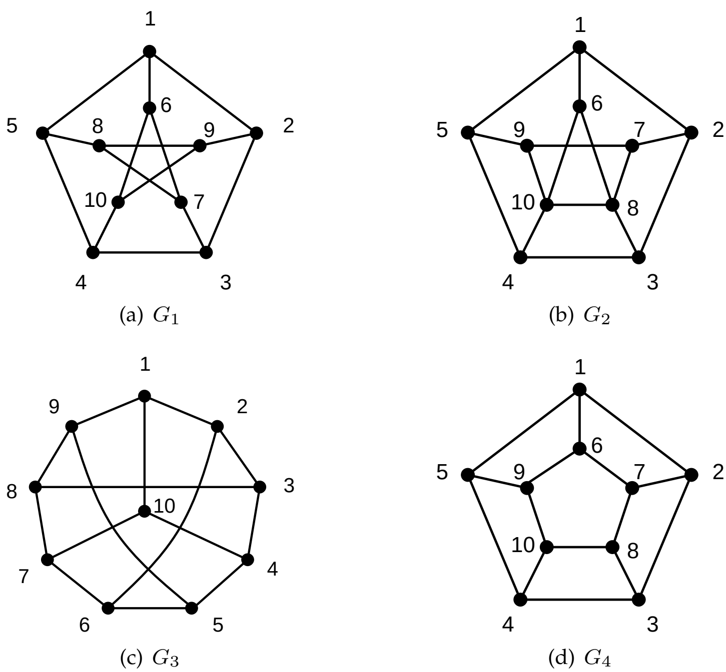
$$G_3 \text{ with } V_3 = \{1,2,3,4,5,6\}\,,\ E_3 = \{12, 23, 34, 14, 45, 56, 26\}$$
$$G_4 \text{ with } V_4 = \{a,b,c,d,e,f\}\,,\ E_4 = \{ab, bc, cd, de, ae, ef, cf\}$$

(c)

$$G_5 \text{ with } V_5 = \{a,b,c,d,e,f,g,h\}\,,\ E_5 = \{ab, bc, cd, ad, ef, fg, gh, he, dh, bf\}$$
$$G_6 \text{ with } V_6 = \{s,t,u,v,w,x,y,z\}\,,\ E_6 = \{st, tu, uv, sv, wx, xy, yz, wz, sw, vz\}$$

Homework Problems

Problem 9.3.
Determine which among the four graphs pictured in the Figures are isomorphic. If two of these graphs are isomorphic, describe an isomorphism between them. If they are not, give a property that is preserved under isomorphism such that one graph has the property, but the other does not. For at least one of the properties you choose, *prove* that it is indeed preserved under isomorphism (you only need prove one of them).



(a) G_1 (b) G_2 (c) G_3 (d) G_4

Figure 9.2: Which graphs are isomorphic?

Problem 9.4. **(a)** For any vertex, v, in a graph, let $N(v)$ be the set of *neighbors* of v, namely, the vertices adjacent to v:

$$N(v) ::= \{u \mid u\text{—}v \text{ is an edge of the graph}\}.$$

Suppose f is an isomorphism from graph G to graph H. Prove that $f(N(v)) = N(f(v))$.

Your proof should follow by simple reasoning using the definitions of isomorphism and neighbors —no pictures or handwaving.

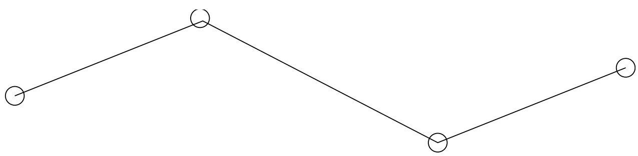
Hint: Prove by a chain of iff's that

$$h \in N(f(v)) \quad \text{iff} \quad h \in f(N(v))$$

for every $h \in V_H$. Use the fact that $h = f(u)$ for some $u \in V_G$.

(b) Conclude that if G and H are isomorphic graphs, then for each $k \in \mathbb{N}$, they have the same number of degree k vertices.

Problem 9.5.
Let's say that a graph has "two ends" if it has exactly two vertices of degree 1 and all its other vertices have degree 2. For example, here is one such graph:



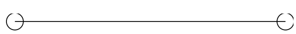
(a) A *line graph* is a graph whose vertices can be listed in a sequence with edges between consecutive vertices only. So the two-ended graph above is also a line graph of length 4.

Prove that the following theorem is false by drawing a counterexample.
False Theorem. *Every two-ended graph is a line graph.*

(b) Point out the first erroneous statement in the following alleged proof of the false theorem. Describe the error as best you can.

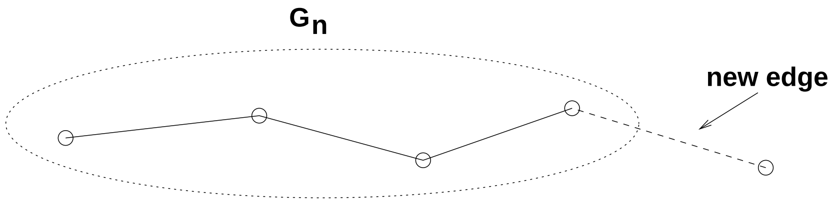
False proof. We use induction. The induction hypothesis is that every two-ended graph with n edges is a path.

Base case ($n = 1$): The only two-ended graph with a single edge consists of two vertices joined by an edge:



Sure enough, this is a line graph.

Inductive case: We assume that the induction hypothesis holds for some $n \geq 1$ and prove that it holds for $n+1$. Let G_n be any two-ended graph with n edges. By the induction assumption, G_n is a line graph. Now suppose that we create a two-ended graph G_{n+1} by adding one more edge to G_n. This can be done in only one way: the new edge must join an endpoint of G_n to a new vertex; otherwise, G_{n+1} would not be two-ended.

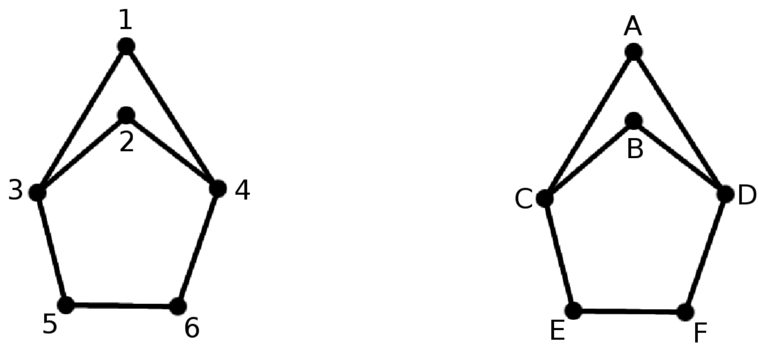


Clearly, G_{n+1} is also a line graph. Therefore, the induction hypothesis holds for all graphs with $n+1$ edges, which completes the proof by induction.

■

Exam Problems

Problem 9.6.
There are four isomorphisms between these two graphs. List them.



Problem 9.7.
A researcher analyzing data on heterosexual sexual behavior in a group of m males and f females found that within the group, the male average number of female partners was 10% larger that the female average number of male partners.

(a) Circle all of the assertions below that are implied by the above information on average numbers of partners:

(i) males exaggerate their number of female partners

(ii) $m = (9/10)f$

(iii) $m = (10/11)f$

(iv) $m = (11/10)f$

(v) there cannot be a perfect matching with each male matched to one of his female partners

(vi) there cannot be a perfect matching with each female matched to one of her male partners

(b) The data shows that approximately 20% of the females were virgins, while only 5% of the males were. The researcher wonders how excluding virgins from the population would change the averages. If he knew graph theory, the researcher would realize that the nonvirgin male average number of partners will be $x(f/m)$ times the nonvirgin female average number of partners. What is x?

9.2 The Stable Marriage Problem

Okay, frequent public reference to derived variables may not help your mating prospects. But they can help with the analysis!

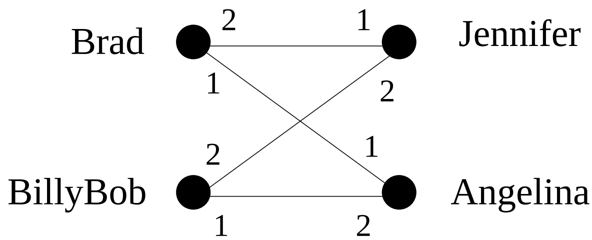
9.2.1 The Problem

Suppose there are a bunch of boys and an equal number of girls that we want to marry off. Each boy has his personal preferences about the girls —in fact, we assume he has a complete list of all the girls ranked according to his preferences, with no ties. Likewise, each girl has a ranked list of all of the boys.

The preferences don't have to be symmetric. That is, Jennifer might like Brad best, but Brad doesn't necessarily like Jennifer best. The goal is to marry off boys and girls: every boy must marry exactly one girl and vice-versa—no polygamy. In mathematical terms, we want the mapping from boys to their wives to be a bijection or *perfect matching*. We'll just call this a "matching," for short.

Here's the difficulty: suppose *every* boy likes Angelina best, and every girl likes Brad best, but Brad and Angelina are married to other people, say Jennifer and Billy Bob. Now *Brad and Angelina prefer each other to their spouses*, which puts their marriages at risk: pretty soon, they're likely to start spending late nights is study sessions together :-) .

This situation is illustrated in the following diagram where the digits "1" and "2" near a boy shows which of the two girls he ranks first and which second, and similarly for the girls:



More generally, in any matching, a boy and girl who are not married to each other and who like each other better than their spouses, is called a *rogue couple*. In the situation above, Brad and Angelina would be a rogue couple.

Having a rogue couple is not a good thing, since it threatens the stability of the marriages. On the other hand, if there are no rogue couples, then for any boy and girl who are not married to each other, at least one likes their spouse better than the other, and so won't be tempted to start an affair.

Definition 9.2.1. A *stable matching* is a matching with no rogue couples.

The question is, given everybody's preferences, how do you find a stable set of marriages? In the example consisting solely of the four people above, we could let Brad and Angelina both have their first choices by marrying each other. Now neither Brad nor Angelina prefers anybody else to their spouse, so neither will be in a rogue couple. This leaves Jen not-so-happily married to Billy Bob, but neither Jen nor Billy Bob can entice somebody else to marry them.

It is something of a surprise that there always is a stable matching among a group of boys and girls, but there is, and we'll shortly explain why. The surprise springs in part from considering the apparently similar "buddy" matching problem. That is, if people can be paired off as buddies, regardless of gender, then a stable matching *may not* be possible. For example, Figure 9.3 shows a situation with a love triangle and a fourth person who is everyone's last choice. In this figure Mergatoid's preferences aren't shown because they don't even matter.

Let's see why there is no stable matching:

Lemma. *There is no stable buddy matching among the four people in Figure* 9.3*.*

Proof. We'll prove this by contradiction.

Assume, for the purposes of contradiction, that there is a stable matching. Then there are two members of the love triangle that are matched. Since preferences in the triangle are symmetric, we may assume in particular, that Robin and Alex are matched. Then the other pair must be Bobby-Joe matched with Mergatoid.

But then there is a rogue couple: Alex likes Bobby-Joe best, and Bobby-Joe prefers Alex to his buddy Mergatoid. That is, Alex and Bobby-Joe are a rogue couple, contradicting the assumed stability of the matching. ∎

So getting a stable *buddy* matching may not only be hard, it may be impossible. But when boys are only allowed to marry girls, and vice versa, then it turns out that a stable matching is not hard to find.

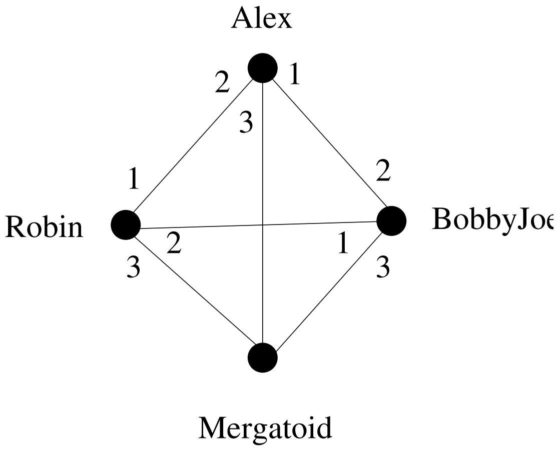


Figure 9.3: Some preferences with no stable buddy matching.

9.2.2 The Mating Ritual

The procedure for finding a stable matching involves a *Mating Ritual* that takes place over several days. The following events happen each day:

Morning: Each girl stands on her balcony. Each boy stands under the balcony of his favorite among the girls on his list, and he serenades her. If a boy has no girls left on his list, he stays home and does his 6.042 homework.

Afternoon: Each girl who has one or more suitors serenading her, says to her favorite among them, "We might get engaged. Come back tomorrow." To the other suitors, she says, "No. I will never marry you! Take a hike!"

Evening: Any boy who is told by a girl to take a hike, crosses that girl off his list.

Termination condition: When every girl has at most one suitor, the ritual ends with each girl marrying her suitor, if she has one.

There are a number of facts about this Mating Ritual that we would like to prove:

- The Ritual has a last day.
- Everybody ends up married.
- The resulting marriages are stable.

9.2.3 A State Machine Model

Before we can prove anything, we should have clear mathematical definitions of what we're talking about. In this section we sketch how to define a rigorous state machine model of the Marriage Problem.

So let's begin by formally defining the problem.

Definition 9.2.2. A *Marriage Problem* consists of two disjoint sets of the same finite size, called the-Boys and the-Girls. The members of the-Boys are called *boys*, and members of the-Girls are called *girls*. For each boy, B, there is a strict total order, $<_B$, on the-Girls, and for each girl, G, there is a strict total order, $<_G$, on the-Boys. If $G_1 <_B G_2$ we say B *prefers* girl G_2 to girl G_1. Similarly, if $B_1 <_G B_2$ we say G *prefers* boy B_2 to boy B_1.

A *marriage assignment* or *perfect matching* is a bijection, w : the-Boys $\rightarrow$ the-Girls. If $B \in$ the-Boys, then $w(B)$ is called B's *wife* in the assignment, and if $G \in$ the-Girls, then $w^{-1}(G)$ is called G's *husband*. A *rogue couple* is a boy, B, and a girl, G, such that B prefers G to his wife, and G prefers B to her husband. An assignment is *stable* if it has no rogue couples. A *solution* to a marriage problem is a stable perfect matching.

To model the Mating Ritual with a state machine, we make a key observation: to determine what happens on any day of the Ritual, all we need to know is which girls are still on which boys' lists on the morning of that day. So we define a state to be some mathematical data structure providing this information. For example, we could define a state to be the "still-has-on-his-list" relation, R, between boys and girls, where $B \ R \ G$ means girl G is still on boy B's list.

We start the Mating Ritual with no girls crossed off. That is, the start state is the *complete bipartite* relation in which every boy is related to every girl.

According to the Mating Ritual, on any given morning, a boy will *serenade* the girl he most prefers among those he has not as yet crossed out. Mathematically, the girl he is serenading is just the maximum among the girls on B's list, ordered by $<_B$. (If the list is empty, he's not serenading anybody.) A girl's *favorite* is just the maximum, under her preference ordering, among the boys serenading her.

Continuing in this way, we could mathematically specify a precise Mating Ritual state machine, but we won't bother. The intended behavior of the Mating Ritual is clear enough that we don't gain much by giving a formal state machine, so we stick to a more memorable description in terms of boys, girls, and their preferences. The point is, though, that it's not hard to define everything using basic mathematical data structures like sets, functions, and relations, if need be.

9.2.4 There is a Marriage Day

It's easy to see why the Mating Ritual has a terminal day when people finally get married. Every day on which the ritual hasn't terminated, at least one boy crosses a girl off his list. (If the ritual hasn't terminated, there must be some girl serenaded by at least two boys, and at least one of them will have to cross her off his list). So starting with n boys and n girls, each of the n boys' lists initially has n girls on it, for a total of n^2 list entries. Since no girl ever gets added to a list, the total number of entries on the lists decreases every day that the Ritual continues, and so the Ritual can continue for at most n^2 days.

9.2.5 They All Live Happily Every After...

We still have to prove that the Mating Ritual leaves everyone in a stable marriage. To do this, we note one very useful fact about the Ritual: if a girl has a favorite boy suitor on some morning of the Ritual, then that favorite suitor will still be serenading her the next morning —because his list won't have changed. So she is sure to have today's favorite boy among her suitors tomorrow. That means she will be able to choose a favorite suitor tomorrow who is at least as desirable to her as today's favorite. So day by day, her favorite suitor can stay the same or get better, never worse. In others words, a girl's favorite is a weakly increasing variable with respect to her preference order on the boys.

Now we can verify the Mating Ritual using a simple invariant predicate, P, that captures what's going on:

> For every girl, G, and every boy, B, if G is crossed off B's list, then G has a suitor whom she prefers over B.

Why is P invariant? Well, we know that G's favorite tomorrow will be at least as desirable to her as her favorite today, and since her favorite today is more desirable than B, tomorrow's favorite will be too.

Notice that P also holds on the first day, since every girl is on every list. So by the Invariant Theorem, we know that P holds on every day that the Mating Ritual runs. Knowing the invariant holds when the Mating Ritual terminates will let us complete the proofs.

Theorem 9.2.3. *Everyone is married by the Mating Ritual.*

Proof. Suppose, for the sake of contradiction, that it is the last day of the Mating Ritual and some boy does not get married. Then he can't be serenading anybody, and so his list must be empty. So by invariant P, every girl has a favorite boy whom she prefers to that boy. In particular, every girl has a favorite boy whom she marries on the last day. So all the girls are married. What's more there is no bigamy: a boy only serenades one girl, so no two girls have the same favorite.

But there are the same number of girls as boys, so all the boys must be married too. ■

Theorem 9.2.4. *The Mating Ritual produces a stable matching.*

Proof. Let Brad be some boy and Jen be any girl that he is *not* married to on the last day of the Mating Ritual. We claim that Brad and Jen are not a rogue couple. Since Brad is an arbitrary boy, it follows that no boy is part of a rogue couple. Hence the marriages on the last day are stable.

To prove the claim, we consider two cases:

Case 1. Jen is not on Brad's list. Then by invariant P, we know that Jen prefers her husband to Brad. So she's not going to run off with Brad: the claim holds in this case.

Case 2. Otherwise, Jen is on Brad's list. But since Brad is not married to Jen, he must be choosing to serenade his wife instead of Jen, so he must prefer his wife. So he's not going to run off with Jen: the claim also holds in this case. ■

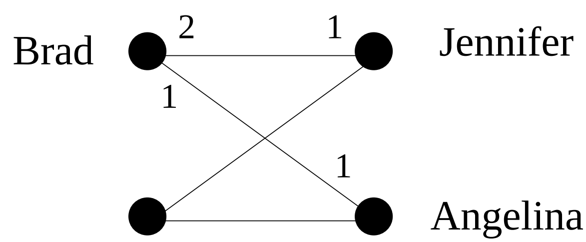
9.2.6 ...Especially the Boys

Who is favored by the Mating Ritual, the boys or the girls? The girls seem to have all the power: they stand on their balconies choosing the finest among their suitors and spurning the rest. What's more, we know their suitors can only change for the better as the Ritual progresses. Similarly, a boy keeps serenading the girl he most prefers among those on his list until he must cross her off, at which point he serenades the next most preferred girl on his list. So from the boy's point of view, the girl he is serenading can only change for the worse. Sounds like a good deal for the girls.

But it's not! The fact is that from the beginning, the boys are serenading their first choice girl, and the desirability of the girl being serenaded decreases only enough to give the boy his most desirable possible spouse. The mating algorithm actually does as well as possible for all the boys and does the worst possible job for the girls.

To explain all this we need some definitions. Let's begin by observing that while the mating algorithm produces one stable matching, there may be other stable matchings among the same set of boys and girls. For example, reversing the roles of boys and girls will often yield a different stable matching among them.

But some spouses might be out of the question in all possible stable matchings. For example, Brad is just not in the realm of possibility for Jennifer, since if you ever pair them, Brad and Angelina will form a rogue couple; here's a picture:



Definition 9.2.5. Given any marriage problem, one person is in another person's *realm of possible spouses* if there is a stable matching in which the two people are married. A person's *optimal spouse* is their most preferred person within their realm of possibility. A person's *pessimal spouse* is their least preferred person in their realm of possibility.

Everybody has an optimal and a pessimal spouse, since we know there is at least one stable matching, namely the one produced by the Mating Ritual. Now here is the shocking truth about the Mating Ritual:

Theorem 9.2.6. *The Mating Ritual marries every boy to his optimal spouse.*

Proof. Assume for the purpose of contradiction that some boy does not get his optimal girl. There must have been a day when he crossed off his optimal girl —otherwise he would still be serenading her or some even more desirable girl.

By the Well Ordering Principle, there must be a *first* day when a boy, call him "Keith," crosses off his optimal girl, Nicole.

According to the rules of the Ritual, Keith crosses off Nicole because Nicole has a favorite suitor, Tom, and

Nicole prefers Tom to Keith (*)

(remember, this is a proof by contradiction :-)).

Now since this is the first day an optimal girl gets crossed off, we know Tom hasn't crossed off his optimal girl. So

Tom ranks Nicole at least as high as his optimal girl. (**)

By the definition of an optimal girl, there must be some stable set of marriages in which Keith gets his optimal girl, Nicole. But then the preferences given in (*) and (**) imply that Nicole and Tom are a rogue couple within this supposedly stable set of marriages (think about it). This is a contradiction. ■

Theorem 9.2.7. *The Mating Ritual marries every girl to her pessimal spouse.*

Proof. Say Nicole and Keith marry each other as a result of the Mating Ritual. By the previous Theorem 9.2.6, Nicole is Keith's optimal spouse, and so in any stable set of marriages,

Keith rates Nicole at least as high as his spouse. (+)

Now suppose for the purpose of contradiction that there is another stable set of marriages where Nicole does worse than Keith. That is, Nicole is married to Tom, and

Nicole prefers Keith to Tom (++)

Then in this stable set of marriages where Nicole is married to Tom, (+) and (++) imply that Nicole and Keith are a rogue couple, contradicting stability. We conclude that Nicole cannot do worse than Keith. ■

9.2.7 Applications

Not surprisingly, a stable matching procedure is used by at least one large dating agency. But although "boy-girl-marriage" terminology is traditional and makes some of the definitions easier to remember (we hope without offending anyone), solutions to the Stable Marriage Problem are widely useful.

The Mating Ritual was first announced in a paper by D. Gale and L.S. Shapley in 1962, but ten years before the Gale-Shapley paper was appeared, and unknown by them, the Ritual was being used to assign residents to hospitals by the National Resident Matching Program (NRMP). The NRMP has, since the turn of the twentieth century, assigned each year's pool of medical school graduates to hospital residencies (formerly called "internships") with hospitals and graduates playing the roles of boys and girls. (In this case there may be multiple boys married to one girl, but there's an easy way to use the Ritual in this situation (see Problem 9.12). Before the Ritual was adopted, there were chronic disruptions and awkward countermeasures taken to preserve assignments of graduates to residencies. The Ritual resolved these problems so successfully, that it was used essentially without change at least through 1989.[2]

The internet infrastructure company, Akamai, also uses a variation of the Gale-Shapley procedure to assign web traffic to servers. In the early days, Akamai used other combinatorial optimization algorithms that got to be too slow as the number of servers (over 20,000 in 2010) reference needed and traffic increased. Akamai switched to Gale-Shapley since it is fast and can be run in a distributed manner. In this case, the web traffic corresponds to the boys and the web servers to the girls. The servers have preferences based on latency and packet loss; the traffic has preferences based on the cost of bandwidth.

9.2.8 Problems

Practice Problems

Problem 9.8.
Four Students want separate assignments to four VI-A Companies. Here are their preference rankings:

Student	Companies
Albert:	HP, Bellcore, AT&T, Draper
Rich:	AT&T, Bellcore, Draper, HP
Megumi:	HP, Draper, AT&T, Bellcore
Justin:	Draper, AT&T, Bellcore, HP

Company	Students
AT&T:	Justin, Albert, Megumi, Rich
Bellcore:	Megumi, Rich, Albert, Justin
HP:	Justin, Megumi, Albert, Rich
Draper:	Rich, Justin, Megumi, Albert

(a) Use the Mating Ritual to find *two* stable assignments of Students to Companies.

(b) Describe a simple procedure to determine whether any given stable marriage problem has a unique solution, that is, only one possible stable matching.

Problem 9.9.
Suppose that Harry is one of the boys and Alice is one of the girls in the *Mating Ritual*. Which of the properties below are preserved invariants? Why?

[2]Much more about the Stable Marriage Problem can be found in the very readable mathematical monograph by Dan Gusfield and Robert W. Irving, The Stable Marriage Problem: Structure and Algorithms, MIT Press, Cambridge, Massachusetts, 1989, 240 pp.

a. Alice is the only girl on Harry's list.

b. There is a girl who does not have any boys serenading her.

c. If Alice is not on Harry's list, then Alice has a suitor that she prefers to Harry.

d. Alice is crossed off Harry's list and Harry prefers Alice to anyone he is serenading.

e. If Alice is on Harry's list, then she prefers to Harry to any suitor she has.

Class Problems

Problem 9.10.
A preserved invariant of the Mating ritual is:

> For every girl, G, and every boy, B, if G is crossed off B's list, then G has a favorite suitor and she prefers him over B.

Use the invariant to prove that the Mating Algorithm produces stable marriages. (Don't look up the proof in the Notes or slides.)

Problem 9.11.
Consider a stable marriage problem with 4 boys and 4 girls and the following partial information about their preferences:

B1:	G1	G2	–	–
B2:	G2	G1	–	–
B3:	–	–	G4	G3
B4:	–	–	G3	G4
G1:	B2	B1	–	–
G2:	B1	B2	–	–
G3:	–	–	B3	B4
G4:	–	–	B4	B3

(a) Verify that

$$(B1, G1), (B2, G2), (B3, G3), (B4, G4)$$

will be a stable matching whatever the unspecified preferences may be.

(b) Explain why the stable matching above is neither boy-optimal nor boy-pessimal and so will not be an outcome of the Mating Ritual.

(c) Describe how to define a set of marriage preferences among n boys and n girls which have at least $2^{n/2}$ stable assignments.

Hint: Arrange the boys into a list of $n/2$ pairs, and likewise arrange the girls into a list of $n/2$ pairs of girls. Choose preferences so that the kth pair of boys ranks the kth pair of girls just below the previous pairs of girls, and likewise for the kth pair of girls. Within the kth pairs, make sure each boy's first choice girl in the pair prefers the other boy in the pair.

Homework Problems

Problem 9.12.
The most famous application of stable matching was in assigning graduating medical students to hospital residencies. Each hospital has a preference ranking of students and each student has a preference order of hospitals, but unlike the setup in the notes where there are an equal number of boys and girls and monogamous marriages, hospitals generally have differing numbers of available residencies, and the total number of residencies may not equal the number of graduating students. Modify the definition of stable matching so it applies in this situation, and explain how to modify the Mating Ritual so it yields stable assignments of students to residencies. No proof is required.

Problem 9.13.
Give an example of a stable matching between 3 boys and 3 girls where no person gets their first choice. Briefly explain why your matching is stable.

Problem 9.14.
In a stable matching between n boys and girls produced by the Mating Ritual, call a person *lucky* if they are matched up with one of their $\lceil n/2 \rceil$ top choices. We will prove:

Theorem. *There must be at least one lucky person.*

To prove this, define the following derived variables for the Mating Ritual:

$q(B) = j$, where j is the rank of the girl that boy B is courting. That is to say, boy B is always courting the jth girl on his list.

$r(G)$ is the number of boys that girl G has rejected.

(a) Let

$$S ::= \sum_{B \in \text{the-Boys}} q(B) - \sum_{G \in \text{the-Girls}} r(G). \tag{9.1}$$

Show that S remains the same from one day to the next in the Mating Ritual.

(b) Prove the Theorem above. (You may assume for simplicity that n is even.)

Hint: A girl is sure to be lucky if she has rejected half the boys.

9.3 Connectedness

9.3.1 Paths and Simple Cycles

Paths in simple graphs are esentially the same as paths in digraphs. We just modify the digraph definitions using undirected edges instead of directed ones. For example, the formal definition of a path in a simple graph is a virtually that same as Definition 7.1.1 of paths in digraphs:

Definition 9.3.1. A *path* in a graph, G, is a sequence of $k \geq 0$ vertices

$$v_0, \ldots, v_k$$

such that v_i—v_{i+1} is an edge of G for all i where $0 \leq i < k$. The path is said to *start* at v_0, to *end* at v_k, and the *length* of the path is defined to be k.

An edge, u—v, is *traversed* n times by the path if there are n different values of i such that v_i—$v_{i+1} = u$—v. The path is *simple*[3] iff all the v_i's are different, that is, if $i \neq j$ implies $v_i \neq v_j$.

For example, the graph in Figure 9.4 has a length 6 simple path A,B,C,D,E,F,G. This is the longest simple path in the graph.

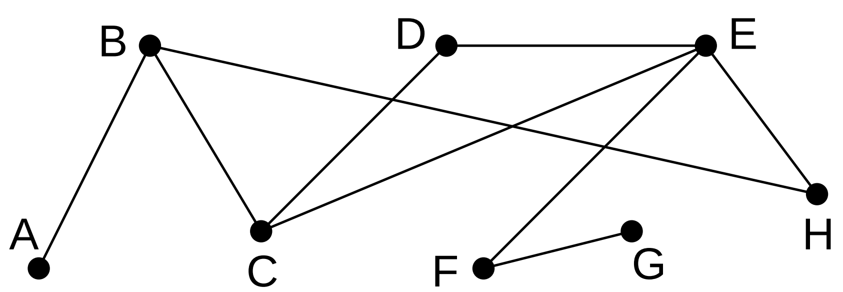


Figure 9.4: *A graph with 3 simple cycles.*

As in digraphs, the length of a path is the total number of times it traverses edges, which is *one less* than its length as a sequence of vertices. For example, the length 6 path A,B,C,D,E,F,G is actually a sequence of seven vertices.

A *cycle* can be described by a path that begins and ends with the same vertex. For example, B,C,D,E,C,B is a cycle in the graph in Figure 9.4. This path suggests that the cycle begins and ends at vertex B, but a cycle isn't intended to have a

[3]Heads up: what we call "paths" are commonly referred to in graph theory texts as "walks," and simple paths are referred to as just "paths". Likewise, what we will call *cycles* and *simple cycles* are commonly called "closed walks" and just "cycles".

beginning and end, and can be described by *any* of the paths that go around it. For example, D,E,C,B,C,D describes this same cycle as though it started and ended at D, and D,C,B,C,E,D describes the same cycle as though it started and ended at D but went in the opposite direction. (By convention, a single vertex is a length 0 cycle beginning and ending at the vertex.)

All the paths that describe the same cycle have the same length which is defined to be the *length of the cycle*. (Note that this implies that going around the same cycle twice is considered to be different than going around it once.)

A *simple* cycle is a cycle that doesn't cross or backtrack on itself. For example, the graph in Figure 9.4 has three simple cycles B,H,E,C,B and C,D,E,C and B,C,D,E,H,B. More precisely, a simple cycle is a cycle that can be described by a path of length at least three whose vertices are all different except for the beginning and end vertices. So in contrast to simple *paths*, the length of a simple *cycle* is the *same* as the number of distinct vertices that appear in it.

From now on we'll stop being picky about distinguishing a cycle from a path that describes it, and we'll just refer to the path as a cycle. [4]

Simple cycles are especially important, so we will give a proper definition of them. Namely, we'll define a simple cycle in G to be a *subgraph* of G that looks like a cycle that doesn't cross itself. Formally:

Definition 9.3.2. A *subgraph*, G', of a graph, G, is a graph whose vertices, V', are a subset of the vertices of G and whose edges are a subset of the edges of G.

Notice that since a subgraph is itself a graph, the endpoints of every edge of G' must be vertices in V'.

Definition 9.3.3. For $n \geq 3$, let C_n be the graph with vertices $1, \ldots, n$ and edges

$$1—2,\ \ 2—3,\ \ \ldots,\ \ (n-1)—n,\ \ n—1.$$

A graph is a *simple cycle* of length n iff it is isomorphic to C_n for some $n \geq 3$. A *simple cycle of a graph*, G, is a subgraph of G that is a simple cycle.

This definition formally captures the idea that simple cycles don't have direction or beginnings or ends.

9.3.2 Connected Components

Definition 9.3.4. Two vertices in a graph are said to be *connected* when there is a path that begins at one and ends at the other. By convention, every vertex is considered to be connected to itself by a path of length zero.

The diagram in Figure 9.5 looks like a picture of three graphs, but is intended to be a picture of *one* graph. This graph consists of three pieces (subgraphs). Each piece by itself is connected, but there are no paths between vertices in different pieces.

[4] Technically speaking, we haven't ever defined what a cycle *is*, only how to describe it with paths. But we won't need an abstract definition of cycle, since all that matters about a cycle is which paths describe it.

Figure 9.5: *One graph with 3 connected components.*

Definition 9.3.5. A graph is said to be *connected* when every pair of vertices are connected.

These connected pieces of a graph are called its *connected components*. A rigorous definition is easy: a connected component is the set of all the vertices connected to some single vertex. So a graph is connected iff it has exactly one connected component. The empty graph on n vertices has n connected components.

9.3.3 How Well Connected?

If we think of a graph as modelling cables in a telephone network, or oil pipelines, or electrical power lines, then we not only want connectivity, but we want connectivity that survives component failure. A graph is called *k-edge connected* if it takes at least k "edge-failures" to disconnect it. More precisely:

Definition 9.3.6. Two vertices in a graph are *k-edge connected* if they remain connected in every subgraph obtained by deleting $k-1$ edges. A graph with at least two vertices is k-edge connected[5] if every two of its vertices are k-edge connected.

So 1-edge connected is the same as connected for both vertices and graphs. Another way to say that a graph is k-edge connected is that every subgraph obtained from it by deleting at most $k-1$ edges is connected. For example, in the graph in Figure 9.4, vertices B and E are 2-edge connected, G and E are 1-edge connected, and no vertices are 3-edge connected. The graph as a whole is only 1-edge connected. More generally, any simple cycle is 2-edge connected, and the complete graph, K_n, is $(n-1)$-edge connected.

If two vertices are connected by k edge-disjoint paths (that is, no two paths traverse the same edge), then they are obviously k-edge connected. A fundamental

[5]The corresponding definition of connectedness based on deleting vertices rather than edges is common in Graph Theory texts and is usually simply called "k-connected" rather than "k-vertex connected."

fact, whose ingenious proof we omit, is Menger's theorem which confirms that the converse is also true: if two vertices are k-edge connected, then there are k edge-disjoint paths connecting them. It even takes some ingenuity to prove this for the case $k = 2$.

9.3.4 Connection by Simple Path

Where there's a path, there's a simple path. This is sort of obvious, but it's easy enough to prove rigorously using the Well Ordering Principle.

Lemma 9.3.7. *If vertex u is connected to vertex v in a graph, then there is a simple path from u to v.*

Proof. Since there is a path from u to v, there must, by the Well-ordering Principle, be a minimum length path from u to v. If the minimum length is zero or one, this minimum length path is itself a simple path from u to v. Otherwise, there is a minimum length path

$$v_0, v_1, \ldots, v_k$$

from $u = v_0$ to $v = v_k$ where $k \geq 2$. We claim this path must be simple. To prove the claim, suppose to the contrary that the path is not simple, that is, some vertex on the path occurs twice. This means that there are integers i, j such that $0 \leq i < j \leq k$ with $v_i = v_j$. Then deleting the subsequence

$$v_{i+1}, \ldots v_j$$

yields a strictly shorter path

$$v_0, v_1, \ldots, v_i, v_{j+1}, v_{j+2}, \ldots, v_k$$

from u to v, contradicting the minimality of the given path. ■

Actually, we proved something stronger:

Corollary 9.3.8. *For any path of length k in a graph, there is a simple path of length* at most *k with the same endpoints.*

9.3.5 The Minimum Number of Edges in a Connected Graph

The following theorem says that a graph with few edges must have many connected components.

Theorem 9.3.9. *Every graph with v vertices and e edges has at least $v - e$ connected components.*

Of course for Theorem 9.3.9 to be of any use, there must be fewer edges than vertices.

Proof. We use induction on the number of edges, e. Let $P(e)$ be the proposition that for every v, every graph with v vertices and e edges has at least $v - e$ connected components.

Base case:($e = 0$). In a graph with 0 edges and v vertices, each vertex is itself a connected component, and so there are exactly $v = v - 0$ connected components. So $P(e)$ holds.

Inductive step: Now we assume that the induction hypothesis holds for every e-edge graph in order to prove that it holds for every $(e + 1)$-edge graph, where $e \geq 0$. Consider a graph, G, with $e + 1$ edges and k vertices. We want to prove that G has at least $v - (e + 1)$ connected components. To do this, remove an arbitrary edge a—b and call the resulting graph G'. By the induction assumption, G' has at least $v - e$ connected components. Now add back the edge a—b to obtain the original graph G. If a and b were in the same connected component of G', then G has the same connected components as G', so G has at least $v - e > v - (e + 1)$ components. Otherwise, if a and b were in different connected components of G', then these two components are merged into one in G, but all other components remain unchanged, reducing the number of components by 1. Therefore, G has at least $(v - e) - 1 = v - (e + 1)$ connected components. So in either case, $P(e + 1)$ holds. This completes the Inductive step. The theorem now follows by induction. ■

Corollary 9.3.10. *Every connected graph with v vertices has at least $v - 1$ edges.*

A couple of points about the proof of Theorem 9.3.9 are worth noticing. First, we used induction on the number of edges in the graph. This is very common in proofs involving graphs, and so is induction on the number of vertices. When you're presented with a graph problem, these two approaches should be among the first you consider. The second point is more subtle. Notice that in the inductive step, we took an arbitrary $(n + 1)$-edge graph, threw out an edge so that we could apply the induction assumption, and then put the edge back. You'll see this shrink-down, grow-back process very often in the inductive steps of proofs related to graphs. This might seem like needless effort; why not start with an n-edge graph and add one more to get an $(n + 1)$-edge graph? That would work fine in this case, but opens the door to a nasty logical error called *buildup error*, illustrated in Problems 9.5 and 9.18. Always use shrink-down, grow-back arguments, and you'll never fall into this trap.

9.3.6 Problems

Class Problems

Problem 9.15.
The n-dimensional hypercube, H_n, is a graph whose vertices are the binary strings of length n. Two vertices are adjacent if and only if they differ in exactly 1 bit. For example, in H_3, vertices `111` and `011` are adjacent because they differ only in the first bit, while vertices `101` and `011` are not adjacent because they differ at both the first and second bits.

(a) Prove that it is impossible to find two spanning trees of H_3 that do not share some edge.

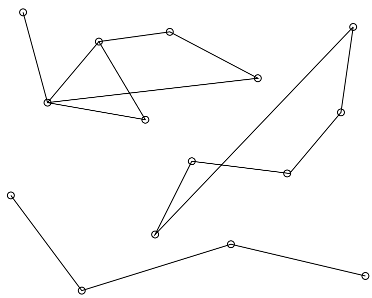
(b) Verify that for any two vertices $x \neq y$ of H_3, there are 3 paths from x to y in H_3, such that, besides x and y, no two of those paths have a vertex in common.

(c) Conclude that the connectivity of H_3 is 3.

(d) Try extending your reasoning to H_4. (In fact, the connectivity of H_n is n for all $n \geq 1$. A proof appears in the problem solution.)

Problem 9.16.
A set, M, of vertices of a graph is a *maximal connected* set if every pair of vertices in the set are connected, and any set of vertices properly containing M will contain two vertices that are not connected.

(a) What are the maximal connected subsets of the following (unconnected) graph?



(b) Explain the connection between maximal connected sets and connected components. Prove it.

Problem 9.17. **(a)** Prove that K_n is $(n-1)$-edge connected for $n > 1$.

Let M_n be a graph defined as follows: begin by taking n graphs with non-overlapping sets of vertices, where each of the n graphs is $(n-1)$-edge connected (they could be disjoint copies of K_n, for example). These will be subgraphs of M_n. Then pick n vertices, one from each subgraph, and add enough edges between pairs of picked vertices that the subgraph of the n picked vertices is also $(n-1)$-edge connected.

(b) Draw a picture of M_4.

(c) Explain why M_n is $(n-1)$-edge connected.

Problem 9.18.
Definition 9.3.5. A graph is *connected* iff there is a path between every pair of its vertices.

False Claim. *If every vertex in a graph has positive degree, then the graph is connected.*

(a) Prove that this Claim is indeed false by providing a counterexample.

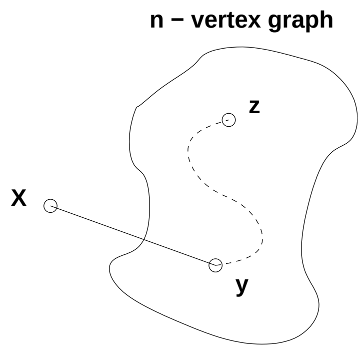
(b) Since the Claim is false, there must be an logical mistake in the following bogus proof. Pinpoint the *first* logical mistake (unjustified step) in the proof.

Bogus proof. We prove the Claim above by induction. Let $P(n)$ be the proposition that if every vertex in an n-vertex graph has positive degree, then the graph is connected.

Base cases: ($n \leq 2$). In a graph with 1 vertex, that vertex cannot have positive degree, so $P(1)$ holds vacuously.

$P(2)$ holds because there is only one graph with two vertices of positive degree, namely, the graph with an edge between the vertices, and this graph is connected.

Inductive step: We must show that $P(n)$ implies $P(n+1)$ for all $n \geq 2$. Consider an n-vertex graph in which every vertex has positive degree. By the assumption $P(n)$, this graph is connected; that is, there is a path between every pair of vertices. Now we add one more vertex x to obtain an $(n+1)$-vertex graph:



All that remains is to check that there is a path from x to every other vertex z. Since x has positive degree, there is an edge from x to some other vertex, y. Thus, we can obtain a path from x to z by going from x to y and then following the path from y to z. This proves $P(n+1)$.

By the principle of induction, $P(n)$ is true for all $n \geq 0$, which proves the Claim.

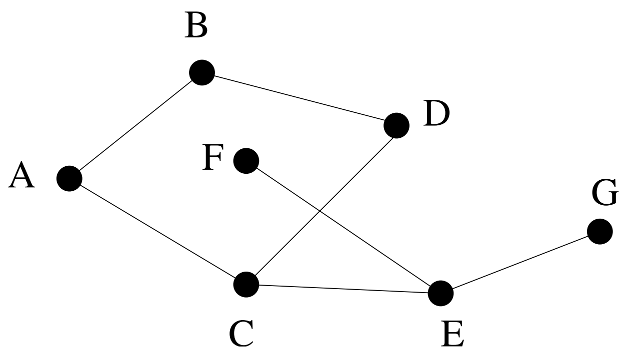
■

Homework Problems

Problem 9.19.
In this problem we'll consider some special cycles in graphs called *Euler circuits*, named after the famous mathematician Leonhard Euler. (Same Euler as for the constant $e \approx 2.718$ —he did a lot of stuff.)

Definition 9.3.11. An Euler circuit of a graph is a cycle which traverses every edge exactly once.

Does the graph in the following figure contain an Euler circuit?



Well, if it did, the edge (E, F) would need to be included. If the path does not start at F then at some point it traverses edge (E, F), and now it is stuck at F since F has no other edges incident to it and an Euler circuit can't traverse (E, F) twice. But then the path could not be a circuit. On the other hand, if the path starts at F, it must then go to E along (E, F), but now it cannot return to F. It again cannot be a circuit. This argument generalizes to show that if a graph has a vertex of degree 1, it cannot contain an Euler circuit.

So how do you tell in general whether a graph has an Euler circuit? At first glance this may seem like a daunting problem (the similar sounding problem of finding a cycle that touches every vertex exactly once is one of those million dollar NP-complete problems known as the *Traveling Salesman Problem*) —but it turns out to be easy.

(a) Show that if a graph has an Euler circuit, then the degree of each of its vertices is even.

In the remaining parts, we'll work out the converse: if the degree of every vertex of a connected finite graph is even, then it has an Euler circuit. To do this, let's define an Euler *path* to be a path that traverses each edge *at most* once.

(b) Suppose that an Euler path in a connected graph does not traverse every edge. Explain why there must be an untraversed edge that is incident to a vertex on the path.

In the remaining parts, let W be the *longest* Euler path in some finite, connected graph.

(c) Show that if W is a cycle, then it must be an Euler circuit.

Hint: part (b)

(d) Explain why all the edges incident to the end of W must already have been traversed by W.

(e) Show that if the end of W was not equal to the start of W, then the degree of the end would be odd.

Hint: part (d)

(f) Conclude that if every vertex of a finite, connected graph has even degree, then it has an Euler circuit.

Homework Problems

Problem 9.20.
An edge is said to *leave* a set of vertices if one end of the edge is in the set and the other end is not.

(a) An n-node graph is said to be *mangled* if there is an edge leaving every set of $\lfloor n/2 \rfloor$ or fewer vertices. Prove the following claim.

Claim. *Every mangled graph is connected.*

An n-node graph is said to be *tangled* if there is an edge leaving every set of $\lceil n/3 \rceil$ or fewer vertices.

(b) Draw a tangled graph that is not connected.

(c) Find the error in the proof of the following

False Claim. *Every tangled graph is connected.*

False proof. The proof is by strong induction on the number of vertices in the graph. Let $P(n)$ be the proposition that if an n-node graph is tangled, then it is connected. In the base case, $P(1)$ is true because the graph consisting of a single node is trivially connected.

For the inductive case, assume $n \geq 1$ and $P(1), \ldots, P(n)$ hold. We must prove $P(n+1)$, namely, that if an $(n+1)$-node graph is tangled, then it is connected.

So let G be a tangled, $(n+1)$-node graph. Choose $\lceil n/3 \rceil$ of the vertices and let G_1 be the tangled subgraph of G with these vertices and G_2 be the tangled subgraph with the rest of the vertices. Note that since $n \geq 1$, the graph G has a least two vertices, and so both G_1 and G_2 contain at least one vertex. Since G_1 and G_2 are tangled, we may assume by strong induction that both are connected. Also, since G is tangled, there is an edge leaving the vertices of G_1 which necessarily connects to a vertex of G_2. This means there is a path between any two vertices of G: a path within one subgraph if both vertices are in the same subgraph, and a path traversing the connecting edge if the vertices are in separate subgraphs. Therefore, the entire graph, G, is connected. This completes the proof of the inductive case, and the Claim follows by strong induction.

■

Problem 9.21.
Let G be the graph formed from C_{2n}, the simple cycle of length $2n$, by connecting every pair of vertices at maximum distance from each other in C_{2n} by an edge in G.

(a) Given two vertices of G find their distance in G.

(b) What is the *diameter* of G, that is, the largest distance between two vertices?

(c) Prove that the graph is not 4-connected.

(d) Prove that the graph is 3-connected.

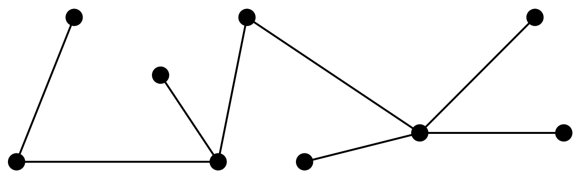
9.4 Trees

Trees are a fundamental data structure in computer science, and there are many kinds, such as rooted, ordered, and binary trees. In this section we focus on the purest kind of tree. Namely, we use the term *tree* to mean a connected graph without simple cycles.

A graph with no simple cycles is called *acyclic*; so trees are acyclic connected graphs.

9.4.1 Tree Properties

Here is an example of a tree:



A vertex of degree at most one is called a *leaf*. In this example, there are 5 leaves. Note that the only case where a tree can have a vertex of degree zero is a graph with a single vertex.

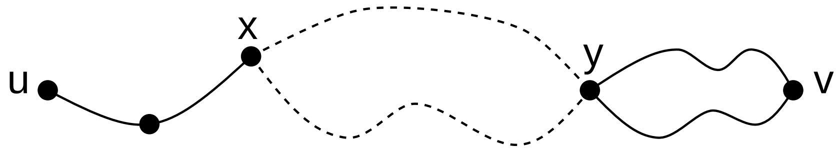
The graph shown above would no longer be a tree if any edge were removed, because it would no longer be connected. The graph would also not remain a tree if any edge were added between two of its vertices, because then it would contain a simple cycle. Furthermore, note that there is a unique path between every pair of vertices. These features of the example tree are actually common to all trees.

Theorem 9.4.1. *Every tree has the following properties:*

1. *Any connected subgraph is a tree.*
2. *There is a unique simple path between every pair of vertices.*
3. *Adding an edge between two vertices creates a cycle.*
4. *Removing any edge disconnects the graph.*
5. *If it has at least two vertices, then it has at least two leaves.*
6. *The number of vertices is one larger than the number of edges.*

Proof. 1. A simple cycle in a subgraph is also a simple cycle in the whole graph, so any subgraph of an acyclic graph must also be acyclic. If the subgraph is also connected, then by definition, it is a tree.

2. There is at least one path, and hence one simple path, between every pair of vertices, because the graph is connected. Suppose that there are two different simple paths between vertices u and v. Beginning at u, let x be the first vertex where the paths diverge, and let y be the next vertex they share. Then there are two simple paths from x to y with no common edges, which defines a simple cycle. This is a contradiction, since trees are acyclic. Therefore, there is exactly one simple path between every pair of vertices.



3. An additional edge u—v together with the unique path between u and v forms a simple cycle.

4. Suppose that we remove edge u—v. Since the tree contained a unique path between u and v, that path must have been u—v. Therefore, when that edge is removed, no path remains, and so the graph is not connected.

5. Let $v_1, \ldots, v_m$ be the sequence of vertices on a longest simple path in the tree. Then $m \geq 2$, since a tree with two vertices must contain at least one edge. There cannot be an edge v_1—v_i for $2 < i \leq m$; otherwise, vertices $v_1, \ldots, v_i$ would from a simple cycle. Furthermore, there cannot be an edge u—v_1 where u is not on the path; otherwise, we could make the path longer. Therefore, the only edge incident to v_1 is v_1—v_2, which means that v_1 is a leaf. By a symmetric argument, v_m is a second leaf.

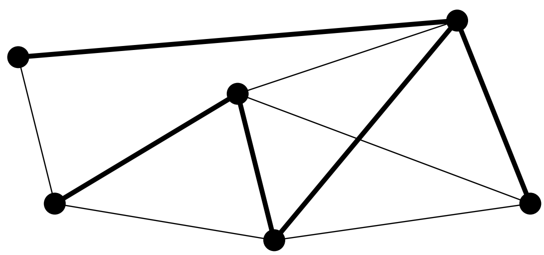
6. We use induction on the number of vertices. For a tree with a single vertex, the claim holds since it has no edges and $0+1=1$ vertex. Now suppose that the claim holds for all n-vertex trees and consider an $(n+1)$-vertex tree, T. Let v be a leaf of the tree. You can verify that deleting a vertex of degree 1 (and its incident edge) from any connected graph leaves a connected subgraph. So by 1., deleting v and its incident edge gives a smaller tree, and this smaller tree has one more vertex than edge by induction. If we re-attach the vertex, v, and its incident edge, then the equation still holds because the number of vertices and number of edges both increase by 1. Thus, the claim holds for T and, by induction, for all trees.

■

Various subsets of these properties provide alternative characterizations of trees, though we won't prove this. For example, a *connected* graph with a number of vertices one larger than the number of edges is necessarily a tree. Also, a graph with unique paths between every pair of vertices is necessarily a tree.

9.4.2 Spanning Trees

Trees are everywhere. In fact, every connected graph contains a subgraph that is a tree with the same vertices as the graph. This is a called a *spanning tree* for the graph. For example, here is a connected graph with a spanning tree highlighted.



Theorem 9.4.2. *Every connected graph contains a spanning tree.*

Proof. Let T be a connected subgraph of G, with the same vertices as G, and with the smallest number of edges possible for such a subgraph. We show that T is acyclic by contradiction. So suppose that T has a cycle with the following edges:

$$v_0 — v_1, v_1 — v_2, \ldots, v_n — v_0$$

Suppose that we remove the last edge, $v_n — v_0$. If a pair of vertices x and y was joined by a path not containing $v_n — v_0$, then they remain joined by that path. On the other hand, if x and y were joined by a path containing $v_n — v_0$, then they remain joined by a path containing the remainder of the cycle. So all the vertices of

G are still connected after we remove an edge from T. This is a contradiction, since T was defined to be a minimum size connected subgraph with all the vertices of G. So T must be acyclic. ■

Conversely, suppose every vertex in a graph, G, has even degree. Let $W = (v_0, \dots, v_n)$ be the longest path in G that traverses every edge *at most* once. Now W must traverse every edge incident to v_n; otherwise, the path could be extended. In particular, the W traverses two of these edges each time it passes through v_n, and it traverses v_{n-1}—v_n at the end. This accounts for an odd number of edges, but the degree of v_n is even by assumption. Therefore, the W must also begin at v_n; that is, $v_0 = v_n$. Suppose that W is not an Euler tour. Because G is a connected graph, we can find an edge not in W but incident to some vertex in W. Call this edge u—v_i. But then we can construct a longer walk:

$$u, u\text{—}v_i, v_i, v_i\text{—}v_{i+1}, \dots, v_{n-1}\text{—}v_n, v_n, v_0\text{—}v_1, \dots, v_{i-1}\text{—}v_i, v_i$$

This contradicts the definition of W, so W must be an Euler tour after all.

Corollary 9.4.3. *A connected graph has an Euler walk if and only if either 0 or 2 vertices have odd degree.*

Hamiltonian cycles are the unruly cousins of Euler tours. A *Hamiltonian cycle* is walk that starts and ends at the same vertex and visits every *vertex* in a graph exactly once. There is no simple characterization of all graphs with a Hamiltonian cycle. In fact, determining whether a given graph has a Hamiltonian cycle is is the same category of problem as the SAT problem of Section 1.4: you get a million dollars for finding an efficient way to determine when a graph has a Hamiltonian cycle —or for proving that no procedure works efficiently on all graphs.

9.4.3 Problems

Class Problems

Problem 9.22.
Procedure $Mark$ starts with a connected, simple graph with all edges unmarked and then marks some edges. At any point in the procedure a path that traverses only marked edges is called a *fully marked* path, and an edge that has no fully marked path between its endpoints is called *eligible*.

Procedure $Mark$ simply keeps marking eligible edges, and terminates when there are none.

Prove that $Mark$ terminates, and that when it does, the set of marked edges forms a spanning tree of the original graph.

Problem 9.23.

Procedure **create-spanning-tree**

Given a simple graph G, keep applying the following operations to the graph until no operation applies:

1. If an edge u—v of G is on a simple cycle, then delete u—v.
2. If vertices u and v of G are not connected, then add the edge u—v.

Assume the vertices of G are the integers $1, 2, \ldots, n$ for some $n \geq 2$. Procedure **create-spanning-tree** can be modeled as a state machine whose states are all possible simple graphs with vertices $1, 2, \ldots, n$. The start state is G, and the final states are the graphs on which no operation is possible.

(a) Let G be the graph with vertices $\{1, 2, 3, 4\}$ and edges

$$\{1—2, 3—4\}$$

What are the possible final states reachable from start state G? Draw them.

(b) Prove that any final state of must be a tree on the vertices.

(c) For any state, G', let e be the number of edges in G', c be the number of connected components it has, and s be the number of simple cycles. For each of the derived variables below, indicate the *strongest* of the properties that it is guaranteed to satisfy, no matter what the starting graph G is and be prepared to briefly explain your answer.

The choices for properties are: *constant*, *strictly increasing*, *strictly decreasing*, *weakly increasing*, *weakly decreasing*, *none of these*. The derived variables are

(i) e

(ii) c

(iii) s

(iv) $e - s$

(v) $c + e$

(vi) $3c + 2e$

(vii) $c + s$

(viii) (c, e), partially ordered coordinatewise (the *product* partial order, Ch. 6.4).

(d) Prove that procedure **create-spanning-tree** terminates. (If your proof depends on one of the answers to part (c), you must prove that answer is correct.)

Problem 9.24.
Prove that a graph is a tree iff it has a unique simple path between any two vertices.

Homework Problems

Problem 9.25. (a) Prove that the average degree of a tree is less than 2.

(b) Suppose every vertex in a graph has degree at least k. Explain why the graph has a simple path of length k.

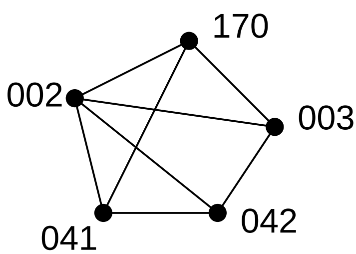
Hint: Consider a longest simple path.

9.5 Coloring Graphs

In section 9.1.2, we used edges to indicate an affinity between two nodes, but having an edge represent a *conflict* between two nodes also turns out to be really useful.

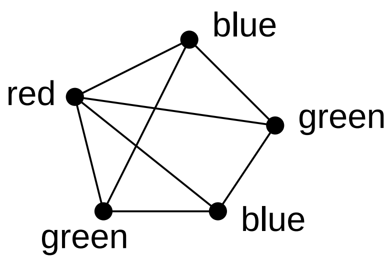
9.6 Modelling Scheduling Conflicts

Each term the MIT Schedules Office must assign a time slot for each final exam. This is not easy, because some students are taking several classes with finals, and a student can take only one test during a particular time slot. The Schedules Office wants to avoid all conflicts. Of course, you can make such a schedule by having every exam in a different slot, but then you would need hundreds of slots for the hundreds of courses, and exam period would run all year! So, the Schedules Office would also like to keep exam period short. The Schedules Office's problem is easy to describe as a graph. There will be a vertex for each course with a final exam, and two vertices will be adjacent exactly when some student is taking both courses. For example, suppose we need to schedule exams for 6.041, 6.042, 6.002, 6.003 and 6.170. The scheduling graph might look like this:



6.002 and 6.042 cannot have an exam at the same time since there are students in both courses, so there is an edge between their nodes. On the other hand, 6.042 and 6.170 can have an exam at the same time if they're taught at the same time (which they sometimes are), since no student can be enrolled in both (that is, no student *should* be enrolled in both when they have a timing conflict). Next, identify each time slot with a color. For example, Monday morning is red, Monday afternoon is blue, Tuesday morning is green, etc.

Assigning an exam to a time slot is now equivalent to coloring the corresponding vertex. The main constraint is that *adjacent vertices must get different colors* — otherwise, some student has two exams at the same time. Furthermore, in order to keep the exam period short, we should try to color all the vertices using as *few different colors as possible*. For our example graph, three colors suffice:



This coloring corresponds to giving one final on Monday morning (red), two Monday afternoon (blue), and two Tuesday morning (green). Can we use fewer than three colors? No! We can't use only two colors since there is a triangle in the graph, and three vertices in a triangle must all have different colors.

This is an example of what is a called a *graph coloring problem*: given a graph G, assign colors to each node such that adjacent nodes have different colors. A color assignment with this property is called a *valid coloring* of the graph —a "*coloring*," for short. A graph G is k*-colorable* if it has a coloring that uses at most k colors.

Definition 9.6.1. The minimum value of k for which a graph, G, has a valid coloring is called its *chromatic number*, $\chi(G)$.

In general, trying to figure out if you can color a graph with a fixed number of colors can take a long time. It's a classic example of a problem for which no fast algorithms are known. In fact, it is easy to check if a coloring works, but it seems really hard to find it (if you figure out how, then you can get a $1 million Clay prize).

9.6.1 Degree-bounded Coloring

There are some simple graph properties that give useful upper bounds on colorings. For example, if we have a bound on the degrees of all the vertices in a graph, then we can easily find a coloring with only one more color than the degree bound.

Theorem 9.6.2. *A graph with maximum degree at most k is $(k+1)$-colorable.*

Unfortunately, if you try induction on k, it will lead to disaster. It is not that it is impossible, just that it is extremely painful and would ruin you if you tried it on an exam. Another option, especially with graphs, is to change what you are inducting on. In graphs, some good choices are n, the number of nodes, or e, the number of edges.

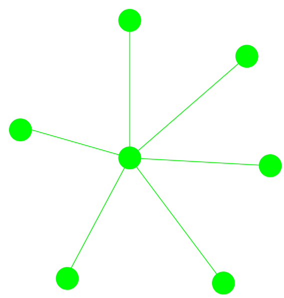
Proof. We use induction on the number of vertices in the graph, which we denote by n. Let $P(n)$ be the proposition that an n-vertex graph with maximum degree at most k is $(k+1)$-colorable.

Base case: $(n = 1)$ A 1-vertex graph has maximum degree 0 and is 1-colorable, so $P(1)$ is true.

Inductive step: Now assume that $P(n)$ is true, and let G be an $(n+1)$-vertex graph with maximum degree at most k. Remove a vertex v (and all edges incident to it), leaving an n-vertex subgraph, H. The maximum degree of H is at most k, and so H is $(k+1)$-colorable by our assumption $P(n)$. Now add back vertex v. We can assign v a color different from all its adjacent vertices, since there are at most k adjacent vertices and $k+1$ colors are available. Therefore, G is $(k+1)$-colorable. This completes the inductive step, and the theorem follows by induction. ■

Sometimes $k+1$ colors is the best you can do. For example, in the complete graph, K_n, every one of its n vertices is adjacent to all the others, so all n must be assigned different colors. Of course n colors is also enough, so $\chi(K_n) = n$. So K_{k+1} is an example where Theorem 9.6.2 gives the best possible bound. This means that Theorem 9.6.2 also gives the best possible bound for *any* graph with degree bounded by k that has K_{k+1} as a subgraph.

But sometimes $k+1$ colors is far from the best that you can do. Here's an example of an n-node star graph for $n = 7$:



In the n-node star graph, the maximum degree is $n-1$, but the star only needs 2 colors!

9.6.2 Why coloring?

One reason coloring problems come all the time is because scheduling conflicts are so common. For example, at Akamai, a new version of software is deployed over each of 20,000 servers every few days. The updates cannot be done at the same time since the servers need to be taken down in order to deploy the software. Also, the servers cannot be handled one at a time, since it would take forever to update them all (each one takes about an hour). Moreover, certain pairs of servers cannot be taken down at the same time since they have common critical functions. This problem was eventually solved by making a 20,000 node conflict graph and coloring it with 8 colors – so only 8 waves of install are needed! Another example comes from the need to assign frequencies to radio stations. If two stations have an overlap in their broadcast area, they can't be given the same frequency. Frequencies are precious and expensive, so you want to minimize the number handed out. This amounts to finding the minimum coloring for a graph whose vertices are the stations and whose edges are between stations with overlapping areas.

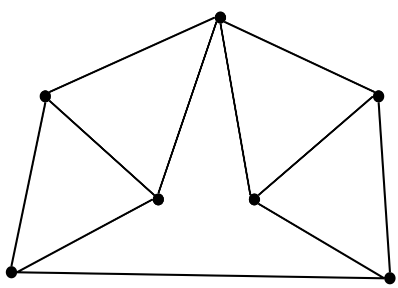
Coloring also comes up in allocating registers for program variables. While a variable is in use, its value needs to be saved in a register, but registers can often be reused for different variables. But two variables need different registers if they are referenced during overlapping intervals of program execution. So register allocation is the coloring problem for a graph whose vertices are the variables; vertices are adjacent if their intervals overlap, and the colors are registers.

Finally, there's the famous map coloring problem stated in Propostion 1.5.4. The question is how many colors are needed to color a map so that adjacent territories get different colors? This is the same as the number of colors needed to color a graph that can be drawn in the plane without edges crossing. A proof that four colors are enough for the *planar* graphs was acclaimed when it was discovered about thirty years ago. Implicit in that proof was a 4-coloring procedure that takes time proportional to the number of vertices in the graph (countries in the map). On the other hand, it's another of those million dollar prize questions to find an efficient procedure to tell if a planar graph really *needs* four colors or if three will actually do the job. But it's always easy to tell if an *arbitrary* graph is 2-colorable, as we show in Section 9.7. Later in Chapter 11, we'll develop enough planar graph theory to present an easy proof at least that planar graphs are 5-colorable.

9.6.3 Problems

Class Problems

Problem 9.26.
Let G be the graph below[6]. Carefully explain why $\chi(G) = 4$.



[6]From *Discrete Mathematics*, Lovász, Pelikan, and Vesztergombi. Springer, 2003. Exercise 13.3.1

Homework Problems

Problem 9.27.
6.042 is often taught using recitations. Suppose it happened that 8 recitations were needed, with two or three staff members running each recitation. The assignment of staff to recitation sections is as follows:

- R1: Eli, Megumi, Rich
- R2: Eli, Stephanie, David
- R3: Megumi, Stav
- R4: Liz, Stephanie, Oscar
- R5: Liz, Tom, David
- R6: Tom, Stav
- R7: Tom, Stephanie
- R8: Megumi, Stav, David

Two recitations can not be held in the same 90-minute time slot if some staff member is assigned to both recitations. The problem is to determine the minimum number of time slots required to complete all the recitations.

(a) Recast this problem as a question about coloring the vertices of a particular graph. Draw the graph and explain what the vertices, edges, and colors represent.

(b) Show a coloring of this graph using the fewest possible colors. What schedule of recitations does this imply?

Problem 9.28.
This problem generalizes the result proved Theorem 9.6.2 that any graph with maximum degree at most w is $(w + 1)$-colorable.

A simple graph, G, is said to have *width*, w, iff its vertices can be arranged in a sequence such that each vertex is adjacent to at most w vertices that precede it in the sequence. If the degree of every vertex is at most w, then the graph obviously has width at most w —just list the vertices in any order.

(a) Describe an example of a graph with 100 vertices, width 3, but *average* degree more than 5. *Hint:* Don't get stuck on this; if you don't see it after five minutes, ask for a hint.

(b) Prove that every graph with width at most w is $(w+1)$-colorable.

(c) Prove that the average degree of a graph of width w is at most $2w$.

Exam Problems

Problem 9.29.
Recall that a *coloring* of a graph is an assignment of a color to each vertex such that no two adjacent vertices have the same color. A k-*coloring* is a coloring that uses at most k colors.

False Claim. *Let G be a graph whose vertex degrees are all $\le k$. If G has a vertex of degree strictly less than k, then G is k-colorable.*

(a) Give a counterexample to the False Claim when $k = 2$.

(b) Underline the exact sentence or part of a sentence that is the first unjustified step in the following "proof" of the False Claim.

False proof. Proof by induction on the number n of vertices:

Induction hypothesis:

$P(n)$::= "Let G be an n-vertex graph whose vertex degrees are all $\le k$. If G also has a vertex of degree strictly less than k, then G is k-colorable."

Base case: $(n = 1)$ G has one vertex, the degree of which is 0. Since G is 1-colorable, $P(1)$ holds.

Inductive step:

We may assume $P(n)$. To prove $P(n+1)$, let G_{n+1} be a graph with $n+1$ vertices whose vertex degrees are all k or less. Also, suppose G_{n+1} has a vertex, v, of degree strictly less than k. Now we only need to prove that G_{n+1} is k-colorable.

To do this, first remove the vertex v to produce a graph, G_n, with n vertices. Let u be a vertex that is adjacent to v in G_{n+1}. Removing v reduces the degree of u by 1. So in G_n, vertex u has degree strictly less than k. Since no edges were added, the vertex degrees of G_n remain $\le k$. So G_n satisfies the conditions of the induction hypothesis, $P(n)$, and so we conclude that G_n is k-colorable.

Now a k-coloring of G_n gives a coloring of all the vertices of G_{n+1}, except for v. Since v has degree less than k, there will be fewer than k colors assigned to the nodes adjacent to v. So among the k possible colors, there will be a color not used to color these adjacent nodes, and this color can be assigned to v to form a k-coloring of G_{n+1}. ■

(c) With a slightly strengthened condition, the preceding proof of the False Claim could be revised into a sound proof of the following Claim:

Claim. *Let G be a graph whose vertex degrees are all $\leq k$. If* $\langle$statement inserted from below$\rangle$ *has a vertex of degree strictly less than k, then G is k-colorable.*

Circle each of the statements below that could be inserted to make the Claim true.

- G is connected and
- G has no vertex of degree zero and
- G does not contain a complete graph on k vertices and
- every connected component of G
- some connected component of G

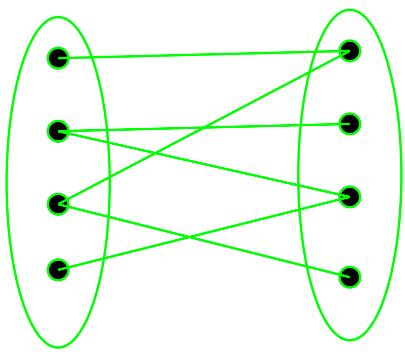
9.7 Bipartite Matchings

9.7.1 Bipartite Graphs

There were two kinds of vertices in the "Sex in America" graph —males and females, and edges only went between the two kinds. Graphs like this come up so frequently they have earned a special name —they are called *bipartite graphs*.

Definition 9.7.1. A *bipartite graph* is a graph together with a partition of its vertices into two sets, L and R, such that every edge is incident to a vertex in L and to a vertex in R.

So every bipartite graph looks something like this:



Now we can immediately see how to color a bipartite graph using only two colors: let all the L vertices be black and all the R vertices be white. Conversely, if a graph is 2-colorable, then it is bipartite with L being the vertices of one color and R the vertices of the other color. In other words, "bipartite" is a synonym for "*2-colorable*."

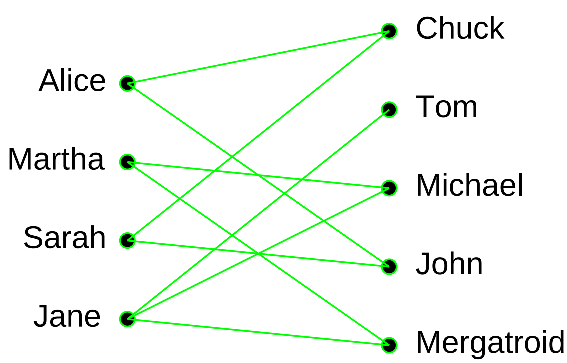
The following Lemma gives another useful characterization of bipartite graphs.

Theorem 9.7.2. *A graph is bipartite iff it has no odd-length cycle.*

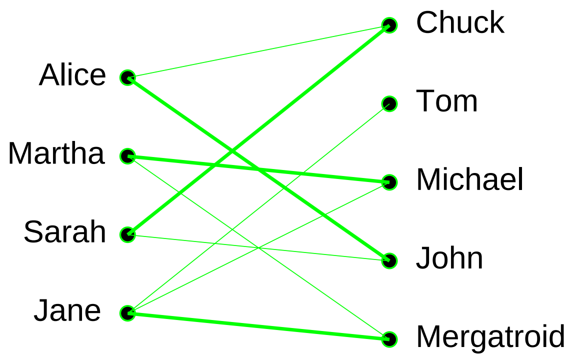
The proof of Theorem 9.7.2 is left to Problem 9.33.

9.7.2 Bipartite Matchings

The *bipartite matching* problem resembles the stable Marriage Problem in that it concerns a set of girls and a set of at least as many boys. There are no preference lists, but each girl does have some boys she likes and others she does not like. In the bipartite matching problem, we ask whether every girl can be paired up with a boy that she likes. Any particular matching problem can be specified by a bipartite graph with a vertex for each girl, a vertex for each boy, and an edge between a boy and a girl iff the girl likes the boy. For example, we might obtain the following graph:



Now a *matching* will mean a way of assigning every girl to a boy so that different girls are assigned to different boys, and a girl is always assigned to a boy she likes. For example, here is one possible matching for the girls:



Hall's Matching Theorem states necessary and sufficient conditions for the existence of a matching in a bipartite graph. It turns out to be a remarkably useful mathematical tool.

9.7.3 The Matching Condition

We'll state and prove Hall's Theorem using girl-likes-boy terminology. Define *the set of boys liked by a given set of girls* to consist of all boys liked by at least one of those girls. For example, the set of boys liked by Martha and Jane consists of Tom, Michael, and Mergatroid. For us to have any chance at all of matching up the girls, the following *matching condition* must hold:

Every subset of girls likes at least as large a set of boys.

For example, we can not find a matching if some 4 girls like only 3 boys. Hall's Theorem says that this necessary condition is actually sufficient; if the matching condition holds, then a matching exists.

Theorem 9.7.3. *A matching for a set of girls G with a set of boys B can be found if and only if the matching condition holds.*

Proof. First, let's suppose that a matching exists and show that the matching condition holds. Consider an arbitrary subset of girls. Each girl likes at least the boy she is matched with. Therefore, every subset of girls likes at least as large a set of boys. Thus, the matching condition holds.

Next, let's suppose that the matching condition holds and show that a matching exists. We use strong induction on $|G|$, the number of girls.

Base Case: ($|G| = 1$) If $|G| = 1$, then the matching condition implies that the lone girl likes at least one boy, and so a matching exists.

Inductive Step: Now suppose that $|G| \geq 2$. There are two cases:

Case 1: Every proper subset of girls likes a *strictly larger* set of boys. In this case, we have some latitude: we pair an arbitrary girl with a boy she likes and send them both away. The matching condition still holds for the remaining boys and girls, so we can match the rest of the girls by induction.

Case 2: Some proper subset of girls $X \subset G$ likes an *equal-size* set of boys $Y \subset B$. We match the girls in X with the boys in Y by induction and send them all away. We can also match the rest of the girls by induction if we show that the matching condition holds for the remaining boys and girls. To check the matching condition for the remaining people, consider an arbitrary subset of the remaining girls $X' \subseteq (G - X)$, and let Y' be the set of remaining boys that they like. We must show that $|X'| \leq |Y'|$. Originally, the combined set of girls $X \cup X'$ liked the set of boys $Y \cup Y'$. So, by the matching condition, we know:

$$|X \cup X'| \leq |Y \cup Y'|$$

We sent away $|X|$ girls from the set on the left (leaving X') and sent away an equal number of boys from the set on the right (leaving Y'). Therefore, it must be that $|X'| \leq |Y'|$ as claimed.

So there is in any case a matching for the girls, which completes the proof of the Inductive step. The theorem follows by induction. ■

The proof of this theorem gives an algorithm for finding a matching in a bipartite graph, albeit not a very efficient one. However, efficient algorithms for finding a matching in a bipartite graph do exist. Thus, if a problem can be reduced to finding a matching, the problem is essentially solved from a computational perspective.

9.7.4 A Formal Statement

Let's restate Hall's Theorem in abstract terms so that you'll not always be condemned to saying, "Now this group of little girls likes at least as many little boys..."

A *matching* in a graph, G, is a set of edges such that no two edges in the set share a vertex. A matching is said to *cover* a set, L, of vertices iff each vertex in L has an edge of the matching incident to it. In any graph, the set $N(S)$, of *neighbors*[7] of some set, S, of vertices is the set of all vertices adjacent to some vertex in S. That is,

$$N(S) ::= \{r \mid s\text{—}r \text{ is an edge for some } s \in S\}.$$

S is called a *bottleneck* if

$$|S| > |N(S)|.$$

Theorem 9.7.4 (Hall's Theorem)**.** *Let G be a bipartite graph with vertex partition L, R. There is matching in G that covers L iff no subset of L is a bottleneck.*

[7]An equivalent definition of $N(S)$ uses relational notation: $N(S)$ is simply the image, SR, of S under the adjacency relation, R, on vertices of the graph.

An Easy Matching Condition

The bipartite matching condition requires that *every* subset of girls has a certain property. In general, verifying that every subset has some property, even if it's easy to check any particular subset for the property, quickly becomes overwhelming because the number of subsets of even relatively small sets is enormous —over a billion subsets for a set of size 30. However, there is a simple property of vertex degrees in a bipartite graph that guarantees a match and is very easy to check. Namely, call a bipartite graph *degree-constrained* if vertex degrees on the left are at least as large as those on the right. More precisely,

Definition 9.7.5. A bipartite graph G with vertex partition L, R is *degree-constrained* if $\deg(l) \geq \deg(r)$ for every $l \in L$ and $r \in R$.

Now we can always find a matching in a degree-constrained bipartite graph.

Lemma 9.7.6. *Every degree-constrained bipartite graph satisifies the matching condition.*

Proof. Let S be any set of vertices in L. The number of edges incident to vertices in S is exactly the sum of the degrees of the vertices in S. Each of these edges is incident to a vertex in $N(S)$ by definition of $N(S)$. So the sum of the degrees of the vertices in $N(S)$ is at least as large as the sum for S. But since the degree of every vertex in $N(S)$ is at most as large as the degree of every vertex in S, there would have to be at least as many terms in the sum for $N(S)$ as in the sum for S. So there have to be at least as many vertices in $N(S)$ as in S, proving that S is not a bottleneck. So there are no bottlenecks, proving that the degree-constrained graph satisifies the matching condition. ■

Of course being degree-constrained is a very strong property, and lots of graphs that aren't degree-constrained have matchings. But we'll see examples of degree-constrained graphs come up naturally in some later applications.

9.7.5 Problems

Class Problems

Problem 9.30.
MIT has a lot of student clubs loosely overseen by the MIT Student Association. Each eligible club would like to delegate one of its members to appeal to the Dean for funding, but the Dean will not allow a student to be the delegate of more than one club. Fortunately, the Association VP took 6.042 and recognizes a matching problem when she sees one.

(a) Explain how to model the delegate selection problem as a bipartite matching problem.

(b) The VP's records show that no student is a member of more than 9 clubs. The VP also knows that to be eligible for support from the Dean's office, a club must have at least 13 members. That's enough for her to guarantee there is a proper delegate selection. Explain. (If only the VP had taken 6.046, *Algorithms*, she could even have found a delegate selection without much effort.)

Problem 9.31.
A *Latin square* is $n \times n$ array whose entries are the number $1, \ldots, n$. These entries satisfy two constraints: every row contains all n integers in some order, and also every column contains all n integers in some order. Latin squares come up frequently in the design of scientific experiments for reasons illustrated by a little story in a footnote[8]

For example, here is a 4×4 Latin square:

1	2	3	4
3	4	2	1
2	1	4	3
4	3	1	2

(a) Here are three rows of what could be part of a 5×5 Latin square:

2	4	5	3	1
4	1	3	2	5
3	2	1	5	4

Fill in the last two rows to extend this "Latin rectangle" to a complete Latin square.

[8]At Guinness brewery in the eary 1900's, W. S. Gosset (a chemist) and E. S. Beavan (a "maltster") were trying to improve the barley used to make the brew. The brewery used different varieties of barley according to price and availability, and their agricultural consultants suggested a different fertilizer mix and best planting month for each variety.

Somewhat sceptical about paying high prices for customized fertilizer, Gosset and Beavan planned a season long test of the influence of fertilizer and planting month on barley yields. For as many months as there were varieties of barley, they would plant one sample of each variety using a different one of the fertilizers. So every month, they would have all the barley varieties planted and all the fertilizers used, which would give them a way to judge the overall quality of that planting month. But they also wanted to judge the fertilizers, so they wanted each fertilizer to be used on each variety during the course of the season. Now they had a little mathematical problem, which we can abstract as follows.

Suppose there are n barley varieties and an equal number of recommended fertilizers. Form an $n \times n$ array with a column for each fertilizer and a row for each planting month. We want to fill in the entries of this array with the integers 1,…,n numbering the barley varieties, so that every row contains all n integers in some order (so every month each variety is planted and each fertilizer is used), and also every column contains all n integers (so each fertilizer is used on all the varieties over the course of the growing season).

(b) Show that filling in the next row of an $n \times n$ Latin rectangle is equivalent to finding a matching in some $2n$-vertex bipartite graph.

(c) Prove that a matching must exist in this bipartite graph and, consequently, a Latin rectangle can always be extended to a Latin square.

Exam Problems

Problem 9.32.
Overworked and over-caffeinated, the TAs decide to oust Albert and teach their own recitations. They will run a recitation session at 4 different times in the same room. There are exactly 20 chairs to which a student can be assigned in each recitation. Each student has provided the TAs with a list of the recitation sessions her schedule allows and no student's schedule conflicts with all 4 sessions. The TAs must assign each student to a chair during recitation at a time she can attend, if such an assignment is possible.

(a) Describe how to model this situation as a matching problem. Be sure to specify what the vertices/edges should be and briefly describe how a matching would determine seat assignments for each student in a recitation that does not conflict with his schedule. (This is a *modeling problem*; we aren't looking for a description of an algorithm to solve the problem.)

(b) Suppose there are 65 students. Given the information provided above, is a matching guaranteed? Briefly explain.

Homework Problems

Problem 9.33.
In this problem you will prove:

Theorem. *A graph G is 2-colorable iff it contains no odd length cycle.*

As usual with "iff" assertions, the proof splits into two proofs: part (a) asks you to prove that the left side of the "iff" implies the right side. The other problem parts prove that the right side implies the left.

(a) Assume the left side and prove the right side. Three to five sentences should suffice.

(b) Now assume the right side. As a first step toward proving the left side, explain why we can focus on a single connected component H within G.

(c) As a second step, explain how to 2-color any tree.

(d) Choose any 2-coloring of a spanning tree, T, of H. Prove that H is 2-colorable by showing that any edge *not* in T must also connect different-colored vertices.

Problem 9.34.
Take a regular deck of 52 cards. Each card has a suit and a value. The suit is one of four possibilities: heart, diamond, club, spade. The value is one of 13 possibilities, $A, 2, 3, \dots, 10, J, Q, K$. There is exactly one card for each of the 4×13 possible combinations of suit and value.

Ask your friend to lay the cards out into a grid with 4 rows and 13 columns. They can fill the cards in any way they'd like. In this problem you will show that you can always pick out 13 cards, one from each column of the grid, so that you wind up with cards of all 13 possible values.

(a) Explain how to model this trick as a bipartite matching problem between the 13 column vertices and the 13 value vertices. Is the graph necessarily degree constrained?

(b) Show that any n columns must contain at least n different values and prove that a matching must exist.

Problem 9.35.
Scholars through the ages have identified *twenty* fundamental human virtues: honesty, generosity, loyalty, prudence, completing the weekly course reading-response, etc. At the beginning of the term, every student in 6.042 possessed exactly *eight* of these virtues. Furthermore, every student was unique; that is, no two students possessed exactly the same set of virtues. The 6.042 course staff must select *one* additional virtue to impart to each student by the end of the term. Prove that there is a way to select an additional virtue for each student so that every student is unique at the end of the term as well.

Suggestion: Use Hall's theorem. Try various interpretations for the vertices on the left and right sides of your bipartite graph.

Chapter 10

Recursive Data Types

Recursive data types play a central role in programming. From a mathematical point of view, recursive data types are what induction is about. Recursive data types are specified by *recursive definitions* that say how to build something from its parts. These definitions have two parts:

- **Base case(s)** that don't depend on anything else.
- **Constructor case(s)** that depend on previous cases.

10.1 Strings of Brackets

Let `brkts` be the set of all strings of square brackets. For example, the following two strings are in `brkts`:

$$[\,]\,]\,[\,[\,[\,[\,[\,]\,] \quad \text{and} \quad [\,[\,[\,]\,]\,[\,]\,]\,[\,] \tag{10.1}$$

Since we're just starting to study recursive data, just for practice we'll formulate `brkts` as a recursive data type,

Definition 10.1.1. The data type, `brkts`, of strings of brackets is defined recursively:

- **Base case:** The *empty string*, λ, is in `brkts`.
- **Constructor case:** If $s \in$ `brkts`, then $s]$ and $s[$ are in `brkts`.

Here we're writing $s]$ to indicate the string that is the sequence of brackets (if any) in the string s, followed by a right bracket; similarly for $s[$.

A string, $s \in$ `brkts`, is called a *matched string* if its brackets "match up" in the usual way. For example, the left hand string above is not matched because its second right bracket does not have a matching left bracket. The string on the right is matched.

We're going to examine several different ways to define and prove properties of matched strings using recursively defined sets and functions. These properties are pretty straighforward, and you might wonder whether they have any particular relevance in computer scientist —other than as a nonnumerical example of recursion. The honest answer is "not much relevance, *any more*." The reason for this is one of the great successes of computer science.

Expression Parsing

During the early development of computer science in the 1950's and 60's, creation of effective programming language compilers was a central concern. A key aspect in processing a program for compilation was expression parsing. The problem was to take in an expression like

$$x + y * z^2 \div y + 7$$

and *put in* the brackets that determined how it should be evaluated —should it be

$$[[x + y] * z^2 \div y] + 7, \text{ or,}$$
$$x + [y * z^2 \div [y + 7]], \text{ or,}$$
$$[x + [y * z^2]] \div [y + 7],$$

or ...?

The Turing award (the "Nobel Prize" of computer science) was ultimately bestowed on Robert Floyd, for, among other things, being discoverer of a simple program that would insert the brackets properly.

In the 70's and 80's, this parsing technology was packaged into high-level compiler-compilers that automatically generated parsers from expression grammars. This automation of parsing was so effective that the subject needed no longer demanded attention. It largely disappeared from the computer science curriculum by the 1990's.

One precise way to determine if a string is matched is to start with 0 and read the string from left to right, adding 1 to the count for each left bracket and subtracting 1 from the count for each right bracket. For example, here are the counts for the two strings above

$$\begin{array}{ccccccccccccc} & [&] &] & [& [& [& [& [&] &] &] &] \\ 0 & 1 & 0 & -1 & 0 & 1 & 2 & 3 & 4 & 3 & 2 & 1 & 0 \end{array}$$

$$\begin{array}{ccccccccccc} & [& [& [&] &] & [&] &] & [&] \\ 0 & 1 & 2 & 3 & 2 & 1 & 2 & 1 & 0 & 1 & 0 \end{array}$$

A string has a *good count* if its running count never goes negative and ends with 0. So the second string above has a good count, but the first one does not because its count went negative at the third step.

Definition 10.1.2. Let

$$\text{GoodCount} ::= \{s \in \texttt{brkts} \mid s \text{ has a good count}\}.$$

The matched strings can now be characterized precisely as this set of strings with good counts. But it turns out to be really useful to characterize the matched strings in another way as well, namely, as a recursive data type:

Definition 10.1.3. Recursively define the set, RecMatch, of strings as follows:

- **Base case:** $\lambda \in \text{RecMatch}$.
- **Constructor case:** If $s, t \in \text{RecMatch}$, then

$$[\, s\,]\, t \in \text{RecMatch}.$$

Here we're writing $[\, s\,]\, t$ to indicate the string that starts with a left bracket, followed by the sequence of brackets (if any) in the string s, followed by a right bracket, and ending with the sequence of brackets in the string t.

Using this definition, we can see that $\lambda \in \text{RecMatch}$ by the Base case, so

$$[\,\lambda]\,\lambda = [\,] \in \text{RecMatch}$$

by the Constructor case. So now,

$$\begin{aligned} [\,\lambda]\,[\,] &= [\,]\,[\,] \in \text{RecMatch} && (\text{letting } s = \lambda, t = [\,]) \\ [\,[\,]\,]\,\lambda &= [\,[\,]\,] \in \text{RecMatch} && (\text{letting } s = [\,], t = \lambda) \\ & [\,[\,]\,]\,[\,] \in \text{RecMatch} && (\text{letting } s = [\,], t = [\,]) \end{aligned}$$

are also strings in RecMatch by repeated applications of the Constructor case. If you haven't seen this kind of definition before, you should try continuing this example to verify that $[\,[\,[\,]\,]\,[\,]\,]\,[\,] \in \text{RecMatch}$

Given the way this section is set up, you might guess that RecMatch = GoodCount, and you'd be right, but it's not completely obvious. The proof is worked out in Problem 10.6.

10.2 Arithmetic Expressions

Expression evaluation is a key feature of programming languages, and recognition of expressions as a recursive data type is a key to understanding how they can be processed.

To illustrate this approach we'll work with a toy example: arithmetic expressions like $3x^2 + 2x + 1$ involving only one variable, "x." We'll refer to the data type of such expressions as Aexp. Here is its definition:

Definition 10.2.1.
- **Base cases:**

1. The variable, x, is in Aexp.
2. The arabic numeral, $\mathtt{k}$, for any nonnegative integer, k, is in Aexp.

- **Constructor cases:** If $e, f \in$ Aexp, then

3. $(e + f) \in$ Aexp. The expression $(e + f)$ is called a *sum*. The Aexp's e and f are called the *components* of the sum; they're also called the *summands*.
4. $(e * f) \in$ Aexp. The expression $(e * f)$ is called a *product*. The Aexp's e and f are called the *components* of the product; they're also called the *multiplier* and *multiplicand*.
5. $-(e) \in$ Aexp. The expression $-(e)$ is called a *negative*.

Notice that Aexp's are fully parenthesized, and exponents aren't allowed. So the Aexp version of the polynomial expression $3x^2 + 2x + 1$ would officially be written as

$$((\mathtt{3} * (x * x)) + ((\mathtt{2} * x) + \mathtt{1})). \tag{10.2}$$

These parentheses and $*$'s clutter up examples, so we'll often use simpler expressions like "$3x^2 + 2x + 1$" instead of (10.2). But it's important to recognize that $3x^2 + 2x + 1$ is not an Aexp; it's an *abbreviation* for an Aexp.

10.3 Structural Induction on Recursive Data Types

Structural induction is a method for proving some property, P, of all the elements of a recursively-defined data type. The proof consists of two steps:

- Prove P for the base cases of the definition.
- Prove P for the constructor cases of the definition, assuming that it is true for the component data items.

A very simple application of structural induction proves that the recursively defined matched strings always have an equal number of left and right brackets. To do this, define a predicate, P, on strings $s \in \mathtt{brkts}$:

$$P(s) ::= \quad s \text{ has an equal number of left and right brackets.}$$

Proof. We'll prove that $P(s)$ holds for all $s \in$ RecMatch by structural induction on the definition that $s \in$ RecMatch, using $P(s)$ as the induction hypothesis.

Base case: $P(\lambda)$ holds because the empty string has zero left and zero right brackets.

Constructor case: For $r = [\ s\ ]\ t$, we must show that $P(r)$ holds, given that $P(s)$ and $P(t)$ holds. So let n_s, n_t be, respectively, the number of left brackets in s and t. So the number of left brackets in r is $1 + n_s + n_t$.

Now from the respective hypotheses $P(s)$ and $P(t)$, we know that the number of right brackets in s is n_s, and likewise, the number of right brackets in t is n_t. So the number of right brackets in r is $1 + n_s + n_t$, which is the same as the number of left brackets. This proves $P(r)$. We conclude by structural induction that $P(s)$ holds for all $s \in$ RecMatch. ■

10.3.1 Functions on Recursively-defined Data Types

Functions on recursively-defined data types can be defined recursively using the same cases as the data type definition. Namely, to define a function, f, on a recursive data type, define the value of f for the base cases of the data type definition, and then define the value of f in each constructor case in terms of the values of f on the component data items.

For example, from the recursive definition of the set, RecMatch, of strings of matched brackets, we define:

Definition 10.3.1. The *depth*, $d(s)$, of a string, $s \in$ RecMatch, is defined recursively by the rules:

- $d(\lambda) ::= 0.$
- $d([\ s\ ]\ t) ::= \max \{d(s) + 1, d(t)\}$

Warning: When a recursive definition of a data type allows the same element to be constructed in more than one way, the definition is said to be *ambiguous*. A function defined recursively from an ambiguous definition of a data type will not be well-defined unless the values specified for the different ways of constructing the element agree.

We were careful to choose an *un*ambiguous definition of RecMatch to ensure that functions defined recursively on the definition would always be well-defined. As an example of the trouble an ambiguous definition can cause, let's consider yet another definition of the matched strings.

Example 10.3.2. Define the set, $M \subseteq \texttt{brkts}$ recursively as follows:

- **Base case:** $\lambda \in M$,
- **Constructor cases:** if $s, t \in M$, then the strings $[\ s\ ]$ and st are also in M.

Quick Exercise: Give an easy proof by structural induction that $M = \text{RecMatch}$.

Since $M = \text{RecMatch}$, and the definition of M seems more straightforward, why didn't we use it? Because the definition of M is ambiguous, while the trickier definition of RecMatch is unambiguous. Does this ambiguity matter? Yes it does. For suppose we defined

$$\begin{aligned} f(\lambda) &::= 1, \\ f(\,\mathbf{[}\, s \,\mathbf{]}\,) &::= 1 + f(s), \\ f(st) &::= (f(s)+1) \cdot (f(t)+1) \qquad\qquad \text{for } st \neq \lambda. \end{aligned}$$

Let a be the string $\mathbf{[\,[\,]\,]} \in M$ built by two successive applications of the first M constructor starting with λ. Next let $b ::= aa$ and $c ::= bb$, each built by successive applications of the second M constructor starting with a.

Alternatively, we can build ba from the second constructor with $s = b$ and $t = a$, and then get to c using the second constructor with $s = ba$ and $t = a$.

Now by these rules, $f(a) = 2$, and $f(b) = (2 + 1)(2 + 1) = 9$. This means that $f(c) = f(bb) = (9 + 1)(9 + 1) = 100$.

But also $f(ba) = (9+1)(2+1) = 27$, so that $f(c) = f(ba\,a) = (27+1)(2+1) = 84$.

The outcome is that $f(c)$ is defined to be both 100 and 84, which shows that the rules defining f are inconsistent.

On the other hand, structural induction remains a sound proof method even for ambiguous recursive definitions, which is why it was easy to prove that $M = \text{RecMatch}$.

10.3.2 Recursive Functions on Nonnegative Integers

The nonnegative integers can be understood as a recursive data type.

Definition 10.3.3. The set, $\mathbb{N}$, is a data type defined recursivly as:

- $0 \in \mathbb{N}$.
- If $n \in \mathbb{N}$, then the *successor*, $n + 1$, of n is in $\mathbb{N}$.

This of course makes it clear that ordinary induction is simply the special case of structural induction on the recursive Definition 10.3.3, This also justifies the familiar recursive definitions of functions on the nonnegative integers. Here are some examples.

The Factorial function. This function is often written "$n!$." You will see a lot of it later in the term. Here we'll use the notation $\text{fac}(n)$:

- $\text{fac}(0) ::= 1$.
- $\text{fac}(n + 1) ::= (n + 1) \cdot \text{fac}(n)$ for $n \geq 0$.

The Fibonacci numbers. Fibonacci numbers arose out of an effort 800 years ago to model population growth. They have a continuing fan club of people captivated by their extraordinary properties. The nth Fibonacci number, fib, can be defined recursively by:

$$\begin{aligned}
\text{fib}(0) &::= 0, \\
\text{fib}(1) &::= 1, \\
\text{fib}(n) &::= \text{fib}(n-1) + \text{fib}(n-2) && \text{for } n \geq 2.
\end{aligned}$$

Here the recursive step starts at $n = 2$ with base cases for 0 and 1. This is needed since the recursion relies on two previous values.

What is $\text{fib}(4)$? Well, $\text{fib}(2) = \text{fib}(1) + \text{fib}(0) = 1$, $\text{fib}(3) = \text{fib}(2) + \text{fib}(1) = 2$, so $\text{fib}(4) = 3$. The sequence starts out $0, 1, 1, 2, 3, 5, 8, 13, 21, \ldots$.

Sum-notation. Let "$S(n)$" abbreviate the expression "$\sum_{i=1}^{n} f(i)$." We can recursively define $S(n)$ with the rules

- $S(0) ::= 0$.
- $S(n+1) ::= f(n+1) + S(n)$ for $n \geq 0$.

Ill-formed Function Definitions

There are some blunders to watch out for when defining functions recursively. Below are some function specifications that resemble good definitions of functions on the nonnegative integers, but they aren't.

$$f_1(n) ::= 2 + f_1(n-1). \tag{10.3}$$

This "definition" has no base case. If some function, f_1, satisfied (10.3), so would a function obtained by adding a constant to the value of f_1. So equation (10.3) does not uniquely define an f_1.

$$f_2(n) ::= \begin{cases} 0, & \text{if } n = 0, \\ f_2(n+1) & \text{otherwise.} \end{cases} \tag{10.4}$$

This "definition" has a base case, but still doesn't uniquely determine f_2. Any function that is 0 at 0 and constant everywhere else would satisfy the specification, so (10.4) also does not uniquely define anything.

In a typical programming language, evaluation of $f_2(1)$ would begin with a recursive call of $f_2(2)$, which would lead to a recursive call of $f_2(3)$, ... with recursive calls continuing without end. This "operational" approach interprets (10.4) as defining a *partial* function, f_2, that is undefined everywhere but 0.

$$f_3(n) ::= \begin{cases} 0, & \text{if } n \text{ is divisible by 2,} \\ 1, & \text{if } n \text{ is divisible by 3,} \\ 2, & \text{otherwise.} \end{cases} \tag{10.5}$$

This "definition" is inconsistent: it requires $f_3(6) = 0$ and $f_3(6) = 1$, so (10.5) doesn't define anything.

A Mysterious Function

Mathematicians have been wondering about this function specification for a while:

$$f_4(n) ::= \begin{cases} 1, & \text{if } n \leq 1, \\ f_4(n/2) & \text{if } n > 1 \text{ is even,} \\ f_4(3n+1) & \text{if } n > 1 \text{ is odd.} \end{cases} \tag{10.6}$$

For example, $f_4(3) = 1$ because

$$f_4(3) ::= f_4(10) ::= f_4(5) ::= f_4(16) ::= f_4(8) ::= f_4(4) ::= f_4(2) ::= f_4(1) ::= 1.$$

The constant function equal to 1 will satisfy (10.6), but it's not known if another function does too. The problem is that the third case specifies $f_4(n)$ in terms of f_4 at arguments larger than n, and so cannot be justified by induction on $\mathbb{N}$. It's known that any f_4 satisfying (10.6) equals 1 for all n up to over a billion.

Quick exercise: Why does the constant function 1 satisfy (10.6)?

10.3.3 Evaluation and Substitution with Aexp's

Evaluating Aexp's

Since the only variable in an Aexp is x, the value of an Aexp is determined by the value of x. For example, if the value of x is 3, then the value of $3x^2 + 2x + 1$ is obviously 34. In general, given any Aexp, e, and an integer value, n, for the variable, x, we can evaluate e to finds its value, $\text{eval}(e, n)$. It's easy, and useful, to specify this evaluation process with a recursive definition.

Definition 10.3.4. The *evaluation function*, $\text{eval} : \text{Aexp} \times \mathbb{Z} \to \mathbb{Z}$, is defined recursively on expressions, $e \in \text{Aexp}$, as follows. Let n be any integer.

- **Base cases:**

  1. Case[e is x]

  $$\text{eval}(x, n) ::= n.$$

  (The value of the variable, x, is given to be n.)

2. Case[e is k]
$$\text{eval}(\texttt{k}, n) ::= k.$$
(The value of the numeral k is the integer k, no matter what value x has.)

- **Constructor cases:**

3. Case[e is $(e_1 + e_2)$]
$$\text{eval}((e_1 + e_2), n) ::= \text{eval}(e_1, n) + \text{eval}(e_2, n).$$

4. Case[e is $(e_1 * e_2)$]
$$\text{eval}((e_1 * e_2), n) ::= \text{eval}(e_1, n) \cdot \text{eval}(e_2, n).$$

5. Case[e is $-(e_1)$]
$$\text{eval}(-(e_1), n) ::= -\text{eval}(e_1, n).$$

For example, here's how the recursive definition of eval would arrive at the value of $3 + x^2$ when x is 2:

$$\begin{aligned}
\text{eval}((\texttt{3} + (x * x)), 2) &= \text{eval}(\texttt{3}, 2) + \text{eval}((x * x), 2) && \text{(by Def 10.3.4.3)} \\
&= 3 + \text{eval}((x * x), 2) && \text{(by Def 10.3.4.2)} \\
&= 3 + (\text{eval}(x, 2) \cdot \text{eval}(x, 2)) && \text{(by Def 10.3.4.4)} \\
&= 3 + (2 \cdot 2) && \text{(by Def 10.3.4.1)} \\
&= 3 + 4 = 7.
\end{aligned}$$

Substituting into Aexp's

Substituting expressions for variables is a standard, important operation. For example the result of substituting the expression $3x$ for x in the $(x(x-1))$ would be $(3x(3x-1)$. We'll use the general notation $\text{subst}(f, e)$ for the result of substituting an Aexp, f, for each of the x's in an Aexp, e. For instance,

$$\text{subst}(3x, x(x-1)) = 3x(3x-1).$$

This substitution function has a simple recursive definition:

Definition 10.3.5. The *substitution function* from Aexp $\times$ Aexp to Aexp is defined recursively on expressions, $e \in$ Aexp, as follows. Let f be any Aexp.

- **Base cases:**

1. Case[e is x]
$$\text{subst}(f, x) ::= f.$$
(The result of substituting f for the variable, x, is just f.)

2. Case[e is $\texttt{k}$]
$$\text{subst}(f, \texttt{k}) ::= \texttt{k}.$$
(The numeral, $\texttt{k}$, has no x's in it to substitute for.)

- **Constructor cases:**

3. Case[e is $(e_1 + e_2)$]
$$\text{subst}(f, (e_1 + e_2))) ::= (\text{subst}(f, e_1) + \text{subst}(f, e_2)).$$

4. Case[e is $(e_1 * e_2)$]
$$\text{subst}(f, (e_1 * e_2))) ::= (\text{subst}(f, e_1) * \text{subst}(f, e_2)).$$

5. Case[e is $-(e_1)$]
$$\text{subst}(f, -(e_1)) ::= -(\text{subst}(f, e_1)).$$

Here's how the recursive definition of the substitution function would find the result of substituting $3x$ for x in the $x(x-1)$:

$$\begin{aligned}
\text{subst}(3x, (x(x-1))) &= \text{subst}(3x, (x * (x + -(1)))) && \text{(unabbreviating)} \\
&= (\text{subst}(3x, x) * \text{subst}(3x, (x + -(1)))) && \text{(by Def 10.3.5 4)} \\
&= (3x * \text{subst}(3x, (x + -(1)))) && \text{(by Def 10.3.5 1)} \\
&= (3x * (\text{subst}(3x, x) + \text{subst}(3x, -(1)))) && \text{(by Def 10.3.5 3)} \\
&= (3x * (3x + -(\text{subst}(3x, 1)))) && \text{(by Def 10.3.5 1 \& 5)} \\
&= (3x * (3x + -(1))) && \text{(by Def 10.3.5 2)} \\
&= 3x(3x-1) && \text{(abbreviation)}
\end{aligned}$$

Now suppose we have to find the value of $\text{subst}(3x, (x(x-1)))$ when $x = 2$. There are two approaches.

First, we could actually do the substitution above to get $3x(3x-1)$, and then we could evaluate $3x(3x-1)$ when $x = 2$, that is, we could recursively calculate $\text{eval}(3x(3x-1), 2)$ to get the final value 30. In programming jargon, this would be called evaluation using the *Substitution Model*. Tracing through the steps in the evaluation, we find that the Substitution Model requires two substitutions for occurrences of x and 5 integer operations: 3 integer multiplications, 1 integer addition, and 1 integer negative operation. Note that in this Substitution Model the multiplication $3 \cdot 2$ was performed twice to get the value of 6 for each of the two occurrences of $3x$.

The other approach is called evaluation using the *Environment Model*. Namely, we evaluate $3x$ when $x = 2$ using just 1 multiplication to get the value 6. Then we evaluate $x(x-1)$ when x has this value 6 to arrive at the value $6 \cdot 5 = 30$. So the Environment Model requires 2 variable lookups and only 4 integer operations: 1

multiplication to find the value of $3x$, another multiplication to find the value $6 \cdot 5$, along with 1 integer addition and 1 integer negative operation.

So the Environment Model approach of calculating

$$\text{eval}(x(x-1), \text{eval}(3x, 2))$$

instead of the Substitution Model approach of calculating

$$\text{eval}(\text{subst}(3x, x(x-1)), 2)$$

is faster. But how do we know that these final values reached by these two approaches always agree? We can prove this easily by structural induction on the definitions of the two approaches. More precisely, what we want to prove is

Theorem 10.3.6. *For all expressions $e, f \in$ Aexp and $n \in \mathbb{Z}$,*

$$\text{eval}(\text{subst}(f, e), n) = \text{eval}(e, \text{eval}(f, n)). \tag{10.7}$$

Proof. The proof is by structural induction on e.[1]

Base cases:

- Case[e is x]

  The left hand side of equation (10.7) equals $\text{eval}(f, n)$ by this base case in Definition 10.3.5 of the substitution function, and the right hand side also equals $\text{eval}(f, n)$ by this base case in Definition 10.3.4 of eval.

- Case[e is $\mathtt{k}$].

  The left hand side of equation (10.7) equals $\mathtt{k}$ by this base case in Definitions 10.3.5 and 10.3.4 of the substitution and evaluation functions. Likewise, the right hand side equals $\mathtt{k}$ by two applications of this base case in the Definition 10.3.4 of eval.

Constructor cases:

- Case[e is $(e_1 + e_2)$]

  By the structural induction hypothesis (10.7), we may assume that for all $f \in$ Aexp and $n \in \mathbb{Z}$,

$$\text{eval}(\text{subst}(f, e_i), n) = \text{eval}(e_i, \text{eval}(f, n)) \tag{10.8}$$

  for $i = 1, 2$. We wish to prove that

$$\text{eval}(\text{subst}(f, (e_1 + e_2)), n) = \text{eval}((e_1 + e_2), \text{eval}(f, n)) \tag{10.9}$$

[1] This is an example of why it's useful to notify the reader what the induction variable is—in this case it isn't n.

But the left hand side of (10.9) equals

$$\text{eval}(\ (\text{subst}(f, e_1) + \text{subst}(f, e_2)),\ n)$$

by Definition 10.3.5.3 of substitution into a sum expression. But this equals

$$\text{eval}(\text{subst}(f, e_1), n) + \text{eval}(\text{subst}(f, e_2), n)$$

by Definition 10.3.4.3 of eval for a sum expression. By induction hypothesis (10.8), this in turn equals

$$\text{eval}(e_1, \text{eval}(f, n)) + \text{eval}(e_2, \text{eval}(f, n)).$$

Finally, this last expression equals the right hand side of (10.9) by Definition 10.3.4.3 of eval for a sum expression. This proves (10.9) in this case.

- e is $(e_1 * e_2)$. Similar.
- e is $-(e_1)$. Even easier.

This covers all the constructor cases, and so completes the proof by structural induction.

■

10.3.4 Problems

Practice Problems

Problem 10.1.

Definition. Consider a new recursive definition, MB_0, of the same set of "matching" brackets strings as MB (definition of MB is provided in the Appendix):

- **Base case:** $\lambda \in \text{MB}_0$.
- **Constructor cases:**

  (i) If s is in MB_0, then $[s]$ is in MB_0.

  (ii) If $s, t \in \text{MB}_0$, $s \neq \lambda$, and $t \neq \lambda$, then st is in MB_0.

(a) Suppose structural induction was being used to prove that $\text{MB}_0 \subseteq \text{MB}$. Circle the one predicate below that would fit the format for a structural induction hypothesis in such a proof.

- $P_0(n) ::= |s| \leq n$ IMPLIES $s \in \text{MB}$.
- $P_1(n) ::= |s| \leq n$ IMPLIES $s \in \text{MB}_0$.
- $P_2(s) ::= s \in \text{MB}$.

- $P_3(s) ::= s \in \text{MB}_0$.
- $P_4(s) ::= \; (s \in \text{MB IMPLIES } s \in \text{MB}_0)$.

(b) The recursive definition MB_0 is *ambiguous*. Verify this by giving two different derivations for the string "[ ][ ][ ]" according to MB_0.

Class Problems

Problem 10.2.
The Elementary 18.01 Functions (F18's) are the set of functions of one real variable defined recursively as follows:

Base cases:

- The identity function, $\text{id}(x) ::= x$ is an F18,
- any constant function is an F18,
- the sine function is an F18,

Constructor cases:
If f, g are F18's, then so are

1. $f + g$, fg, e^g (the constant e),
2. the inverse function $f^{(-1)}$,
3. the composition $f \circ g$.

(a) Prove that the function $1/x$ is an F18.

Warning: Don't confuse $1/x = x^{-1}$ with the inverse, $\text{id}^{(-1)}$ of the identity function $\text{id}(x)$. The inverse $\text{id}^{(-1)}$ is equal to id.

(b) Prove by Structural Induction on this definition that the Elementary 18.01 Functions are *closed under taking derivatives*. That is, show that if $f(x)$ is an F18, then so is $f' ::= df/dx$. (Just work out 2 or 3 of the most interesting constructor cases; you may skip the less interesting ones.)

Problem 10.3.
Here is a simple recursive definition of the set, E, of even integers:

Definition. Base case: $0 \in E$.
Constructor cases: If $n \in E$, then so are $n + 2$ and $-n$.

Provide similar simple recursive definitions of the following sets:

(a) The set $S ::= \{2^k 3^m 5^n \mid k, m, n \in \mathbb{N}\}$.

(b) The set $T ::= \{2^k 3^{2k+m} 5^{m+n} \mid k, m, n \in \mathbb{N}\}$.

(c) The set $L ::= \{(a, b) \in \mathbb{Z}^2 \mid 3 \mid (a - b)\}$.

Let L' be the set defined by the recursive definition you gave for L in the previous part. Now if you did it right, then $L' = L$, but maybe you made a mistake. So let's check that you got the definition right.

(d) Prove by structural induction on your definition of L' that

$$L' \subseteq L.$$

(e) Confirm that you got the definition right by proving that

$$L \subseteq L'.$$

(f) See if you can give an *unambiguous* recursive definition of L.

Problem 10.4.
Let p be the string [ ] . A string of brackets is said to be *erasable* iff it can be reduced to the empty string by repeatedly erasing occurrences of p. For example, here's how to erase the string [[[ ]] [ ]] [ ] :

[[[ ]] [ ]] [ ] $\rightarrow$ [[ ]] $\rightarrow$ [ ] $\rightarrow \lambda$.

On the other hand the string [ ]] [[[[[ ]] is not erasable because when we try to erase, we get stuck:

[ ]] [[[[[ ]] $\rightarrow$] [[[[ ] $\rightarrow$] [[[$\not\rightarrow$

Let Erasable be the set of erasable strings of brackets. Let RecMatch be the recursive data type of strings of *matched* brackets given in Definition 10.3.7.

(a) Use structural induction to prove that

$$\text{RecMatch} \subseteq \text{Erasable}.$$

(b) Supply the missing parts of the following proof that

$$\text{Erasable} \subseteq \text{RecMatch}.$$

Proof. We prove by induction on the length, n, of strings, x, that if $x \in \text{Erasable}$, then $x \in \text{RecMatch}$. The induction predicate is

$$P(n) ::= \forall x \in \text{Erasable}. \left(|x| \leq n \text{ IMPLIES } x \in \text{RecMatch}\right)$$

Base case:

What is the base case? Prove that P is true in this case.

Inductive step: To prove $P(n+1)$, suppose $|x| \leq n+1$ and $x \in \text{Erasable}$. We need only show that $x \in \text{RecMatch}$. Now if $|x| < n + 1$, then the induction hypothesis,

$P(n)$, implies that $x \in$ RecMatch, so we only have to deal with x of length exactly $n+1$.

Let's say that a string y is an *erase* of a string z iff y is the result of erasing a single occurrence of p in z.

Since $x \in$ Erasable and has positive length, there must be an erase, $y \in$ Erasable, of x. So $|y| = n-1$, and since $y \in$ Erasable, we may assume by induction hypothesis that $y \in$ RecMatch.

Now we argue by cases:

Case (y is the empty string).

Prove that $x \in$ RecMatch in this case.

Case ($y = [\ s\ ]\ t$ for some strings $s, t \in$ RecMatch.) Now we argue by subcases.

- **Subcase** (x is of the form $[\ s'\ ]\ t$ where s is an erase of s').
  Since $s \in$ RecMatch, it is erasable by part (b), which implies that $s' \in$ Erasable. But $|s'| < |x|$, so by induction hypothesis, we may assume that $s' \in$ RecMatch. This shows that x is the result of the constructor step of RecMatch, and therefore $x \in$ RecMatch.
- **Subcase** (x is of the form $[\ s\ ]\ t'$ where t is an erase of t').
  Prove that $x \in$ RecMatch in this subcase.
- **Subcase**($x = p[\ s\ ]\ t$).
  Prove that $x \in$ RecMatch in this subcase.

The proofs of the remaining subcases are just like this last one. **List these remaining subcases.**

This completes the proof by induction on n, so we conclude that $P(n)$ holds for all $n \in \mathbb{N}$. Therefore $x \in$ RecMatch for every string $x \in$ Erasable. That is,

$$\text{Erasable} \subseteq \text{RecMatch and hence Erasable} = \text{RecMatch}.$$

■

Problem 10.5.

Definition. The recursive data type, binary-2PTG, of *binary trees* with leaf labels, L, is defined recursively as follows:

- **Base case:** $\langle \texttt{leaf}, l \rangle \in$ binary-2PTG, for all labels $l \in L$.
- **Constructor case:** If $G_1, G_2 \in$ binary-2PTG, then

$$\langle \texttt{bintree}, G_1, G_2 \rangle \in \text{binary-2PTG}.$$

The *size*, $|G|$, of $G \in$ binary-2PTG is defined recursively on this definition by:

- **Base case:**
$$|\langle \texttt{leaf}, l \rangle| ::= 1, \quad \text{for all } l \in L.$$

- **Constructor case:**
$$|\langle \texttt{bintree}, G_1, G_2 \rangle| ::= |G_1| + |G_2| + 1.$$

For example, for the size of the binary-2PTG, G, pictured in Figure 10.1, is 7.

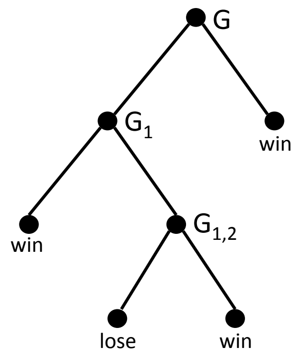


Figure 10.1: *A picture of a binary tree w.*

(a) Write out (using angle brackets and labels `bintree`, `leaf`, etc.) the binary-2PTG, G, pictured in Figure 10.1.

The value of flatten(G) for $G \in$ binary-2PTG is the sequence of labels in L of the leaves of G. For example, for the binary-2PTG, G, pictured in Figure 10.1,

$$\text{flatten}(G) = (\texttt{win}, \texttt{lose}, \texttt{win}, \texttt{win}).$$

(b) Give a recursive definition of flatten. (You may use the operation of *concatenation* (append) of two sequences.)

(c) Prove by structural induction on the definitions of flatten and size that

$$2 \cdot \text{length}(\text{flatten}(G)) = |G| + 1. \tag{10.10}$$

Homework Problems

Problem 10.6.

Definition 10.3.7. The set, RecMatch, of strings of matching brackets, is defined recursively as follows:

- **Base case:** $\lambda \in$ RecMatch.
- **Constructor case:** If $s, t \in$ RecMatch, then

$$[\, s \,]\, t \in \text{RecMatch}.$$

There is a simple test to determine whether a string of brackets is in RecMatch: starting with zero, read the string from left to right adding one for each left bracket and -1 for each right bracket. A string has a *good count* when the count never goes negative and is back to zero by the end of the string. Let GoodCount be the bracket strings with good counts.

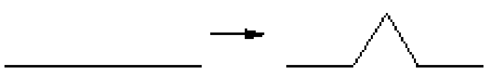
(a) Prove that GoodCount contains RecMatch by structural induction on the definition of RecMatch.

(b) Conversely, prove that RecMatch contains GoodCount.

Problem 10.7.

Fractals are example of a mathematical object that can be defined recursively. In this problem, we consider the Koch snowflake. Any Koch snowflake can be constructed by the following recursive definition.

- Base Case: An equilateral triangle with a positive integer side length is a Koch snowflake.
- Recursive case: Let K be a Koch snowflake, and let l be a line segment on the snowflake. Remove the middle third of l, and replace it with two line segments of the same length as is done below:



The resulting figure is also a Koch snowflake.

Prove by structural induction that the area inside any Koch snowflake is of the form $q\sqrt{3}$, where q is a rational number.

10.4 Games as a Recursive Data Type

Chess, Checkers, and Tic-Tac-Toe are examples of *two-person terminating games of perfect information*, —2PTG's for short. These are games in which two players alternate moves that depend only on the visible board position or state of the game. "Perfect information" means that the players know the complete state of the game at each move. (Most card games are *not* games of perfect information because neither player can see the other's hand.) "Terminating" means that play cannot go on forever —it must end after a finite number of moves.[2]

We will define 2PTG's as a recursive data type. To see how this will work, let's use the game of Tic-Tac-Toe as an example.

10.4.1 Tic-Tac-Toe

Tic-Tac-Toe is a game for young children. There are two players who alternately write the letters "X" and "O" in the empty boxes of a 3×3 grid. Three copies of the same letter filling a row, column, or diagonal of the grid is called a *tic-tac-toe*, and the first player who gets a tic-tac-toe of their letter wins the game.

We're now going give a precise mathematical definition of the Tic-Tac-Toe *game tree* as a recursive data type.

Here's the idea behind the definition: at any point in the game, the "board position" is the pattern of X's and O's on the 3×3 grid. From any such Tic-Tac-Toe pattern, there are a number of next patterns that might result from a move. For example, from the initial empty grid, there are nine possible next patterns, each with a single X in some grid cell and the other eight cells empty. From any of these patterns, there are eight possible next patterns gotten by placing an O in an empty cell. These move possibilities are given by the game tree for Tic-Tac-Toe indicated in Figure 10.2.

Definition 10.4.1. A Tic-Tac-Toe *pattern* is a 3×3 grid each of whose 9 cells contains either the single letter, X, the single letter, O, or is empty.

A pattern, Q, is a *possible next pattern after P*, providing P has no tic-tac-toes and

- if P has an equal number of X's and O's, and Q is the same as P except that a cell that was empty in P has an X in Q, or
- if P has one more X than O's, and Q is the same as P except that a cell that was empty in P has an O in Q.

If P is a Tic-Tac-Toe pattern, and P has no next patterns, then the *terminated Tic-Tac-Toe game trees* at P are

- $\langle P, \langle \texttt{win} \rangle \rangle$, if P has a tic-tac-toe of X's.

[2] Since board positions can repeat in chess and checkers, termination is enforced by rules that prevent any position from being repeated more than a fixed number of times. So the "state" of these games is the board position *plus* a record of how many times positions have been reached.

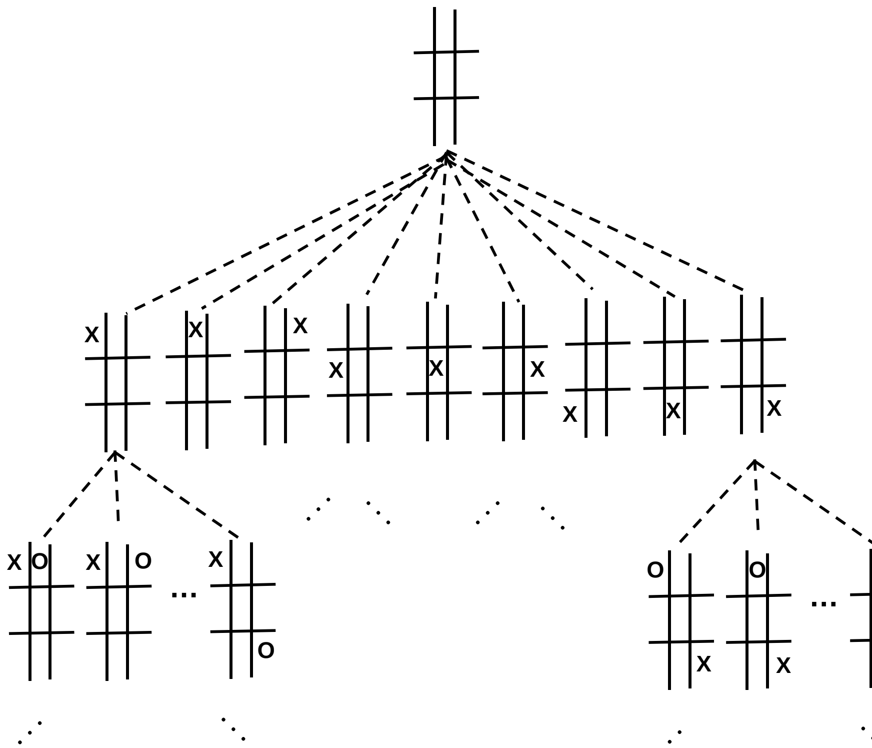


Figure 10.2: The Top of the Game Tree for Tic-Tac-Toe.

- $\langle P, \langle \texttt{lose} \rangle \rangle$, if P has a tic-tac-toe of O's.
- $\langle P, \langle \texttt{tie} \rangle \rangle$, otherwise.

The *Tic-Tac-Toe game trees starting at P* are defined recursively:

Base Case: A terminated Tic-Tac-Toe game tree at P is a Tic-Tac-Toe game tree starting at P.

Constructor case: If P is a non-terminated Tic-Tac-Toe pattern, then the Tic-Tac-Toe game tree starting at P consists of P and the set of all game trees starting at possible next patterns after P.

For example, if

$$P_0 = \begin{array}{c|c|c} O & X & O \\ \hline X & O & X \\ \hline X & & \end{array}$$

$$Q_1 = \begin{array}{c|c|c} O & X & O \\ \hline X & O & X \\ \hline X & & O \end{array}$$

$$Q_2 = \begin{array}{c|c|c} O & X & O \\ \hline X & O & X \\ \hline X & O & \end{array}$$

$$R = \begin{array}{c|c|c} O & X & O \\ \hline X & O & X \\ \hline X & O & X \end{array}$$

the game tree starting at P_0 is pictured in Figure 10.3.

Game trees are usually pictured in this way with the starting pattern (referred to as the "root" of the tree) at the top and lines connecting the root to the game trees that start at each possible next pattern. The "leaves" at the bottom of the tree (trees grow upside down in computer science) correspond to terminated games. A path from the root to a leaf describes a complete *play* of the game. (In English, "game" can be used in two senses: first we can say that Chess is a game, and second we can play a game of Chess. The first usage refers to the data type of Chess game trees, and the second usage refers to a "play.")

10.4.2 Infinite Tic-Tac-Toe Games

At any point in a Tic-Tac-Toe game, there are at most nine possible next patterns, and no play can continue for more than nine moves. But we can expand Tic-Tac-Toe into a larger game by running a 5-game tournament: play Tic-Tac-Toe five times and the tournament winner is the player who wins the most individual games. A 5-game tournament can run for as many as 45 moves.

It's not much of generalization to have an *n-game Tic-Tac-Toe tournament*. But then comes a generalization that sounds simple but can be mind-boggling: consolidate all these different size tournaments into a single game we can call *Tournament-Tic-Tac-Toe (T^4)*. The first player in a game of T^4 chooses any integer $n > 0$. Then

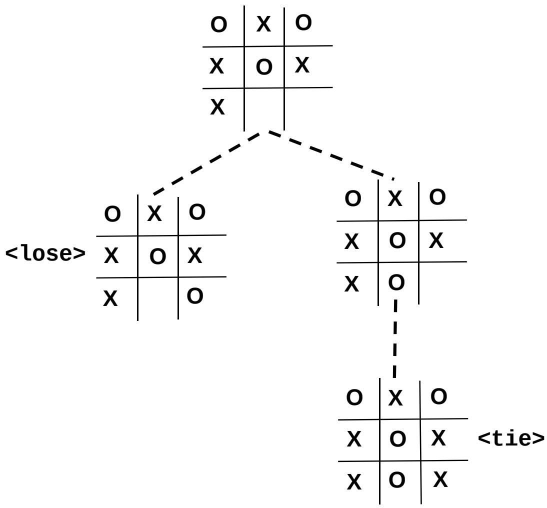


Figure 10.3: Game Tree for the Tic-Tac-Toe game starting at P_0.

the players play an n-game tournament. Now we can no longer say how long a T^4 play can take. In fact, there are T^4 plays that last as long as you might like: if you want a game that has a play with, say, nine billion moves, just have the first player choose n equal to one billion. This should make it clear the game tree for T^4 is infinite.

But still, it's obvious that every possible T^4 play will stop. That's because after the first player chooses a value for n, the game can't continue for more than $9n$ moves. So it's not possible to keep playing forever even though the game tree is infinite.

This isn't very hard to understand, but there is an important difference between any given n-game tournament and T^4: even though every play of T^4 must come to an end, there is no longer any initial bound on how many moves it might be before the game ends —a play might end after 9 moves, or $9(2001)$ moves, or $9(10^{10}+1)$ moves. It just can't continue forever.

Now that we recognize T^4 as a 2PTG, we can go on to a *meta-*T^4 game, where the first player chooses a number, $m > 0$, of T^4 games to play, and then the second player gets the first move in each of the individual T^4 games to be played.

Then, of course, there's meta-meta-T^4....

10.4.3 Two Person Terminating Games

Familiar games like Tic-Tac-Toe, Checkers, and Chess can all end in ties, but for simplicity we'll only consider win/lose games —no "everybody wins"-type games at MIT. :-) But everything we show about win/lose games will extend easily to games with ties, and more generally to games with outcomes that have different payoffs.

Like Tic-Tac-Toe, or Tournament-Tic-Tac-Toe, the idea behind the definition of 2PTG's as a recursive data type is that making a move in a 2PTG leads to the start of a subgame. In other words, given any set of games, we can make a new game whose first move is to pick a game to play from the set.

So what defines a game? For Tic-Tac-Toe, we used the patterns and the rules of Tic-Tac-Toe to determine the next patterns. But once we have a complete game tree, we don't really need the pattern labels: the root of a game tree itself can play the role of a "board position" with its possible "next positions" determined by the roots of its subtrees. So any game is defined by its game tree. This leads to the following very simple —perhaps deceptively simple —general definition.

Definition 10.4.2. The 2PTG, *game trees for two-person terminating games of perfect information* are defined recursively as follows:

- **Base cases:**
$$\begin{aligned}\langle \texttt{leaf}, \texttt{win}\rangle &\in \text{2PTG, and}\\ \langle \texttt{leaf}, \texttt{lose}\rangle &\in \text{2PTG.}\end{aligned}$$

- **Constructor case:** If $\mathcal{G}$ is a nonempty set of 2PTG's, then G is a 2PTG, where
$$G ::= \langle \texttt{tree}, \mathcal{G}\rangle .$$

The game trees in $\mathcal{G}$ are called the possible *next moves* from G.

These games are called "terminating" because, even though a 2PTG may be a (very) infinite datum like Tournament^2-Tic-Tac-Toe, every play of a 2PTG must terminate. This is something we can now prove, after we give a precise definition of "play":

Definition 10.4.3. A *play* of a 2PTG, G, is a (potentially infinite) sequence of 2PTG's starting with G and such that if G_1 and G_2 are consecutive 2PTG's in the play, then G_2 is a possible next move of G_1.

If a 2PTG has no infinite play, it is called a *terminating* game.

Theorem 10.4.4. *Every 2PTG is terminating.*

Proof. By structural induction on the definition of a 2PTG, G, with induction hypothesis

$$G \text{ is terminating.}$$

Base case: If $G = \langle \texttt{leaf}, \texttt{win} \rangle$ or $G = \langle \texttt{leaf}, \texttt{lose} \rangle$ then the only possible play of G is the length one sequence consisting of G. Hence G terminates.

Constructor case: For $G = \langle \texttt{tree}, \mathcal{G} \rangle$, we must show that G is terminating, given the Induction Hypothesis that *every* $G' \in \mathcal{G}$ is terminating.

But any play of G is, by definition, a sequence starting with G and followed by a play starting with some $G_0 \in \mathcal{G}$. But G_0 is terminating, so the play starting at G_0 is finite, and hence so is the play starting at G.

This completes the structural induction, proving that every 2PTG, G, is terminating. ■

10.4.4 Game Strategies

A key question about a game is whether a player has a winning strategy. A *strategy* for a player in a game specifies which move the player should make at any point in the game. A *winning* strategy ensures that the player will win no matter what moves the other player makes.

In Tic-Tac-Toe for example, most elementary school children figure out strategies for both players that each ensure that the game ends with no tic-tac-toes, that is, it ends in a tie. Of course the first player can win if his opponent plays childishly, but not if the second player follows the proper strategy. In more complicated games like Checkers or Chess, it's not immediately clear that anyone has a winning strategy, even if we agreed to count ties as wins for the second player.

But structural induction makes it easy to prove that in any 2PTG, *somebody* has the winning strategy!

Theorem 10.4.5. ***Fundamental Theorem for Two-Person Games:*** *For every two-person terminating game of perfect information, there is a winning strategy for one of the players.*

Proof. The proof is by structural induction on the definition of a 2PTG, G. The induction hypothesis is that there is a winning strategy for G.

Base cases:

1. $G = \langle \texttt{leaf}, \texttt{win} \rangle$. Then the first player has the winning strategy: "make the winning move."

2. $G = \langle \texttt{leaf}, \texttt{lose} \rangle$. Then the second player has a winning strategy: "Let the first player make the losing move."

Constructor case: Suppose $G = \langle \texttt{tree}, \mathcal{G} \rangle$. By structural induction, we may assume that some player has a winning strategy for each $G' \in \mathcal{G}$. There are two cases to consider:

- some $G_0 \in \mathcal{G}$ has a winning strategy for its second player. Then the first player in G has a winning strategy: make the move to G_0 and then follow the second player's winning strategy in G_0.

- every $G' \in \mathcal{G}$ has a winning strategy for its first player. Then the second player in G has a winning strategy: if the first player's move in G is to $G_0 \in \mathcal{G}$, then follow the winning strategy for the first player in G_0.

So in any case, one of the players has a winning strategy for G, which completes the proof of the constructor case.

It follows by structural induction that there is a winning strategy for every 2PTG, G. ∎

Notice that although Theorem 10.4.5 guarantees a winning strategy, its proof gives no clue which player has it. For most familiar 2PTG's like Chess, Go, ..., no one knows which player has a winning strategy.[3]

10.4.5 Problems

Homework Problems

Problem 10.8.

Define *2-person 50-point games of perfect information*50-PG's, recursively as follows:

Base case: An integer, k, is a 50-PG for $-50 \leq k \leq 50$. This 50-PG called the *terminated game with payoff* k. A *play* of this 50-PG is the length one integer sequence, k.

Constructor case: If $G_0, \ldots, G_n$ is a finite sequence of 50-PG's for some $n \in \mathbb{N}$, then the following game, G, is a 50-PG: the possible first moves in G are the choice of an integer i between 0 and n, the possible second moves in G are the possible first moves in G_i, and the rest of the game G proceeds as in G_i.

[3] Checkers used to be in this list, but there has been a recent announcement that each player has a strategy that forces a tie. (reference TBA)

A *play* of the 50-PG, G, is a sequence of nonnegative integers starting with a possible move, i, of G, followed by a play of G_i. If the play ends at the game terminated game, k, then k is called the *payoff* of the play.

There are two players in a 50-PG who make moves alternately. The objective of one player (call him the *max*-player) is to have the play end with as high a payoff as possible, and the other player (called the *min*-player) aims to have play end with as low a payoff as possible.

Given which of the players moves first in a game, a strategy for the max-player is said to *ensure* the payoff, k, if play ends with a payoff of at least k, no matter what moves the min-player makes. Likewise, a strategy for the min-player is said to *hold down* the payoff to k, if play ends with a payoff of at most k, no matter what moves the max-player makes.

A 50-PG is said to have *max value*, k, if the max-player has a strategy that ensures payoff k, and the min-player has a strategy that holds down the payoff to k, when the *max-player moves first*. Likewise, the 50-PG has *min value*, k, if the max-player has a strategy that ensures k, and the min-player has a strategy that holds down the payoff to k, when the *min-player moves first*.

The *Fundamental Theorem* for 2-person 50-point games of perfect information is that is that every game has both a max value and a min value. (Note: the two values are usually different.)

What this means is that there's no point in playing a game: if the max player gets the first move, the min-player should just pay the max-player the max value of the game without bothering to play (a negative payment means the max-player is paying the min-player). Likewise, if the min-player gets the first move, the min-player should just pay the max-player the min value of the game.

(a) Prove this Fundamental Theorem for 50-valued 50-PG's by structural induction.

(b) A meta-50-PG game has as possible first moves the choice of *any* 50-PG to play. Meta-50-PG games aren't any harder to understand than 50-PG's, but there is one notable difference, they have an infinite number of possible first moves. We could also define meta-meta-50-PG's in which the first move was a choice of any 50-PG *or* the meta-50-PG game to play. In meta-meta-50-PG's there are an infinite number of possible first *and* second moves. And then there's meta$^3 -$ 50-PG . . . .

To model such infinite games, we could have modified the recursive definition of 50-PG's to allow first moves that choose any one of an infinite sequence

$$G_0, G_1, \ldots, G_n, G_{n+1}, \ldots$$

of 50-PG's. Now a 50-PG can be a mind-bendingly infinite datum instead of a finite one.

Do these infinite 50-PG's still have max and min values? In particular, do you think it would be correct to use structural induction as in part (a) to prove a Fundamental Theorem for such infinite 50-PG's? Offer an answer to this question, and briefly indicate why you believe in it.

10.5 Induction in Computer Science

Induction is a powerful and widely applicable proof technique, which is why we've devoted two entire chapters to it. Strong induction and its special case of ordinary induction are applicable to any kind of thing with nonnegative integer sizes –which is a awful lot of things, including all step-by-step computational processes.

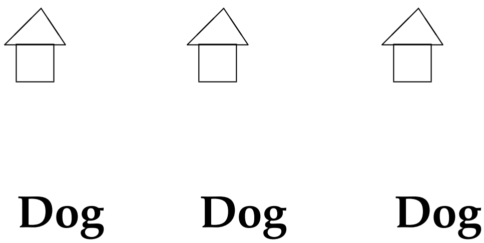
Structural induction then goes beyond natural number counting by offering a simple, natural approach to proving things about recursive computation and recursive data types. This makes it a technique every computer scientist should embrace.

Chapter 11

Planar Graphs

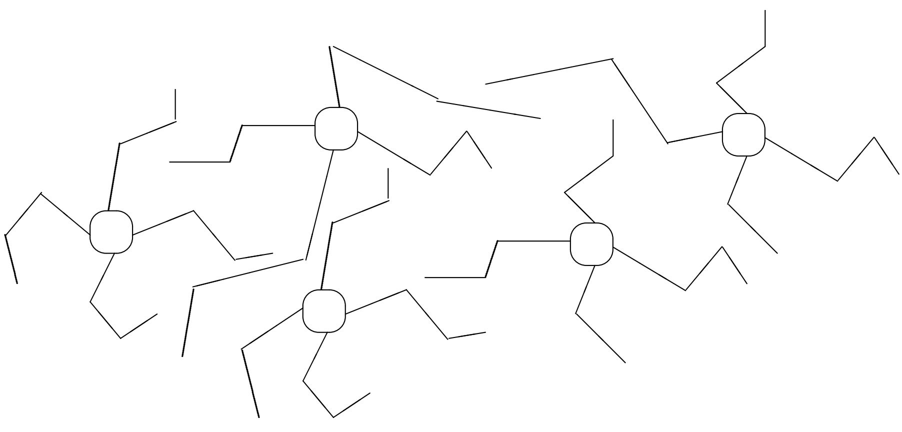
11.1 Drawing Graphs in the Plane

Here are three dogs and three houses.



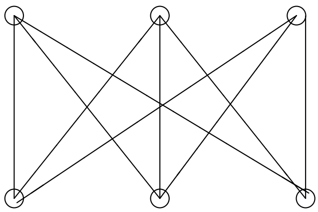
Can you find a path from each dog to each house such that no two paths intersect?

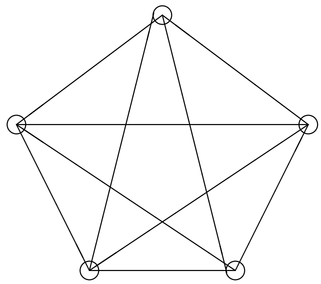
A *quadapus* is a little-known animal similar to an octopus, but with four arms. Here are five quadapi resting on the seafloor:



Can each quadapus simultaneously shake hands with every other in such a way that no arms cross?

Informally, a *planar graph* is a graph that can be drawn in the plane so that no edges cross, as in a map of showing the borders of countries or states. Thus, these two puzzles are asking whether the graphs below are planar; that is, whether they can be redrawn so that no edges cross. The first graph is called the *complete bipartite graph*, $K_{3,3}$, and the second is K_5.





In each case, the answer is, "No— but almost!" In fact, each drawing *would* be possible if any single edge were removed.

Planar graphs have applications in circuit layout and are helpful in displaying graphical data, for example, program flow charts, organizational charts, and scheduling conflicts. We will treat them as a recursive data type and use structural induction to establish their basic properties. Then we'll be able to describe a simple recursive procedure to color any planar graph with *five* colors, and also prove that there is no uniform way to place n satellites around the globe unless $n = 4, 6, 8, 12$, or 20.

When wires are arranged on a surface, like a circuit board or microchip, crossings require troublesome three-dimensional structures. When Steve Wozniak designed the disk drive for the early Apple II computer, he struggled mightly to achieve a nearly planar design:

> For two weeks, he worked late each night to make a satisfactory design. When he was finished, he found that if he moved a connector he could cut down on feedthroughs, making the board more reliable. To make that move, however, he had to start over in his design. This time it only took twenty hours. He then saw another feedthrough that could be eliminated, and again started over on his design. "The final design was generally recognized by computer engineers as brilliant and was by engineering aesthetics beautiful. Woz later said, 'It's something you can only do if you're the engineer and the PC board layout person yourself. That was an artistic layout. The board has virtually no feedthroughs.'"[a]

[a]From apple2history.org which in turn quotes *Fire in the Valley* by Freiberger and Swaine.

11.2 Continuous & Discrete Faces

Planar graphs are graphs that can be drawn in the plane —like familiar maps of countries or states. "Drawing" the graph means that each vertex of the graph corresponds to a distinct point in the plane, and if two vertices are adjacent, their vertices are connected by a smooth, non-self-intersecting curve. None of the curves may "cross" —the only points that may appear on more than one curve are the vertex points. These curves are the boundaries of connected regions of the plane called the *continuous faces* of the drawing.

For example, the drawing in Figure 11.1 has four continuous faces. Face IV, which extends off to infinity in all directions, is called the *outside face*.

This definition of planar graphs is perfectly precise, but completely unsatisfying: it invokes smooth curves and continuous regions of the plane to define a property of a discrete data type. So the first thing we'd like to find is a discrete data type that represents planar drawings.

The clue to how to do this is to notice that the vertices along the boundary of each of the faces in Figure 11.1 form a simple cycle. For example, labeling the vertices as in Figure 11.2, the simple cycles for the face boundaries are

`abca     abda     bcdb     acda.`

Since every edge in the drawing appears on the boundaries of exactly two continuous faces, every edge of the simple graph appears on exactly two of the simple cycles.

Vertices around the boundaries of states and countries in an ordinary map are

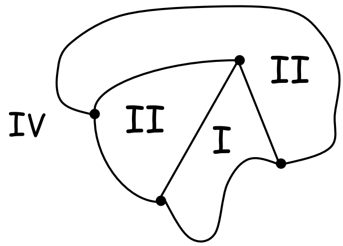


Figure 11.1: A Planar Drawing with Four Faces.

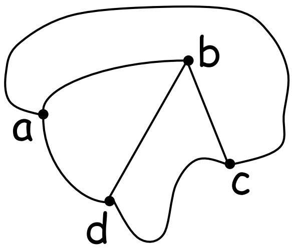


Figure 11.2: The Drawing with Labelled Vertices.

always simple cycles, but oceans are slightly messier. The ocean boundary is the set of all boundaries of islands and continents in the ocean; it is a *set* of simple cycles (this can happen for countries too —like Bangladesh). But this happens because islands (and the two parts of Bangladesh) are not connected to each other. So we can dispose of this complication by treating each connected component separately.

But general planar graphs, even when they are connected, may be a bit more complicated than maps. For example a planar graph may have a "bridge," as in Figure 11.3. Now the cycle around the outer face is

$$\texttt{abcefgecda}.$$

This is not a simple cycle, since it has to traverse the bridge c—e twice.

Planar graphs may also have "dongles," as in Figure 11.4. Now the cycle around the inner face is

$$\texttt{rstvxyxvwvtur},$$

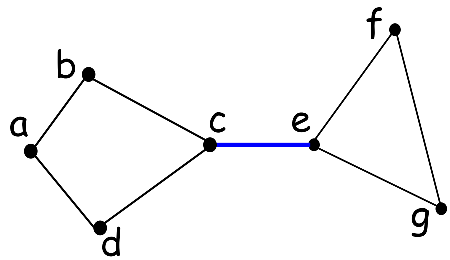


Figure 11.3: A Planar Drawing with a *Bridge*.

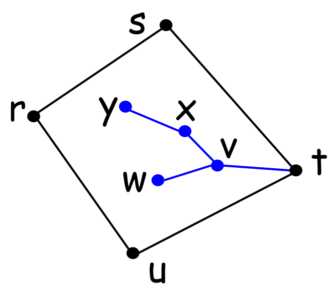


Figure 11.4: A Planar Drawing with a *Dongle*.

because it has to traverse *every* edge of the dongle twice —once "coming" and once "going."

But bridges and dongles are really the only complications, which leads us to the discrete data type of *planar embeddings* that we can use in place of continuous planar drawings. Namely, we'll define a planar embedding recursively to be the set of boundary-tracing cycles we could get drawing one edge after another.

11.3 Planar Embeddings

By thinking of the process of drawing a planar graph edge by edge, we can give a useful recursive definition of planar embeddings.

Definition 11.3.1. A *planar embedding* of a *connected* graph consists of a nonempty set of cycles of the graph called the *discrete faces* of the embedding. Planar embeddings are defined recursively as follows:

- **Base case:** If G is a graph consisting of a single vertex, v, then a planar embedding of G has one discrete face, namely the length zero cycle, v.

- **Constructor Case:** (split a face) Suppose G is a connected graph with a planar embedding, and suppose a and b are distinct, nonadjacent vertices of G that appear on some discrete face, γ, of the planar embedding. That is, γ is a cycle of the form

  $$a \ldots b \cdots a.$$

  Then the graph obtained by adding the edge a—b to the edges of G has a planar embedding with the same discrete faces as G, except that face γ is replaced by the two discrete faces[1]

  $$a \ldots ba \quad \text{and} \quad ab \cdots a,$$

  as illustrated in Figure 11.5.

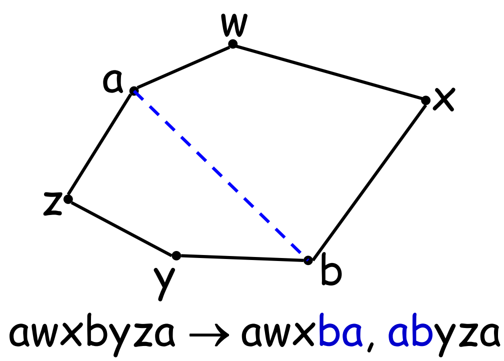


Figure 11.5: The Split a Face Case.

- **Constructor Case:** (add a bridge) Suppose G and H are connected graphs with planar embeddings and disjoint sets of vertices. Let a be a vertex on a discrete face, γ, in the embedding of G. That is, γ is of the form

  $$a \ldots a.$$

[1] There is one exception to this rule. If G is a line graph beginning with a and ending with b, then the cycles into which γ splits are actually the same. That's because adding edge a—b creates a simple cycle graph, C_n, that divides the plane into an "inner" and an "outer" region with the same border. In order to maintain the correspondence between continuous faces and discrete faces, we have to allow two "copies" of this same cycle to count as discrete faces. But since this is the only situation in which two faces are actually the same cycle, this exception is better explained in a footnote than mentioned explicitly in the definition.

Similarly, let b be a vertex on a discrete face, δ, in the embedding of H, so δ is of the form

$$b \cdots b.$$

Then the graph obtained by connecting G and H with a new edge, a—b, has a planar embedding whose discrete faces are the union of the discrete faces of G and H, except that faces γ and δ are replaced by one new face

$$a \ldots ab \cdots ba.$$

This is illustrated in Figure 11.6, where the faces of G and H are:

G : {axyza, axya, ayza} H : {btuvwb, btvwb, tuvt},

and after adding the bridge a—b, there is a single connected graph with faces

{axyzabtuvwba, axya, ayza, btvwb, tuvt}.

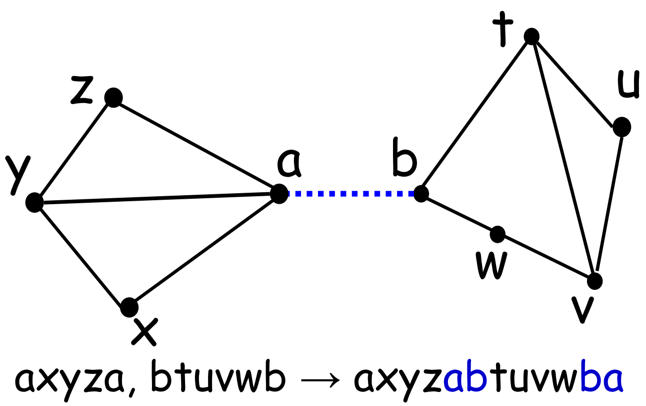


Figure 11.6: The Add Bridge Case.

An arbitrary graph is *planar* iff each of its connected components has a planar embedding.

11.4 What outer face?

Notice that the definition of planar embedding does not distinguish an "outer" face. There really isn't any need to distinguish one.

In fact, a planar embedding could be drawn with any given face on the outside. An intuitive explanation of this is to think of drawing the embedding on a *sphere* instead of the plane. Then any face can be made the outside face by "puncturing" that face of the sphere, stretching the puncture hole to a circle around the rest of the faces, and flattening the circular drawing onto the plane.

So pictures that show different "outside" boundaries may actually be illustrations of the same planar embedding.

This is what justifies the "add bridge" case in a planar embedding: whatever face is chosen in the embeddings of each of the disjoint planar graphs, we can draw a bridge between them without needing to cross any other edges in the drawing, because we can assume the bridge connects two "outer" faces.

11.5 Euler's Formula

The value of the recursive definition is that it provides a powerful technique for proving properties of planar graphs, namely, structural induction.

One of the most basic properties of a connected planar graph is that its number of vertices and edges determines the number of faces in every possible planar embedding:

Theorem 11.5.1 (Euler's Formula)**.** *If a connected graph has a planar embedding, then*

$$v - e + f = 2$$

where v is the number of vertices, e is the number of edges, and f is the number of faces.

For example, in Figure 11.1, $|V| = 4$, $|E| = 6$, and $f = 4$. Sure enough, $4-6+4 = 2$, as Euler's Formula claims.

Proof. The proof is by structural induction on the definition of planar embeddings. Let $P(\mathcal{E})$ be the proposition that $v - e + f = 2$ for an embedding, $\mathcal{E}$.

Base case: ($\mathcal{E}$ is the one vertex planar embedding). By definition, $v = 1$, $e = 0$, and $f = 1$, so $P(\mathcal{E})$ indeed holds.

Constructor case: (split a face) Suppose G is a connected graph with a planar embedding, and suppose a and b are distinct, nonadjacent vertices of G that appear on some discrete face, $\gamma = a \ldots b \cdots a$, of the planar embedding.

Then the graph obtained by adding the edge a—b to the edges of G has a planar embedding with one more face and one more edge than G. So the quantity $v - e + f$ will remain the same for both graphs, and since by structural induction this quantity is 2 for G's embedding, it's also 2 for the embedding of G with the added edge. So P holds for the constructed embedding.

Constructor case: (add bridge) Suppose G and H are connected graphs with planar embeddings and disjoint sets of vertices. Then connecting these two graphs with a bridge merges the two bridged faces into a single face, and leaves all other faces unchanged. So the bridge operation yields a planar embedding of a connected graph with $v_G + v_H$ vertices, $e_G + e_H + 1$ edges, and $f_G + f_H - 1$ faces. But

$$\begin{aligned}
&(v_G + v_H) - (e_G + e_H + 1) + (f_G + f_H - 1) \\
&= (v_G - e_G + f_G) + (v_H - e_H + f_H) - 2 \\
&= (2) + (2) - 2 && \text{(by structural induction hypothesis)} \\
&= 2.
\end{aligned}$$

So $v - e + f$ remains equal to 2 for the constructed embedding. That is, P also holds in this case.

This completes the proof of the constructor cases, and the theorem follows by structural induction. ■

11.6 Number of Edges versus Vertices

Like Euler's formula, the following lemmas follow by structural induction directly from the definition of planar embedding.

Lemma 11.6.1. *In a planar embedding of a connected graph, each edge is traversed once by each of two different faces, or is traversed exactly twice by one face.*

Lemma 11.6.2. *In a planar embedding of a connected graph with at least three vertices, each face is of length at least three.*

Corollary 11.6.3. *Suppose a connected planar graph has $v \geq 3$ vertices and e edges. Then*

$$e \leq 3v - 6.$$

Proof. By definition, a connected graph is planar iff it has a planar embedding. So suppose a connected graph with v vertices and e edges has a planar embedding with f faces. By Lemma 11.6.1, every edge is traversed exactly twice by the face boundaries. So the sum of the lengths of the face boundaries is exactly $2e$. Also by Lemma 11.6.2, when $v \geq 3$, each face boundary is of length at least three, so this sum is at least $3f$. This implies that

$$3f \leq 2e. \tag{11.1}$$

But $f = e - v + 2$ by Euler's formula, and substituting into (11.1) gives

$$\begin{aligned}
3(e - v + 2) &\leq 2e \\
e - 3v + 6 &\leq 0 \\
e &\leq 3v - 6
\end{aligned}$$

■

Corollary 11.6.3 lets us prove that the quadapi can't all shake hands without crossing. Representing quadapi by vertices and the necessary handshakes by edges, we get the complete graph, K_5. Shaking hands without crossing amounts to showing that K_5 is planar. But K_5 is connected, has 5 vertices and 10 edges, and $10 > 3 \cdot 5 - 6$. This violates the condition of Corollary 11.6.3 required for K_5 to be planar, which proves

Lemma 11.6.4. *K_5 is not planar.*

Another consequence is

Lemma 11.6.5. *Every planar graph has a vertex of degree at most five.*

Proof. If every vertex had degree at least 6, then the sum of the vertex degrees is at least $6v$, but since the sum equals $2e$, we have $e \geq 3v$ contradicting the fact that $e \leq 3v - 6 < 3v$ by Corollary 11.6.3. ■

11.7 Planar Subgraphs

If you draw a graph in the plane by repeatedly adding edges that don't cross, you clearly could add the edges in any other order and still wind up with the same drawing. This is so basic that we might presume that our recursively defined planar embeddings have this property. But that wouldn't be fair: we really need to prove it. After all, the recursive definition of planar embedding was pretty technical —maybe we got it a little bit wrong, with the result that our embeddings don't have this basic draw-in-any-order property.

Now any ordering of edges can be obtained just by repeatedly switching the order of successive edges, and if you think about the recursive definition of embedding for a minute, you should realize that you can switch *any* pair of successive edges if you can just switch the last two. So it all comes down to the following lemma.

Lemma 11.7.1. *Suppose that, starting from some embeddings of planar graphs with disjoint sets of vertices, it is possible by two successive applications of constructor operations to add edges e and then f to obtain a planar embedding, $\mathcal{F}$. Then starting from the same embeddings, it is also possible to obtain $\mathcal{F}$ by adding f and then e with two successive applications of constructor operations.*

We'll leave the proof of Lemma 11.7.1 to Problem 11.6.

Corollary 11.7.2. *Suppose that, starting from some embeddings of planar graphs with disjoint sets of vertices, it is possible to add a sequence of edges $e_0, e_1, \ldots, e_n$ by successive applications of constructor operations to obtain a planar embedding, $\mathcal{F}$. Then starting from the same embeddings, it is also possible to obtain $\mathcal{F}$ by applications of constructor operations that successively add any permutation[2] of the edges $e_0, e_1, \ldots, e_n$.*

[2]If $\pi : \{0, 1, \ldots, n\} \to \{0, 1, \ldots, n\}$ is a bijection, then the sequence $e_{\pi(0)}, e_{\pi(1)}, \ldots, e_{\pi(n)}$ is called a *permutation* of the sequence $e_0, e_1, \ldots, e_n$.

Corollary 11.7.3. *Deleting an edge from a planar graph leaves a planar graph.*

Proof. By Corollary 11.7.2, we may assume the deleted edge was the last one added in constructing an embedding of the graph. So the embedding to which this last edge was added must be an embedding of the graph without that edge. ■

Since we can delete a vertex by deleting all its incident edges, Corollary 11.7.3 immediately implies

Corollary 11.7.4. *Deleting a vertex from a planar graph, along with all its incident edges of course, leaves another planar graph.*

A *subgraph* of a graph, G, is any graph whose set of vertices is a subset of the vertices of G and whose set of edges is a subset of the set of edges of G. So we can summarize Corollaries 11.7.3 and 11.7.4 and their consequences in a Theorem.

Theorem 11.7.5. *Any subgraph of a planar graph is planar.*

11.8 Planar 5-Colorability

We need to know one more property of planar graphs in order to prove that planar graphs are 5-colorable.

Lemma 11.8.1. *Merging two adjacent vertices of a planar graph leaves another planar graph.*

Here merging two adjacent vertices, n_1 and n_2 of a graph means deleting the two vertices and then replacing them by a new "merged" vertex, m, adjacent to all the vertices that were adjacent to either of n_1 or n_2, as illustrated in Figure 11.7.

Lemma 11.8.1 can be proved by structural induction, but the proof is kind of boring, and we hope you'll be relieved that we're going to omit it. (If you insist, we can add it to the next problem set.)

Now we've got all the simple facts we need to prove 5-colorability.

Theorem 11.8.2. *Every planar graph is five-colorable.*

Proof. The proof will be by strong induction on the number, v, of vertices, with induction hypothesis:

> Every planar graph with v vertices is five-colorable.

Base cases ($v \leq 5$): immediate.

Inductive case: Suppose G is a planar graph with $v + 1$ vertices. We will describe a five-coloring of G.

First, choose a vertex, g, of G with degree at most 5; Lemma 11.6.5 guarantees there will be such a vertex.

Case 1 ($\deg(g) < 5$): Deleting g from G leaves a graph, H, that is planar by Lemma 11.7.4, and, since H has v vertices, it is five-colorable by induction hypothesis. Now define a five coloring of G as follows: use the five-coloring of H for all

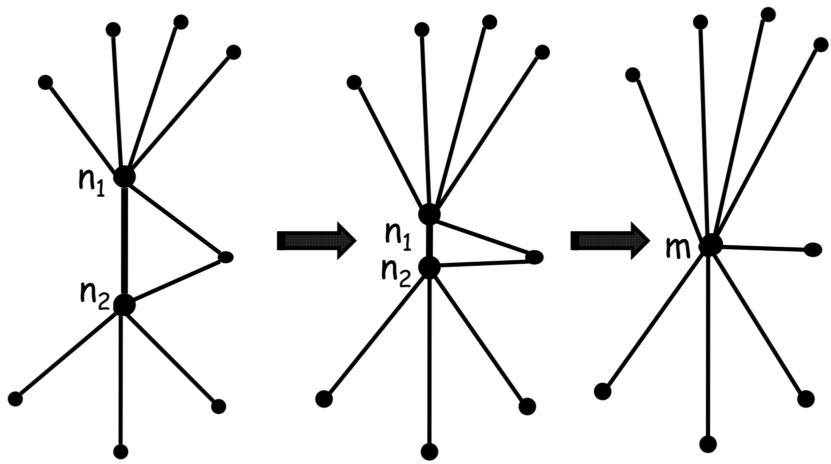


Figure 11.7: Merging adjacent vertices n_1 and n_2 into new vertex, m.

the vertices besides g, and assign one of the five colors to g that is not the same as the color assigned to any of its neighbors. Since there are fewer than 5 neighbors, there will always be such a color available for g.

Case 2 (deg $(g) = 5$): If the five neighbors of g in G were all adjacent to each other, then these five vertices would form a nonplanar subgraph isomorphic to K_5, contradicting Theorem 11.7.5. So there must be two neighbors, n_1 and n_2, of g that are not adjacent. Now merge n_1 and g into a new vertex, m, as in Figure 11.7. In this new graph, n_2 is adjacent to m, and the graph is planar by Lemma 11.8.1. So we can then merge m and n_2 into a another new vertex, m', resulting in a new graph, G', which by Lemma 11.8.1 is also planar. Now G' has $v - 1$ vertices and so is five-colorable by the induction hypothesis.

Now define a five coloring of G as follows: use the five-coloring of G' for all the vertices besides g, n_1 and n_2. Next assign the color of m' in G' to be the color of the neighbors n_1 and n_2. Since n_1 and n_2 are not adjacent in G, this defines a proper five-coloring of G except for vertex g. But since these two neighbors of g have the same color, the neighbors of g have been colored using fewer than five colors altogether. So complete the five-coloring of G by assigning one of the five colors to g that is not the same as any of the colors assigned to its neighbors.

■

A graph obtained from a graph, G, be repeatedly deleting vertices, deleting edges, and merging adjacent vertices is called a *minor* of G. Since K_5 and $K_{3,3}$ are not planar, Lemmas 11.7.3, 11.7.4, and 11.8.1 immediately imply:

Corollary 11.8.3. *A graph which has K_5 or $K_{3,3}$ as a minor is not planar.*

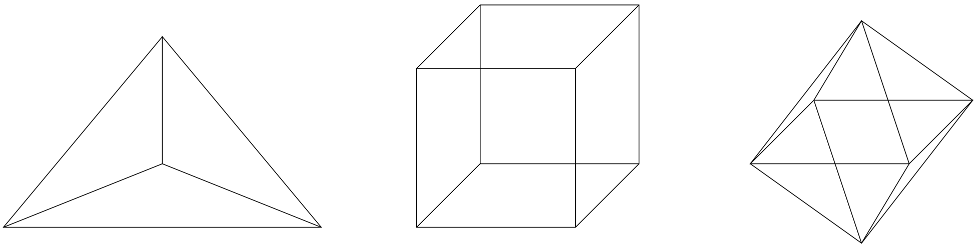
We don't have time to prove it, but the converse of Corollary 11.8.3 is also true. This gives the following famous, very elegant, and purely discrete characterization of planar graphs:

Theorem 11.8.4 (Kuratowksi)**.** *A graph is not planar iff it has K_5 or $K_{3,3}$ as a minor.*

11.9 Classifying Polyhedra

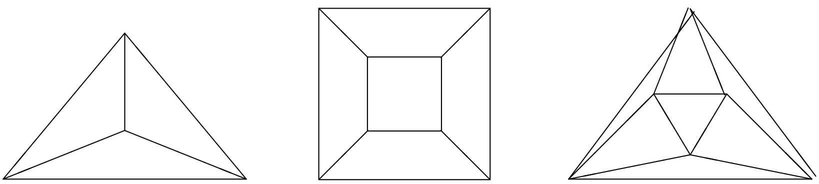
The Pythagoreans had two great mathematical secrets, the irrationality of $\sqrt{2}$ and a geometric construct that we're about to rediscover!

A *polyhedron* is a convex, three-dimensional region bounded by a finite number of polygonal faces. If the faces are identical regular polygons and an equal number of polygons meet at each corner, then the polyhedron is *regular*. Three examples of regular polyhedra are shown below: the tetrahedron, the cube, and the octahedron.



We can determine how many more regular polyhedra there are by thinking about planarity. Suppose we took *any* polyhedron and placed a sphere inside it. Then we could project the polyhedron face boundaries onto the sphere, which would give an image that was a planar graph embedded on the sphere, with the images of the corners of the polyhedron corresponding to vertices of the graph. But we've already observed that embeddings on a sphere are the same as embeddings on the plane, so Euler's formula for planar graphs can help guide our search for regular polyhedra.

For example, planar embeddings of the three polyhedra above look like this:



Let m be the number of faces that meet at each corner of a polyhedron, and let n be the number of sides on each face. In the corresponding planar graph, there are m edges incident to each of the v vertices. Since each edge is incident to two vertices, we know:

$$mv = 2e$$

Also, each face is bounded by n edges. Since each edge is on the boundary of two faces, we have:

$$nf = 2e$$

Solving for v and f in these equations and then substituting into Euler's formula gives:

$$\frac{2e}{m} - e + \frac{2e}{n} = 2$$

which simplifies to

$$\frac{1}{m} + \frac{1}{n} = \frac{1}{e} + \frac{1}{2} \tag{11.2}$$

This last equation (11.2) places strong restrictions on the structure of a polyhedron. Every nondegenerate polygon has at least 3 sides, so $n \geq 3$. And at least 3 polygons must meet to form a corner, so $m \geq 3$. On the other hand, if either n or m were 6 or more, then the left side of the equation could be at most $1/3 + 1/6 = 1/2$, which is less than the right side. Checking the finitely-many cases that remain turns up only five solutions. For each valid combination of n and m, we can compute the associated number of vertices v, edges e, and faces f. And polyhedra with these properties do actually exist:

n	m	v	e	f	polyhedron
3	3	4	6	4	tetrahedron
4	3	8	12	6	cube
3	4	6	12	8	octahedron
3	5	12	30	20	icosahedron
5	3	20	30	12	dodecahedron

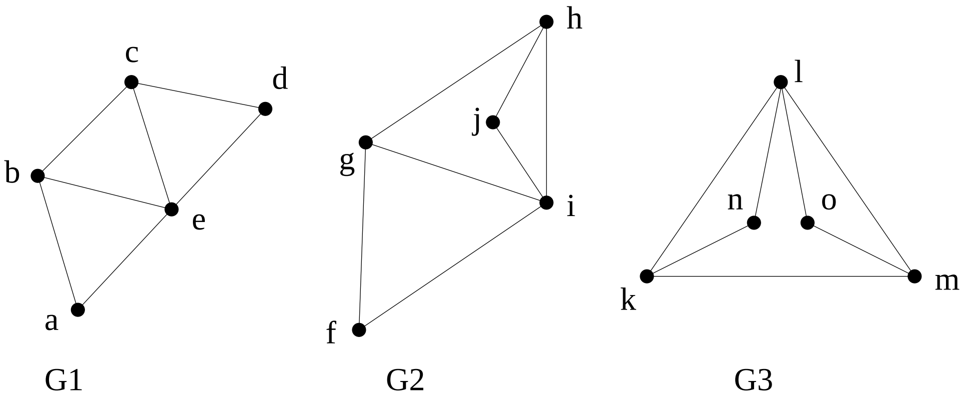
The last polyhedron in this list, the dodecahedron, was the other great mathematical secret of the Pythagorean sect. These five, then, are the only possible regular polyhedra.

So if you want to put more than 20 geocentric satellites in orbit so that they *uniformly* blanket the globe —tough luck!

11.9.1 Problems

Exam Problems

Problem 11.1.



(a) Describe an isomorphism between graphs G_1 and G_2, and another isomorphism between G_2 and G_3.

(b) Why does part (a) imply that there is an isomorphism between graphs G_1 and G_3?

Let G and H be planar graphs. An embedding E_G of G is isomorphic to an embedding E_H of H iff there is an isomorphism from G to H that also maps each face of E_G to a face of E_H.

(c) One of the embeddings pictured above is not isomorphic to either of the others. Which one? Briefly explain why.

(d) Explain why all embeddings of two isomorphic planar graphs must have the same number of faces.

Class Problems

Problem 11.2.
Figures 1–4 show different pictures of planar graphs.

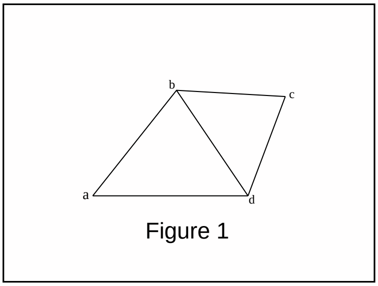


Figure 1

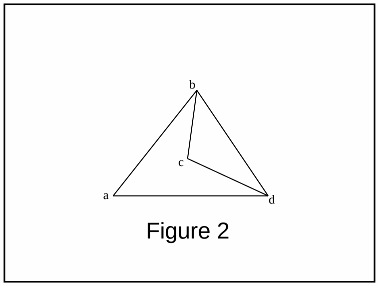


Figure 2

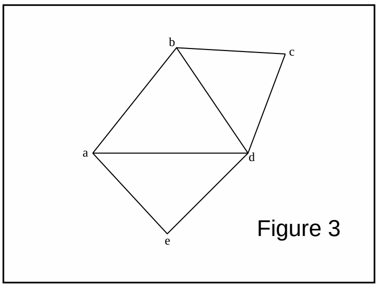


Figure 3

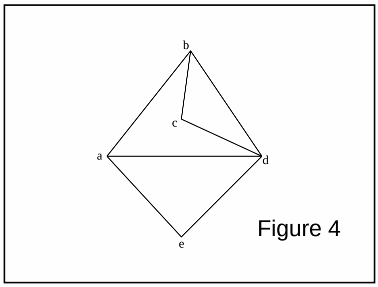


Figure 4

1

(a) For each picture, describe its discrete faces (simple cycles that define the region borders).

(b) Which of the pictured graphs are isomorphic? Which pictures represent the same *planar embedding*? – that is, they have the same discrete faces.

(c) Describe a way to construct the embedding in Figure 4 according to the recursive Definition 11.3.1 of planar embedding. For each application of a constructor rule, be sure to indicate the faces (cycles) to which the rule was applied and the cycles which result from the application.

Problem 11.3. **(a)** Show that if a connected planar graph with more than two vertices is bipartite, then

$$e \leq 2v - 4. \tag{11.3}$$

Hint: Similar to the proof of Corollary 11.6.3 that for planar graphs $e \leq 3v - 6$.

(b) Conclude that that $K_{3,3}$ is not planar. ($K_{3,3}$ is the graph with six vertices and an edge from each of the first three vertices to each of the last three.)

Problem 11.4.
Prove the following assertions by structural induction on the definition of planar embedding.

(a) In a planar embedding of a graph, each edge is traversed a total of two times by the faces of the embedding.

(b) In a planar embedding of a connected graph with at least three vertices, each face is of length at least three.

Homework Problems

Problem 11.5.
A simple graph is *triangle-free* when it has no simple cycle of length three.

(a) Prove for any connected triangle-free planar graph with $v > 2$ vertices and e edges, $e \leq 2v - 4$.

Hint: Similar to the proof that $e \leq 3v - 6$. Use Problem 11.4.

(b) Show that any connected triangle-free planar graph has at least one vertex of degree three or less.

(c) Prove by induction on the number of vertices that any connected triangle-free planar graph is 4-colorable.

Hint: use part (b).

Problem 11.6. **(a)** Prove Lemma 11.7.1. *Hint:* There are four cases to analyze, depending on which two constructor operations are applied to add e and then f. Structural induction is not needed.

(b) Prove Corollary 11.7.2.

Hint: By induction on the number of switches of adjacent elements needed to convert the sequence 0,1,...,n into a permutation $\pi(0), \pi(1), \ldots, \pi(n)$.

Chapter 12

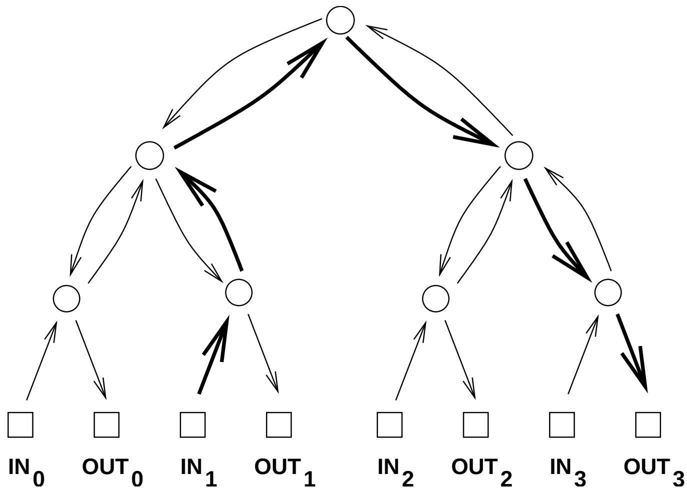
Communication Networks

12.1 Communication Networks

Modeling communication networks is an important application of digraphs in computer science. In this such models, vertices represent computers, processors, and switches; edges will represent wires, fiber, or other transmission lines through which data flows. For some communication networks, like the internet, the corresponding graph is enormous and largely chaotic. Highly structured networks, by contrast, find application in telephone switching systems and the communication hardware inside parallel computers. In this chapter, we'll look at some of the nicest and most commonly used structured networks.

12.2 Complete Binary Tree

Let's start with a *complete binary tree*. Here is an example with 4 inputs and 4 outputs.



The kinds of communication networks we consider aim to transmit packets of data between computers, processors, telephones, or other devices. The term *packet* refers to some roughly fixed-size quantity of data— 256 bytes or 4096 bytes or whatever. In this diagram and many that follow, the squares represent *terminals*, sources and destinations for packets of data. The circles represent *switches*, which direct packets through the network. A switch receives packets on incoming edges and relays them forward along the outgoing edges. Thus, you can imagine a data packet hopping through the network from an input terminal, through a sequence of switches joined by directed edges, to an output terminal.

Recall that there is a unique simple path between every pair of vertices in a tree. So the natural way to route a packet of data from an input terminal to an output in the complete binary tree is along the corresponding directed path. For example, the route of a packet traveling from input 1 to output 3 is shown in bold.

12.3 Routing Problems

Communication networks are supposed to get packets from inputs to outputs, with each packet entering the network at its own input switch and arriving at its own output switch. We're going to consider several different communication network designs, where each network has N inputs and N outputs; for convenience, we'll assume N is a power of two.

Which input is supposed to go where is specified by a permutation of $\{0, 1, \ldots, N-1\}$. So a permutation, π, defines a *routing problem*: get a packet that starts at input i to output $\pi(i)$. A *routing*, P, that *solves* a routing problem, π, is a set of paths from each input to its specified output. That is, P is a set of n paths, P_i, for $i = 0 \ldots, N-1$, where P_i goes from input i to output $\pi(i)$.

12.4 Network Diameter

The delay between the time that a packets arrives at an input and arrives at its designated output is a critical issue in communication networks. Generally this delay is proportional to the length of the path a packet follows. Assuming it takes one time unit to travel across a wire, the delay of a packet will be the number of wires it crosses going from input to output.

Generally packets are routed to go from input to output by the shortest path possible. With a shortest path routing, the worst case delay is the distance between the input and output that are farthest apart. This is called the *diameter* of the network. In other words, the diameter of a network[1] is the maximum length of any shortest path between an input and an output. For example, in the complete binary tree above, the distance from input 1 to output 3 is six. No input and output are farther apart than this, so the diameter of this tree is also six.

More generally, the diameter of a complete binary tree with N inputs and outputs is $2\log N+2$. (All logarithms in this lecture— and in most of computer science —are base 2.) This is quite good, because the logarithm function grows very slowly. We could connect up $2^{10} = 1024$ inputs and outputs using a complete binary tree and the worst input-output delay for any packet would be this diameter, namely, $2\log(2^{10}) + 2 = 22$.

12.4.1 Switch Size

One way to reduce the diameter of a network is to use larger switches. For example, in the complete binary tree, most of the switches have three incoming edges and three outgoing edges, which makes them 3×3 switches. If we had 4×4 switches, then we could construct a complete *ternary* tree with an even smaller diameter. In principle, we could even connect up all the inputs and outputs via a single monster $N \times N$ switch.

This isn't very productive, however, since we've just concealed the original network design problem inside this abstract switch. Eventually, we'll have to design the internals of the monster switch using simpler components, and then we're right back where we started. So the challenge in designing a communication network is figuring out how to get the functionality of an $N \times N$ switch using fixed size, elementary devices, like 3×3 switches.

12.5 Switch Count

Another goal in designing a communication network is to use as few switches as possible. The number of switches in a complete binary tree is $1+2+4+8+\cdots+N$, since there is 1 switch at the top (the "root switch"), 2 below it, 4 below those, and

[1] The usual definition of *diameter* for a general *graph* (simple or directed) is the largest distance between *any* two vertices, but in the context of a communication network we're only interested in the distance between inputs and outputs, not between arbitrary pairs of vertices.

so forth. By the formula (3.16) for geometric sums, the total number of switches is $2N - 1$, which is nearly the best possible with 3×3 switches.

12.6 Network Latency

We'll sometimes be choosing routings through a network that optimize some quantity besides delay. For example, in the next section we'll be trying to minimize packet congestion. When we're not minimizing delay, shortest routings are not always the best, and in general, the delay of a packet will depend on how it is routed. For any routing, the most delayed packet will be the one that follows the longest path in the routing. The length of the longest path in a routing is called its *latency*.

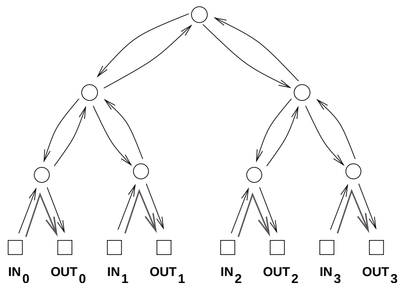
The latency of a *network* depends on what's being optimized. It is measured by assuming that optimal routings are always chosen in getting inputs to their specified outputs. That is, for each routing problem, π, we choose an optimal routing that solves π. Then *network latency* is defined to be the largest routing latency among these optimal routings. Network latency will equal network diameter if routings are always chosen to optimize delay, but it may be significantly larger if routings are chosen to optimize something else.

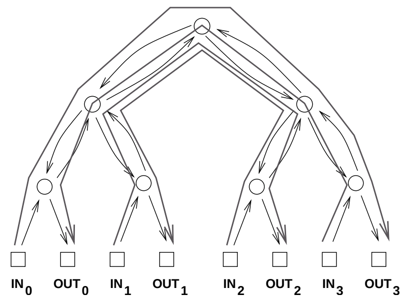
For the networks we consider below, paths from input to output are uniquely determined (in the case of the tree) or all paths are the same length, so network latency will always equal network diameter.

12.7 Congestion

The complete binary tree has a fatal drawback: the root switch is a bottleneck. At best, this switch must handle an enormous amount of traffic: every packet traveling from the left side of the network to the right or vice-versa. Passing all these packets through a single switch could take a long time. At worst, if this switch fails, the network is broken into two equal-sized pieces.

For example, if the routing problem is given by the identity permutation, $\text{Id}(i) ::= i$, then there is an easy routing, P, that solves the problem: let P_i be the path from input i up through one switch and back down to output i. On the other hand, if the problem was given by $\pi(i) ::= (N - 1) - i$, then in *any* solution, Q, for π, each path Q_i beginning at input i must eventually loop all the way up through the root switch and then travel back down to output $(N - 1) - i$. These two situations are illustrated below.





We can distinguish between a "good" set of paths and a "bad" set based on congestion. The *congestion* of a routing, P, is equal to the largest number of paths in P that pass through a single switch. For example, the congestion of the routing on the left is 1, since at most 1 path passes through each switch. However, the congestion of the routing on the right is 4, since 4 paths pass through the root switch (and the two switches directly below the root). Generally, lower congestion is better since packets can be delayed at an overloaded switch.

By extending the notion of congestion to networks, we can also distinguish between "good" and "bad" networks with respect to bottleneck problems. For each routing problem, π, for the network, we assume a routing is chosen that optimizes congestion, that is, that has the minimum congestion among all routings that solve π. Then the largest congestion that will ever be suffered by a switch will be the maximum congestion among these optimal routings. This "maximin" congestion is called the *congestion of the network*.

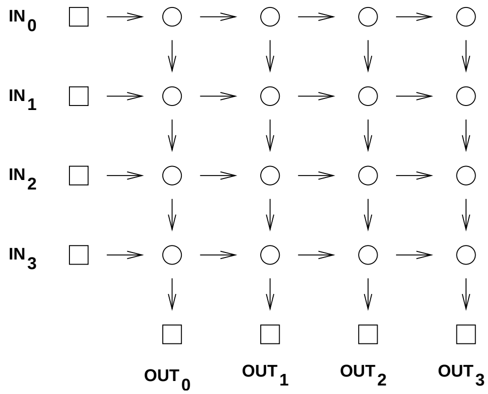
So for the complete binary tree, the worst permutation would be $\pi(i) ::= (N-1)-i$. Then in every possible solution for π, *every* packet, would have to follow a path passing through the root switch. Thus, the max congestion of the complete binary tree is N —which is horrible!

Let's tally the results of our analysis so far:

network	diameter	switch size	# switches	congestion
complete binary tree	$2\log N+2$	3×3	$2N-1$	N

12.8 2-D Array

Let's look at an another communication network. This one is called a *2-dimensional array* or *grid*.



Here there are four inputs and four outputs, so $N = 4$.

The diameter in this example is 8, which is the number of edges between input 0 and output 3. More generally, the diameter of an array with N inputs and outputs is $2N$, which is much worse than the diameter of $2 \log N + 2$ in the complete binary tree. On the other hand, replacing a complete binary tree with an array almost eliminates congestion.

Theorem 12.8.1. *The congestion of an N-input array is 2.*

Proof. First, we show that the congestion is at most 2. Let π be any permutation. Define a solution, P, for π to be the set of paths, P_i, where P_i goes to the right from input i to column $\pi(i)$ and then goes down to output $\pi(i)$. Thus, the switch in row i and column j transmits at most two packets: the packet originating at input i and the packet destined for output j.

Next, we show that the congestion is at least 2. This follows because in any routing problem, π, where $\pi(0) = 0$ and $\pi(N-1) = N-1$, two packets must pass through the lower left switch. ■

As with the tree, the network latency when minimizing congestion is the same as the diameter. That's because all the paths between a given input and output are the same length.

Now we can record the characteristics of the 2-D array.

network	diameter	switch size	# switches	congestion
complete binary tree	$2 \log N + 2$	3×3	$2N - 1$	N
2-D array	$2N$	2×2	N^2	2

The crucial entry here is the number of switches, which is N^2. This is a major defect of the 2-D array; a network of size $N = 1000$ would require a *million* 2×2 switches! Still, for applications where N is small, the simplicity and low congestion of the array make it an attractive choice.

12.9 Butterfly

The Holy Grail of switching networks would combine the best properties of the complete binary tree (low diameter, few switches) and of the array (low congestion). The *butterfly* is a widely-used compromise between the two.

A good way to understand butterfly networks is as a recursive data type. The recursive definition works better if we define just the switches and their connections, omitting the terminals. So we recursively define F_n to be the switches and connections of the butterfly net with $N ::= 2^n$ input and output switches.

The base case is F_1 with 2 input switches and 2 output switches connected as in Figure 12.1.

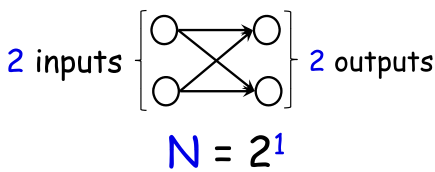


Figure 12.1: F_1, the Butterfly Net switches with $N = 2^1$.

In the constructor step, we construct F_{n+1} with 2^{n+1} inputs and outputs out of two F_n nets connected to a new set of 2^{n+1} input switches, as shown in as in Figure 12.2. That is, the ith and $2^n + i$th new input switches are each connected to the same two switches, namely, to the ith input switches of each of two F_n components for $i = 1, \ldots, 2^n$. The output switches of F_{n+1} are simply the output switches of each of the F_n copies.

So F_{n+1} is laid out in columns of height 2^{n+1} by adding one more column of switches to the columns in F_n. Since the construction starts with two columns when $n = 1$, the F_{n+1} switches are arrayed in $n + 1$ columns. The total number of switches is the height of the columns times the number of columns, namely, $2^{n+1}(n+1)$. Remembering that $n = \log N$, we conclude that the Butterfly Net with

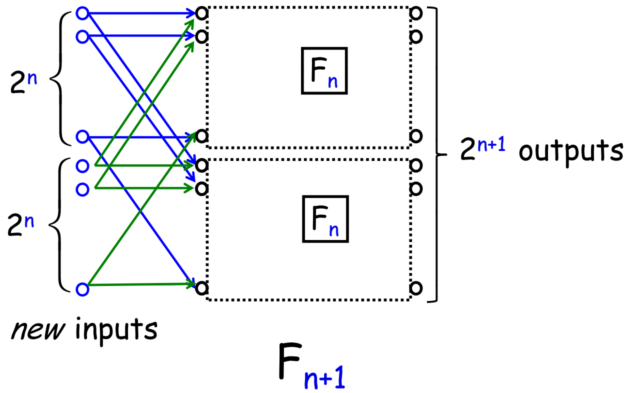


Figure 12.2: F_{n+1}, the Butterfly Net switches with 2^{n+1} inputs and outputs.

N inputs has $N(\log N + 1)$ switches.

Since every path in F_{n+1} from an input switch to an output is the same length, namely, $n+1$, the diameter of the Butterfly net with 2^{n+1} inputs is this length plus two because of the two edges connecting to the terminals (square boxes) —one edge from input terminal to input switch (circle) and one from output switch to output terminal.

There is an easy recursive procedure to route a packet through the Butterfly Net. In the base case, there is obviously only one way to route a packet from one of the two inputs to one of the two outputs. Now suppose we want to route a packet from an input switch to an output switch in F_{n+1}. If the output switch is in the "top" copy of F_n, then the first step in the route must be from the input switch to the unique switch it is connected to in the top copy; the rest of the route is determined by recursively routing the rest of the way in the top copy of F_n. Likewise, if the output switch is in the "bottom" copy of F_n, then the first step in the route must be to the switch in the bottom copy, and the rest of the route is determined by recursively routing in the bottom copy of F_n. In fact, this argument shows that the routing is *unique*: there is exactly one path in the Butterfly Net from each input to each output, which implies that the network latency when minimizing congestion is the same as the diameter.

The congestion of the butterfly network is about $\sqrt{N}$, more precisely, the congestion is $\sqrt{N}$ if N is an even power of 2 and $\sqrt{N/2}$ if N is an odd power of 2. A simple proof of this appears in Problem12.8.

Let's add the butterfly data to our comparison table:

network	diameter	switch size	# switches	congestion
complete binary tree	$2\log N + 2$	3×3	$2N - 1$	N
2-D array	$2N$	2×2	N^2	2
butterfly	$\log N + 2$	2×2	$N(\log(N) + 1)$	$\sqrt{N}$ or $\sqrt{N/2}$

The butterfly has lower congestion than the complete binary tree. And it uses fewer switches and has lower diameter than the array. However, the butterfly does not capture the best qualities of each network, but rather is a compromise somewhere between the two. So our quest for the Holy Grail of routing networks goes on.

12.10 Beneš Network

In the 1960's, a researcher at Bell Labs named Beneš had a remarkable idea. He obtained a marvelous communication network with congestion 1 by placing *two* butterflies back-to-back. This amounts to recursively growing *Beneš nets* by adding both inputs and outputs at each stage. Now we recursively define B_n to be the switches and connections (without the terminals) of the Beneš net with $N ::= 2^n$ input and output switches.

The base case, B_1, with 2 input switches and 2 output switches is exactly the same as F_1 in Figure 12.1.

In the constructor step, we construct B_{n+1} out of two B_n nets connected to a new set of 2^{n+1} input switches *and also* a new set of 2^{n+1} output switches. This is illustrated in Figure 12.3.

Namely, the ith and $2^n + i$th new input switches are each connected to the same two switches, namely, to the ith input switches of each of two B_n components for $i = 1, \ldots, 2^n$, exactly as in the Butterfly net. In addition, the ith and $2^n + i$th new *output* switches are connected to the same two switches, namely, to the ith output switches of each of two B_n components.

Now B_{n+1} is laid out in columns of height 2^{n+1} by adding two more columns of switches to the columns in B_n. So the B_{n+1} switches are arrayed in $2(n + 1)$ columns. The total number of switches is the number of columns times the height of the columns, namely, $2(n + 1)2^{n+1}$.

All paths in B_{n+1} from an input switch to an output are the same length, namely, $2(n + 1) - 1$, and the diameter of the Beneš net with 2^{n+1} inputs is this length plus two because of the two edges connecting to the terminals.

So Beneš has doubled the number of switches and the diameter, of course, but completely eliminates congestion problems! The proof of this fact relies on a clever induction argument that we'll come to in a moment. Let's first see how the Beneš

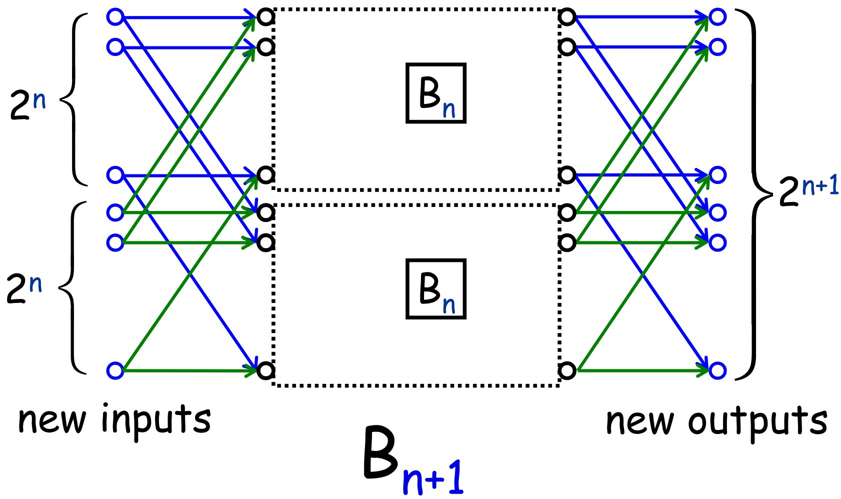


Figure 12.3: B_{n+1}, the Beneš Net switches with 2^{n+1} inputs and outputs.

network stacks up:

network	diameter	switch size	# switches	congestion
complete binary tree	$2 \log N + 2$	3×3	$2N - 1$	N
2-D array	$2N$	2×2	N^2	2
butterfly	$\log N + 2$	2×2	$N(\log(N) + 1)$	$\sqrt{N}$ or $\sqrt{N/2}$
Beneš	$2 \log N + 1$	2×2	$2N \log N$	1

The Beneš network has small size and diameter, and completely eliminates congestion. The Holy Grail of routing networks is in hand!

Theorem 12.10.1. *The congestion of the N-input Beneš network is 1.*

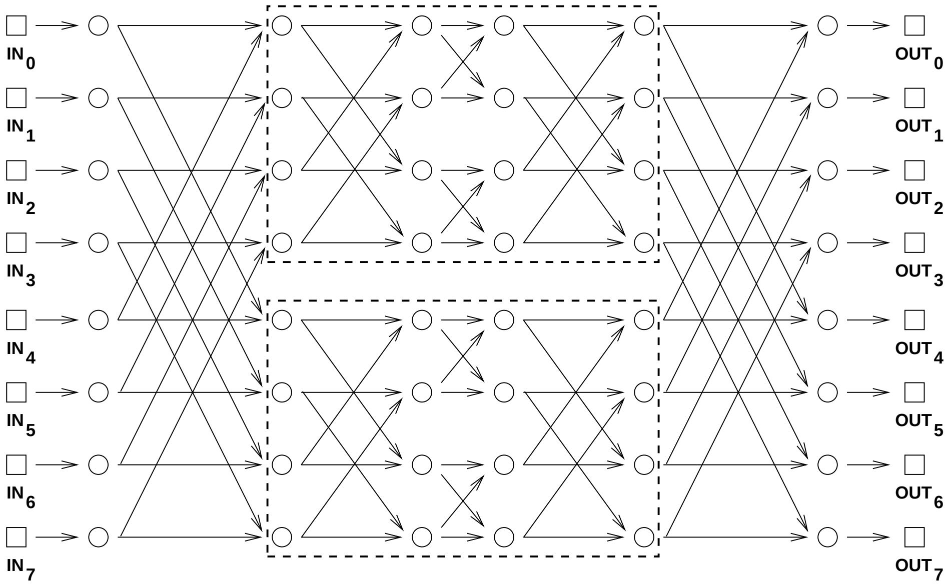
Proof. By induction on n where $N = 2^n$. So the induction hypothesis is

$$P(n) ::= \text{the congestion of } B_n \text{ is } 1.$$

Base case ($n = 1$): $B_1 = F_1$ and the unique routings in F_1 have congestion 1.

Inductive step: We assume that the congestion of an $N = 2^n$-input Beneš network is 1 and prove that the congestion of a $2N$-input Beneš network is also 1.

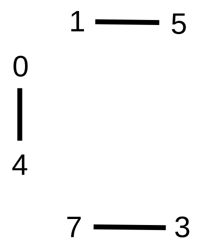
Digression. Time out! Let's work through an example, develop some intuition, and then complete the proof. In the Beneš network shown below with $N = 8$ inputs and outputs, the two 4-input/output subnetworks are in dashed boxes.



By the inductive assumption, the subnetworks can each route an arbitrary permutation with congestion 1. So if we can guide packets safely through just the first and last levels, then we can rely on induction for the rest! Let's see how this works in an example. Consider the following permutation routing problem:

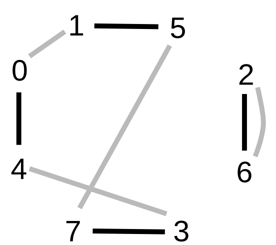
$$\begin{aligned} \pi(0) &= 1 & \qquad \pi(4) &= 3 \\ \pi(1) &= 5 & \pi(5) &= 6 \\ \pi(2) &= 4 & \pi(6) &= 0 \\ \pi(3) &= 7 & \pi(7) &= 2 \end{aligned}$$

We can route each packet to its destination through either the upper subnetwork or the lower subnetwork. However, the choice for one packet may constrain the choice for another. For example, we can not route both packet 0 *and* packet 4 through the same network since that would cause two packets to collide at a single switch, resulting in congestion. So one packet must go through the upper network and the other through the lower network. Similarly, packets 1 and 5, 2 and 6, and 3 and 7 must be routed through different networks. Let's record these constraints in a graph. The vertices are the 8 packets. If two packets must pass through different networks, then there is an edge between them. Thus, our constraint graph looks like this:



Notice that at most one edge is incident to each vertex.

The output side of the network imposes some further constraints. For example, the packet destined for output 0 (which is packet 6) and the packet destined for output 4 (which is packet 2) can not both pass through the same network; that would require both packets to arrive from the same switch. Similarly, the packets destined for outputs 1 and 5, 2 and 6, and 3 and 7 must also pass through different switches. We can record these additional constraints in our graph with gray edges:



Notice that at most one new edge is incident to each vertex. The two lines drawn between vertices 2 and 6 reflect the two different reasons why these packets must be routed through different networks. However, we intend this to be a simple graph; the two lines still signify a single edge.

Now here's the key insight: *a 2-coloring of the graph corresponds to a solution to the routing problem*. In particular, suppose that we could color each vertex either red or blue so that adjacent vertices are colored differently. Then all constraints are satisfied if we send the red packets through the upper network and the blue packets through the lower network.

The only remaining question is whether the constraint graph is 2-colorable, which is easy to verify:

Lemma 12.10.2. *Prove that if the edges of a graph can be grouped into two sets such that every vertex has at most 1 edge from each set incident to it, then the graph is 2-colorable.*

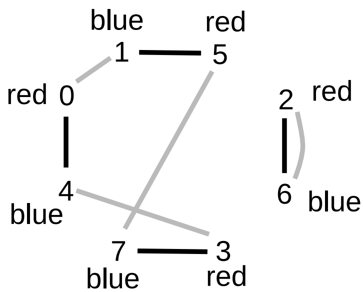
Proof. Since the two sets of edges may overlap, let's call an edge that is in both sets a *doubled edge*.

We know from Theorem 9.7.2 that all we have to do is show that every cycle has even length. There are two cases:

Case 1: [The cycle contains a doubled edge.] No other edge can be incident to either of the endpoints of a doubled edge, since that endpoint would then be incident to two edges from the same set. So a cycle traversing a doubled edge has nowhere to go but back and forth along the edge an even number of times.

Case 2: [No edge on the cycle is doubled.] Since each vertex is incident to at most one edge from each set, any path with no doubled edges must traverse successive edges that alternate from one set to the other. In particular, a cycle must traverse a path of alternating edges that begins and ends with edges from different sets. This means the cycle has to be of even length. ■

For example, here is a 2-coloring of the constraint graph:



The solution to this graph-coloring problem provides a start on the packet routing problem:

We can complete the routing in the two smaller Beneš networks by induction! Back to the proof. **End of Digression.**

Let π be an arbitrary permutation of $\{0, 1, \ldots, N-1\}$. Let G be the graph whose vertices are packet numbers $0, 1, \ldots, N-1$ and whose edges come from the union of these two sets:

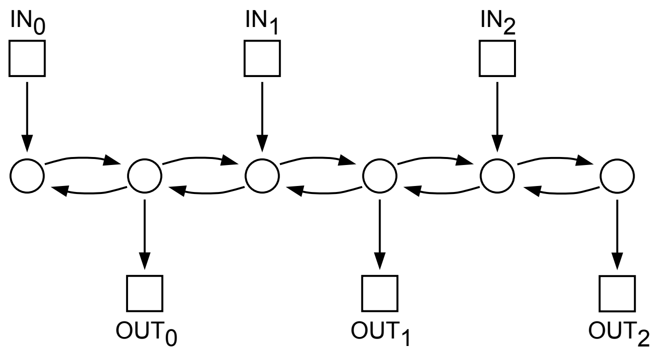
$$E_1 ::= \{u\text{—}v \mid |u - v| = N/2\}, \text{ and}$$
$$E_2 ::= \{u\text{—}w \mid |\pi(u) - \pi(w)| = N/2\}.$$

Now any vertex, u, is incident to at most two edges: a unique edge $u\text{—}v \in E_1$ and a unique edge $u\text{—}w \in E_2$. So according to Lemma 12.10.2, there is a 2-coloring for the vertices of G. Now route packets of one color through the upper subnetwork and packets of the other color through the lower subnetwork. Since for each edge in E_1, one vertex goes to the upper subnetwork and the other to the lower subnetwork, there will not be any conflicts in the first level. Since for each edge in E_2, one vertex comes from the upper subnetwork and the other from the lower subnetwork, there will not be any conflicts in the last level. We can complete the routing within each subnetwork by the induction hypothesis $P(n)$. ■

12.10.1 Problems

Exam Problems

Problem 12.1.
Consider the following communication network:



(a) What is the max congestion? ________

(b) Give an input/output permutation, π_0, that forces maximum congestion:

$\pi_0(0) =$ ______ $\pi_0(1) =$ ______ $\pi_0(2) =$ ______

(c) Give an input/output permutation, π_1, that allows *minimum* congestion:

$\pi_1(0) =$ ______ $\pi_1(1) =$ ______ $\pi_1(2) =$ ______

(d) What is the latency for the permutation π_1? (If you could not find π_1, just choose a permutation and find its latency.) ________

Class Problems

Problem 12.2.
The Beneš network has a max congestion of 1; that is, every permutation can be routed in such a way that a single packet passes through each switch. Let's work through an example. A Beneš network of size $N = 8$ is attached.

(a) Within the Beneš network of size $N = 8$, there are two subnetworks of size $N = 4$. Put boxes around these. Hereafter, we'll refer to these as the *upper* and *lower* subnetworks.

(b) Now consider the following permutation routing problem:

$$\begin{aligned} \pi(0) &= 3 & \qquad \pi(4) &= 2 \\ \pi(1) &= 1 & \qquad \pi(5) &= 0 \\ \pi(2) &= 6 & \qquad \pi(6) &= 7 \\ \pi(3) &= 5 & \qquad \pi(7) &= 4 \end{aligned}$$

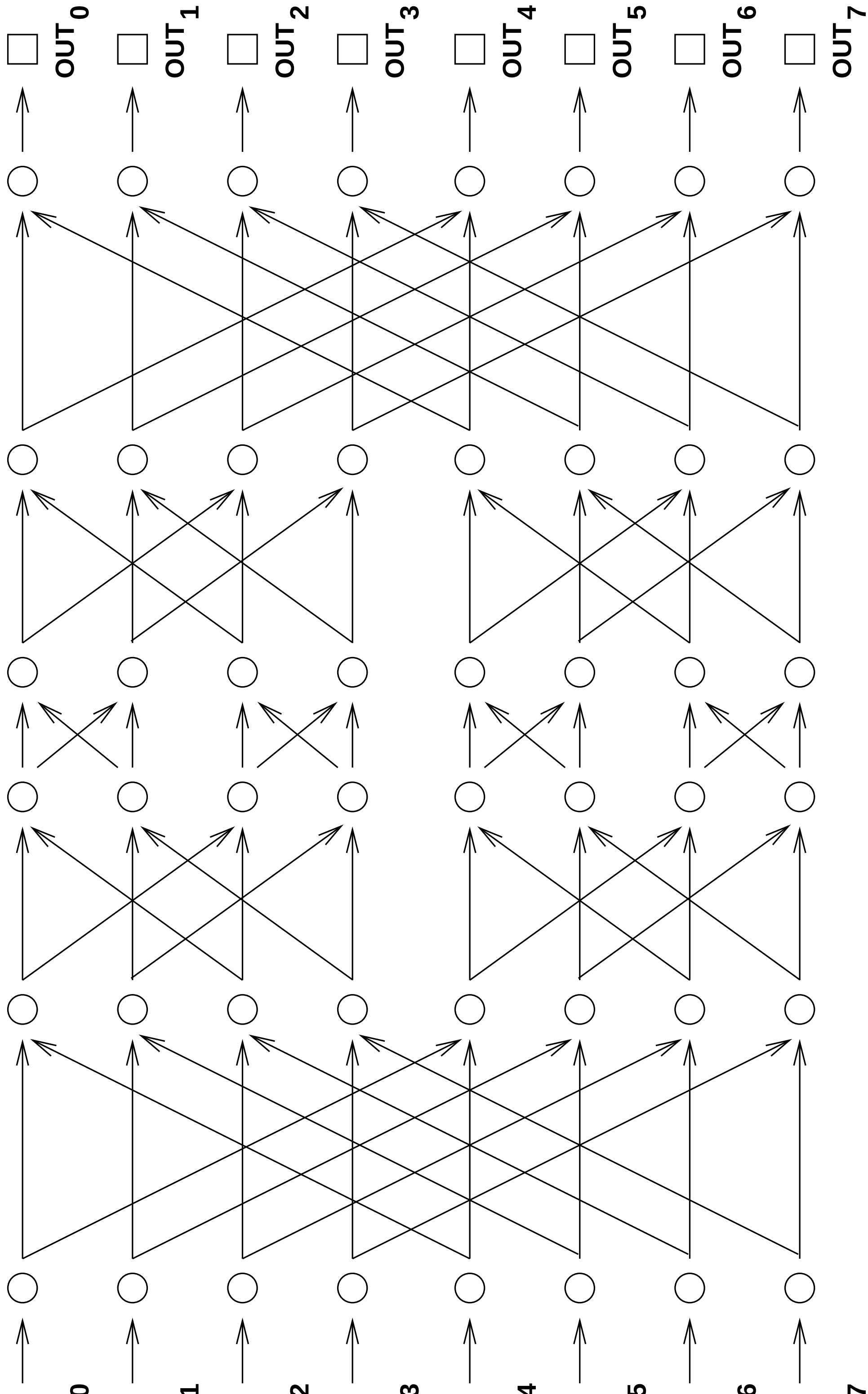
Each packet must be routed through either the upper subnetwork or the lower subnetwork. Construct a graph with vertices 0, 1, . . . , 7 and draw a *dashed* edge between each pair of packets that can not go through the same subnetwork because a collision would occur in the second column of switches.

(c) Add a *solid* edge in your graph between each pair of packets that can not go through the same subnetwork because a collision would occur in the next-to-last column of switches.

(d) Color the vertices of your graph red and blue so that adjacent vertices get different colors. Why must this be possible, regardless of the permutation π?

(e) Suppose that red vertices correspond to packets routed through the upper subnetwork and blue vertices correspond to packets routed through the lower subnetwork. On the attached copy of the Beneš network, highlight the first and last edge traversed by each packet.

(f) All that remains is to route packets through the upper and lower subnetworks. One way to do this is by applying the procedure described above recursively on each subnetwork. However, since the remaining problems are small, see if you can complete all the paths on your own.


OUT0
OUT1
OUT2
OUT3
OUT4
OUT5
OUT6
OUT7

Problem 12.3.
A *multiple binary-tree network* has n inputs and n outputs, where n is a power of 2. Each input is connected to the root of a binary tree with $n/2$ leaves and with edges pointing away from the root. Likewise, each output is connected to the root of a binary tree with $n/2$ leaves and with edges pointing toward the root.

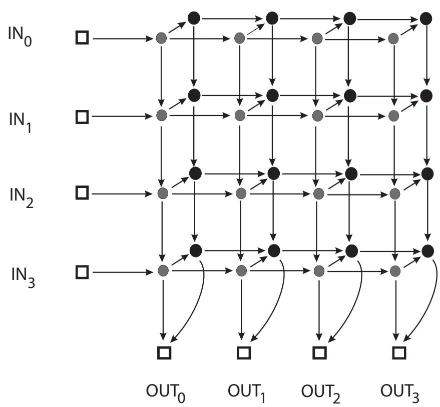
Two edges point from each leaf of an input tree, and each of these edges points to a leaf of an output tree. The matching of leaf edges is arranged so that for every input and output tree, there is an edge from a leaf of the input tree to a leaf of the output tree, and every output tree leaf has exactly two edges pointing to it.

(a) Draw such a multiple binary-tree net for $n = 4$.

(b) Fill in the table, and explain your entries.

# switches	switch size	diameter	max congestion

Problem 12.4.
The n-input *2-D Array* network was shown to have congestion 2. An n-input *2-Layer Array* consisting of two n-input 2-D Arrays connected as pictured below for $n = 4$.



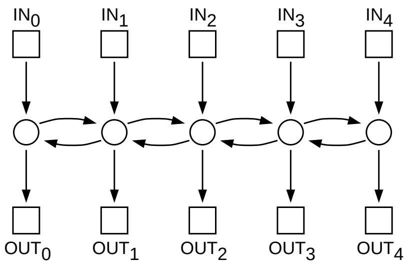
In general, an n-input 2-Layer Array has two layers of switches, with each layer connected like an n-input 2-D Array. There is also an edge from each switch in the first layer to the corresponding switch in the second layer. The inputs of the 2-Layer Array enter the left side of the first layer, and the n outputs leave from the bottom row of either layer.

(a) For any given input-output permutation, there is a way to route packets that achieves congestion 1. Describe how to route the packets in this way.

(b) What is the latency of a routing designed to minimize latency?

(c) Explain why the congestion of any minimum latency (CML) routing of packets through this network is greater than the network's congestion.

Problem 12.5.
A *5-path* communication network is shown below. From this, it's easy to see what an n-path network would be. Fill in the table of properties below, and be prepared to justify your answers.

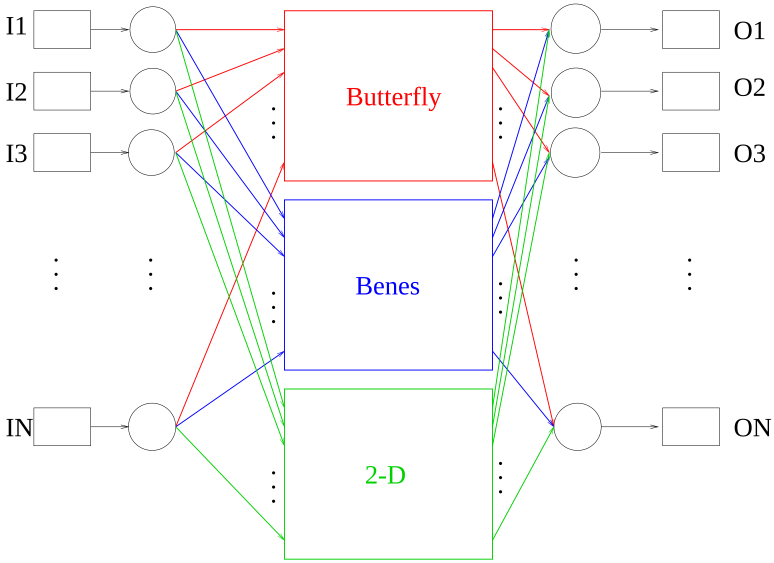


5-Path

network	# switches	switch size	diameter	max congestion
5-path				
n-path				

Problem 12.6.
Tired of being a TA, Megumi has decided to become famous by coming up with a new, better communication network design. Her network has the following specifications: every input node will be sent to a Butterfly network, a Benes network and a 2D Grid network. At the end, the outputs of all three networks will converge on the new output.

In the Megumi-net a minimum latency routing does not have minimum congestion. The *latency for min-congestion* (*LMC*) of a net is the best bound on latency achievable using routings that minimize congestion. Likewise, the *congestion for min-latency* (*CML*) is the best bound on congestion achievable using routings that minimize latency.



Fill in the following chart for Megumi's new net and explain your answers.

network	diameter	# switches	congestion	LMC	CML
Megumi's net					

Homework Problems

Problem 12.7.
Louis Reasoner figures that, wonderful as the Beneš network may be, the butterfly network has a few advantages, namely: fewer switches, smaller diameter, and an easy way to route packets through it. So Louis designs an N-input/output network he modestly calls a *Reasoner-net* with the aim of combining the best features of both the butterfly and Beneš nets:

> The ith input switch in a Reasoner-net connects to two switches, a_i and b_i, and likewise, the jth output switch has two switches, y_j and z_j, connected to it. Then the Reasoner-net has an N-input Beneš network connected using the a_i switches as input switches and the y_j switches as its output switches. The Reasoner-net also has an N-input butterfly net connected using the b_i switches as inputs and¡ the z_j switches as outputs.

In the Reasoner-net a minimum latency routing does not have minimum congestion. The *latency for min-congestion* (*LMC*) of a net is the best bound on latency achievable using routings that minimize congestion. Likewise, the *congestion for min-latency* (*CML*) is the best bound on congestion achievable using routings that minimize latency.

Fill in the following chart for the Reasoner-net and briefly explain your answers.

diameter	**switch size(s)**	**# switches**	**congestion**	**LMC**	**CML**

Problem 12.8.
Show that the congestion of the butterfly net, F_n, is exactly $\sqrt{N}$ when n is even.

Hint:

- There is a unique path from each input to each output, so the congestion is the maximum number of messages passing through a vertex for any routing problem.
- If v is a vertex in column i of the butterfly network, there is a path from exactly 2^i input vertices to v and a path from v to exactly 2^{n-i} output vertices.
- At which column of the butterfly network must the congestion be worst? What is the congestion of the topmost switch in that column of the network?

Part III

Counting

Chapter 13

Sums & Asymptotics

13.1 The Value of an Annuity

Would you prefer a million dollars today or \$50,000 a year for the rest of your life? On the one hand, instant gratification is nice. On the other hand, the total dollars received at \$50K per year is much larger if you live long enough.

Formally, this is a question about the value of an annuity. An *annuity* is a financial instrument that pays out a fixed amount of money at the beginning of every year for some specified number of years. In particular, an n-year, m-payment annuity pays m dollars at the start of each year for n years. In some cases, n is finite, but not always. Examples include lottery payouts, student loans, and home mortgages. There are even Wall Street people who specialize in trading annuities.

A key question is what an annuity is worth. For example, lotteries often pay out jackpots over many years. Intuitively, $\$50,000$ a year for 20 years ought to be worth less than a million dollars right now. If you had all the cash right away, you could invest it and begin collecting interest. But what if the choice were between $\$50,000$ a year for 20 years and a *half* million dollars today? Now it is not clear which option is better.

In order to answer such questions, we need to know what a dollar paid out in the future is worth today. To model this, let's assume that money can be invested at a fixed annual interest rate p. We'll assume an 8% rate[1] for the rest of the discussion.

Here is why the interest rate p matters. Ten dollars invested today at interest rate p will become $(1+p) \cdot 10 = 10.80$ dollars in a year, $(1+p)^2 \cdot 10 \approx 11.66$ dollars in two years, and so forth. Looked at another way, ten dollars paid out a year from now are only really worth $1/(1+p) \cdot 10 \approx 9.26$ dollars today. The reason is that if we

[1] U.S. interest rates have dropped steadily for several years, and ordinary bank deposits now earn around 1.5%. But just a few years ago the rate was 8%; this rate makes some of our examples a little more dramatic. The rate has been as high as 17% in the past thirty years.

In Japan, the standard interest rate is near zero%, and on a few occasions in the past few years has even been slightly negative. It's a mystery why the Japanese populace keeps any money in their banks.

had the $\$9.26$ today, we could invest it and would have $\$10.00$ in a year anyway. Therefore, p determines the value of money paid out in the future.

13.1.1 The Future Value of Money

So for an n-year, m-payment annuity, the first payment of m dollars is truly worth m dollars. But the second payment a year later is worth only $m/(1+p)$ dollars. Similarly, the third payment is worth $m/(1+p)^2$, and the n-th payment is worth only $m/(1+p)^{n-1}$. The total value, V, of the annuity is equal to the sum of the payment values. This gives:

$$\begin{aligned} V &= \sum_{i=1}^{n} \frac{m}{(1+p)^{i-1}} \\ &= m \cdot \sum_{j=0}^{n-1} \left(\frac{1}{1+p}\right)^j && \text{(substitute } j ::= i-1\text{)} \\ &= m \cdot \sum_{j=0}^{n-1} x^j && \text{(substitute } x = \frac{1}{1+p}\text{).} \qquad (13.1) \end{aligned}$$

The summation in (13.1) is a geometric sum that has a *closed form*, making the evaluation a lot easier, namely[2],

$$\sum_{i=0}^{n-1} x^i = \frac{1-x^n}{1-x}. \qquad (13.2)$$

(The phrase "closed form" refers to a mathematical expression without any summation or product notation.)

Equation (13.2) was proved by induction in Problem 3.17, but, as is often the case, the proof by induction gave no hint about how the formula was found in the first place. So we'll take this opportunity to explain where it comes from. The trick is to let S be the value of the sum and then observe what $-xS$ is:

$$\begin{array}{rcllllll} S &=& 1 & +x & +x^2 & +x^3 & + \cdots & +x^{n-1} \\ -xS &=& & -x & -x^2 & -x^3 & - \cdots & -x^{n-1} - x^n. \end{array}$$

Adding these two equations gives:

$$S - xS = 1 - x^n,$$

so

$$S = \frac{1-x^n}{1-x}.$$

We'll look further into this method of proof in a few weeks when we introduce *generating functions* in Chapter 16.

[2]To make this equality hold for $x = 0$, we adopt the convention that $0^0 ::= 1$.

13.1.2 Closed Form for the Annuity Value

So now we have a simple formula for V, the value of an annuity that pays m dollars at the start of each year for n years.

$$\begin{aligned} V &= m\frac{1-x^n}{1-x} && \text{(by (13.1) and (13.2))} && (13.3) \\ &= m\frac{1+p-(1/(1+p))^{n-1}}{p} && (x = 1/(1+p)). && (13.4) \end{aligned}$$

The formula (13.4) is much easier to use than a summation with dozens of terms. For example, what is the real value of a winning lottery ticket that pays $\$50,000$ per year for 20 years? Plugging in $m = \$50,000$, $n = 20$, and $p = 0.08$ gives $V \approx \$530,180$. So because payments are deferred, the million dollar lottery is really only worth about a half million dollars! This is a good trick for the lottery advertisers!

13.1.3 Infinite Geometric Series

The question we began with was whether you would prefer a million dollars today or $\$50,000$ a year for the rest of your life. Of course, this depends on how long you live, so optimistically assume that the second option is to receive $\$50,000$ a year *forever*. This sounds like infinite money! But we can compute the value of an annuity with an infinite number of payments by taking the limit of our geometric sum in (13.2) as n tends to infinity.

Theorem 13.1.1. *If* $|x| < 1$, *then*

$$\sum_{i=0}^{\infty} x^i = \frac{1}{1-x}.$$

Proof.

$$\begin{aligned} \sum_{i=0}^{\infty} x^i &::= \lim_{n\to\infty} \sum_{i=0}^{n-1} x^i \\ &= \lim_{n\to\infty} \frac{1-x^n}{1-x} && \text{(by (13.2))} \\ &= \frac{1}{1-x}. \end{aligned}$$

The final line follows from that fact that $\lim_{n\to\infty} x^n = 0$ when $|x| < 1$. ■

In our annuity problem, $x = 1/(1+p) < 1$, so Theorem 13.1.1 applies, and we get

$$\begin{aligned} V &= m \cdot \sum_{j=0}^{\infty} x^j && \text{(by (13.1))} \\ &= m \cdot \frac{1}{1-x} && \text{(by Theorem 13.1.1)} \\ &= m \cdot \frac{1+p}{p} && (x = 1/(1+p)). \end{aligned}$$

Plugging in $m = \$50,000$ and $p = 0.08$, the value, V, is only $\$675,000$. Amazingly, a million dollars today is worth much more than $\$50,000$ paid every year forever! Then again, if we had a million dollars today in the bank earning 8% interest, we could take out and spend $\$80,000$ a year forever. So on second thought, this answer really isn't so amazing.

13.1.4 Problems

Class Problems

Problem 13.1.
You've seen this neat trick for evaluating a geometric sum:

$$\begin{aligned} S &= 1 + z + z^2 + \ldots + z^n \\ zS &= z + z^2 + \ldots + z^n + z^{n+1} \\ S - zS &= 1 - z^{n+1} \\ S &= \frac{1 - z^{n+1}}{1 - z} \end{aligned}$$

Use the same approach to find a closed-form expression for this sum:

$$T = 1z + 2z^2 + 3z^3 + \ldots + nz^n$$

Homework Problems

Problem 13.2.
Is a Harvard degree really worth more than an MIT degree?! Let us say that a person with a Harvard degree starts with \$40,000 and gets a \$20,000 raise every year after graduation, whereas a person with an MIT degree starts with \$30,000, but gets a 20% raise every year. Assume inflation is a fixed 8% every year. That is, \$1.08 a year from now is worth \$1.00 today.

(a) How much is a Harvard degree worth today if the holder will work for n years following graduation?

(b) How much is an MIT degree worth in this case?

(c) If you plan to retire after twenty years, which degree would be worth more?

Problem 13.3.
Suppose you deposit \$100 into your MIT Credit Union account today, \$99 in one month from now, \$98 in two months from now, and so on. Given that the interest rate is constantly 0.3% per month, how long will it take to save \$5,000?

13.2 Book Stacking

Suppose you have a pile of books and you want to stack them on a table in some off-center way so the top book sticks out past books below it. How far past the edge of the table do you think you could get the top book to go without having the stack fall over? Could the top book stick out completely beyond the edge of table?

Most people's first response to this question—sometimes also their second and third responses—is "No, the top book will never get completely past the edge of the table." But in fact, you can get the top book to stick out as far as you want: one booklength, two booklengths, any number of booklengths!

13.2.1 Formalizing the Problem

We'll approach this problem recursively. How far past the end of the table can we get one book to stick out? It won't tip as long as its center of mass is over the table, so we can get it to stick out half its length, as shown in Figure 13.1.

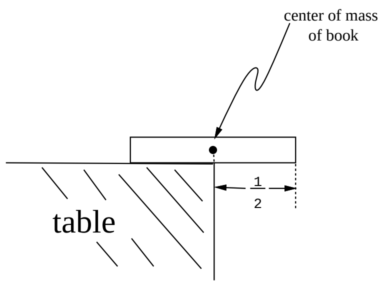


Figure 13.1: One book can overhang half a book length.

Now suppose we have a stack of books that will stick out past the table edge without tipping over—call that a *stable* stack. Let's define the *overhang* of a stable stack to be the largest horizontal distance from the center of mass of the stack to the furthest edge of a book. If we place the center of mass of the stable stack at the edge of the table as in Figure 13.2, that's how far we can get a book in the stack to stick out past the edge.

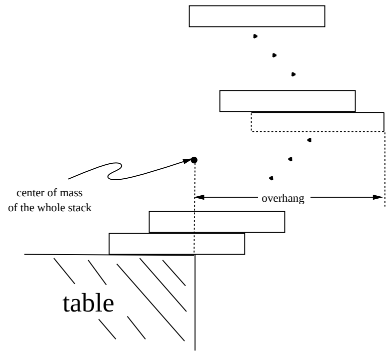


Figure 13.2: Overhanging the edge of the table.

So we want a formula for the maximum possible overhang, B_n, achievable with a stack of n books.

We've already observed that the overhang of one book is $1/2$ a book length. That is,

$$B_1 = \frac{1}{2}.$$

Now suppose we have a stable stack of $n+1$ books with maximum overhang. If the overhang of the n books on top of the bottom book was not maximum, we could get a book to stick out further by replacing the top stack with a stack of n books with larger overhang. So the maximum overhang, B_{n+1}, of a stack of $n+1$ books is obtained by placing a maximum overhang stable stack of n books on top of the bottom book. And we get the biggest overhang for the stack of $n+1$ books by placing the center of mass of the n books right over the edge of the bottom book as in Figure 13.3.

So we know where to place the $n+1$st book to get maximum overhang, and all we have to do is calculate what it is. The simplest way to do that is to let the center of mass of the top n books be the origin. That way the horizontal coordinate of the center of mass of the whole stack of $n+1$ books will equal the increase in the overhang. But now the center of mass of the bottom book has horizontal coordinate $1/2$, so the horizontal coordinate of center of mass of the whole stack of $n+1$ books is

$$\frac{0 \cdot n + (1/2) \cdot 1}{n+1} = \frac{1}{2(n+1)}.$$

In other words,

$$B_{n+1} = B_n + \frac{1}{2(n+1)}, \tag{13.5}$$

as shown in Figure 13.3.

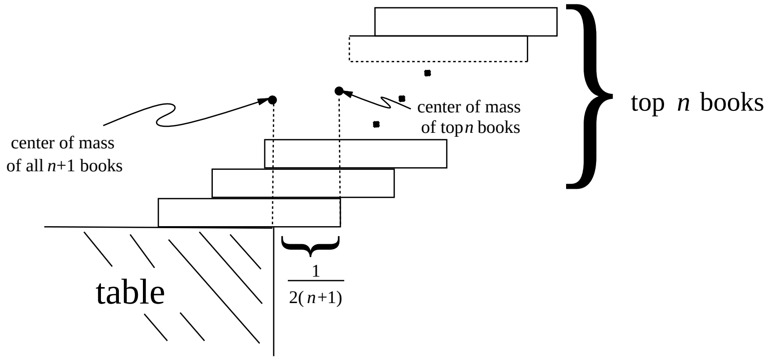


Figure 13.3: Additional overhang with $n+1$ books.

Expanding equation (13.5), we have

$$\begin{aligned} B_{n+1} &= B_{n-1} + \frac{1}{2n} + \frac{1}{2(n+1)} \\ &= B_1 + \frac{1}{2\cdot 2} + \cdots + \frac{1}{2n} + \frac{1}{2(n+1)} \\ &= \frac{1}{2}\sum_{i=1}^{n+1} \frac{1}{i}. \end{aligned} \tag{13.6}$$

The nth *Harmonic number*, H_n, is defined to be

Definition 13.2.1.

$$H_n ::= \sum_{i=1}^{n} \frac{1}{i}.$$

So (13.6) means that

$$B_n = \frac{H_n}{2}.$$

The first few Harmonic numbers are easy to compute. For example, $H_4 = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} = \frac{25}{12}$. The fact that H_4 is greater than 2 has special significance; it implies that the total extension of a 4-book stack is greater than one full book! This is the situation shown in Figure 13.4.

13.2.2 Evaluating the Sum—The Integral Method

It would be nice to answer questions like, "How many books are needed to build a stack extending 100 book lengths beyond the table?" One approach to this question

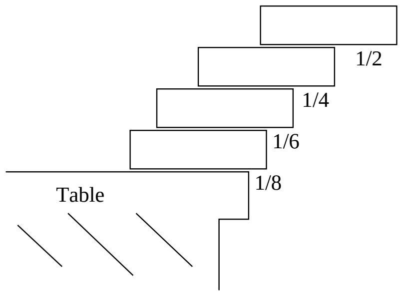


Figure 13.4: Stack of four books with maximum overhang.

would be to keep computing Harmonic numbers until we found one exceeding 200. However, as we will see, this is not such a keen idea.

Such questions would be settled if we could express H_n in a closed form. Unfortunately, no closed form is known, and probably none exists. As a second best, however, we can find closed forms for very good approximations to H_n using the Integral Method. The idea of the Integral Method is to bound terms of the sum above and below by simple functions as suggested in Figure 13.5. The integrals of these functions then bound the value of the sum above and below.

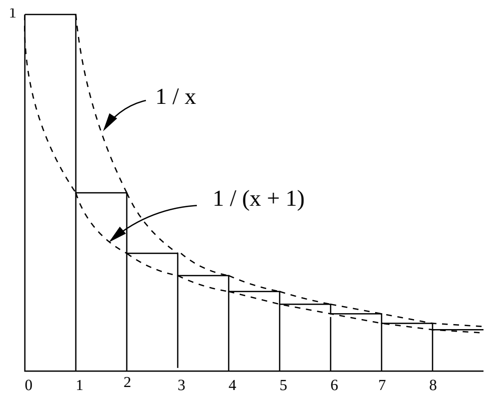


Figure 13.5: *This figure illustrates the Integral Method for bounding a sum. The area under the "stairstep" curve over the interval* $[0, n]$ *is equal to* $H_n = \sum_{i=1}^{n} 1/i$*. The function* $1/x$ *is everywhere greater than or equal to the stairstep and so the integral of* $1/x$ *over this interval is an upper bound on the sum. Similarly,* $1/(x+1)$ *is everywhere less than or equal to the stairstep and so the integral of* $1/(x+1)$ *is a lower bound on the sum.*

The Integral Method gives the following upper and lower bounds on the harmonic number H_n:

$$H_n \leq 1 + \int_1^n \frac{1}{x}\, dx = 1 + \ln n \tag{13.7}$$

$$H_n \geq \int_0^n \frac{1}{x+1}\, dx = \int_1^{n+1} \frac{1}{x}\, dx = \ln(n+1). \tag{13.8}$$

These bounds imply that the harmonic number H_n is around $\ln n$.

But $\ln n$ grows —slowly —but without bound. That means we can get books to overhang *any distance* past the edge of the table by piling them high enough! For example, to build a stack extending three book lengths beyond the table, we need a number of books n so that $H_n \geq 6$. By inequality (13.8), this means we want

$$H_n \geq \ln(n+1) \geq 6,$$

so $n \geq e^6 - 1$ books will work, that is, 403 books will be enough to get a three book overhang. Actual calculation of H_6 shows that 227 books is the smallest number that will work.

13.2.3 More about Harmonic Numbers

In the preceding section, we showed that H_n is about $\ln n$. An even better approximation is known:

$$H_n = \ln n + \gamma + \frac{1}{2n} + \frac{1}{12n^2} + \frac{\epsilon(n)}{120n^4}$$

Here γ is a value $0.577215664\ldots$ called *Euler's constant*, and $\epsilon(n)$ is between 0 and 1 for all n. We will not prove this formula.

Asymptotic Equality

The shorthand $H_n \sim \ln n$ is used to indicate that the leading term of H_n is $\ln n$. More precisely:

Definition 13.2.2. For functions $f, g : \mathbb{R} \to \mathbb{R}$, we say f is *asymptotically equal* to g, in symbols,

$$f(x) \sim g(x)$$

iff

$$\lim_{x \to \infty} f(x)/g(x) = 1.$$

It's tempting to might write $H_n \sim \ln n + \gamma$ to indicate the two leading terms, but it is not really right. According to Definition 13.2.2, $H_n \sim \ln n + c$ where c is *any constant*. The correct way to indicate that γ is the second-largest term is $H_n - \ln n \sim \gamma$.

The reason that the $\sim$ notation is useful is that often we do not care about lower order terms. For example, if $n = 100$, then we can compute $H(n)$ to great precision using only the two leading terms:

$$|H_n - \ln n - \gamma| \leq \left| \frac{1}{200} - \frac{1}{120000} + \frac{1}{120 \cdot 100^4} \right| < \frac{1}{200}.$$

13.2.4 Problems

Class Problems

Problem 13.4.
An explorer is trying to reach the Holy Grail, which she believes is located in a desert shrine d days walk from the nearest oasis. In the desert heat, the explorer must drink continuously. She can carry at most 1 gallon of water, which is enough for 1 day. However, she is free to make multiple trips carrying up to a gallon each time to create water caches out in the desert.

For example, if the shrine were $2/3$ of a day's walk into the desert, then she could recover the Holy Grail after two days using the following strategy. She leaves the oasis with 1 gallon of water, travels $1/3$ day into the desert, caches $1/3$ gallon, and then walks back to the oasis— arriving just as her water supply runs out. Then she picks up another gallon of water at the oasis, walks $1/3$ day into the desert, tops off her water supply by taking the $1/3$ gallon in her cache, walks the remaining $1/3$ day to the shrine, grabs the Holy Grail, and then walks for $2/3$ of a day back to the oasis— again arriving with no water to spare.

But what if the shrine were located farther away?

(a) What is the most distant point that the explorer can reach and then return to the oasis if she takes a total of only 1 gallon from the oasis?

(b) What is the most distant point the explorer can reach and still return to the oasis if she takes a total of only 2 gallons from the oasis? No proof is required; just do the best you can.

(c) The explorer will travel using a recursive strategy to go far into the desert and back drawing a total of n gallons of water from the oasis. Her strategy is to build up a cache of $n - 1$ gallons, plus enough to get home, a certain fraction of a day's distance into the desert. On the last delivery to the cache, instead of returning home, she proceeds recursively with her $n - 1$ gallon strategy to go farther into the desert and return to the cache. At this point, the cache has just enough water left to get her home.

Prove that with n gallons of water, this strategy will get her $H_n/2$ days into the desert and back, where H_n is the nth Harmonic number:

$$H_n ::= \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}.$$

Conclude that she can reach the shrine, however far it is from the oasis.

(d) Suppose that the shrine is $d = 10$ days walk into the desert. Use the asymptotic approximation $H_n \sim \ln n$ to show that it will take more than a million years for the explorer to recover the Holy Grail.

Problem 13.5.
There is a number a such that $\sum_{i=1}^{\infty} i^p$ converges iff $p < a$. What is the value of a? Prove it.

Homework Problems

Problem 13.6.
There is a bug on the edge of a 1-meter rug. The bug wants to cross to the other side of the rug. It crawls at 1 cm per second. However, at the end of each second, a malicious first-grader named Mildred Anderson *stretches* the rug by 1 meter. Assume that her action is instantaneous and the rug stretches uniformly. Thus, here's what happens in the first few seconds:

- The bug walks 1 cm in the first second, so 99 cm remain ahead.
- Mildred stretches the rug by 1 meter, which doubles its length. So now there are 2 cm behind the bug and 198 cm ahead.
- The bug walks another 1 cm in the next second, leaving 3 cm behind and 197 cm ahead.
- Then Mildred strikes, stretching the rug from 2 meters to 3 meters. So there are now $3 \cdot (3/2) = 4.5$ cm behind the bug and $197 \cdot (3/2) = 295.5$ cm ahead.
- The bug walks another 1 cm in the third second, and so on.

Your job is to determine this poor bug's fate.

(a) During second i, what *fraction* of the rug does the bug cross?

(b) Over the first n seconds, what fraction of the rug does the bug cross altogether? Express your answer in terms of the Harmonic number H_n.

(c) The known universe is thought to be about $3 \cdot 10^{10}$ light years in diameter. How many universe diameters must the bug travel to get to the end of the rug?

13.3 Finding Summation Formulas

The Integral Method offers a way to derive formulas like for the sum of consecutive integers,

$$\sum_{i=1}^{n} i = n(n+1)/2,$$

or for the sum of squares,

$$\begin{aligned}\sum_{i=1}^{n} i^2 &= \frac{(2n+1)(n+1)n}{6}\\ &= \frac{n^3}{3} + \frac{n^2}{2} + \frac{n}{6}.\end{aligned} \tag{13.9}$$

These equations appeared in Chapter 3.1 as equations (3.1) and (3.3) where they were proved using the Well-ordering Principle. But those proofs did not explain how someone figured out in the first place that these were the formulas to prove.

Here's how the Integral Method leads to the sum-of-squares formula, for example. First, get a quick estimate of the sum:

$$\int_0^n x^2\,dx \le \sum_{i=1}^{n} i^2 \le \int_0^n (x+1)^2\,dx,$$

so

$$n^3/3 \le \sum_{i=1}^{n} i^2 \le (n+1)^3/3 - 1/3. \tag{13.10}$$

and the upper and lower bounds (13.10) imply that

$$\sum_{i=1}^{n} i^2 \sim n^3/3.$$

To get an exact formula, we then guess the general form of the solution. Where we are uncertain, we can add parameters $a, b, c, \ldots$. For example, we might make the guess:

$$\sum_{i=1}^{n} i^2 = an^3 + bn^2 + cn + d.$$

If the guess is correct, then we can determine the parameters a, b, c, and d by plugging in a few values for n. Each such value gives a linear equation in a, b, c, and d. If we plug in enough values, we may get a linear system with a unique solution. Applying this method to our example gives:

$$\begin{array}{lll} n=0 & \text{implies} & 0 = d\\ n=1 & \text{implies} & 1 = a+b+c+d\\ n=2 & \text{implies} & 5 = 8a+4b+2c+d\\ n=3 & \text{implies} & 14 = 27a+9b+3c+d.\end{array}$$

Solving this system gives the solution $a = 1/3$, $b = 1/2$, $c = 1/6$, $d = 0$. Therefore, *if* our initial guess at the form of the solution was correct, then the summation is equal to $n^3/3 + n^2/2 + n/6$, which matches equation (13.9).

The point is that if the desired formula turns out to be a polynomial, then once you get an estimate of the *degree* of the polynomial —by the Integral Method or any other way —all the coefficients of the polynomial can be found automatically.

Be careful! This method let's you discover formulas, but it doesn't guarantee they are right! After obtaining a formula by this method, it's important to go back and *prove* it using induction or some other method, because if the initial guess at the solution was not of the right form, then the resulting formula will be completely wrong!

13.3.1 Double Sums

Sometimes we have to evaluate sums of sums, otherwise known as *double summations*. This can be easy: evaluate the inner sum, replace it with a closed form, and then evaluate the outer sum which no longer has a summation inside it. For example,

$$\begin{aligned}
&\sum_{n=0}^{\infty}\left(y^n \sum_{i=0}^{n} x^i\right) \\
&= \sum_{n=0}^{\infty}\left(y^n \frac{1-x^{n+1}}{1-x}\right) && \text{(geometric sum formula (13.2))} \\
&= \frac{\sum_{n=0}^{\infty} y^n}{1-x} - \frac{\sum_{n=0}^{\infty} y^n x^{n+1}}{1-x} \\
&= \frac{1}{(1-y)(1-x)} - \frac{x\sum_{n=0}^{\infty}(xy)^n}{1-x} && \text{(infinite geometric sum, Theorem 13.1.1)} \\
&= \frac{1}{(1-y)(1-x)} - \frac{x}{(1-xy)(1-x)} && \text{(infinite geometric sum, Theorem 13.1.1)} \\
&= \frac{(1-xy)-x(1-y)}{(1-xy)(1-y)(1-x)} \\
&= \frac{1-x}{(1-xy)(1-y)(1-x)} \\
&= \frac{1}{(1-xy)(1-y)}.
\end{aligned}$$

When there's no obvious closed form for the inner sum, a special trick that is often useful is to try *exchanging the order of summation.* For example, suppose we want to compute the sum of the harmonic numbers

$$\sum_{k=1}^{n} H_k = \sum_{k=1}^{n}\sum_{j=1}^{k} 1/j$$

For intuition about this sum, we can try the integral method:

$$\sum_{k=1}^{n} H_k \approx \int_1^n \ln x \, dx \approx n \ln n - n.$$

Now let's look for an exact answer. If we think about the pairs (k, j) over which we are summing, they form a triangle:

		j 1	2	3	4	5	$\dots$	n
k	1	1						
	2	1	$1/2$					
	3	1	$1/2$	$1/3$				
	4	1	$1/2$	$1/3$	$1/4$			
		$\dots$						
	n	1	$1/2$		$\dots$			$1/n$

The summation above is summing each row and then adding the row sums. Instead, we can sum the columns and then add the column sums. Inspecting the table we see that this double sum can be written as

$$
\begin{aligned}
\sum_{k=1}^{n} H_k &= \sum_{k=1}^{n} \sum_{j=1}^{k} 1/j \\
&= \sum_{j=1}^{n} \sum_{k=j}^{n} 1/j \\
&= \sum_{j=1}^{n} 1/j \sum_{k=j}^{n} 1 \\
&= \sum_{j=1}^{n} \frac{1}{j}(n-j+1) \\
&= \sum_{j=1}^{n} \frac{n-j+1}{j} \\
&= \sum_{j=1}^{n} \frac{n+1}{j} - \sum_{j=1}^{n} \frac{j}{j} \\
&= (n+1) \sum_{j=1}^{n} \frac{1}{j} - \sum_{j=1}^{n} 1 \\
&= (n+1)H_n - n. \qquad (13.11)
\end{aligned}
$$

13.4 Stirling's Approximation

The familiar factorial notation, $n!$, is an abbreviation for the product

$$\prod_{i=1}^{n} i.$$

This is by far the most common product in discrete mathematics. In this section we describe a good closed-form estimate of $n!$ called *Stirling's Approximation*. Unfortunately, all we can do is estimate: there is no closed form for $n!$ —though proving so would take us beyond the scope of this text.

13.4.1 Products to Sums

A good way to handle a product is often to convert it into a sum by taking the logarithm. In the case of factorial, this gives

$$\begin{aligned}\ln(n!) &= \ln(1 \cdot 2 \cdot 3 \cdots (n-1) \cdot n) \\ &= \ln 1 + \ln 2 + \ln 3 + \cdots + \ln(n-1) + \ln n \\ &= \sum_{i=1}^{n} \ln i.\end{aligned}$$

We've not seen a summation containing a logarithm before! Fortunately, one tool that we used in evaluating sums is still applicable: the Integral Method. We can bound the terms of this sum with $\ln x$ and $\ln(x+1)$ as shown in Figure 13.6. This gives bounds on $\ln(n!)$ as follows:

$$\begin{array}{rcl} \int_1^n \ln x \, dx \leq & \sum_{i=1}^n \ln i & \leq \int_0^n \ln(x+1) \, dx \\ n \ln(\frac{n}{e}) + 1 \leq & \sum_{i=1}^n \ln i & \leq (n+1) \ln\left(\frac{n+1}{e}\right) + 1 \\ \left(\frac{n}{e}\right)^n e \leq & n! & \leq \left(\frac{n+1}{e}\right)^{n+1} e. \end{array}$$

The second line follows from the first by completing the integrations. The third line is obtained by exponentiating.

So $n!$ behaves something like the closed form formula $(n/e)^n$. A more careful analysis yields an unexpected closed form formula that is asymptotically exact:

Lemma (Stirling's Formula)**.**

$$n! \sim \left(\frac{n}{e}\right)^n \sqrt{2\pi n}, \tag{13.12}$$

Stirling's Formula describes how $n!$ behaves in the limit, but to use it effectively, we need to know how close it is to the limit for different values of n. That information is given by the bounding formulas:

Fact (Stirling's Approximation)**.**

$$\sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{1/(12n+1)} \leq n! \leq \sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{1/12n}.$$

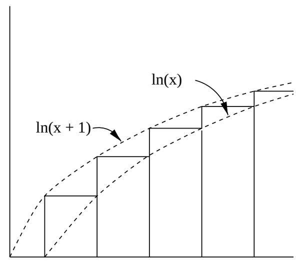


Figure 13.6: *This figure illustrates the Integral Method for bounding the sum* $\sum_{i=1}^{n} \ln i$.

Stirling's Approximation implies the asymptotic formula (13.12), since $e^{1/(12n+1)}$ and $e^{1/12n}$ both approach 1 as n grows large. These inequalities can be verified by induction, but the details are nasty.

The bounds in Stirling's formula are very tight. For example, if $n = 100$, then Stirling's bounds are:

$$100! \geq \sqrt{200\pi}\left(\frac{100}{e}\right)^{100} e^{1/1201}$$
$$100! \leq \sqrt{200\pi}\left(\frac{100}{e}\right)^{100} e^{1/1200}$$

The only difference between the upper bound and the lower bound is in the final term. In particular $e^{1/1201} \approx 1.00083299$ and $e^{1/1200} \approx 1.00083368$. As a result, the upper bound is no more than $1 + 10^{-6}$ times the lower bound. This is amazingly tight! Remember Stirling's formula; we will use it often.

13.5 Asymptotic Notation

Asymptotic notation is a shorthand used to give a quick measure of the behavior of a function $f(n)$ as n grows large.

13.5.1 Little Oh

The asymptotic notation, $\sim$, of Definition 13.2.2 is a binary relation indicating that two functions grow at the *same* rate. There is also a binary relation indicating that one function grows at a significantly *slower* rate than another. Namely,

Definition 13.5.1. For functions $f, g : \mathbb{R} \to \mathbb{R}$, with g nonnegative, we say f is *asymptotically smaller* than g, in symbols,

$$f(x) = o(g(x)),$$

iff

$$\lim_{x\to\infty} f(x)/g(x) = 0.$$

For example, $1000x^{1.9} = o(x^2)$, because $1000x^{1.9}/x^2 = 1000/x^{0.1}$ and since $x^{0.1}$ goes to infinity with x and 1000 is constant, we have $\lim_{x\to\infty} 1000x^{1.9}/x^2 = 0$. This argument generalizes directly to yield

Lemma 13.5.2. *$x^a = o(x^b)$ for all nonnegative constants $a < b$.*

Using the familiar fact that $\log x < x$ for all $x > 1$, we can prove

Lemma 13.5.3. *$\log x = o(x^\epsilon)$ for all $\epsilon > 0$ and $x > 1$.*

Proof. Choose $\epsilon > \delta > 0$ and let $x = z^\delta$ in the inequality $\log x < x$. This implies

$$\log z < z^\delta/\delta = o(z^\epsilon) \qquad \text{by Lemma 13.5.2.} \tag{13.13}$$

■

Corollary 13.5.4. *$x^b = o(a^x)$ for any $a, b \in \mathbb{R}$ with $a > 1$.*

Proof. From (13.13),

$$\log z < z^\delta/\delta$$

for all $z > 1$, $\delta > 0$. Hence

$$\begin{aligned}
(e^b)^{\log z} &< (e^b)^{z^\delta/\delta} \\
z^b &< \left(e^{\log a (b/\log a)}\right)^{z^\delta/\delta} \\
&= a^{(b/\delta \log a) z^\delta} \\
&< a^z
\end{aligned}$$

for all z such that

$$(b/\delta \log a) z^\delta < z.$$

But choosing $\delta < 1$, we know $z^\delta = o(z)$, so this last inequality holds for all large enough z. ■

Lemma 13.5.3 and Corollary 13.5.4 can also be proved easily in several other ways, for example, using L'Hopital's Rule or the McLaurin Series for $\log x$ and e^x. Proofs can be found in most calculus texts.

13.5.2 Big Oh

Big Oh is the most frequently used asymptotic notation. It is used to give an upper bound on the growth of a function, such as the running time of an algorithm.

Definition 13.5.5. Given nonnegative functions $f, g : \mathbb{R} \to \mathbb{R}$, we say that

$$f = O(g)$$

iff

$$\limsup_{x \to \infty} f(x)/g(x) < \infty.$$

This definition[3] makes it clear that

Lemma 13.5.6. *If $f = o(g)$ or $f \sim g$, then $f = O(g)$.*

Proof. $\lim f/g = 0$ or $\lim f/g = 1$ implies $\lim f/g < \infty$. ■

It is easy to see that the converse of Lemma 13.5.6 is not true. For example, $2x = O(x)$, but $2x \not\sim x$ and $2x \neq o(x)$.

The usual formulation of Big Oh spells out the definition of $\limsup$ without mentioning it. Namely, here is an equivalent definition:

Definition 13.5.7. Given functions $f, g : \mathbb{R} \to \mathbb{R}$, we say that

$$f = O(g)$$

iff there exists a constant $c \geq 0$ and an x_0 such that for all $x \geq x_0$, $|f(x)| \leq cg(x)$.

This definition is rather complicated, but the idea is simple: $f(x) = O(g(x))$ means $f(x)$ is less than or equal to $g(x)$, except that we're willing to ignore a constant factor, namely, c, and to allow exceptions for small x, namely, $x < x_0$.

We observe,

Lemma 13.5.8. *If $f = o(g)$, then it is* not *true that $g = O(f)$.*

Proof.

$$\lim_{x \to \infty} \frac{g(x)}{f(x)} = \frac{1}{\lim_{x \to \infty} f(x)/g(x)} = \frac{1}{0} = \infty,$$

so $g \neq O(f)$.

■

[3] We can't simply use the limit as $x \to \infty$ in the definition of $O()$, because if $f(x)/g(x)$ oscillates between, say, 3 and 5 as x grows, then $f = O(g)$ because $f \leq 5g$, but $\lim_{x \to \infty} f(x)/g(x)$ does not exist. So instead of limit, we use the technical notion of $\limsup$. In this oscillating case, $\limsup_{x \to \infty} f(x)/g(x) = 5$.

The precise definition of $\limsup$ is

$$\limsup_{x \to \infty} h(x) ::= \lim_{x \to \infty} \text{lub}_{y \geq x} h(y),$$

where "lub" abbreviates "least upper bound."

Proposition 13.5.9. $100x^2 = O(x^2)$.

Proof. Choose $c = 100$ and $x_0 = 1$. Then the proposition holds, since for all $x \geq 1$, $\left|100x^2\right| \leq 100x^2$. ■

Proposition 13.5.10. $x^2 + 100x + 10 = O(x^2)$.

Proof. $(x^2 + 100x + 10)/x^2 = 1 + 100/x + 10/x^2$ and so its limit as x approaches infinity is $1+0+0 = 1$. So in fact, $x^2+100x+10 \sim x^2$, and therefore $x^2+100x+10 = O(x^2)$. Indeed, it's conversely true that $x^2 = O(x^2 + 100x + 10)$. ■

Proposition 13.5.10 generalizes to an arbitrary polynomial:

Proposition 13.5.11. *For* $a_k \neq 0$, $a_k x^k + a_{k-1}x^{k-1} + \cdots + a_1 x + a_0 = O(x^k)$.

We'll omit the routine proof.

Big Oh notation is especially useful when describing the running time of an algorithm. For example, the usual algorithm for multiplying $n \times n$ matrices requires proportional to n^3 operations in the worst case. This fact can be expressed concisely by saying that the running time is $O(n^3)$. So this asymptotic notation allows the speed of the algorithm to be discussed without reference to constant factors or lower-order terms that might be machine specific. In this case there is another, ingenious matrix multiplication procedure that requires $O(n^{2.55})$ operations. This procedure will therefore be much more efficient on large enough matrices. Unfortunately, the $O(n^{2.55})$-operation multiplication procedure is almost never used because it happens to be less efficient than the usual $O(n^3)$ procedure on matrices of practical size.

13.5.3 Theta

Definition 13.5.12.

$$f = \Theta(g) \quad \text{iff} \quad f = O(g) \text{ and } g = O(f).$$

The statement $f = \Theta(g)$ can be paraphrased intuitively as "f and g are equal to within a constant factor."

The value of these notations is that they highlight growth rates and allow suppression of distracting factors and low-order terms. For example, if the running time of an algorithm is

$$T(n) = 10n^3 - 20n^2 + 1,$$

then

$$T(n) = \Theta(n^3).$$

In this case, we would say that T *is of order* n^3 or that $T(n)$ *grows cubically*.

Another such example is

$$\pi^2 3^{x-7} + \frac{(2.7x^{113} + x^9 - 86)^4}{\sqrt{x}} - 1.08^{3x} = \Theta(3^x).$$

Just knowing that the running time of an algorithm is $\Theta(n^3)$, for example, is useful, because if n doubles we can predict that the running time will *by and large*[4] increase by a factor of at most 8 for large n. In this way, Theta notation preserves information about the scalability of an algorithm or system. Scalability is, of course, a big issue in the design of algorithms and systems.

13.5.4 Pitfalls with Big Oh

There is a long list of ways to make mistakes with Big Oh notation. This section presents some of the ways that Big Oh notation can lead to ruin and despair.

The Exponential Fiasco

Sometimes relationships involving Big Oh are not so obvious. For example, one might guess that $4^x = O(2^x)$ since 4 is only a constant factor larger than 2. This reasoning is incorrect, however; actually 4^x grows much faster than 2^x.

Proposition 13.5.13. $4^x \neq O(2^x)$

Proof. $2^x/4^x = 2^x/(2^x 2^x) = 1/2^x$. Hence, $\lim_{x\to\infty} 2^x/4^x = 0$, so in fact $2^x = o(4^x)$. We observed earlier that this implies that $4^x \neq O(2^x)$. ■

Constant Confusion

Every constant is $O(1)$. For example, $17 = O(1)$. This is true because if we let $f(x) = 17$ and $g(x) = 1$, then there exists a $c > 0$ and an x_0 such that $|f(x)| \leq cg(x)$. In particular, we could choose c = 17 and $x_0 = 1$, since $|17| \leq 17 \cdot 1$ for all $x \geq 1$. We can construct a false theorem that exploits this fact.

False Theorem 13.5.14.

$$\sum_{i=1}^{n} i = O(n)$$

False proof. Define $f(n) = \sum_{i=1}^{n} i = 1 + 2 + 3 + \cdots + n$. Since we have shown that every constant i is $O(1)$, $f(n) = O(1) + O(1) + \cdots + O(1) = O(n)$. ■

Of course in reality $\sum_{i=1}^{n} i = n(n+1)/2 \neq O(n)$.

The error stems from confusion over what is meant in the statement $i = O(1)$. For any *constant* $i \in \mathbb{N}$ it is true that $i = O(1)$. More precisely, if f is any constant function, then $f = O(1)$. But in this False Theorem, i is not constant but ranges over a set of values 0,1,…,n that depends on n.

And anyway, we should not be adding $O(1)$'s as though they were numbers. We never even defined what $O(g)$ means by itself; it should only be used in the context "$f = O(g)$" to describe a relation between functions f and g.

---

[4]Since $\Theta(n^3)$ only implies that the running time, $T(n)$, is between cn^3 and dn^3 for constants $0 < c < d$, the time $T(2n)$ could regularly exceed $T(n)$ by a factor as large as $8d/c$. The factor is sure to be close to 8 for all large n only if $T(n) \sim n^3$.

Lower Bound Blunder

Sometimes people incorrectly use Big Oh in the context of a lower bound. For example, they might say, "The running time, $T(n)$, is at least $O(n^2)$," when they probably mean something like "$O(T(n)) = n^2$," or more properly, "$n^2 = O(T(n))$."

Equality Blunder

The notation $f = O(g)$ is too firmly entrenched to avoid, but the use of "=" is really regrettable. For example, if $f = O(g)$, it seems quite reasonable to write $O(g) = f$. But doing so might tempt us to the following blunder: because $2n = O(n)$, we can say $O(n) = 2n$. But $n = O(n)$, so we conclude that $n = O(n) = 2n$, and therefore $n = 2n$. To avoid such nonsense, we will never write "$O(f) = g$."

13.5.5 Problems

Practice Problems

Problem 13.7.
Let $f(n) = n^3$. For each function $g(n)$ in the table below, indicate which of the indicated asymptotic relations hold.

$g(n)$	$f = O(g)$	$f = o(g)$	$g = O(f)$	$g = o(f)$
$6 - 5n - 4n^2 + 3n^3$				
$n^3 \log n$				
$(\sin(\pi n/2) + 2)\, n^3$				
$n^{\sin(\pi n/2)+2}$				
$\log n!$				
$e^{0.2n} - 100n^3$				

Homework Problems

Problem 13.8. **(a)** Prove that $\log x < x$ for all $x > 1$ (requires elementary calculus).

(b) Prove that the relation, R, on functions such that $f \; R \; g$ iff $f = o(g)$ is a strict partial order.

(c) Prove that $f \sim g$ iff $f = g + h$ for some function $h = o(g)$.

Problem 13.9.
Indicate which of the following holds for each pair of functions $(f(n), g(n))$ in the table below. Assume $k \geq 1$, $\epsilon > 0$, and $c > 1$ are constants. Pick the four table entries you consider to be the most challenging or interesting and justify your answers to these.

$f(n)$	$g(n)$	$f = O(g)$	$f = o(g)$	$g = O(f)$	$g = o(f)$	$f = \Theta(g)$	$f \sim g$
2^n	$2^{n/2}$						
$\sqrt{n}$	$n^{\sin n\pi/2}$						
$\log(n!)$	$\log(n^n)$						
n^k	c^n						
$\log^k n$	n^ϵ						

Problem 13.10.
Let f, g be nonnegative real-valued functions such that $\lim_{x\to\infty} f(x) = \infty$ and $f \sim g$.

(a) Give an example of f, g such that $\text{NOT}(2^f \sim 2^g)$.

(b) Prove that $\log f \sim \log g$.

(c) Use Stirling's formula to prove that in fact

$$\log(n!) \sim n \log n$$

Class Problems

Problem 13.11.
Give an elementary proof (without appealing to Stirling's formula) that $\log(n!) = \Theta(n \log n)$.

Problem 13.12.
Recall that for functions f, g on $\mathbb{N}$, $f = O(g)$ iff

$$\exists c \in \mathbb{N}\, \exists n_0 \in \mathbb{N}\, \forall n \geq n_0 \quad c \cdot g(n) \geq |f(n)|\,. \tag{13.14}$$

For each pair of functions below, determine whether $f = O(g)$ and whether $g = O(f)$. In cases where one function is O() of the other, indicate the *smallest nonnegative integer*, c, and for that smallest c, the *smallest corresponding nonnegative integer* n_0 ensuring that condition (13.14) applies.

(a) $f(n) = n^2, g(n) = 3n$.

$f = O(g)$ YES NO If YES, $c =$ ________, n_0 = ________

$g = O(f)$ YES NO If YES, $c =$ ________, n_0 = ________

(b) $f(n) = (3n - 7)/(n + 4), g(n) = 4$

$f = O(g)$ YES NO If YES, $c =$ ________, n_0 = ________

$g = O(f)$ YES NO If YES, $c =$ ________, n_0 = ________

(c) $f(n) = 1 + (n \sin(n\pi/2))^2, g(n) = 3n$

$f = O(g)$	YES	NO	If yes, $c =$ _______ n_0 = _______
$g = O(f)$	YES	NO	If yes, $c =$ _______ n_0 = _______

Problem 13.13.

False Claim.

$$2^n = O(1). \tag{13.15}$$

Explain why the claim is false. Then identify and explain the mistake in the following bogus proof.

Bogus proof. The proof by induction on n where the induction hypothesis, $P(n)$, is the assertion (13.15).

base case: $P(0)$ holds trivially.

inductive step: We may assume $P(n)$, so there is a constant $c > 0$ such that $2^n \leq c \cdot 1$. Therefore,

$$2^{n+1} = 2 \cdot 2^n \leq (2c) \cdot 1,$$

which implies that $2^{n+1} = O(1)$. That is, $P(n+1)$ holds, which completes the proof of the inductive step.

We conclude by induction that $2^n = O(1)$ for all n. That is, the exponential function is bounded by a constant.

■

Problem 13.14.

(a) Define a function $f(n)$ such that $f = \Theta(n^2)$ and $\text{NOT}(f \sim n^2)$.

(b) Define a function $g(n)$ such that $g = O(n^2)$, $g \neq \Theta(n^2)$ and $g \neq o(n^2)$.

Chapter 14

Counting

14.1 Why Count?

Are there two different subsets of the ninety 25-digit numbers shown below that have the same sum —for example, maybe the sum of the numbers in the first column is equal to the sum of the numbers in the second column?

0020480135385502964448038	3171004832173501394113017
5763257331083479647409398	8247331000042995311646021
0489445991866915676240992	3208234421597368647019265
5800949123548989122628663	8496243997123475922766310
1082662032430379651370981	3437254656355157864869113
6042900801199280218026001	8518399140676002660747477
1178480894769706178994993	3574883393058653923711365
6116171789137737896701405	8543691283470191452333763
1253127351683239693851327	3644909946040480189969149
6144868973001582369723512	8675309258374137092461352
1301505129234077811069011	3790044132737084094417246
6247314593851169234746152	8694321112363996867296665
1311567111143866433882194	3870332127437971355322815
6814428944266874963488274	8772321203608477245851154
1470029452721203587686214	4080505804577801451363100
6870852945543886849147881	8791422161722582546341091
1578271047286257499433886	4167283461025702348124920
6914955508120950093732397	9062628024592126283973285
1638243921852176243192354	4235996831123777788211249
6949632451365987152423541	9137845566925526349897794
1763580219131985963102365	4670939445749439042111220
7128211143613619828415650	9153762966803189291934419
1826227795601842231029694	4815379351865384279613427
7173920083651862307925394	9270880194077636406984249
1843971862675102037201420	4837052948212922604442190
7215654874211755676220587	9324301480722103490379204
2396951193722134526177237	5106389423855018550671530
7256932847164391040233050	9436090832146695147140581
2781394568268599801096354	5142368192004769218069910
7332822657075235431620317	9475308159734538249013238
2796605196713610405408019	5181234096130144084041856
7426441829541573444964139	9492376623917486974923202
2931016394761975263190347	5198267398125617994391348
7632198126531809327186321	9511972558779880288252979
2933458058294405155197296	5317592940316231219758372
7712154432211912882310511	9602413424619187112552264
3075514410490975920315348	5384358126771794128356947

7858918664240262356610010	9631217114906129219461111
8149436716871371161932035	3157693105325111284321993
3111474985252793452860017	5439211712248901995423441
7898156786763212963178679	9908189853102753335981319
3145621587936120118438701	5610379826092838192760458
8147591017037573337848616	9913237476341764299813987
3148901255628881103198549	5632317555465228677676044
5692168374637019617423712	8176063831682536571306791

Finding two subsets with the same sum may seem like an silly puzzle, but solving problems like this turns out to be useful, for example in finding good ways to fit packages into shipping containers and in decoding secret messages.

The answer to the question turns out to be "yes." Of course this would be easy to confirm just by showing two subsets with the same sum, but that turns out to be kind of hard to do. So before we put a lot of effort into finding such a pair, it would be nice to be sure there were some. Fortunately, it *is* very easy to see why there is such a pair —or at least it will be easy once we have developed a few simple rules for counting things.

The Contest to Find Two Sets with the Same Sum

One term one of us authors offered a $100 prize to the first student to actually *find* two different subsets of the above ninety 25-digit numbers that have the same sum. We didn't expect to have to pay off this bet, but we underestimated the ingenuity and initiative of the students. One computer science major wrote a program that cleverly searched only among a reasonably small set of "plausible" sets, sorted them by their sums, and actually found a couple with the same sum. He won the prize. A few days later, a math major figured out how to reformulate the sum problem as a "lattice basis reduction" problem; then he found a software package implementing an efficient basis reduction procedure, and using it, he very quickly found lots of pairs of subsets with the same sum. He didn't win the prize, but he got a standing ovation from the class —staff included.

Counting seems easy enough: 1, 2, 3, 4, etc. This direct approach works well for counting simple things —like your toes —and may be the only approach for extremely complicated things with no identifiable structure. However, subtler methods can help you count many things in the vast middle ground, such as:

- The number of different ways to select a dozen doughnuts when there are five varieties available.

- The number of 16-bit numbers with exactly 4 ones.

Counting is useful in computer science for several reasons:

- Determining the time and storage required to solve a computational problem —a central objective in computer science —often comes down to solving a counting problem.
- Counting is the basis of probability theory, which plays a central role in all sciences, including computer science.
- Two remarkable proof techniques, the "pigeonhole principle" and "combinatorial proof," rely on counting. These lead to a variety of interesting and useful insights.

We're going to present a lot of rules for counting. These rules are actually theorems, but most of them are pretty obvious anyway, so we're not going to focus on proving them. Our objective is to teach you simple counting as a practical skill, like integration.

14.2 Counting One Thing by Counting Another

How do you count the number of people in a crowded room? You could count heads, since for each person there is exactly one head. Alternatively, you could count ears and divide by two. Of course, you might have to adjust the calculation if someone lost an ear in a pirate raid or someone was born with three ears. The point here is that you can often *count one thing by counting another*, though some fudge factors may be required.

In more formal terms, every counting problem comes down to determining the size of some set. The *size* or *cardinality* of a finite set, S, is the number of elements in it and is denoted $|S|$. In these terms, we're claiming that we can often find the size of one set by finding the size of a related set. We've already seen a general statement of this idea in the Mapping Rule of Lemma 5.5.2.

14.2.1 The Bijection Rule

We've already implicitly used the Bijection Rule of Lemma 3 a lot. For example, when we studied Stable Marriage and Bipartite Matching, we assumed the obvious fact that if we can pair up all the girls at a dance with all the boys, then there must be an equal number of each. If we needed to be explicit about using the Bijection Rule, we could say that A was the set of boys, B was the set of girls, and the bijection between them was how they were paired.

The Bijection Rule acts as a magnifier of counting ability; if you figure out the size of one set, then you can immediately determine the sizes of many other sets via bijections. For example, let's return to two sets mentioned earlier:

$A =$ all ways to select a dozen doughnuts when five varieties are available

$B =$ all 16-bit sequences with exactly 4 ones

Let's consider a particular element of set A:

$$\underbrace{0\,0}_{\text{chocolate}} \quad \underbrace{}_{\text{lemon-filled}} \quad \underbrace{0\,0\,0\,0\,0\,0}_{\text{sugar}} \quad \underbrace{0\,0}_{\text{glazed}} \quad \underbrace{0\,0}_{\text{plain}}$$

We've depicted each doughnut with a 0 and left a gap between the different varieties. Thus, the selection above contains two chocolate doughnuts, no lemon-filled, six sugar, two glazed, and two plain. Now let's put a 1 into each of the four gaps:

$$\underbrace{0\,0}_{\text{chocolate}} \quad 1 \quad \underbrace{}_{\text{lemon-filled}} \quad 1 \quad \underbrace{0\,0\,0\,0\,0\,0}_{\text{sugar}} \quad 1 \quad \underbrace{0\,0}_{\text{glazed}} \quad 1 \quad \underbrace{0\,0}_{\text{plain}}$$

We've just formed a 16-bit number with exactly 4 ones— an element of B!

This example suggests a bijection from set A to set B: map a dozen doughnuts consisting of:

c chocolate, l lemon-filled, s sugar, g glazed, and p plain

to the sequence:

$$\underbrace{0 \ldots 0}_{c} \quad 1 \quad \underbrace{0 \ldots 0}_{l} \quad 1 \quad \underbrace{0 \ldots 0}_{s} \quad 1 \quad \underbrace{0 \ldots 0}_{g} \quad 1 \quad \underbrace{0 \ldots 0}_{p}$$

The resulting sequence always has 16 bits and exactly 4 ones, and thus is an element of B. Moreover, the mapping is a bijection; every such bit sequence is mapped to by exactly one order of a dozen doughnuts. Therefore, $|A| = |B|$ by the Bijection Rule!

This demonstrates the magnifying power of the bijection rule. We managed to prove that two very different sets are actually the same size— even though we don't know exactly how big either one is. But as soon as we figure out the size of one set, we'll immediately know the size of the other.

This particular bijection might seem frighteningly ingenious if you've not seen it before. But you'll use essentially this same argument over and over, and soon you'll consider it routine.

14.2.2 Counting Sequences

The Bijection Rule lets us count one thing by counting another. This suggests a general strategy: get really good at counting just a *few* things and then use bijections to count *everything else*. This is the strategy we'll follow. In particular, we'll get really good at counting *sequences*. When we want to determine the size of some other set T, we'll find a bijection from T to a set of sequences S. Then we'll use our super-ninja sequence-counting skills to determine $|S|$, which immediately gives us $|T|$. We'll need to hone this idea somewhat as we go along, but that's pretty much the plan!

14.2.3 The Sum Rule

Linus allocates his big sister Lucy a quota of 20 crabby days, 40 irritable days, and 60 generally surly days. On how many days can Lucy be out-of-sorts one way or another? Let set C be her crabby days, I be her irritable days, and S be the generally surly. In these terms, the answer to the question is $|C \cup I \cup S|$. Now assuming that she is permitted at most one bad quality each day, the size of this union of sets is given by the *Sum Rule*:

Rule 1 (Sum Rule)**.** *If $A_1, A_2, \ldots, A_n$ are* disjoint *sets, then:*

$$|A_1 \cup A_2 \cup \ldots \cup A_n| = |A_1| + |A_2| + \ldots + |A_n|$$

Thus, according to Linus' budget, Lucy can be out-of-sorts for:

$$\begin{aligned} |C \cup I \cup S| &= |C| + |I| + |S| \\ &= 20 + 40 + 60 \\ &= 120 \text{ days} \end{aligned}$$

Notice that the Sum Rule holds only for a union of *disjoint* sets. Finding the size of a union of intersecting sets is a more complicated problem that we'll take up later.

14.2.4 The Product Rule

The *Product Rule* gives the size of a product of sets. Recall that if $P_1, P_2, \ldots, P_n$ are sets, then

$$P_1 \times P_2 \times \ldots \times P_n$$

is the set of all sequences whose first term is drawn from P_1, second term is drawn from P_2 and so forth.

Rule 2 (Product Rule)**.** *If $P_1, P_2, \ldots P_n$ are sets, then:*

$$|P_1 \times P_2 \times \ldots \times P_n| = |P_1| \cdot |P_2| \cdots |P_n|$$

Unlike the sum rule, the product rule does not require the sets $P_1, \ldots, P_n$ to be disjoint. For example, suppose a *daily diet* consists of a breakfast selected from set B, a lunch from set L, and a dinner from set D:

$$\begin{aligned} B &= \{\text{pancakes, bacon and eggs, bagel, Doritos}\} \\ L &= \{\text{burger and fries, garden salad, Doritos}\} \\ D &= \{\text{macaroni, pizza, frozen burrito, pasta, Doritos}\} \end{aligned}$$

Then $B \times L \times D$ is the set of all possible daily diets. Here are some sample elements:

$$\begin{gathered} (\text{pancakes, burger and fries, pizza}) \\ (\text{bacon and eggs, garden salad, pasta}) \\ (\text{Doritos, Doritos, frozen burrito}) \end{gathered}$$

The Product Rule tells us how many different daily diets are possible:

$$\begin{aligned} |B \times L \times D| &= |B| \cdot |L| \cdot |D| \\ &= 4 \cdot 3 \cdot 5 \\ &= 60 \end{aligned}$$

14.2.5 Putting Rules Together

Few counting problems can be solved with a single rule. More often, a solution is a flurry of sums, products, bijections, and other methods. Let's look at some examples that bring more than one rule into play.

Counting Passwords

The sum and product rules together are useful for solving problems involving passwords, telephone numbers, and license plates. For example, on a certain computer system, a valid password is a sequence of between six and eight symbols. The first symbol must be a letter (which can be lowercase or uppercase), and the remaining symbols must be either letters or digits. How many different passwords are possible?

Let's define two sets, corresponding to valid symbols in the first and subsequent positions in the password.

$$\begin{aligned} F &= \{a, b, \ldots, z, A, B, \ldots, Z\} \\ S &= \{a, b, \ldots, z, A, B, \ldots, Z, 0, 1, \ldots, 9\} \end{aligned}$$

In these terms, the set of all possible passwords is:

$$(F \times S^5) \cup (F \times S^6) \cup (F \times S^7)$$

Thus, the length-six passwords are in set $F \times S^5$, the length-seven passwords are in $F \times S^6$, and the length-eight passwords are in $F \times S^7$. Since these sets are disjoint, we can apply the Sum Rule and count the total number of possible passwords as follows:

$$\begin{aligned} \left|(F \times S^5) \cup (F \times S^6) \cup (F \times S^7)\right| &= \left|F \times S^5\right| + \left|F \times S^6\right| + \left|F \times S^7\right| && \text{Sum Rule} \\ &= |F| \cdot |S|^5 + |F| \cdot |S|^6 + |F| \cdot |S|^7 && \text{Product Rule} \\ &= 52 \cdot 62^5 + 52 \cdot 62^6 + 52 \cdot 62^7 \\ &\approx 1.8 \cdot 10^{14} \text{ different passwords} \end{aligned}$$

Subsets of an n-element Set

How many different subsets of an n-element set X are there? For example, the set $X = \{x_1, x_2, x_3\}$ has eight different subsets:

$$\begin{array}{cccc} \emptyset & \{x_1\} & \{x_2\} & \{x_1, x_2\} \\ \{x_3\} & \{x_1, x_3\} & \{x_2, x_3\} & \{x_1, x_2, x_3\} \end{array}$$

There is a natural bijection from subsets of X to n-bit sequences. Let $x_1, x_2, \ldots, x_n$ be the elements of X. Then a particular subset of X maps to the sequence $(b_1, \ldots, b_n)$ where $b_i = 1$ if and only if x_i is in that subset. For example, if $n = 10$, then the subset $\{x_2, x_3, x_5, x_7, x_{10}\}$ maps to a 10-bit sequence as follows:

$$\begin{array}{rcccccccccccc} \text{subset:} & \{ & & x_2, & x_3, & & x_5, & & x_7, & & & x_{10} & \} \\ \text{sequence:} & (& 0, & 1, & 1, & 0, & 1, & 0, & 1, & 0, & 0, & 1 &) \end{array}$$

We just used a bijection to transform the original problem into a question about sequences —*exactly according to plan!* Now if we answer the sequence question, then we've solved our original problem as well.

But how many different n-bit sequences are there? For example, there are 8 different 3-bit sequences:

$$\begin{array}{cccc} (0,0,0) & (0,0,1) & (0,1,0) & (0,1,1) \\ (1,0,0) & (1,0,1) & (1,1,0) & (1,1,1) \end{array}$$

Well, we can write the set of all n-bit sequences as a product of sets:

$$\underbrace{\{0,1\} \times \{0,1\} \times \ldots \times \{0,1\}}_{n \text{ terms}} = \{0,1\}^n$$

Then Product Rule gives the answer:

$$\begin{aligned} |\{0,1\}^n| &= |\{0,1\}|^n \\ &= 2^n \end{aligned}$$

This means that the number of subsets of an n-element set X is also 2^n. We'll put this answer to use shortly.

14.2.6 Problems

Practice Problems

Problem 14.1.
How many ways are there to select k out of n books on a shelf so that there are always at least 3 unselected books between selected books? (Assume n is large enough for this to be possible.)

Class Problems

Problem 14.2.
A license plate consists of either:

- 3 letters followed by 3 digits (standard plate)
- 5 letters (vanity plate)

- 2 characters – letters or numbers (big shot plate)

Let L be the set of all possible license plates.

(a) Express L in terms of

$$\mathcal{A} = \{A, B, C, \ldots, Z\}$$
$$\mathcal{D} = \{0, 1, 2, \ldots, 9\}$$

using unions ($\cup$) and set products ($\times$).

(b) Compute $|L|$, the number of different license plates, using the sum and product rules.

Problem 14.3. **(a)** How many of the billion numbers in the range from 1 to 10^9 contain the digit 1? (*Hint:* How many don't?)

(b) There are 20 books arranged in a row on a shelf. Describe a bijection between ways of choosing 6 of these books so that no two adjacent books are selected and 15-bit strings with exactly 6 ones.

Problem 14.4.

(a) Let $\mathcal{S}_{n,k}$ be the possible nonnegative integer solutions to the inequality

$$x_1 + x_2 + \cdots + x_k \leq n. \tag{14.1}$$

That is

$$\mathcal{S}_{n,k} ::= \left\{(x_1, x_2, \ldots, x_k) \in \mathbb{N}^k \mid \text{(14.1) is true}\right\}.$$

Describe a bijection between $\mathcal{S}_{n,k}$ and the set of binary strings with n zeroes and k ones.

(b) Let $\mathcal{L}_{n,k}$ be the length k weakly increasing sequences of nonnegative integers $\leq n$. That is

$$\mathcal{L}_{n,k} ::= \left\{(y_1, y_2, \ldots, y_k) \in \mathbb{N}^k \mid y_1 \leq y_2 \leq \cdots \leq y_k \leq n\right\}.$$

Describe a bijection between $\mathcal{L}_{n,k}$ and $\mathcal{S}_{n,k}$.

Problem 14.5.
An n-vertex *numbered tree* is a tree whose vertex set is $\{1, 2, \ldots, n\}$ for some $n > 2$. We define the *code* of the numbered tree to be a sequence of $n - 2$ integers from 1 to n obtained by the following recursive process:

If there are more than two vertices left, write down the *father* of the largest leaf[a], delete this *leaf*, and continue this process on the resulting smaller tree.

If there are only two vertices left, then stop —the code is complete.

[a]The necessarily unique node adjacent to a leaf is called its *father*.

For example, the codes of a couple of numbered trees are shown in the Figure 14.1.

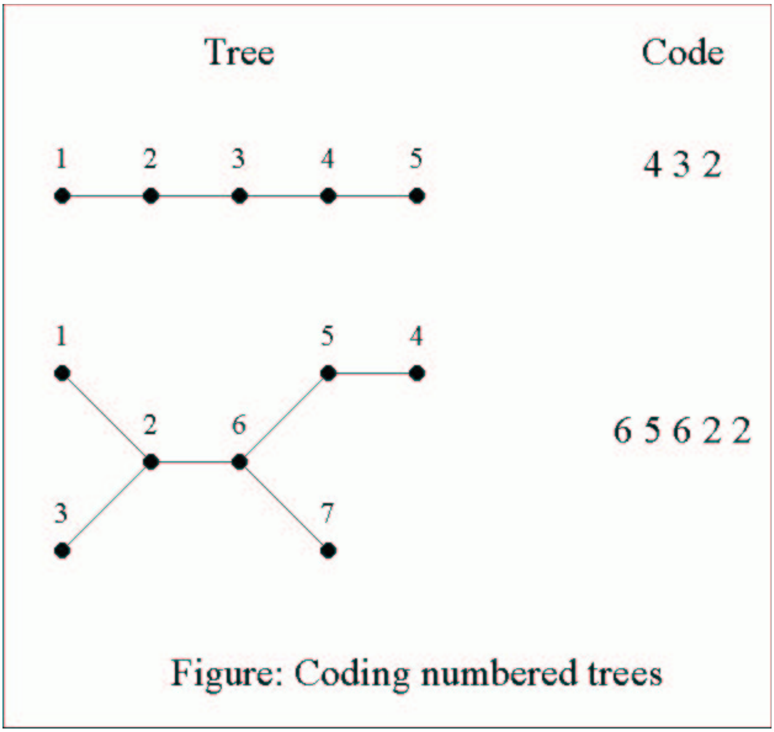


Figure 14.1:

(a) Describe a procedure for reconstructing a numbered tree from its code.

(b) Conclude there is a bijection between the n-vertex numbered trees and $\{1, \ldots, n\}^{n-2}$, and state how many n-vertex numbered trees there are.

Homework Problems

Problem 14.6.
Answer the following questions with a number or a simple formula involving factorials and binomial coefficients. Briefly explain your answers.

(a) How many ways are there to order the 26 letters of the alphabet so that no two of the vowels `a`, `e`, `i`, `o`, `u` appear consecutively and the last letter in the ordering is not a vowel?

Hint: Every vowel appears to the left of a consonant.

(b) How many ways are there to order the 26 letters of the alphabet so that there are *at least two* consonants immediately following each vowel?

(c) In how many different ways can $2n$ students be paired up?

(d) Two n-digit sequences of digits 0,1,...,9 are said to be of the *same type* if the digits of one are a permutation of the digits of the other. For $n = 8$, for example, the sequences `03088929` and `00238899` are the same type. How many types of n-digit integers are there?

Problem 14.7.
In a standard 52-card deck, each card has one of thirteen *rank*s in the set, R, and one of four *suit*s in the set, S, where

$$R ::= \{A, 2, \ldots, 10, J, Q, K\},$$
$$S ::= \{\clubsuit, \diamondsuit, \heartsuit, \spadesuit\}.$$

A 5-card *hand* is a set of five distinct cards from the deck.

For each part describe a bijection between a set that can easily be counted using the Product and Sum Rules of Ch. 14.2, and the set of hands matching the specification. *Give bijections, not numerical answers.*

For instance, consider the set of 5-card hands containing all 4 suits. Each such hand must have 2 cards of one suit. We can describe a bijection between such hands and the set $S \times R_2 \times R^3$ where R_2 is the set of two-element subsets of R. Namely, an element

$$(s, \{r_1, r_2\}, (r_3, r_4, r_5)) \in S \times R_2 \times R^3$$

indicates

1. the repeated suit, $s \in S$,
2. the set, $\{r_1, r_2\} \in R_2$, of ranks of the cards of suit, s, and
3. the ranks (r_3, r_4, r_5) of remaining three cards, listed in increasing suit order where $\clubsuit \prec \diamondsuit \prec \heartsuit \prec \spadesuit$.

For example,

$$(\clubsuit, \{10, A\}, (J, J, 2)) \longleftrightarrow \{A\clubsuit, 10\clubsuit, J\diamondsuit, J\heartsuit, 2\spadesuit\}.$$

(a) A single pair of the same rank (no 3-of-a-kind, 4-of-a-kind, or second pair).

(b) Three or more aces.

14.3 The Pigeonhole Principle

Here is an old puzzle:

> A drawer in a dark room contains red socks, green socks, and blue socks. How many socks must you withdraw to be sure that you have a matching pair?

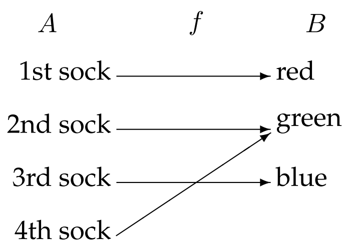
For example, picking out three socks is not enough; you might end up with one red, one green, and one blue. The solution relies on the *Pigeonhole Principle*, which is a friendly name for the contrapositive of the injective case 2 of the Mapping Rule of Lemma 5.5.2. Let's write it down:

> If $|X| > |Y|$, then no total function[1] $f : X \to Y$ is injective.

And now rewrite it again to eliminate the word "injective."

Rule 3 (Pigeonhole Principle)**.** *If $|X| > |Y|$, then for every total function $f : X \to Y$, there exist two different elements of X that are mapped to the same element of Y.*

What this abstract mathematical statement has to do with selecting footwear under poor lighting conditions is maybe not obvious. However, let A be the set of socks you pick out, let B be the set of colors available, and let f map each sock to its color. The Pigeonhole Principle says that if $|A| > |B| = 3$, then at least two elements of A (that is, at least two socks) must be mapped to the same element of B (that is, the same color). For example, one possible mapping of four socks to three colors is shown below.



[1]This Mapping Rule actually applies even if f is a total injective *relation*.

Therefore, four socks are enough to ensure a matched pair.

Not surprisingly, the pigeonhole principle is often described in terms of pigeons:

> *If there are more pigeons than holes they occupy, then at least two pigeons must be in the same hole.*

In this case, the pigeons form set A, the pigeonholes are set B, and f describes which hole each pigeon flies into.

Mathematicians have come up with many ingenious applications for the pigeonhole principle. If there were a cookbook procedure for generating such arguments, we'd give it to you. Unfortunately, there isn't one. One helpful tip, though: when you try to solve a problem with the pigeonhole principle, the key is to clearly identify three things:

1. The set A (the pigeons).
2. The set B (the pigeonholes).
3. The function f (the rule for assigning pigeons to pigeonholes).

14.3.1 Hairs on Heads

There are a number of generalizations of the pigeonhole principle. For example:

Rule 4 (Generalized Pigeonhole Principle)**.** *If $|X| > k \cdot |Y|$, then every total function $f : X \to Y$ maps at least $k+1$ different elements of X to the same element of Y.*

For example, if you pick two people at random, surely they are extremely unlikely to have *exactly* the same number of hairs on their heads. However, in the remarkable city of Boston, Massachusetts there are actually *three* people who have exactly the same number of hairs! Of course, there are many bald people in Boston, and they all have zero hairs. But we're talking about non-bald people; say a person is non-bald if they have at least ten thousand hairs on their head.

Boston has about 500,000 non-bald people, and the number of hairs on a person's head is at most 200,000. Let A be the set of non-bald people in Boston, let $B = \{10,000, 10,001, \dots, 200,000\}$, and let f map a person to the number of hairs on his or her head. Since $|A| > 2\,|B|$, the Generalized Pigeonhole Principle implies that at least three people have exactly the same number of hairs. We don't know who they are, but we know they exist!

14.3.2 Subsets with the Same Sum

We asserted that two different subsets of the ninety 25-digit numbers listed on the first page have the same sum. This actually follows from the Pigeonhole Principle. Let A be the collection of all subsets of the 90 numbers in the list. Now the sum of any subset of numbers is at most $90 \cdot 10^{25}$, since there are only 90 numbers and every

25-digit number is less than 10^{25}. So let B be the set of integers $\{0, 1, \ldots, 90 \cdot 10^{25}\}$, and let f map each subset of numbers (in A) to its sum (in B).

We proved that an n-element set has 2^n different subsets. Therefore:

$$\begin{aligned}|A| &= 2^{90}\\ &\geq 1.237 \times 10^{27}\end{aligned}$$

On the other hand:

$$\begin{aligned}|B| &= 90 \cdot 10^{25} + 1\\ &\leq 0.901 \times 10^{27}\end{aligned}$$

Both quantities are enormous, but $|A|$ is a bit greater than $|B|$. This means that f maps at least two elements of A to the same element of B. In other words, by the Pigeonhole Principle, two different subsets must have the same sum!

Notice that this proof gives no indication *which* two sets of numbers have the same sum. This frustrating variety of argument is called a *nonconstructive proof*.

Sets with Distinct Subset Sums

How can we construct a set of n positive integers such that all its subsets have *distinct* sums? One way is to use powers of two:

$$\{1, 2, 4, 8, 16\}$$

This approach is so natural that one suspects all other such sets must involve larger numbers. (For example, we could safely replace 16 by 17, but not by 15.) Remarkably, there are examples involving *smaller* numbers. Here is one:

$$\{6, 9, 11, 12, 13\}$$

One of the top mathematicans of the Twentieth Century, Paul Erdős, conjectured in 1931 that there are no such sets involving *significantly* smaller numbers. More precisely, he conjectured that the largest number must be $> c2^n$ for some constant $c > 0$. He offered \$500 to anyone who could prove or disprove his conjecture, but the problem remains unsolved.

14.3.3 Problems

Class Problems

Problem 14.8.
Solve the following problems using the pigeonhole principle. For each problem,

try to identify the *pigeons*, the *pigeonholes*, and a *rule* assigning each pigeon to a pigeonhole.

(a) Every MIT ID number starts with a 9 (we think). Suppose that each of the 75 students in 6.042 sums the nine digits of his or her ID number. Explain why two people must arrive at the same sum.

(b) In every set of 100 integers, there exist two whose difference is a multiple of 37.

(c) For any five points inside a unit square (not on the boundary), there are two points at distance *less than* $1/\sqrt{2}$.

(d) Show that if $n+1$ numbers are selected from $\{1, 2, 3, \ldots, 2n\}$, two must be consecutive, that is, equal to k and $k+1$ for some k.

Homework Problems

Problem 14.9.
Pigeon Huntin'

(a) Show that any odd integer x in the range $10^9 < x < 2 \cdot 10^9$ containing all ten digits $0, 1, \ldots, 9$ must have consecutive even digits. *Hint:* What can you conclude about the parities of the first and last digit?

(b) Show that there are 2 vertices of equal degree in any finite undirected graph with $n \geq 2$ vertices. *Hint:* Cases conditioned upon the existence of a degree zero vertex.

Problem 14.10.
Show that for any set of 201 positive integers less than 300, there must be two whose quotient is a power of three (with no remainder).

14.4 The Generalized Product Rule

We realize everyone has been working pretty hard this term, and we're considering awarding some prizes for *truly exceptional* coursework. Here are some possible categories:

Best Administrative Critique We asserted that the quiz was closed-book. On the cover page, one strong candidate for this award wrote, "There is no book."

Awkward Question Award "Okay, the left sock, right sock, and pants are in an antichain, but how— even with assistance— could I put on all three at once?"

Best Collaboration Statement Inspired by a student who wrote "I worked alone" on Quiz 1.

In how many ways can, say, three different prizes be awarded to n people? This is easy to answer using our strategy of translating the problem about awards into a problem about sequences. Let P be the set of n people taking the course. Then there is a bijection from ways of awarding the three prizes to the set $P^3 ::= P \times P \times P$. In particular, the assignment:

"person x wins prize #1, y wins prize #2, and z wins prize #3"

maps to the sequence (x, y, z). By the Product Rule, we have $\left|P^3\right| = |P|^3 = n^3$, so there are n^3 ways to award the prizes to a class of n people.

But what if the three prizes must be awarded to *different* students? As before, we could map the assignment

"person x wins prize #1, y wins prize #2, and z wins prize #3"

to the triple $(x, y, z) \in P^3$. But this function is *no longer a bijection*. For example, no valid assignment maps to the triple (Dave, Dave, Becky) because Dave is not allowed to receive two awards. However, there *is* a bijection from prize assignments to the set:

$$S = \left\{(x, y, z) \in P^3 \mid x, y, \text{ and } z \text{ are different people}\right\}$$

This reduces the original problem to a problem of counting sequences. Unfortunately, the Product Rule is of no help in counting sequences of this type because the entries depend on one another; in particular, they must all be different. However, a slightly sharper tool does the trick.

Rule 5 (Generalized Product Rule)**.** *Let S be a set of length-k sequences. If there are:*

- *n_1 possible first entries,*
- *n_2 possible second entries for each first entry,*
- *n_3 possible third entries for each combination of first and second entries, etc.*

then:

$$|S| = n_1 \cdot n_2 \cdot n_3 \cdots n_k$$

In the awards example, S consists of sequences (x, y, z). There are n ways to choose x, the recipient of prize #1. For each of these, there are $n - 1$ ways to choose y, the recipient of prize #2, since everyone except for person x is eligible. For each combination of x and y, there are $n - 2$ ways to choose z, the recipient of prize #3, because everyone except for x and y is eligible. Thus, according to the Generalized Product Rule, there are

$$|S| = n \cdot (n - 1) \cdot (n - 2)$$

ways to award the 3 prizes to different people.

14.4.1 Defective Dollars

A dollar is *defective* if some digit appears more than once in the 8-digit serial number. If you check your wallet, you'll be sad to discover that defective dollars are all-too-common. In fact, how common are *nondefective* dollars? Assuming that the digit portions of serial numbers all occur equally often, we could answer this question by computing:

$$\text{fraction dollars that are nondefective} = \frac{\text{\# of serial \#'s with all digits different}}{\text{total \# of serial \#'s}}$$

Let's first consider the denominator. Here there are no restrictions; there are are 10 possible first digits, 10 possible second digits, 10 third digits, and so on. Thus, the total number of 8-digit serial numbers is 10^8 by the Product Rule.

Next, let's turn to the numerator. Now we're not permitted to use any digit twice. So there are still 10 possible first digits, but only 9 possible second digits, 8 possible third digits, and so forth. Thus, by the Generalized Product Rule, there are

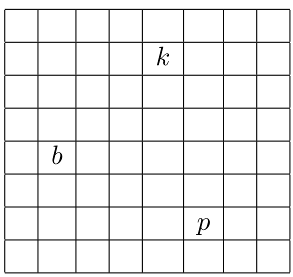
$$\begin{aligned} 10 \cdot 9 \cdot 8 \cdot 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 &= \frac{10!}{2} \\ &= 1,814,400 \end{aligned}$$

serial numbers with all digits different. Plugging these results into the equation above, we find:

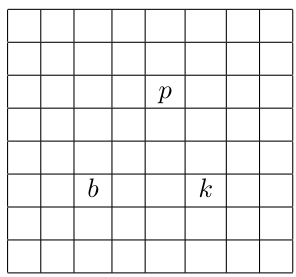
$$\begin{aligned} \text{fraction dollars that are nondefective} &= \frac{1,814,400}{100,000,000} \\ &= 1.8144\% \end{aligned}$$

14.4.2 A Chess Problem

In how many different ways can we place a pawn (p), a knight (k), and a bishop (b) on a chessboard so that no two pieces share a row or a column? A valid configuration is shown below on the left, and an invalid configuration is shown on the right.



valid



invalid

First, we map this problem about chess pieces to a question about sequences. There is a bijection from configurations to sequences

$$(r_p, c_p, r_k, c_k, r_b, c_b)$$

where r_p, r_k, and r_b are distinct rows and c_p, c_k, and c_b are distinct columns. In particular, r_p is the pawn's row, c_p is the pawn's column, r_k is the knight's row, etc. Now we can count the number of such sequences using the Generalized Product Rule:

- r_p is one of 8 rows
- c_p is one of 8 columns
- r_k is one of 7 rows (any one but r_p)
- c_k is one of 7 columns (any one but c_p)
- r_b is one of 6 rows (any one but r_p or r_k)
- c_b is one of 6 columns (any one but c_p or c_k)

Thus, the total number of configurations is $(8 \cdot 7 \cdot 6)^2$.

14.4.3 Permutations

A *permutation* of a set S is a sequence that contains every element of S exactly once. For example, here are all the permutations of the set $\{a, b, c\}$:

$$\begin{array}{ccc} (a, b, c) & (a, c, b) & (b, a, c) \\ (b, c, a) & (c, a, b) & (c, b, a) \end{array}$$

How many permutations of an n-element set are there? Well, there are n choices for the first element. For each of these, there are $n - 1$ remaining choices for the second element. For every combination of the first two elements, there are $n - 2$ ways to choose the third element, and so forth. Thus, there are a total of

$$n \cdot (n-1) \cdot (n-2) \cdots 3 \cdot 2 \cdot 1 = n!$$

permutations of an n-element set. In particular, this formula says that there are $3! = 6$ permuations of the 3-element set $\{a, b, c\}$, which is the number we found above.

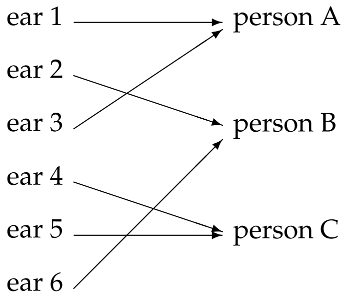
Permutations will come up again in this course approximately 1.6 bazillion times. In fact, permutations are the reason why factorial comes up so often and why we taught you Stirling's approximation:

$$n! \sim \sqrt{2\pi n} \left(\frac{n}{e}\right)^n$$

14.5 The Division Rule

Counting ears and dividing by two is a silly way to count the number of people in a room, but this approach is representative of a powerful counting principle.

A *k-to-1 function* maps exactly k elements of the domain to every element of the codomain. For example, the function mapping each ear to its owner is 2-to-1:



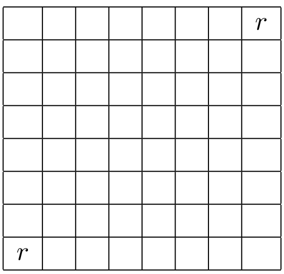
Similarly, the function mapping each finger to its owner is 10-to-1, and the function mapping each finger and toe to its owner is 20-to-1. The general rule is:

Rule 6 (Division Rule)**.** *If $f : A \to B$ is k-to-1, then $|A| = k \cdot |B|$.*

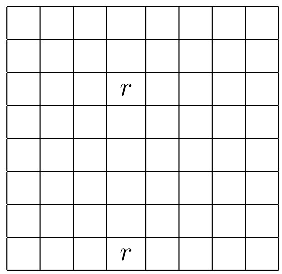
For example, suppose A is the set of ears in the room and B is the set of people. There is a 2-to-1 mapping from ears to people, so by the Division Rule $|A| = 2 \cdot |B|$ or, equivalently, $|B| = |A|/2$, expressing what we knew all along: the number of people is half the number of ears. Unlikely as it may seem, many counting problems are made much easier by initially counting every item multiple times and then correcting the answer using the Division Rule. Let's look at some examples.

14.5.1 Another Chess Problem

In how many different ways can you place two identical rooks on a chessboard so that they do not share a row or column? A valid configuration is shown below on the left, and an invalid configuration is shown on the right.



valid



invalid

Let A be the set of all sequences

$$(r_1, c_1, r_2, c_2)$$

where r_1 and r_2 are distinct rows and c_1 and c_2 are distinct columns. Let B be the set of all valid rook configurations. There is a natural function f from set A to set B; in particular, f maps the sequence (r_1, c_1, r_2, c_2) to a configuration with one rook in row r_1, column c_1 and the other rook in row r_2, column c_2.

But now there's a snag. Consider the sequences:

$$(1, 1, 8, 8) \qquad \text{and} \qquad (8, 8, 1, 1)$$

The first sequence maps to a configuration with a rook in the lower-left corner and a rook in the upper-right corner. The second sequence maps to a configuration with a rook in the upper-right corner and a rook in the lower-left corner. The problem is that those are two different ways of describing the *same* configuration! In fact, this arrangement is shown on the left side in the diagram above.

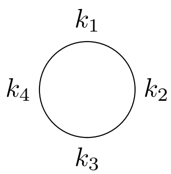
More generally, the function f maps exactly two sequences to *every* board configuration; that is f is a 2-to-1 function. Thus, by the quotient rule, $|A| = 2 \cdot |B|$. Rearranging terms gives:

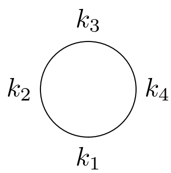
$$\begin{aligned} |B| &= \frac{|A|}{2} \\ &= \frac{(8 \cdot 7)^2}{2} \end{aligned}$$

On the second line, we've computed the size of A using the General Product Rule just as in the earlier chess problem.

14.5.2 Knights of the Round Table

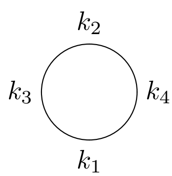
In how many ways can King Arthur seat n different knights at his round table? Two seatings are considered equivalent if one can be obtained from the other by rotation. For example, the following two arrangements are equivalent:





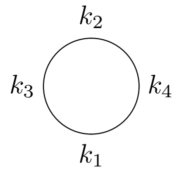
Let A be all the permutations of the knights, and let B be the set of all possible seating arrangements at the round table. We can map each permutation in set A to a circular seating arrangement in set B by seating the first knight in the permutation anywhere, putting the second knight to his left, the third knight to the left of the second, and so forth all the way around the table. For example:

$(k_2, k_4, k_1, k_3) \quad \longrightarrow$



This mapping is actually an n-to-1 function from A to B, since all n cyclic shifts of the original sequence map to the same seating arrangement. In the example, $n = 4$ different sequences map to the same seating arrangement:

$$\begin{array}{c} (k_2, k_4, k_1, k_3) \\ (k_4, k_1, k_3, k_2) \\ (k_1, k_3, k_2, k_4) \\ (k_3, k_2, k_4, k_1) \end{array} \longrightarrow$$



Therefore, by the division rule, the number of circular seating arrangements is:

$$\begin{aligned} |B| &= \frac{|A|}{n} \\ &= \frac{n!}{n} \\ &= (n-1)! \end{aligned}$$

Note that $|A| = n!$ since there are $n!$ permutations of n knights.

14.5.3 Problems

Class Problems

Problem 14.11.
Your 6.006 tutorial has 12 students, who are supposed to break up into 4 groups of 3 students each. Your TA has observed that the students waste too much time trying to form balanced groups, so he decided to pre-assign students to groups and email the group assignments to his students.

(a) Your TA has a list of the 12 students in front of him, so he divides the list into consecutive groups of 3. For example, if the list is ABCDEFGHIJKL, the TA would define a sequence of four groups to be $(\{A, B, C\}, \{D, E, F\}, \{G, H, I\}, \{J, K, L\})$. This way of forming groups defines a mapping from a list of twelve students to a sequence of four groups. This is a k-to-1 mapping for what k?

(b) A group assignment specifies which students are in the same group, but not any order in which the groups should be listed. If we map a sequence of 4 groups,

$$(\{A, B, C\}, \{D, E, F\}, \{G, H, I\}, \{J, K, L\}),$$

into a group assignment

$$\{\{A, B, C\}, \{D, E, F\}, \{G, H, I\}, \{J, K, L\}\},$$

this mapping is j-to-1 for what j?

(c) How many group assignments are possible?

(d) In how many ways can $3n$ students be broken up into n groups of 3?

Problem 14.12.
A pizza house is having a promotional sale. Their commercial reads:

> We offer 9 different toppings for your pizza! Buy 3 large pizzas at the regular price, and you can get each one with as many different toppings as you wish, absolutely free. That's $22,369,621$ different ways to choose your pizzas!

The ad writer was a former Harvard student who had evaluated the formula $(2^9)^3/3!$ on his calculator and gotten close to $22,369,621$. Unfortunately, $(2^9)^3/3!$ is obviously not an integer, so clearly something is wrong. What mistaken reasoning might have led the ad writer to this formula? Explain how to fix the mistake and get a correct formula.

Problem 14.13.
Answer the following quesions using the Generalized Product Rule.

(a) Next week, I'm going to get really fit! On day 1, I'll exercise for 5 minutes. On each subsequent day, I'll exercise 0, 1, 2, or 3 minutes more than the previous day. For example, the number of minutes that I exercise on the seven days of next week might be 5, 6, 9, 9, 9, 11, 12. How many such sequences are possible?

(b) An *r-permutation* of a set is a sequence of r distinct elements of that set. For example, here are all the 2-permutations of $\{a, b, c, d\}$:

$$\begin{array}{ccc} (a,b) & (a,c) & (a,d) \\ (b,a) & (b,c) & (b,d) \\ (c,a) & (c,b) & (c,d) \\ (d,a) & (d,b) & (d,c) \end{array}$$

How many r-permutations of an n-element set are there? Express your answer using factorial notation.

(c) How many $n \times n$ matrices are there with *distinct* entries drawn from $\{1, \ldots, p\}$, where $p \geq n^2$?

Exam Problems

Problem 14.14.
Suppose that two identical 52-card decks are mixed together. Write a simple formula for the number of 104-card double-deck mixes that are possible.

14.6 Counting Subsets

How many k-element subsets of an n-element set are there? This question arises all the time in various guises:

- In how many ways can I select 5 books from my collection of 100 to bring on vacation?
- How many different 13-card Bridge hands can be dealt from a 52-card deck?
- In how many ways can I select 5 toppings for my pizza if there are 14 available toppings?

This number comes up so often that there is a special notation for it:

$$\binom{n}{k} ::= \text{ the number of } k\text{-element subsets of an } n\text{-element set.}$$

The expression $\binom{n}{k}$ is read "n choose k." Now we can immediately express the answers to all three questions above:

- I can select 5 books from 100 in $\binom{100}{5}$ ways.
- There are $\binom{52}{13}$ different Bridge hands.
- There are $\binom{14}{5}$ different 5-topping pizzas, if 14 toppings are available.

14.6.1 The Subset Rule

We can derive a simple formula for the n-choose-k number using the Division Rule. We do this by mapping any permutation of an n-element set $\{a_1, \ldots, a_n\}$ into a k-element subset simply by taking the first k elements of the permutation. That is, the permutation $a_1 a_2 \ldots a_n$ will map to the set $\{a_1, a_2, \ldots, a_k\}$.

Notice that any other permutation with the same first k elements $a_1, \ldots, a_k$ *in any order* and the same remaining elements $n - k$ elements *in any order* will also map to this set. What's more, a permutation can only map to $\{a_1, a_2, \ldots, a_k\}$ if its first k elements are the elements $a_1, \ldots, a_k$ in some order. Since there are $k!$ possible permutations of the first k elements and $(n-k)!$ permutations of the remaining elements, we conclude from the Product Rule that exactly $k!(n-k)!$ permutations of the n-element set map to the the particular subset, S. In other words, the mapping from permutations to k-element subsets is $k!(n-k)!$-to-1.

But we know there are $n!$ permutations of an n-element set, so by the Division Rule, we conclude that

$$n! = k!(n-k)!\binom{n}{k}$$

which proves:

Rule 7 (Subset Rule)**.** *The number,*

$$\binom{n}{k},$$

of k-element subsets of an n-element set is

$$\frac{n!}{k!\,(n-k)!}.$$

Notice that this works even for 0-element subsets: $n!/0!n! = 1$. Here we use the fact that $0!$ is a *product* of 0 terms, which by convention equals 1. (A *sum* of zero terms equals 0.)

14.6.2 Bit Sequences

How many n-bit sequences contain exactly k ones? We've already seen the straightforward bijection between subsets of an n-element set and n-bit sequences. For example, here is a 3-element subset of $\{x_1, x_2, \ldots, x_8\}$ and the associated 8-bit sequence:

$$\begin{array}{cccccccccc} \{ & x_1, & & & x_4, & x_5 & & & & \} \\ (& 1, & 0, & 0, & 1, & 1, & 0, & 0, & 0 &) \end{array}$$

Notice that this sequence has exactly 3 ones, each corresponding to an element of the 3-element subset. More generally, the n-bit sequences corresponding to a k-element subset will have exactly k ones. So by the Bijection Rule,

The number of n-bit sequences with exactly k ones is $\binom{n}{k}$.

14.7 Sequences with Repetitions

14.7.1 Sequences of Subsets

Choosing a k-element subset of an n-element set is the same as splitting the set into a pair of subsets: the first subset of size k and the second subset consisting of the remaining $n - k$ elements. So the Subset Rule can be understood as a rule for counting the number of such splits into pairs of subsets.

We can generalize this to splits into more than two subsets. Namely, let A be an n-element set and $k_1, k_2, \ldots, k_m$ be nonnegative integers whose sum is n. A *$(k_1, k_2, \ldots, k_m)$-split of A* is a sequence

$$(A_1, A_2, \ldots, A_m)$$

where the A_i are pairwise disjoint[2] subsets of A and $|A_i| = k_i$ for $i = 1, \ldots, m$.

[2] That is $A_i \cap A_j = \emptyset$ whenever $i \neq j$. Another way to say this is that no element appears in more than one of the A_i's.

The same reasoning used to explain the Subset Rule extends directly to a rule for counting the number of splits into subsets of given sizes.

Rule 8 (Subset Split Rule)**.** *The number of $(k_1, k_2, \ldots, k_m)$-splits of an n-element set is*

$$\binom{n}{k_1, \ldots, k_m} ::= \frac{n!}{k_1!\, k_2!\, \cdots k_m!}$$

The proof of this Rule is essentially the same as for the Subset Rule. Namely, we map any permutation $a_1 a_2 \ldots a_n$ of an n-element set, A, into a $(k_1, k_2, \ldots, k_m)$-split by letting the 1st subset in the split be the first k_1 elements of the permutation, the 2nd subset of the split be the next k_2 elements, ..., and the mth subset of the split be the final k_m elements of the permutation. This map is a $k_1!\, k_2!\, \cdots\, k_m!$-to-1 from the $n!$ permutations to the $(k_1, k_2, \ldots, k_m)$-splits of A, and the Subset Split Rule now follows from the Division Rule.

14.7.2 The Bookkeeper Rule

We can also generalize our count of n-bit sequences with k-ones to counting length n sequences of letters over an alphabet with more than two letters. For example, how many sequences can be formed by permuting the letters in the 10-letter word BOOKKEEPER?

Notice that there are 1 B, 2 O's, 2 K's, 3 E's, 1 P, and 1 R in BOOKKEEPER. This leads to a straightforward bijection between permutations of BOOKKEEPER and (1,2,2,3,1,1)-splits of $\{1, \ldots, n\}$. Namely, map a permutation to the sequence of sets of positions where each of the different letters occur.

For example, in the permutation BOOKKEEPER itself, the B is in the 1st position, the O's occur in the 2nd and 3rd positions, K's in 4th and 5th, the E's in the 6th, 7th and 9th, P in the 8th, and R is in the 10th position, so BOOKKEEPER maps to

$$(\{1\}, \{2, 3\}, \{4, 5\}, \{6, 7, 9\}, \{8\}, \{10\}).$$

From this bijection and the Subset Split Rule, we conclude that the number of ways to rearrange the letters in the word BOOKKEEPER is:

$$\frac{\overbrace{10!}^{\text{total letters}}}{\underbrace{1!}_{\text{B's}}\ \underbrace{2!}_{\text{O's}}\ \underbrace{2!}_{\text{K's}}\ \underbrace{3!}_{\text{E's}}\ \underbrace{1!}_{\text{P's}}\ \underbrace{1!}_{\text{R's}}}$$

This example generalizes directly to an exceptionally useful counting principle which we will call the

Rule 9 (Bookkeeper Rule)**.** *Let $l_1, \ldots, l_m$ be distinct elements. The number of sequences with k_1 occurrences of l_1, and k_2 occurrences of l_2, ..., and k_m occurrences of l_m is*

$$\frac{(k_1 + k_2 + \ldots + k_m)!}{k_1!\, k_2!\, \ldots\, k_m!}$$

Example. 20-Mile Walks.

I'm planning a 20-mile walk, which should include 5 northward miles, 5 eastward miles, 5 southward miles, and 5 westward miles. How many different walks are possible?

There is a bijection between such walks and sequences with 5 N's, 5 E's, 5 S's, and 5 W's. By the Bookkeeper Rule, the number of such sequences is:

$$\frac{20!}{5!^4}$$

14.7.3 A Word about Words

Someday you might refer to the Subset Split Rule or the Bookkeeper Rule in front of a roomful of colleagues and discover that they're all staring back at you blankly. This is not because they're dumb, but rather because we made up the name "Bookkeeper Rule". However, the rule is excellent and the name is apt, so we suggest that you play through: "You know? The Bookkeeper Rule? Don't you guys know *anything???*"

The Bookkeeper Rule is sometimes called the "formula for permutations with indistinguishable objects." The size k subsets of an n-element set are sometimes called *k-combinations*. Other similar-sounding descriptions are "combinations with repetition, permutations with repetition, r-permutations, permutations with indistinguishable objects," and so on. However, the counting rules we've taught you are sufficient to solve all these sorts of problems without knowing this jargon, so we won't burden you with it.

14.7.4 Problems

Class Problems

Problem 14.15.
The Tao of BOOKKEEPER: we seek enlightenment through contemplation of the word $BOOKKEEPER$.

(a) In how many ways can you arrange the letters in the word $POKE$?

(b) In how many ways can you arrange the letters in the word BO_1O_2K? Observe that we have subscripted the O's to make them distinct symbols.

(c) Suppose we map arrangements of the letters in BO_1O_2K to arrangements of the letters in $BOOK$ by erasing the subscripts. Indicate with arrows how the arrangements on the left are mapped to the arrangements on the right.

O_2BO_1K
KO_2BO_1
O_1BO_2K
KO_1BO_2
BO_1O_2K
BO_2O_1K
...

$BOOK$
$OBOK$
$KOBO$
...

(d) What kind of mapping is this, young grasshopper?

(e) In light of the Division Rule, how many arrangements are there of $BOOK$?

(f) Very good, young master! How many arrangements are there of the letters in $KE_1E_2PE_3R$?

(g) Suppose we map each arrangement of $KE_1E_2PE_3R$ to an arrangement of $KEEPER$ by erasing subscripts. List all the different arrangements of $KE_1E_2PE_3R$ that are mapped to $REPEEK$ in this way.

(h) What kind of mapping is this?

(i) So how many arrangements are there of the letters in $KEEPER$?

(j) *Now you are ready to face the BOOKKEEPER!*
How many arrangements of $BO_1O_2K_1K_2E_1E_2PE_3R$ are there?

(k) How many arrangements of $BOOK_1K_2E_1E_2PE_3R$ are there?

(l) How many arrangements of $BOOKKE_1E_2PE_3R$ are there?

(m) How many arrangements of $BOOKKEEPER$ are there?

Remember well what you have learned: subscripts on, subscripts off.
This is the Tao of Bookkeeper.

(n) How many arrangements of $VOODOODOLL$ are there?

(o) How many length 52 sequences of digits contain exactly 17 two's, 23 fives, and 12 nines?

14.8 Magic Trick

There is a Magician and an Assistant. The Assistant goes into the audience with a deck of 52 cards while the Magician looks away.[3]

[3] There are 52 cards in a standard deck. Each card has a *suit* and a *rank*. There are four suits:

♠(spades) ♡(hearts) ♣(clubs) ♢(diamonds)

Five audience members each select one card from the deck. The Assistant then gathers up the five cards and holds up four of them so the Magician can see them. The Magician concentrates for a short time and then correctly names the secret, fifth card!

Since we don't really believe the Magician can read minds, we know the Assistant has somehow communicated the secret card to the Magician. Since real Magicians and Assistants are not to be trusted, we can expect that the Assistant would illegitimately signal the Magician with coded phrases or body language, but they don't have to cheat in this way. In fact, the Magician and Assistant could be kept out of sight of each other while some audience member holds up the 4 cards designated by the Assistant for the Magician to see.

Of course, without cheating, there is still an obvious way the Assistant can communicate to the Magician: he can choose any of the $4! = 24$ permutations of the 4 cards as the order in which to hold up the cards. However, this alone won't quite work: there are 48 cards remaining in the deck, so the Assistant doesn't have enough choices of orders to indicate exactly what the secret card is (though he could narrow it down to two cards).

14.8.1 The Secret

The method the Assistant can use to communicate the fifth card exactly is a nice application of what we know about counting and matching.

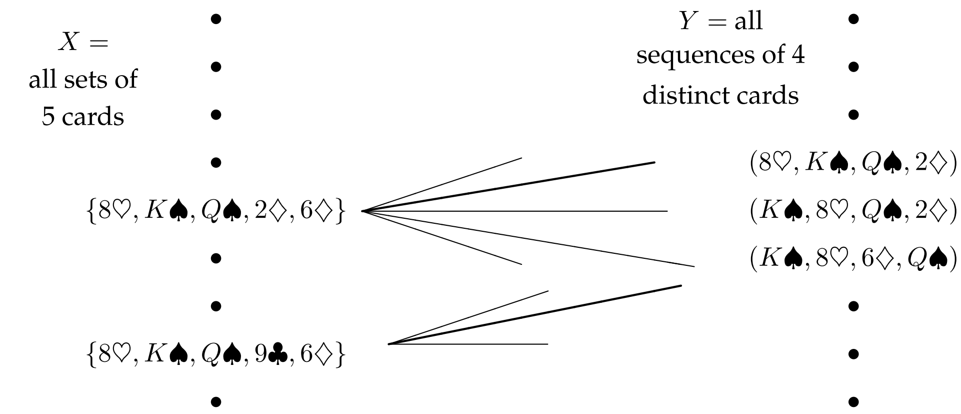
The Assistant really has another legitimate way to communicate: he can choose *which of the five cards to keep hidden*. Of course, it's not clear how the Magician could determine which of these five possibilities the Assistant selected by looking at the four visible cards, but there is a way, as we'll now explain.

The problem facing the Magician and Assistant is actually a bipartite matching problem. Put all the *sets* of 5 cards in a collection, X, on the left. And put all the sequences of 4 distinct cards in a collection, Y, on the right. These are the two sets of vertices in the bipartite graph. There is an edge between a set of 5 cards and a sequence of 4 if every card in the sequence is also in the set. In other words, if the audience selects a set of cards, then the Assistant must reveal a sequence of cards that is adjacent in the bipartite graph. Some edges are shown in the diagram below.

And there are 13 ranks, listed here from lowest to highest:

$$\overset{\text{Ace}}{A}, 2, 3, 4, 5, 6, 7, 8, 9, \overset{\text{Jack}}{J}, \overset{\text{Queen}}{Q}, \overset{\text{King}}{K}$$

Thus, for example, $8\heartsuit$ is the 8 of hearts and $A\spadesuit$ is the ace of spades.



For example,

$$\{8\heartsuit, K\spadesuit, Q\spadesuit, 2\diamondsuit, 6\diamondsuit\} \tag{14.2}$$

is an element of X on the left. If the audience selects this set of 5 cards, then there are many different 4-card sequences on the right in set Y that the Assistant could choose to reveal, including $(8\heartsuit, K\spadesuit, Q\spadesuit, 2\diamondsuit)$, $(K\spadesuit, 8\heartsuit, Q\spadesuit, 2\diamondsuit)$, and $(K\spadesuit, 8\heartsuit, 6\diamondsuit, Q\spadesuit)$.

What the Magician and his Assistant need to perform the trick is a *matching* for the X vertices. If they agree in advance on some matching, then when the audience selects a set of 5 cards, the Assistant reveals the matching sequence of 4 cards. The Magician uses the reverse of the matching to find the audience's chosen set of 5 cards, and so he can name the one not already revealed.

For example, suppose the Assistant and Magician agree on a matching containing the two bold edges in the diagram above. If the audience selects the set

$$\{8\heartsuit, K\spadesuit, Q\spadesuit, 9\clubsuit, 6\diamondsuit\}, \tag{14.3}$$

then the Assistant reveals the corresponding sequence

$$(K\spadesuit, 8\heartsuit, 6\diamondsuit, Q\spadesuit). \tag{14.4}$$

Using the matching, the Magician sees that the hand (14.3) is matched to the sequence (14.4), so he can name the one card in the corresponding set not already revealed, namely, the $9\clubsuit$. Notice that the fact that the sets are *matched*, that is, that different sets are paired with *distinct* sequences, is essential. For example, if the audience picked the previous hand (14.2), it would be possible for the Assistant to reveal the same sequence (14.4), but he better not do that: if he did, then the Magician would have no way to tell if the remaining card was the $9\clubsuit$ or the $2\diamondsuit$.

So how can we be sure the needed matching can be found? The reason is that each vertex on the left has degree $5 \cdot 4! = 120$, since there are five ways to select the card kept secret and there are $4!$ permutations of the remaining 4 cards. In addition, each vertex on the right has degree 48, since there are 48 possibilities for the fifth card. So this graph is *degree-constrained* according to Definition 9.7.5, and therefore satisfies Hall's matching condition.

In fact, this reasoning show that the Magician could still pull off the trick if 120 cards were left instead of 48, that is, the trick would work with a deck as large as 124 different cards —without any magic!

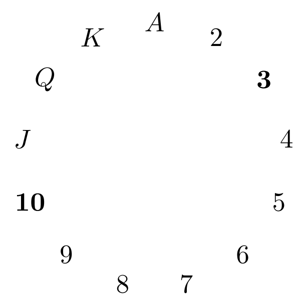
14.8.2 The Real Secret

But wait a minute! It's all very well in principle to have the Magician and his Assistant agree on a matching, but how are they supposed to remember a matching with $\binom{52}{5} = 2,598,960$ edges? For the trick to work in practice, there has to be a way to match hands and card sequences mentally and on the fly.

We'll describe one approach. As a running example, suppose that the audience selects:

$$10\heartsuit \quad 9\diamondsuit \quad 3\heartsuit \quad Q\spadesuit \quad J\diamondsuit$$

- The Assistant picks out two cards of the same suit. In the example, the assistant might choose the $3\heartsuit$ and $10\heartsuit$.
- The Assistant locates the ranks of these two cards on the cycle shown below:



  For any two distinct ranks on this cycle, one is always between 1 and 6 hops clockwise from the other. For example, the $3\heartsuit$ is 6 hops clockwise from the $10\heartsuit$.
- The more counterclockwise of these two cards is revealed first, and the other becomes the secret card. Thus, in our example, the $10\heartsuit$ would be revealed, and the $3\heartsuit$ would be the secret card. Therefore:
  - The suit of the secret card is the same as the suit of the first card revealed.
  - The rank of the secret card is between 1 and 6 hops clockwise from the rank of the first card revealed.
- All that remains is to communicate a number between 1 and 6. The Magician and Assistant agree beforehand on an ordering of all the cards in the deck from smallest to largest such as:

$$A\clubsuit\ A\diamondsuit\ A\heartsuit\ A\spadesuit\ 2\clubsuit\ 2\diamondsuit\ 2\heartsuit\ 2\spadesuit\ \ldots\ K\heartsuit\ K\spadesuit$$

The order in which the last three cards are revealed communicates the number according to the following scheme:

$$\begin{array}{lcccl} (& \text{small}, & \text{medium}, & \text{large} &) = 1 \\ (& \text{small}, & \text{large}, & \text{medium} &) = 2 \\ (& \text{medium}, & \text{small}, & \text{large} &) = 3 \\ (& \text{medium}, & \text{large}, & \text{small} &) = 4 \\ (& \text{large}, & \text{small}, & \text{medium} &) = 5 \\ (& \text{large}, & \text{medium}, & \text{small} &) = 6 \end{array}$$

In the example, the Assistant wants to send 6 and so reveals the remaining three cards in large, medium, small order. Here is the complete sequence that the Magician sees:

$$10\heartsuit \quad Q\spadesuit \quad J\diamondsuit \quad 9\diamondsuit$$

- The Magician starts with the first card, $10\heartsuit$, and hops 6 ranks clockwise to reach $3\heartsuit$, which is the secret card!

So that's how the trick can work with a standard deck of 52 cards. On the other hand, Hall's Theorem implies that the Magician and Assistant can *in principle* perform the trick with a deck of up to 124 cards. It turns out that there is a method which they could actually learn to use with a reasonable amount of practice for a 124 card deck (see *The Best Card Trick* by Michael Kleber).

14.8.3 Same Trick with Four Cards?

Suppose that the audience selects only *four* cards and the Assistant reveals a sequence of *three* to the Magician. Can the Magician determine the fourth card?

Let X be all the sets of four cards that the audience might select, and let Y be all the sequences of three cards that the Assistant might reveal. Now, on one hand, we have

$$|X| = \binom{52}{4} = 270,725$$

by the Subset Rule. On the other hand, we have

$$|Y| = 52 \cdot 51 \cdot 50 = 132,600$$

by the Generalized Product Rule. Thus, by the Pigeonhole Principle, the Assistant must reveal the *same* sequence of three cards for at least

$$\left\lceil \frac{270,725}{132,600} \right\rceil = 3$$

different four-card hands. This is bad news for the Magician: if he sees that sequence of three, then there are at least three possibilities for the fourth card which he cannot distinguish. So there is no legitimate way for the Assistant to communicate exactly what the fourth card is!

14.8.4 Problems

Class Problems

Problem 14.16. **(a)** Show that the Magician could not pull off the trick with a deck larger than 124 cards.

Hint: Compare the number of 5-card hands in an n-card deck with the number of 4-card sequences.

(b) Show that, in principle, the Magician could pull off the Card Trick with a deck of 124 cards.

Hint: Hall's Theorem and degree-constrained (9.7.5) graphs.

Problem 14.17.
The Magician can determine the 5th card in a poker hand when his Assisant reveals the other 4 cards. Describe a similar method for determining 2 hidden cards in a hand of 9 cards when your Assisant reveals the other 7 cards.

Homework Problems

Problem 14.18.
Section 14.8.3 explained why it is not possible to perform a four-card variant of the hidden-card magic trick with one card hidden. But the Magician and her Assistant are determined to find a way to make a trick like this work. They decide to change the rules slightly: instead of the Assistant lining up the three unhidden cards for the Magician to see, he will line up all four cards with one card face down and the other three visible. We'll call this the *face-down four-card trick*.

For example, suppose the audience members had selected the cards $9\heartsuit$, $10\diamondsuit$, $A\clubsuit$, $5\clubsuit$. Then the Assistant could choose to arrange the 4 cards in any order so long as one is face down and the others are visible. Two possibilities are:

$A\clubsuit$	?	$10\diamondsuit$	$5\clubsuit$
?	$5\clubsuit$	$9\heartsuit$	$10\diamondsuit$

(a) Explain why there must be a bipartite matching which will in theory allow the Magician and Assistant to perform the face-down four-card trick.

(b) There is actually a simple way to perform the face-down four-card trick.

Case 1. *there are two cards with the same suit*: Say there are two $\spadesuit$ cards. The Assistant proceeds as in the original card trick: he puts one of the $\spadesuit$ cards *face up as the first card*. He will place the second $\spadesuit$ card *face down*. He then uses a permutation of the face down card and the remaining two face up cards to code the offset of the face down card from the first card.

Case 2. *all four cards have different suits*: Assign numbers $0, 1, 2, 3$ to the four suits in some agreed upon way. The Assistant computes, s, the sum modulo 4 of the ranks of the four cards, and chooses the card with suit s to be placed *face down as the first card*. He then uses a permutation of the remaining three face-up cards to code the rank of the face down card.[a]

[a] This elegant method was devised in Fall '09 by student Katie E Everett.

Explain how in Case 2. the Magician can determine the face down card from the cards the Assistant shows her.

(c) Explain how any method for performing the face-down four-card trick can be adapted to perform the regular (5-card hand, show 4 cards) with a 52-card deck consisting of the usual 52 cards along with a 53rd card call the *joker*.

14.9 Counting Practice: Poker Hands

Five-Card Draw is a card game in which each player is initially dealt a *hand*, a subset of 5 cards. (Then the game gets complicated, but let's not worry about that.) The number of different hands in Five-Card Draw is the number of 5-element subsets of a 52-element set, which is 52 choose 5:

$$\text{total \# of hands} = \binom{52}{5} = 2{,}598{,}960$$

Let's get some counting practice by working out the number of hands with various special properties.

14.9.1 Hands with a Four-of-a-Kind

A *Four-of-a-Kind* is a set of four cards with the same rank. How many different hands contain a Four-of-a-Kind? Here are a couple examples:

$$\begin{array}{lccccccr} \{ & 8\spadesuit, & 8\diamondsuit, & Q\heartsuit, & 8\heartsuit, & 8\clubsuit & \} \\ \{ & A\clubsuit, & 2\clubsuit, & 2\heartsuit, & 2\diamondsuit, & 2\spadesuit & \} \end{array}$$

As usual, the first step is to map this question to a sequence-counting problem. A hand with a Four-of-a-Kind is completely described by a sequence specifying:

1. The rank of the four cards.

2. The rank of the extra card.

3. The suit of the extra card.

Thus, there is a bijection between hands with a Four-of-a-Kind and sequences consisting of two distinct ranks followed by a suit. For example, the three hands above are associated with the following sequences:

$$\begin{array}{lcllllll}
(8, Q, \heartsuit) & \leftrightarrow & \{ & 8\spadesuit, & 8\diamondsuit, & 8\heartsuit, & 8\clubsuit, & Q\heartsuit & \} \\
(2, A, \clubsuit) & \leftrightarrow & \{ & 2\clubsuit, & 2\heartsuit, & 2\diamondsuit, & 2\spadesuit, & A\clubsuit & \}
\end{array}$$

Now we need only count the sequences. There are 13 ways to choose the first rank, 12 ways to choose the second rank, and 4 ways to choose the suit. Thus, by the Generalized Product Rule, there are $13 \cdot 12 \cdot 4 = 624$ hands with a Four-of-a-Kind. This means that only 1 hand in about 4165 has a Four-of-a-Kind; not surprisingly, this is considered a very good poker hand!

14.9.2 Hands with a Full House

A *Full House* is a hand with three cards of one rank and two cards of another rank. Here are some examples:

$$\begin{array}{lllllll}
\{ & 2\spadesuit, & 2\clubsuit, & 2\diamondsuit, & J\clubsuit, & J\diamondsuit & \} \\
\{ & 5\diamondsuit, & 5\clubsuit, & 5\heartsuit, & 7\heartsuit, & 7\clubsuit & \}
\end{array}$$

Again, we shift to a problem about sequences. There is a bijection between Full Houses and sequences specifying:

1. The rank of the triple, which can be chosen in 13 ways.

2. The suits of the triple, which can be selected in $\binom{4}{3}$ ways.

3. The rank of the pair, which can be chosen in 12 ways.

4. The suits of the pair, which can be selected in $\binom{4}{2}$ ways.

The example hands correspond to sequences as shown below:

$$\begin{array}{lcllllll}
(2, \{\spadesuit, \clubsuit, \diamondsuit\}, J, \{\clubsuit, \diamondsuit\}) & \leftrightarrow & \{ & 2\spadesuit, & 2\clubsuit, & 2\diamondsuit, & J\clubsuit, & J\diamondsuit & \} \\
(5, \{\diamondsuit, \clubsuit, \heartsuit\}, 7, \{\heartsuit, \clubsuit\}) & \leftrightarrow & \{ & 5\diamondsuit, & 5\clubsuit, & 5\heartsuit, & 7\heartsuit, & 7\clubsuit & \}
\end{array}$$

By the Generalized Product Rule, the number of Full Houses is:

$$13 \cdot \binom{4}{3} \cdot 12 \cdot \binom{4}{2}$$

We're on a roll— but we're about to hit a speedbump.

14.9.3 Hands with Two Pairs

How many hands have *Two Pairs*; that is, two cards of one rank, two cards of another rank, and one card of a third rank? Here are examples:

$$\begin{array}{lllllll} \{ & 3\diamondsuit, & 3\spadesuit, & Q\diamondsuit, & Q\heartsuit, & A\clubsuit & \} \\ \{ & 9\heartsuit, & 9\diamondsuit, & 5\heartsuit, & 5\clubsuit, & K\spadesuit & \} \end{array}$$

Each hand with Two Pairs is described by a sequence consisting of:

1. The rank of the first pair, which can be chosen in 13 ways.
2. The suits of the first pair, which can be selected $\binom{4}{2}$ ways.
3. The rank of the second pair, which can be chosen in 12 ways.
4. The suits of the second pair, which can be selected in $\binom{4}{2}$ ways.
5. The rank of the extra card, which can be chosen in 11 ways.
6. The suit of the extra card, which can be selected in $\binom{4}{1} = 4$ ways.

Thus, it might appear that the number of hands with Two Pairs is:

$$13 \cdot \binom{4}{2} \cdot 12 \cdot \binom{4}{2} \cdot 11 \cdot 4$$

Wrong answer! The problem is that there is *not* a bijection from such sequences to hands with Two Pairs. This is actually a 2-to-1 mapping. For example, here are the pairs of sequences that map to the hands given above:

$$\begin{array}{lll} (3, \{\diamondsuit, \spadesuit\}, Q, \{\diamondsuit, \heartsuit\}, A, \clubsuit) & \searrow & \\ & & \{ \; 3\diamondsuit, \; 3\spadesuit, \; Q\diamondsuit, \; Q\heartsuit, \; A\clubsuit \; \} \\ (Q, \{\diamondsuit, \heartsuit\}, 3, \{\diamondsuit, \spadesuit\}, A, \clubsuit) & \nearrow & \\ (9, \{\heartsuit, \diamondsuit\}, 5, \{\heartsuit, \clubsuit\}, K, \spadesuit) & \searrow & \\ & & \{ \; 9\heartsuit, \; 9\diamondsuit, \; 5\heartsuit, \; 5\clubsuit, \; K\spadesuit \; \} \\ (5, \{\heartsuit, \clubsuit\}, 9, \{\heartsuit, \diamondsuit\}, K, \spadesuit) & \nearrow & \end{array}$$

The problem is that nothing distinguishes the first pair from the second. A pair of 5's and a pair of 9's is the same as a pair of 9's and a pair of 5's. We avoided this difficulty in counting Full Houses because, for example, a pair of 6's and a triple of kings is different from a pair of kings and a triple of 6's.

We ran into precisely this difficulty last time, when we went from counting arrangements of *different* pieces on a chessboard to counting arrangements of two *identical* rooks. The solution then was to apply the Division Rule, and we can do the same here. In this case, the Division rule says there are twice as many sequences as hands, so the number of hands with Two Pairs is actually:

$$\frac{13 \cdot \binom{4}{2} \cdot 12 \cdot \binom{4}{2} \cdot 11 \cdot 4}{2}$$

Another Approach

The preceding example was disturbing! One could easily overlook the fact that the mapping was 2-to-1 on an exam, fail the course, and turn to a life of crime. You can make the world a safer place in two ways:

1. Whenever you use a mapping $f : A \to B$ to translate one counting problem to another, check that the same number elements in A are mapped to each element in B. If k elements of A map to each of element of B, then apply the Division Rule using the constant k.

2. As an extra check, try solving the same problem in a different way. Multiple approaches are often available— and all had better give the same answer! (Sometimes different approaches give answers that *look* different, but turn out to be the same after some algebra.)

We already used the first method; let's try the second. There is a bijection between hands with two pairs and sequences that specify:

1. The ranks of the two pairs, which can be chosen in $\binom{13}{2}$ ways.

2. The suits of the lower-rank pair, which can be selected in $\binom{4}{2}$ ways.

3. The suits of the higher-rank pair, which can be selected in $\binom{4}{2}$ ways.

4. The rank of the extra card, which can be chosen in 11 ways.

5. The suit of the extra card, which can be selected in $\binom{4}{1} = 4$ ways.

For example, the following sequences and hands correspond:

$$\begin{array}{ccc} (\{3, Q\}, \{\diamondsuit, \spadesuit\}, \{\diamondsuit, \heartsuit\}, A, \clubsuit) & \leftrightarrow & \{\ 3\diamondsuit,\ 3\spadesuit,\ Q\diamondsuit,\ Q\heartsuit,\ A\clubsuit\ \} \\ (\{9, 5\}, \{\heartsuit, \clubsuit\}, \{\heartsuit, \diamondsuit\}, K, \spadesuit) & \leftrightarrow & \{\ 9\heartsuit,\ 9\diamondsuit,\ 5\heartsuit,\ 5\clubsuit,\ K\spadesuit\ \} \end{array}$$

Thus, the number of hands with two pairs is:

$$\binom{13}{2} \cdot \binom{4}{2} \cdot \binom{4}{2} \cdot 11 \cdot 4$$

This is the same answer we got before, though in a slightly different form.

14.9.4 Hands with Every Suit

How many hands contain at least one card from every suit? Here is an example of such a hand:

$$\{\ 7\diamondsuit,\ K\clubsuit,\ 3\diamondsuit,\ A\heartsuit,\ 2\spadesuit\ \}$$

Each such hand is described by a sequence that specifies:

1. The ranks of the diamond, the club, the heart, and the spade, which can be selected in $13 \cdot 13 \cdot 13 \cdot 13 = 13^4$ ways.

2. The suit of the extra card, which can be selected in 4 ways.
3. The rank of the extra card, which can be selected in 12 ways.

For example, the hand above is described by the sequence:

$$(7, K, A, 2, \diamondsuit, 3) \quad \leftrightarrow \quad \{ \quad 7\diamondsuit, \quad K\clubsuit, \quad A\heartsuit, \quad 2\spadesuit, \quad 3\diamondsuit \quad \}$$

Are there other sequences that correspond to the same hand? There is one more! We could equally well regard either the $3\diamondsuit$ or the $7\diamondsuit$ as the extra card, so this is actually a 2-to-1 mapping. Here are the two sequences corresponding to the example hand:

$$\begin{array}{ll} (7, K, A, 2, \diamondsuit, 3) & \searrow \\ & \qquad \{ \quad 7\diamondsuit, \quad K\clubsuit, \quad A\heartsuit, \quad 2\spadesuit, \quad 3\diamondsuit \quad \} \\ (3, K, A, 2, \diamondsuit, 7) & \nearrow \end{array}$$

Therefore, the number of hands with every suit is:

$$\frac{13^4 \cdot 4 \cdot 12}{2}$$

14.9.5 Problems

Class Problems

Problem 14.19.
Solve the following counting problems by defining an appropriate mapping (bijective or k-to-1) between a set whose size you know and the set in question.

(a) How many different ways are there to select a dozen donuts if four varieties are available?

(b) In how many ways can Mr. and Mrs. Grumperson distribute 13 identical pieces of coal to their two —no, three! —children for Christmas?

(c) How many solutions over the nonnegative integers are there to the inequality:

$$x_1 + x_2 + \ldots + x_{10} \quad \leq \quad 100$$

(d) We want to count step-by-step paths between points in the plane with integer coordinates. Ony two kinds of step are allowed: a right-step which increments the x coordinate, and an up-step which increments the y coordinate.

(i) How many paths are there from $(0, 0)$ to $(20, 30)$?

(ii) How many paths are there from $(0, 0)$ to $(20, 30)$ that go through the point $(10, 10)$?

(iii) How many paths are there from $(0,0)$ to $(20,30)$ that do *not* go through either of the points $(10,10)$ and $(15,20)$?

Hint: Let P be the set of paths from $(0,0)$ to $(20,30)$, N_1 be the paths in P that go through $(10,10)$ and N_2 be the paths in P that go through $(15,20)$.

Problem 14.20.
Solve the following counting problems. Define an appropriate mapping (bijective or k-to-1) between a set whose size you know and the set in question.

(a) An independent living group is hosting nine new candidates for membership. Each candidate must be assigned a task: 1 must wash pots, 2 must clean the kitchen, 3 must clean the bathrooms, 1 must clean the common area, and 2 must serve dinner. Write a multinomial coefficient for the number of ways this can be done.

(b) Write a multinomial coefficient for the number of nonnegative integer solutions for the equation:

$$x_1 + x_2 + x_3 + x_4 + x_5 = 8. \tag{14.5}$$

(c) How many nonnegative integers less than 1,000,000 have exactly one digit equal to 9 and have a sum of digits equal to 17?

Exam Problems

Problem 14.21.
Here are the solutions to the next 10 problem parts, in no particular order.

$$n^m \qquad m^n \qquad \frac{n!}{(n-m)!} \qquad \binom{n+m}{m} \qquad \binom{n-1+m}{m} \qquad \binom{n-1+m}{n} \qquad 2^{mn}$$

(a) How many solutions over the natural numbers are there to the inequality $x_1 + x_2 + \cdots + x_n \leq m$? _________

(b) How many length m words can be formed from an n-letter alphabet, if no letter is used more than once? _________

(c) How many length m words can be formed from an n-letter alphabet, if letters can be reused? _________

(d) How many binary relations are there from set A to set B when $|A| = m$ and $|B| = n$? ________

(e) How many injections are there from set A to set B, where $|A| = m$ and $|B| = n \geq m$? ________

(f) How many ways are there to place a total of m distinguishable balls into n distinguishable urns, with some urns possibly empty or with several balls? ________

(g) How many ways are there to place a total of m indistinguishable balls into n distinguishable urns, with some urns possibly empty or with several balls? ________

(h) How many ways are there to put a total of m distinguishable balls into n distinguishable urns with at most one ball in each urn? ________

14.10 Inclusion-Exclusion

How big is a union of sets? For example, suppose there are 60 math majors, 200 EECS majors, and 40 physics majors. How many students are there in these three departments? Let M be the set of math majors, E be the set of EECS majors, and P be the set of physics majors. In these terms, we're asking for $|M \cup E \cup P|$.

The Sum Rule says that the size of union of *disjoint* sets is the sum of their sizes:

$$|M \cup E \cup P| = |M| + |E| + |P| \qquad \text{(if } M, E, \text{ and } P \text{ are disjoint)}$$

However, the sets M, E, and P might *not* be disjoint. For example, there might be a student majoring in both math and physics. Such a student would be counted twice on the right side of this equation, once as an element of M and once as an element of P. Worse, there might be a triple-major[4] counted *three* times on the right side!

Our last counting rule determines the size of a union of sets that are not necessarily disjoint. Before we state the rule, let's build some intuition by considering some easier special cases: unions of just two or three sets.

14.10.1 Union of Two Sets

For two sets, S_1 and S_2, the *Inclusion-Exclusion Rule* is that the size of their union is:

$$|S_1 \cup S_2| = |S_1| + |S_2| - |S_1 \cap S_2| \tag{14.6}$$

[4]...though not at MIT anymore.

Intuitively, each element of S_1 is accounted for in the first term, and each element of S_2 is accounted for in the second term. Elements in *both* S_1 and S_2 are counted *twice*— once in the first term and once in the second. This double-counting is corrected by the final term.

We can capture this double-counting idea in a precise way by decomposing the union of S_1 and S_2 into three disjoint sets, the elements in each set but not the other, and the elements in both:

$$S_1 \cup S_2 = (S_1 - S_2) \cup (S_2 - S_1) \cup (S_1 \cap S_2). \tag{14.7}$$

Similarly, we can decompose each of S_1 and S_2 into the elements exclusively in each set and the elements in both:

$$S_1 = (S_1 - S_2) \cup (S_1 \cap S_2), \tag{14.8}$$
$$S_2 = (S_2 - S_1) \cup (S_1 \cap S_2). \tag{14.9}$$

Now we have from (14.8) and (14.9)

$$\begin{aligned} |S_1| + |S_2| &= (|S_1 - S_2| + |S_1 \cap S_2|) + (|S_2 - S_1| + |S_1 \cap S_2|) \\ &= |S_1 - S_2| + |S_2 - S_1| + 2\,|S_1 \cap S_2|\,, \end{aligned} \tag{14.10}$$

which shows the double-counting of $S_1 \cap S_2$ in the sum. On the other hand, we have from (14.7)

$$|S_1 \cup S_2| = |S_1 - S_2| + |S_2 - S_1| + |S_1 \cap S_2|\,. \tag{14.11}$$

Subtracting (14.11) from (14.10), we get

$$(|S_1| + |S_2|) - |S_1 \cup S_2| = |S_1 \cap S_2|$$

which proves (14.6).

14.10.2 Union of Three Sets

So how many students are there in the math, EECS, and physics departments? In other words, what is $|M \cup E \cup P|$ if:

$$\begin{aligned} |M| &= 60 \\ |E| &= 200 \\ |P| &= 40 \end{aligned}$$

The size of a union of three sets is given by a more complicated Inclusion-Exclusion formula:

$$\begin{aligned} |S_1 \cup S_2 \cup S_3| = {} & |S_1| + |S_2| + |S_3| \\ & - |S_1 \cap S_2| - |S_1 \cap S_3| - |S_2 \cap S_3| \\ & + |S_1 \cap S_2 \cap S_3| \end{aligned}$$

Remarkably, the expression on the right accounts for each element in the union of S_1, S_2, and S_3 exactly once. For example, suppose that x is an element of all three sets. Then x is counted three times (by the $|S_1|$, $|S_2|$, and $|S_3|$ terms), subtracted off three times (by the $|S_1 \cap S_2|$, $|S_1 \cap S_3|$, and $|S_2 \cap S_3|$ terms), and then counted once more (by the $|S_1 \cap S_2 \cap S_3|$ term). The net effect is that x is counted just once.

So we can't answer the original question without knowing the sizes of the various intersections. Let's suppose that there are:

4	math - EECS double majors
3	math - physics double majors
11	EECS - physics double majors
2	triple majors

Then $|M \cap E| = 4 + 2$, $|M \cap P| = 3 + 2$, $|E \cap P| = 11 + 2$, and $|M \cap E \cap P| = 2$. Plugging all this into the formula gives:

$$\begin{aligned}
|M \cup E \cup P| &= |M| + |E| + |P| - |M \cap E| - |M \cap P| - |E \cap P| + |M \cap E \cap P| \\
&= 60 + 200 + 40 - 6 - 5 - 13 + 2 \\
&= 278
\end{aligned}$$

Sequences with 42, 04, or 60

In how many permutations of the set $\{0, 1, 2, \ldots, 9\}$ do either 4 and 2, 0 and 4, or 6 and 0 appear consecutively? For example, none of these pairs appears in:

$$(7, 2, 9, 5, 4, 1, 3, 8, 0, 6)$$

The 06 at the end doesn't count; we need 60. On the other hand, both 04 and 60 appear consecutively in this permutation:

$$(7, 2, 5, \underline{6}, \underline{0}, \underline{4}, 3, 8, 1, 9)$$

Let P_{42} be the set of all permutations in which 42 appears; define P_{60} and P_{04} similarly. Thus, for example, the permutation above is contained in both P_{60} and P_{04}. In these terms, we're looking for the size of the set $P_{42} \cup P_{04} \cup P_{60}$.

First, we must determine the sizes of the individual sets, such as P_{60}. We can use a trick: group the 6 and 0 together as a single symbol. Then there is a natural bijection between permutations of $\{0, 1, 2, \ldots 9\}$ containing 6 and 0 consecutively and permutations of:

$$\{60, 1, 2, 3, 4, 5, 7, 8, 9\}$$

For example, the following two sequences correspond:

$$(7, 2, 5, \underline{6}, \underline{0}, 4, 3, 8, 1, 9) \qquad \leftrightarrow \qquad (7, 2, 5, \underline{60}, 4, 3, 8, 1, 9)$$

There are $9!$ permutations of the set containing 60, so $|P_{60}| = 9!$ by the Bijection Rule. Similarly, $|P_{04}| = |P_{42}| = 9!$ as well.

Next, we must determine the sizes of the two-way intersections, such as $P_{42} \cap P_{60}$. Using the grouping trick again, there is a bijection with permutations of the set:

$$\{42, 60, 1, 3, 5, 7, 8, 9\}$$

Thus, $|P_{42} \cap P_{60}| = 8!$. Similarly, $|P_{60} \cap P_{04}| = 8!$ by a bijection with the set:

$$\{604, 1, 2, 3, 5, 7, 8, 9\}$$

And $|P_{42} \cap P_{04}| = 8!$ as well by a similar argument. Finally, note that $|P_{60} \cap P_{04} \cap P_{42}| = 7!$ by a bijection with the set:

$$\{6042, 1, 3, 5, 7, 8, 9\}$$

Plugging all this into the formula gives:

$$|P_{42} \cup P_{04} \cup P_{60}| = 9! + 9! + 9! - 8! - 8! - 8! + 7!$$

14.10.3 Union of *n* Sets

The size of a union of n sets is given by the following rule.

Rule 10 (Inclusion-Exclusion)**.**

$$|S_1 \cup S_2 \cup \cdots \cup S_n| =$$

	the sum of the sizes of the individual sets
minus	*the sizes of all two-way intersections*
plus	*the sizes of all three-way intersections*
minus	*the sizes of all four-way intersections*
plus	*the sizes of all five-way intersections, etc.*

The formulas for unions of two and three sets are special cases of this general rule.

This way of expressing Inclusion-Exclusion is easy to understand and nearly as precise as expressing it in mathematical symbols, but we'll need the symbolic version below, so let's work on deciphering it now.

We already have a standard notation for the sum of sizes of the individual sets, namely,

$$\sum_{i=1}^{n} |S_i| .$$

A "two-way intersection" is a set of the form $S_i \cap S_j$ for $i \neq j$. We regard $S_j \cap S_i$ as the same two-way intersection as $S_i \cap S_j$, so we can assume that $i < j$. Now we can express the sum of the sizes of the two-way intersections as

$$\sum_{1 \leq i < j \leq n} |S_i \cap S_j| .$$

Similarly, the sum of the sizes of the three-way intersections is

$$\sum_{1 \leq i < j < k \leq n} |S_i \cap S_j \cap S_k| .$$

These sums have alternating signs in the Inclusion-Exclusion formula, with the sum of the k-way intersections getting the sign $(-1)^{k-1}$. This finally leads to a symbolic version of the rule:

Rule (Inclusion-Exclusion)**.**

$$\begin{aligned}
\left| \bigcup_{i=1}^{n} S_i \right| = & \sum_{i=1}^{n} |S_i| \\
& - \sum_{1 \leq i < j \leq n} |S_i \cap S_j| \\
& + \sum_{1 \leq i < j < k \leq n} |S_i \cap S_j \cap S_k| + \cdots \\
& + (-1)^{n-1} \left| \bigcap_{i=1}^{n} S_i \right| .
\end{aligned}$$

14.10.4 Computing Euler's Function

We will now use Inclusion-Exclusion to calculate Euler's function, $\phi(n)$. By definition, $\phi(n)$ is the number of nonnegative integers less than a positive integer n that are relatively prime to n. But the set, S, of nonnegative integers less than n that are *not* relatively prime to n will be easier to count.

Suppose the prime factorization of n is $p_1^{e_1} \cdots p_m^{e_m}$ for distinct primes p_i. This means that the integers in S are precisely the nonnegative integers less than n that are divisible by at least one of the p_i's. So, letting C_i be the set of nonnegative integers less than n that are divisible by p_i, we have

$$S = \bigcup_{i=1}^{m} C_i.$$

We'll be able to find the size of this union using Inclusion-Exclusion because the intersections of the C_i's are easy to count. For example, $C_1 \cap C_2 \cap C_3$ is the set of nonnegative integers less than n that are divisible by each of p_1, p_2 and p_3. But since the p_i's are distinct primes, being divisible by each of these primes is that same as being divisible by their product. Now observe that if r is a positive divisor of n, then exactly n/r nonnegative integers less than n are divisible by r, namely, $0, r, 2r, \ldots, ((n/r) - 1)r$. So exactly $n/p_1p_2p_3$ nonnegative integers less than n are divisible by all three primes p_1, p_2, p_3. In other words,

$$|C_1 \cap C_2 \cap C_3| = \frac{n}{p_1p_2p_3}.$$

So reasoning this way about all the intersections among the C_i's and applying Inclusion-Exclusion, we get

$$\begin{aligned}
|S| &= \left|\bigcup_{i=1}^{m} C_i\right| \\
&= \sum_{i=1}^{m} |C_i| - \sum_{1\le i<j\le m} |C_i \cap C_j| + \sum_{1\le i<j<k\le m} |C_i \cap C_j \cap C_k| - \cdots + (-1)^{m-1}\left|\bigcap_{i=1}^{m} C_i\right| \\
&= \sum_{i=1}^{m} \frac{n}{p_i} - \sum_{1\le i<j\le m} \frac{n}{p_i p_j} + \sum_{1\le i<j<k\le m} \frac{n}{p_i p_j p_k} - \cdots + (-1)^{m-1}\frac{n}{p_1 p_2 \cdots p_n} \\
&= n\left(\sum_{i=1}^{m} \frac{1}{p_i} - \sum_{1\le i<j\le m} \frac{1}{p_i p_j} + \sum_{1\le i<j<k\le m} \frac{1}{p_i p_j p_k} - \cdots + (-1)^{m-1}\frac{1}{p_1 p_2 \cdots p_n}\right)
\end{aligned}$$

But $\phi(n) = n - |S|$ by definition, so

$$\begin{aligned}
\phi(n) &= n\left(1 - \sum_{i=1}^{m} \frac{1}{p_i} + \sum_{1\le i<j\le m} \frac{1}{p_i p_j} - \sum_{1\le i<j<k\le m} \frac{1}{p_i p_j p_k} + \cdots + (-1)^{m}\frac{1}{p_1 p_2 \cdots p_n}\right) \\
&= n \prod_{i=1}^{m}\left(1 - \frac{1}{p_i}\right). \qquad (14.12)
\end{aligned}$$

Notice that in case $n = p^k$ for some prime, p, then (14.12) simplifies to

$$\phi(p^k) = p^k\left(1 - \frac{1}{p}\right) = p^k - p^{k-1}$$

as claimed in chapter 4.

Quick Question: Why does equation (14.12) imply that

$$\phi(ab) = \phi(a)\phi(b)$$

for relatively prime integers $a, b > 1$, as claimed in Theorem 4.7.1.(a)?

14.10.5 Problems

Practice Problems

Problem 14.22.

The working days in the next year can be numbered 1, 2, 3, ..., 300. I'd like to avoid as many as possible.

- On even-numbered days, I'll say I'm sick.
- On days that are a multiple of 3, I'll say I was stuck in traffic.

- On days that are a multiple of 5, I'll refuse to come out from under the blankets.

In total, how many work days will I *avoid* in the coming year?

Class Problems

Problem 14.23.
A certain company wants to have security for their computer systems. So they have given everyone a name and password. A length 10 word containing each of the characters:

a, d, e, f, i, l, o, p, r, s,

is called a *cword*. A password will be a cword which does not contain any of the subwords "fails", "failed", or "drop".

For example, the following two words are passwords:

adefiloprs, srpolifeda,

but the following three cwords are not:

a**drop**eflis, **failedrop**s, **drop**e**fails**.

(a) How many cwords contain the subword "drop"?

(b) How many cwords contain both "drop" and "fails"?

(c) Use the Inclusion-Exclusion Principle to find a simple formula for the number of passwords.

Homework Problems

Problem 14.24.
How many paths are there from point $(0,0)$ to $(50,50)$ if every step increments one coordinate and leaves the other unchanged? How many are there when there are impassable boulders sitting at points $(10,11)$ and $(21,20)$? (You do not have to calculate the number explicitly; your answer may be an expression involving binomial coefficients.)

Hint: Count the number of paths going through $(10,11)$, the number through $(21,20)$, and use Inclusion-Exclusion.

Problem 14.25.
A *derangement* is a permutation $(x_1, x_2, \ldots, x_n)$ of the set $\{1, 2, \ldots, n\}$ such that $x_i \neq i$ for all i. For example, $(2,3,4,5,1)$ is a derangement, but $(2,1,3,5,4)$ is not because 3 appears in the third position. The objective of this problem is to count derangements.

It turns out to be easier to start by counting the permutations that are *not* derangements. Let S_i be the set of all permutations $(x_1, x_2, \ldots, x_n)$ that are not derangements because $x_i = i$. So the set of non-derangements is

$$\bigcup_{i=1}^{n} S_i.$$

(a) What is $|S_i|$?

(b) What is $|S_i \cap S_j|$ where $i \neq j$?

(c) What is $|S_{i_1} \cap S_{i_2} \cap \cdots \cap S_{i_k}|$ where $i_1, i_2, \ldots, i_k$ are all distinct?

(d) Use the inclusion-exclusion formula to express the number of non-derangements in terms of sizes of possible intersections of the sets $S_1, \ldots, S_n$.

(e) How many terms in the expression in part (d) have the form $|S_{i_1} \cap S_{i_2} \cap \cdots \cap S_{i_k}|$?

(f) Combine your answers to the preceding parts to prove the number of non-derangements is:

$$n!\left(\frac{1}{1!} - \frac{1}{2!} + \frac{1}{3!} - \cdots \pm \frac{1}{n!}\right).$$

Conclude that the number of derangements is

$$n!\left(1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + \cdots \pm \frac{1}{n!}\right).$$

(g) As n goes to infinity, the number of derangements approaches a constant fraction of all permutations. What is that constant? *Hint:*

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

Problem 14.26.
How many of the numbers $2, \ldots, n$ are prime? The Inclusion-Exclusion Principle offers a useful way to calculate the answer when n is large. Actually, we will use Inclusion-Exclusion to count the number of *composite* (nonprime) integers from 2 to n. Subtracting this from $n - 1$ gives the number of primes.

Let C_n be the set of composites from 2 to n, and let A_m be the set of numbers in the range $m + 1, \ldots, n$ that are divisible by m. Notice that by definition, $A_m = \emptyset$ for $m \geq n$. So

$$C_n = \bigcup_{i=2}^{n-1} A_i. \tag{14.13}$$

(a) Verify that if $m \mid k$, then $A_m \supseteq A_k$.

(b) Explain why the right hand side of (14.13) equals

$$\bigcup_{\text{primes } p \leq \sqrt{n}} A_p. \tag{14.14}$$

(c) Explain why $|A_m| = \lfloor n/m \rfloor - 1$ for $m \geq 2$.

(d) Consider any two relatively prime numbers $p, q \leq n$. What is the one number in $(A_p \cap A_q) - A_{p \cdot q}$?

(e) Let $\mathcal{P}$ be a finite set of at least two primes. Give a simple formula for

$$\left| \bigcap_{p \in \mathcal{P}} A_p \right|.$$

(f) Use the Inclusion-Exclusion principle to obtain a formula for $|C_{150}|$ in terms the sizes of intersections among the sets $A_2, A_3, A_5, A_7, A_{11}$. (Omit the intersections that are empty; for example, any intersection of more than three of these sets must be empty.)

(g) Use this formula to find the number of primes up to 150.

14.11 Binomial Theorem

Counting gives insight into one of the basic theorems of algebra. A *binomial* is a sum of two terms, such as $a + b$. Now consider its 4th power, $(a + b)^4$.

If we multiply out this 4th power expression completely, we get

$$\begin{aligned}
(a+b)^4 \quad = \quad & \quad aaaa + aaab + aaba + aabb \\
& + abaa + abab + abba + abbb \\
& + baaa + baab + baba + babb \\
& + bbaa + bbab + bbba + bbbb
\end{aligned}$$

Notice that there is one term for every sequence of a's and b's. So there are 2^4 terms, and the number of terms with k copies of b and $n - k$ copies of a is:

$$\frac{n!}{k!\,(n-k)!} = \binom{n}{k}$$

by the Bookkeeper Rule. Now let's group equivalent terms, such as $aaab = aaba = abaa = baaa$. Then the coefficient of $a^{n-k}b^k$ is $\binom{n}{k}$. So for $n = 4$, this means:

$$(a+b)^4 = \binom{4}{0} \cdot a^4 b^0 + \binom{4}{1} \cdot a^3 b^1 + \binom{4}{2} \cdot a^2 b^2 + \binom{4}{3} \cdot a^1 b^3 + \binom{4}{4} \cdot a^0 b^4$$

In general, this reasoning gives the Binomial Theorem:

Theorem 14.11.1 (Binomial Theorem)**.** *For all $n \in \mathbb{N}$ and $a, b \in \mathbb{R}$:*

$$(a+b)^n = \sum_{k=0}^{n} \binom{n}{k} a^{n-k} b^k$$

The expression $\binom{n}{k}$ is often called a "binomial coefficient" in honor of its appearance here.

This reasoning about binomials extends nicely to *multinomials*, which are sums of two or more terms. For example, suppose we wanted the coefficient of

$$bo^2k^2e^3pr$$

in the expansion of $(b+o+k+e+p+r)^{10}$. Each term in this expansion is a product of 10 variables where each variable is one of b, o, k, e, p, or r. Now, the coefficient of $bo^2k^2e^3pr$ is the number of those terms with exactly 1 b, 2 o's, 2 k's, 3 e's, 1 p, and 1 r. And the number of such terms is precisely the number of rearrangments of the word BOOKKEEPER:

$$\binom{10}{1,2,2,3,1,1} = \frac{10!}{1!\,2!\,2!\,3!\,1!\,1!}.$$

The expression on the left is called a "multinomial coefficient." This reasoning extends to a general theorem.

Definition 14.11.2. For $n, k_1, \ldots, k_m \in naturals$, such that $k_1 + k_2 + \cdots + k_m = n$, define the *multinomial coefficient*

$$\binom{n}{k_1, k_2, \ldots, k_m} ::= \frac{n!}{k_1!\,k_2!\,\ldots k_m!}.$$

Theorem 14.11.3 (Multinomial Theorem)**.** *For all $n \in \mathbb{N}$ and $z_1, \ldots z_m \in \mathbb{R}$:*

$$(z_1 + z_2 + \cdots + z_m)^n = \sum_{\substack{k_1,\ldots,k_m \in \mathbb{N} \\ k_1+\cdots+k_m = n}} \binom{n}{k_1, k_2, \ldots, k_m} z_1^{k_1} z_2^{k_2} \cdots z_m^{k_m}$$

You'll be better off remembering the reasoning behind the Multinomial Theorem rather than this ugly formal statement.

14.11.1 Problems

Practice Problems

Problem 14.27.
Find the coefficients of $x^{10}y^5$ in $(19x + 4y)^{15}$

Class Problems

Problem 14.28.
Find the coefficients of

(a) x^5 in $(1+x)^{11}$

(b) x^8y^9 in $(3x+2y)^{17}$

(c) a^6b^6 in $(a^2+b^3)^5$

Problem 14.29. **(a)** Use the Multinomial Theorem 14.11.3 to prove that

$$(x_1 + x_2 + \cdots + x_n)^p \equiv x_1^p + x_2^p + \cdots + x_n^p \pmod{p} \tag{14.15}$$

for all primes p. (Do not prove it using Fermat's "little" Theorem. The point of this problem is to offer an independent proof of Fermat's theorem.)

Hint: Explain why $\binom{p}{k_1, k_2, \ldots, k_n}$ is divisible by p if all the k_i's are positive integers less than p.

(b) Explain how (14.15) immediately proves Fermat's Little Theorem 4.6.4: $n^{p-1} \equiv 1 \pmod{p}$ when n is not a multiple of p.

Homework Problems

Problem 14.30.
The *degree sequence* of a simple graph is the weakly decreasing sequence of degrees of its vertices. For example, the degree sequence for the 5-vertex numbered tree pictured in the Figure 14.1 in Problem 14.5 is $(2,2,2,1,1)$ and for the 7-vertex tree it is $(3,3,2,1,1,1,1)$.

We're interested in counting how many numbered trees there are with a given degree sequence. We'll do this using the bijection defined in Problem 14.5 between n-vertex numbered trees and length $n-2$ code words whose characters are integers between 1 and n.

The *occurrence number* for a character in a word is the number of times that the character occurs in the word. For example, in the word `65622`, the occurrence number for `6` is two, and the occurrence number for `5` is one. The *occurrence sequence* of a word is the weakly decreasing sequence of occurrence numbers of characters in the word. The occurrence sequence for this word is $(2,2,1)$ because it has two occurrences of each of the characters `6` and `2`, and one occurrence of `5`.

(a) There is simple relationship between the degree sequence of an n-vertex numbered tree and the occurrence sequence of its code. Describe this relationship and explain why it holds. Conclude that counting n-vertex numbered trees with a given degree sequence is the same as counting the number of length $n-2$ code words with a given occurrence sequence.

Hint: How many times does a vertex of degree, d, occur in the code?

For simplicity, let's focus on counting 9-vertex numbered trees with a given degree sequence. By part (a), this is the same as counting the number of length 7 code words with a given occurrence sequence.

Any length 7 code word has a *pattern*, which is another length 7 word over the alphabet `a,b,c,d,e,f,g` that has the same occurrence sequence.

(b) How many length 7 patterns are there with three occurrences of `a`, two occurrences of `b`, and one occurrence of `c` and `d`?

(c) How many ways are there to assign occurrence numbers to integers $1, 2, \ldots, 9$ so that a code word with those occurrence numbers would have the occurrence sequence $3, 2, 1, 1, 0, 0, 0, 0, 0$?

In general, to find the pattern of a code word, list its characters in decreasing order by *number of occurrences*, and list characters with the same number of occurrences in decreasing order. Then replace successive characters in the list by successive letters `a,b,c,d,e,f,g`. The code word `2468751`, for example, has the pattern `fecabdg`, which is obtained by replacing its characters `8,7,6,5,4,2,1` by `a,b,c,d,e,f,g`, respectively. The code word `2449249` has pattern `caabcab`, which is obtained by replacing its characters `4,9,2` by `a,b,c`, respectively.

(d) What length 7 code word has three occurrences of `7`, two occurrences of `8`, one occurrence each of `2` and `9`, and pattern `abacbad`?

(e) Explain why the number of 9-vertex numbered trees with degree sequence $(4, 3, 2, 2, 1, 1, 1, 1, 1)$ is the product of the answers to parts (b) and (c).

14.12 Combinatorial Proof

Suppose you have n different T-shirts, but only want to keep k. You could equally well select the k shirts you want to keep or select the complementary set of $n - k$ shirts you want to throw out. Thus, the number of ways to select k shirts from among n must be equal to the number of ways to select $n - k$ shirts from among n. Therefore:

$$\binom{n}{k} = \binom{n}{n-k}$$

This is easy to prove algebraically, since both sides are equal to:

$$\frac{n!}{k!\,(n-k)!}$$

But we didn't really have to resort to algebra; we just used counting principles.

Hmm.

14.12.1 Boxing

Jay, famed Math for Computer Science Teaching Assistant, has decided to try out for the US Olympic boxing team. After all, he's watched all of the *Rocky* movies and spent hours in front of a mirror sneering, "Yo, you wanna piece a' *me*?!" Jay figures that n people (including himself) are competing for spots on the team and only k will be selected. As part of maneuvering for a spot on the team, he needs to work out how many different teams are possible. There are two cases to consider:

- Jay *is* selected for the team, and his $k-1$ teammates are selected from among the other $n-1$ competitors. The number of different teams that can be formed in this way is:
$$\binom{n-1}{k-1}$$
- Jay is *not* selected for the team, and all k team members are selected from among the other $n-1$ competitors. The number of teams that can be formed this way is:
$$\binom{n-1}{k}$$

All teams of the first type contain Jay, and no team of the second type does; therefore, the two sets of teams are disjoint. Thus, by the Sum Rule, the total number of possible Olympic boxing teams is:

$$\binom{n-1}{k-1} + \binom{n-1}{k}$$

Jeremy, equally-famed Teaching Assistant, thinks Jay isn't so tough and so he might as well also try out. He reasons that n people (including himself) are trying out for k spots. Thus, the number of ways to select the team is simply:

$$\binom{n}{k}$$

Jeremy and Jay each correctly counted the number of possible boxing teams; thus, their answers must be equal. So we know:

$$\binom{n-1}{k-1} + \binom{n-1}{k} = \binom{n}{k}$$

This is called *Pascal's Identity*. And we proved it *without any algebra!* Instead, we relied purely on counting techniques.

14.12.2 Finding a Combinatorial Proof

A *combinatorial proof* is an argument that establishes an algebraic fact by relying on counting principles. Many such proofs follow the same basic outline:

1. Define a set S.
2. Show that $|S| = n$ by counting one way.

3. Show that $|S| = m$ by counting another way.
4. Conclude that $n = m$.

In the preceding example, S was the set of all possible Olympic boxing teams. Jay computed

$$|S| = \binom{n-1}{k-1} + \binom{n-1}{k}$$

by counting one way, and Jeremy computed

$$|S| = \binom{n}{k}$$

by counting another. Equating these two expressions gave Pascal's Identity.

More typically, the set S is defined in terms of simple sequences or sets rather than an elaborate story. Here is less colorful example of a combinatorial argument.

Theorem 14.12.1.

$$\sum_{r=0}^{n} \binom{n}{r} \binom{2n}{n-r} = \binom{3n}{n}$$

Proof. We give a combinatorial proof. Let S be all n-card hands that can be dealt from a deck containing n red cards (numbered $1, \ldots, n$) and $2n$ black cards (numbered $1, \ldots, 2n$). First, note that every $3n$-element set has

$$|S| = \binom{3n}{n}$$

n-element subsets.

From another perspective, the number of hands with exactly r red cards is

$$\binom{n}{r} \binom{2n}{n-r}$$

since there are $\binom{n}{r}$ ways to choose the r red cards and $\binom{2n}{n-r}$ ways to choose the $n - r$ black cards. Since the number of red cards can be anywhere from 0 to n, the total number of n-card hands is:

$$|S| = \sum_{r=0}^{n} \binom{n}{r} \binom{2n}{n-r}$$

Equating these two expressions for $|S|$ proves the theorem. ■

Combinatorial proofs are almost magical. Theorem 14.12.1 looks pretty scary, but we proved it without any algebraic manipulations at all. The key to constructing a combinatorial proof is choosing the set S properly, which can be tricky. Generally, the simpler side of the equation should provide some guidance. For example, the right side of Theorem 14.12.1 is $\binom{3n}{n}$, which suggests choosing S to be all n-element subsets of some $3n$-element set.

14.12.3 Problems

Class Problems

Problem 14.31.
According to the Multinomial theorem, $(w+x+y+z)^n$ can be expressed as a sum of terms of the form

$$\binom{n}{r_1, r_2, r_3, r_4} w^{r_1} x^{r_2} y^{r_3} z^{r_4}.$$

(a) How many terms are there in the sum?

(b) The sum of these multinomial coefficients has an easily expressed value. What is it?

$$\sum_{\substack{r_1+r_2+r_3+r_4=n,\\ r_i \in \mathbb{N}}} \binom{n}{r_1,\ r_2,\ r_3,\ r_4} = ? \tag{14.16}$$

Hint: How many terms are there when $(w+x+y+z)^n$ is expressed as a sum of monomials in w, x, y, z *before* terms with like powers of these variables are collected together under a single coefficient?

Problem 14.32. (a) Give a combinatorial proof of the following theorem:

$$n2^{n-1} = \sum_{k=1}^{n} k \binom{n}{k} \tag{14.17}$$

Hint: Let S be the set of all length-n sequences of 0's, 1's and a single *.

(b) Now prove (14.17) algebraically by applying the Binomial Theorem to $(1+x)^n$ and taking derivatives.

Homework Problems

Problem 14.33.
Prove the following identity by algebraic manipulation and by giving a combinatorial argument:

$$\binom{n}{r}\binom{r}{k} = \binom{n}{k}\binom{n-k}{r-k}$$

Problem 14.34. (a) Find a combinatorial (*not* algebraic) proof that

$$\sum_{i=0}^{n} \binom{n}{i} = 2^n.$$

(b) Below is a combinatorial proof of an equation. What is the equation?

Proof. Stinky Peterson owns n newts, t toads, and s slugs. Conveniently, he lives in a dorm with $n + t + s$ other students. (The students are distinguishable, but creatures of the same variety are not distinguishable.) Stinky wants to put one creature in each neighbor's bed. Let W be the set of all ways in which this can be done.

On one hand, he could first determine who gets the slugs. Then, he could decide who among his remaining neighbors has earned a toad. Therefore, $|W|$ is equal to the expression on the left.

On the other hand, Stinky could first decide which people deserve newts and slugs and then, from among those, determine who truly merits a newt. This shows that $|W|$ is equal to the expression on the right.

Since both expressions are equal to $|W|$, they must be equal to each other. ■

(Combinatorial proofs are real proofs. They are not only rigorous, but also convey an intuitive understanding that a purely algebraic argument might not reveal. However, combinatorial proofs are usually less colorful than this one.)

Problem 14.35.
According to the Multinomial Theorem 14.11.3, $(x_1+x_2+...+x_k)^n$ can be expressed as a sum of terms of the form

$$\binom{n}{r_1, r_2, ..., r_k} x_1^{r_1} x_2^{r_2} ... x_k^{r_k}.$$

(a) How many terms are there in the sum?

(b) The sum of these multinomial coefficients has an easily expressed value:

$$\sum_{\substack{r_1+r_2+...+r_k=n,\\ r_i\in\mathbb{N}}} \binom{n}{r_1,\ r_2,\ ...,\ r_k} = k^n \tag{14.18}$$

Give a combinatorial proof of this identity.

Hint: How many terms are there when $(x_1 + x_2 + ... + x_k)^n$ is expressed as a sum of monomials in x_i *before* terms with like powers of these variables are collected together under a single coefficient?

Chapter 15

Generating Functions

Generating Functions are one of the most surprising and useful inventions in Discrete Math. Roughly speaking, generating functions transform problems about *sequences* into problems about *functions*. This is great because we've got piles of mathematical machinery for manipulating functions. Thanks to generating functions, we can apply all that machinery to problems about sequences. In this way, we can use generating functions to solve all sorts of counting problems. There is a huge chunk of mathematics concerning generating functions, so we will only get a taste of the subject.

In this chapter, we'll put sequences in angle brackets to more clearly distinguish them from the many other mathematical expressions floating around.

The *ordinary generating function* for $\langle g_0, g_1, g_2, g_3 \ldots \rangle$ is the power series:

$$G(x) = g_0 + g_1 x + g_2 x^2 + g_3 x^3 + \cdots .$$

There are a few other kinds of generating functions in common use, but ordinary generating functions are enough to illustrate the power of the idea, so we'll stick to them. So from now on *generating function* will mean the ordinary kind.

A generating function is a "formal" power series in the sense that we usually regard x as a placeholder rather than a number. Only in rare cases will we actually evaluate a generating function by letting x take a real number value, so we generally ignore the issue of convergence.

Throughout this chapter, we'll indicate the correspondence between a sequence and its generating function with a double-sided arrow as follows:

$$\langle g_0, g_1, g_2, g_3, \ldots \rangle \longleftrightarrow g_0 + g_1 x + g_2 x^2 + g_3 x^3 + \cdots$$

For example, here are some sequences and their generating functions:

$$\begin{aligned}
\langle 0, 0, 0, 0, \ldots \rangle &\longleftrightarrow 0 + 0x + 0x^2 + 0x^3 + \cdots = 0 \\
\langle 1, 0, 0, 0, \ldots \rangle &\longleftrightarrow 1 + 0x + 0x^2 + 0x^3 + \cdots = 1 \\
\langle 3, 2, 1, 0, \ldots \rangle &\longleftrightarrow 3 + 2x + 1x^2 + 0x^3 + \cdots = 3 + 2x + x^2
\end{aligned}$$

The pattern here is simple: the ith term in the sequence (indexing from 0) is the coefficient of x^i in the generating function.

Recall that the sum of an infinite geometric series is:

$$1 + z + z^2 + z^3 + \cdots = \frac{1}{1-z}$$

This equation does not hold when $|z| \geq 1$, but as remarked, we don't worry about convergence issues. This formula gives closed form generating functions for a whole range of sequences. For example:

$$\begin{aligned}
\langle 1, 1, 1, 1, \ldots \rangle \quad &\longleftrightarrow \quad 1 + x + x^2 + x^3 + \cdots &&= \frac{1}{1-x} \\
\langle 1, -1, 1, -1, \ldots \rangle \quad &\longleftrightarrow \quad 1 - x + x^2 - x^3 + x^4 - \cdots &&= \frac{1}{1+x} \\
\left\langle 1, a, a^2, a^3, \ldots \right\rangle \quad &\longleftrightarrow \quad 1 + ax + a^2x^2 + a^3x^3 + \cdots &&= \frac{1}{1-ax} \\
\langle 1, 0, 1, 0, 1, 0, \ldots \rangle \quad &\longleftrightarrow \quad 1 + x^2 + x^4 + x^6 + \cdots &&= \frac{1}{1-x^2}
\end{aligned}$$

15.1 Operations on Generating Functions

The magic of generating functions is that we can carry out all sorts of manipulations on sequences by performing mathematical operations on their associated generating functions. Let's experiment with various operations and characterize their effects in terms of sequences.

15.1.1 Scaling

Multiplying a generating function by a constant scales every term in the associated sequence by the same constant. For example, we noted above that:

$$\langle 1, 0, 1, 0, 1, 0, \ldots \rangle \longleftrightarrow 1 + x^2 + x^4 + x^6 + \cdots = \frac{1}{1-x^2}$$

Multiplying the generating function by 2 gives

$$\frac{2}{1-x^2} = 2 + 2x^2 + 2x^4 + 2x^6 + \cdots$$

which generates the sequence:

$$\langle 2, 0, 2, 0, 2, 0, \ldots \rangle$$

Rule 11 (Scaling Rule)**.** *If*

$$\langle f_0, f_1, f_2, \ldots \rangle \longleftrightarrow F(x),$$

then

$$\langle cf_0,\ cf_1,\ cf_2,\ \ldots \rangle \longleftrightarrow c \cdot F(x).$$

The idea behind this rule is that:

$$\begin{aligned}
\langle cf_0, cf_1, cf_2, \ldots \rangle &\longleftrightarrow cf_0 + cf_1x + cf_2x^2 + \cdots \\
&= c \cdot (f_0 + f_1x + f_2x^2 + \cdots) \\
&= cF(x)
\end{aligned}$$

15.1.2 Addition

Adding generating functions corresponds to adding the two sequences term by term. For example, adding two of our earlier examples gives:

$$\begin{array}{rcccccccccl}
 & \langle & 1, & 1, & 1, & 1, & 1, & 1, & \ldots & \rangle & \longleftrightarrow \dfrac{1}{1-x} \\[2ex]
+ & \langle & 1, & -1, & 1, & -1, & 1, & -1, & \ldots & \rangle & \longleftrightarrow \dfrac{1}{1+x} \\[1ex]
\hline
 & \langle & 2, & 0, & 2, & 0, & 2, & 0, & \ldots & \rangle & \longleftrightarrow \dfrac{1}{1-x} + \dfrac{1}{1+x}
\end{array}$$

We've now derived two different expressions that both generate the sequence $\langle 2, 0, 2, 0, \ldots \rangle$. They are, of course, equal:

$$\frac{1}{1-x} + \frac{1}{1+x} = \frac{(1+x) + (1-x)}{(1-x)(1+x)} = \frac{2}{1-x^2}$$

Rule 12 (Addition Rule)**.** *If*

$$\begin{aligned}
\langle f_0, f_1, f_2, \ldots \rangle &\longleftrightarrow F(x), \qquad \textit{and} \\
\langle g_0, g_1, g_2, \ldots \rangle &\longleftrightarrow G(x),
\end{aligned}$$

then

$$\langle f_0 + g_0,\ f_1 + g_1,\ f_2 + g_2,\ \ldots \rangle \longleftrightarrow F(x) + G(x).$$

The idea behind this rule is that:

$$\begin{aligned}
\langle f_0 + g_0,\ f_1 + g_1,\ f_2 + g_2,\ \ldots \rangle &\longleftrightarrow \sum_{n=0}^{\infty} (f_n + g_n)x^n \\
&= \left(\sum_{n=0}^{\infty} f_nx^n \right) + \left(\sum_{n=0}^{\infty} g_nx^n \right) \\
&= F(x) + G(x)
\end{aligned}$$

15.1.3 Right Shifting

Let's start over again with a simple sequence and its generating function:

$$\langle 1, 1, 1, 1, \dots \rangle \longleftrightarrow \frac{1}{1-x}$$

Now let's *right-shift* the sequence by adding k leading zeros:

$$\begin{aligned}
\langle \underbrace{0, 0, \dots, 0}_{k \text{ zeroes}}, 1, 1, 1, \dots \rangle \quad &\longleftrightarrow \quad x^k + x^{k+1} + x^{k+2} + x^{k+3} + \cdots \\
&= \quad x^k \cdot (1 + x + x^2 + x^3 + \cdots) \\
&= \quad \frac{x^k}{1-x}
\end{aligned}$$

Evidently, adding k leading zeros to the sequence corresponds to multiplying the generating function by x^k. This holds true in general.

Rule 13 (Right-Shift Rule)**.** *If* $\langle f_0, f_1, f_2, \dots \rangle \longleftrightarrow F(x)$*, then:*

$$\langle \underbrace{0, 0, \dots, 0}_{k \text{ zeroes}}, f_0, f_1, f_2, \dots \rangle \longleftrightarrow x^k \cdot F(x)$$

The idea behind this rule is that:

$$\begin{aligned}
\langle \overbrace{0, 0, \dots, 0}^{k \text{ zeroes}}, f_0, f_1, f_2, \dots \rangle \quad &\longleftrightarrow \quad f_0 x^k + f_1 x^{k+1} + f_2 x^{k+2} + \cdots \\
&= \quad x^k \cdot (f_0 + f_1 x + f_2 x^2 + f_3 x^3 + \cdots) \\
&= \quad x^k \cdot F(x)
\end{aligned}$$

15.1.4 Differentiation

What happens if we take the *derivative* of a generating function? As an example, let's differentiate the now-familiar generating function for an infinite sequence of 1's.

$$\begin{aligned}
\frac{d}{dx}(1 + x + x^2 + x^3 + x^4 + \cdots) \quad &= \quad \frac{d}{dx}\left(\frac{1}{1-x}\right) \\
1 + 2x + 3x^2 + 4x^3 + \cdots \quad &= \quad \frac{1}{(1-x)^2} \qquad (15.1) \\
\langle 1, 2, 3, 4, \dots \rangle \quad &\longleftrightarrow \quad \frac{1}{(1-x)^2}
\end{aligned}$$

We found a generating function for the sequence $\langle 1, 2, 3, 4, \dots \rangle$ of positive integers!

In general, differentiating a generating function has two effects on the corresponding sequence: each term is multiplied by its index and the entire sequence is shifted left one place.

Rule 14 (Derivative Rule)**.** *If*

$$\langle f_0, f_1, f_2, f_3, \ldots \rangle \longleftrightarrow F(x),$$

then

$$\langle f_1, 2f_2, 3f_3, \ldots \rangle \longleftrightarrow F'(x).$$

The idea behind this rule is that:

$$\begin{aligned} \langle f_1, 2f_2, 3f_3, \ldots \rangle \longleftrightarrow\ & f_1 + 2f_2x + 3f_3x^2 + \cdots \\ &= \frac{d}{dx}\left(f_0 + f_1x + f_2x^2 + f_3x^3 + \cdots\right) \\ &= \frac{d}{dx}\, F(x) \end{aligned}$$

The Derivative Rule is very useful. In fact, there is frequent, independent need for each of differentiation's two effects, multiplying terms by their index and left-shifting one place. Typically, we want just one effect and must somehow cancel out the other. For example, let's try to find the generating function for the sequence of squares, $\langle 0, 1, 4, 9, 16, \ldots \rangle$. If we could start with the sequence $\langle 1, 1, 1, 1, \ldots \rangle$ and multiply each term by its index two times, then we'd have the desired result:

$$\langle 0 \cdot 0,\ 1 \cdot 1,\ 2 \cdot 2,\ 3 \cdot 3,\ \ldots \rangle = \langle 0, 1, 4, 9, \ldots \rangle$$

A challenge is that differentiation not only multiplies each term by its index, but also shifts the whole sequence left one place. However, the Right-Shift Rule 13 tells how to cancel out this unwanted left-shift: multiply the generating function by x.

Our procedure, therefore, is to begin with the generating function for $\langle 1, 1, 1, 1, \ldots \rangle$, differentiate, multiply by x, and then differentiate and multiply by x once more.

$$\begin{aligned} \langle 1, 1, 1, 1, \ldots \rangle &\longleftrightarrow \frac{1}{1-x} \\ \langle 1, 2, 3, 4, \ldots \rangle &\longleftrightarrow \frac{d}{dx}\,\frac{1}{1-x} = \frac{1}{(1-x)^2} \\ \langle 0, 1, 2, 3, \ldots \rangle &\longleftrightarrow x \cdot \frac{1}{(1-x)^2} = \frac{x}{(1-x)^2} \\ \langle 1, 4, 9, 16, \ldots \rangle &\longleftrightarrow \frac{d}{dx}\,\frac{x}{(1-x)^2} = \frac{1+x}{(1-x)^3} \\ \langle 0, 1, 4, 9, \ldots \rangle &\longleftrightarrow x \cdot \frac{1+x}{(1-x)^3} = \frac{x(1+x)}{(1-x)^3} \end{aligned}$$

Thus, the generating function for squares is:

$$\frac{x(1+x)}{(1-x)^3} \tag{15.2}$$

15.1.5 Products

Rule 15 (Product Rule)**.** *If*

$$\langle a_0, a_1, a_2, \dots \rangle \longleftrightarrow A(x), \quad \textit{and} \quad \langle b_0, b_1, b_2, \dots \rangle \longleftrightarrow B(x),$$

then

$$\langle c_0,\ c_1,\ c_2,\ \dots \rangle \longleftrightarrow A(x) \cdot B(x),$$

where

$$c_n ::= a_0 b_n + a_1 b_{n-1} + a_2 b_{n-2} + \cdots + a_n b_0.$$

To understand this rule, let

$$C(x) ::= A(x) \cdot B(x) = \sum_{n=0}^{\infty} c_n x^n.$$

We can evaluate the product $A(x) \cdot B(x)$ by using a table to identify all the cross-terms from the product of the sums:

	b_0x^0	b_1x^1	b_2x^2	b_3x^3	$\dots$
a_0x^0	$a_0b_0x^0$	$a_0b_1x^1$	$a_0b_2x^2$	$a_0b_3x^3$	$\dots$
a_1x^1	$a_1b_0x^1$	$a_1b_1x^2$	$a_1b_2x^3$	$\dots$	
a_2x^2	$a_2b_0x^2$	$a_2b_1x^3$	$\dots$		
a_3x^3	$a_3b_0x^3$	$\dots$			
$\vdots$	$\dots$				

Notice that all terms involving the same power of x lie on a /-sloped diagonal. Collecting these terms together, we find that the coefficient of x^n in the product is the sum of all the terms on the $(n+1)$st diagonal, namely,

$$a_0 b_n + a_1 b_{n-1} + a_2 b_{n-2} + \cdots + a_n b_0. \tag{15.3}$$

This expression (15.3) may be familiar from a signal processing course; the sequence $\langle c_0, c_1, c_2, \dots \rangle$ is called the *convolution* of sequences $\langle a_0, a_1, a_2, \dots \rangle$ and $\langle b_0, b_1, b_2, \dots \rangle$.

15.2 The Fibonacci Sequence

Sometimes we can find nice generating functions for more complicated sequences. For example, here is a generating function for the Fibonacci numbers:

$$\langle 0, 1, 1, 2, 3, 5, 8, 13, 21, \dots \rangle \longleftrightarrow \frac{x}{1 - x - x^2}$$

The Fibonacci numbers may seem like a fairly nasty bunch, but the generating function is simple!

We're going to derive this generating function and then use it to find a closed form for the nth Fibonacci number. The techniques we'll use are applicable to a large class of recurrence equations.

15.2.1 Finding a Generating Function

Let's begin by recalling the definition of the Fibonacci numbers:

$$\begin{aligned}
f_0 &= 0 \\
f_1 &= 1 \\
f_n &= f_{n-1} + f_{n-2} \qquad \text{(for } n \geq 2\text{)}
\end{aligned}$$

We can expand the final clause into an infinite sequence of equations. Thus, the Fibonacci numbers are defined by:

$$\begin{aligned}
f_0 &= 0 \\
f_1 &= 1 \\
f_2 &= f_1 + f_0 \\
f_3 &= f_2 + f_1 \\
f_4 &= f_3 + f_2 \\
&\vdots
\end{aligned}$$

Now the overall plan is to *define* a function $F(x)$ that generates the sequence on the left side of the equality symbols, which are the Fibonacci numbers. Then we *derive* a function that generates the sequence on the right side. Finally, we equate the two and solve for $F(x)$. Let's try this. First, we define:

$$F(x) = f_0 + f_1 x + f_2 x^2 + f_3 x^3 + f_4 x^4 + \cdots$$

Now we need to derive a generating function for the sequence:

$$\langle 0,\ 1,\ f_1 + f_0,\ f_2 + f_1,\ f_3 + f_2,\ \ldots \rangle$$

One approach is to break this into a sum of three sequences for which we know generating functions and then apply the Addition Rule:

$$\begin{array}{rccccccclcl}
 & \langle & 0, & 1, & 0, & 0, & 0, & \ldots & \rangle & \longleftrightarrow & x \\
 & \langle & 0, & f_0, & f_1, & f_2, & f_3, & \ldots & \rangle & \longleftrightarrow & xF(x) \\
+ & \langle & 0, & 0, & f_0, & f_1, & f_2, & \ldots & \rangle & \longleftrightarrow & x^2F(x) \\
\hline
 & \langle & 0, & 1 + f_0, & f_1 + f_0, & f_2 + f_1, & f_3 + f_2, & \ldots & \rangle & \longleftrightarrow & x + xF(x) + x^2F(x)
\end{array}$$

This sequence is almost identical to the right sides of the Fibonacci equations. The one blemish is that the second term is $1 + f_0$ instead of simply 1. However, this amounts to nothing, since $f_0 = 0$ anyway.

Now if we equate $F(x)$ with the new function $x + xF(x) + x^2F(x)$, then we're implicitly writing down *all* of the equations that define the Fibonacci numbers in one fell swoop:

$$\begin{array}{ccccccccc} F(x) & = & f_0 + & f_1 & x + & f_2 & x^2 + & f_3 & x^3 + \cdots \\ \shortparallel & & \shortparallel & \shortparallel & & \shortparallel & & \shortparallel & \\ x + xF(x) + x^2F(x) & = & 0 & + (1 + f_0) & x + & (f_1 + f_0) & x^2 + & (f_2 + f_1) & x^3 + \cdots \end{array}$$

Solving for $F(x)$ gives the generating function for the Fibonacci sequence:

$$F(x) = x + xF(x) + x^2F(x)$$

so

$$F(x) = \frac{x}{1 - x - x^2}.$$

Sure enough, this is the simple generating function we claimed at the outset.

15.2.2 Finding a Closed Form

Why should one care about the generating function for a sequence? There are several answers, but here is one: if we can find a generating function for a sequence, then we can often find a closed form for the nth coefficient— which can be pretty useful! For example, a closed form for the coefficient of x^n in the power series for $x/(1 - x - x^2)$ would be an explicit formula for the nth Fibonacci number.

So our next task is to extract coefficients from a generating function. There are several approaches. For a generating function that is a ratio of polynomials, we can use the method of *partial fractions*, which you learned in calculus. Just as the terms in a partial fraction expansion are easier to integrate, the coefficients of those terms are easy to compute.

Let's try this approach with the generating function for Fibonacci numbers. First, we factor the denominator:

$$1 - x - x^2 = (1 - \alpha_1 x)(1 - \alpha_2 x)$$

where $\alpha_1 = \frac{1}{2}(1 + \sqrt{5})$ and $\alpha_2 = \frac{1}{2}(1 - \sqrt{5})$. Next, we find A_1 and A_2 which satisfy:

$$\frac{x}{1 - x - x^2} = \frac{A_1}{1 - \alpha_1 x} + \frac{A_2}{1 - \alpha_2 x}$$

We do this by plugging in various values of x to generate linear equations in A_1 and A_2. We can then find A_1 and A_2 by solving a linear system. This gives:

$$A_1 = \frac{1}{\alpha_1 - \alpha_2} = \frac{1}{\sqrt{5}}$$
$$A_2 = \frac{-1}{\alpha_1 - \alpha_2} = -\frac{1}{\sqrt{5}}$$

Substituting into the equation above gives the partial fractions expansion of $F(x)$:

$$\frac{x}{1-x-x^2} = \frac{1}{\sqrt{5}}\left(\frac{1}{1-\alpha_1 x} - \frac{1}{1-\alpha_2 x}\right)$$

Each term in the partial fractions expansion has a simple power series given by the geometric sum formula:

$$\begin{aligned}
\frac{1}{1-\alpha_1 x} &= 1 + \alpha_1 x + \alpha_1^2 x^2 + \cdots \\
\frac{1}{1-\alpha_2 x} &= 1 + \alpha_2 x + \alpha_2^2 x^2 + \cdots
\end{aligned}$$

Substituting in these series gives a power series for the generating function:

$$\begin{aligned}
F(x) &= \frac{1}{\sqrt{5}}\left(\frac{1}{1-\alpha_1 x} - \frac{1}{1-\alpha_2 x}\right) \\
&= \frac{1}{\sqrt{5}}\left((1 + \alpha_1 x + \alpha_1^2 x^2 + \cdots) - (1 + \alpha_2 x + \alpha_2^2 x^2 + \cdots)\right),
\end{aligned}$$

so

$$\begin{aligned}
f_n &= \frac{\alpha_1^n - \alpha_2^n}{\sqrt{5}} \\
&= \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right)
\end{aligned}$$

This formula may be scary and astonishing —it's not even obvious that its value is an integer —but it's very useful. For example, it provides (via the repeated squaring method) a much more efficient way to compute Fibonacci numbers than crunching through the recurrence, and it also clearly reveals the exponential growth of these numbers.

15.2.3 Problems

Class Problems

Problem 15.1.

The famous mathematician, Fibonacci, has decided to start a rabbit farm to fill up his time while he's not making new sequences to torment future college students. Fibonacci starts his farm on month zero (being a mathematician), and at the start of month one he receives his first pair of rabbits. Each pair of rabbits takes a month to mature, and after that breeds to produce one new pair of rabbits each month. Fibonacci decides that in order never to run out of rabbits or money, every time a batch of new rabbits is born, he'll sell a number of newborn pairs equal to the total number of pairs he had three months earlier. Fibonacci is convinced that this way he'll never run out of stock.

(a) Define the number, r_n, of pairs of rabbits Fibonacci has in month n, using a recurrence relation. That is, define r_n in terms of various r_i where $i < n$.

(b) Let $R(x)$ be the generating function for rabbit pairs,

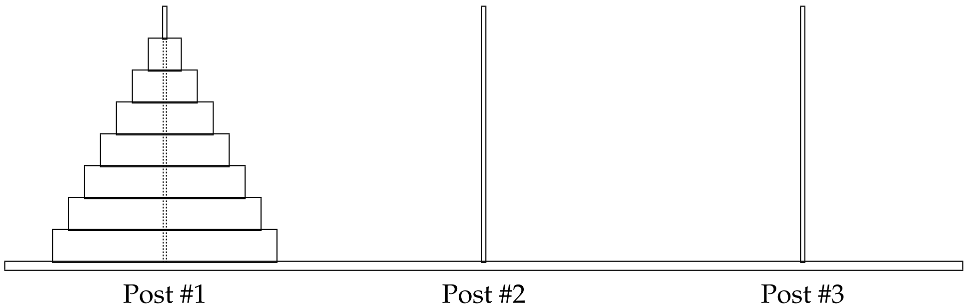
$$R(x) ::= r_0 + r_1 x + r_2 x^2 + \cdot.$$

Express $R(x)$ as a quotient of polynomials.

(c) Find a partial fraction decomposition of the generating function $R(x)$.

(d) Finally, use the partial fraction decomposition to come up with a closed form expression for the number of pairs of rabbits Fibonacci has on his farm on month n.

Problem 15.2.
Less well-known than the Towers of Hanoi —but no less fascinating —are the Towers of Sheboygan. As in Hanoi, the puzzle in Sheboygan involves 3 posts and n disks of different sizes. Initially, all the disks are on post #1:



The objective is to transfer all n disks to post #2 via a sequence of moves. A move consists of removing the top disk from one post and dropping it onto another post with the restriction that a larger disk can never lie above a smaller disk. Furthermore, a local ordinance requires that *a disk can be moved only from a post to the next post on its right —or from post #3 to post #1.* Thus, for example, moving a disk directly from post #1 to post #3 is not permitted.

(a) One procedure that solves the Sheboygan puzzle is defined recursively: to move an initial stack of n disks to the next post, move the top stack of $n-1$ disks to the furthest post by moving it to the next post two times, then move the big, nth disk to the next post, and finally move the top stack another two times to land on top of the big disk. Let s_n be the number of moves that this procedure uses. Write a simple linear recurrence for s_n.

(b) Let $S(x)$ be the generating function for the sequence $\langle s_0, s_1, s_2, \ldots \rangle$. Carefully Show that

$$S(x) = \frac{x}{(1-x)(1-4x)}.$$

(c) Give a simple formula for s_n.

(d) A better (indeed optimal, but we won't prove this) procedure to solve the Towers of Sheboygan puzzle can be defined in terms of two mutually recursive procedures, procedure $P_1(n)$ for moving a stack of n disks 1 pole forward, and $P_2(n)$ for moving a stack of n disks 2 poles forward. This is trivial for $n = 0$. For $n > 0$, define:

$P_1(n)$: Apply $P_2(n-1)$ to move the top $n-1$ disks two poles forward to the third pole. Then move the remaining big disk once to land on the second pole. Then apply $P_2(n-1)$ again to move the stack of $n-1$ disks two poles forward from the third pole to land on top of the big disk.

$P_2(n)$: Apply $P_2(n-1)$ to move the top $n-1$ disks two poles forward to land on the third pole. Then move the remaining big disk to the second pole. Then apply $P_1(n-1)$ to move the stack of $n-1$ disks one pole forward to land on the first pole. Now move the big disk 1 pole forward again to land on the third pole. Finally, apply $P_2(n-1)$ again to move the stack of $n-1$ disks two poles forward to land on the big disk.

Let t_n be the number of moves needed to solve the Sheboygan puzzle using procedure $P_1(n)$. Show that

$$t_n = 2t_{n-1} + 2t_{n-2} + 3, \tag{15.4}$$

for $n > 1$.

Hint: Let s_n be the number of moves used by procedure $P_2(n)$. Express each of t_n and s_n as linear combinations of t_{n-1} and s_{n-1} and solve for t_n.

(e) Derive values a, b, c, α, β such that

$$t_n = a\alpha^n + b\beta^n + c.$$

Conclude that $t_n = o(s_n)$.

Homework Problems

Problem 15.3.
Taking derivatives of generating functions is another useful operation. This is done termwise, that is, if

$$F(x) = f_0 + f_1x + f_2x^2 + f_3x^3 + \cdots ,$$

then

$$F'(x) ::= f_1 + 2f_2x + 3f_3x^2 + \cdots .$$

For example,

$$\frac{1}{(1-x)^2} = \left(\frac{1}{(1-x)}\right)' = 1 + 2x + 3x^2 + \cdots$$

so

$$H(x) ::= \frac{x}{(1-x)^2} = 0 + 1x + 2x^2 + 3x^3 + \cdots$$

is the generating function for the sequence of nonegative integers. Therefore

$$\frac{1+x}{(1-x)^3} = H'(x) = 1 + 2^2x + 3^2x^2 + 4^2x^3 + \cdots ,$$

so

$$\frac{x^2+x}{(1-x)^3} = xH'(x) = 0 + 1x + 2^2x^2 + 3^2x^3 + \cdots + n^2x^n + \cdots$$

is the generating function for the nonegative integer squares.

(a) Prove that for all $k \in \mathbb{N}$, the generating function for the nonnegative integer kth powers is a quotient of polynomials in x. That is, for all $k \in \mathbb{N}$ there are polynomials $R_k(x)$ and $S_k(x)$ such that

$$[x^n]\left(\frac{R_k(x)}{S_k(x)}\right) = n^k. \tag{15.5}$$

Hint: Observe that the derivative of a quotient of polynomials is also a quotient of polynomials. It is not necessary work out explicit formulas for R_k and S_k to prove this part.

(b) Conclude that if $f(n)$ is a function on the nonnegative integers defined recursively in the form

$$f(n) = af(n-1) + bf(n-2) + cf(n-3) + p(n)\alpha^n$$

where the $a, b, c, \alpha \in \mathbb{C}$ and p is a polynomial with complex coefficients, then the generating function for the sequence $f(0), f(1), f(2), \ldots$ will be a quotient of polynomials in x, and hence there is a closed form expression for $f(n)$.

Hint: Consider

$$\frac{R_k(\alpha x)}{S_k(\alpha x)}$$

Problem 15.4.

Generating functions provide an interesting way to count the number of strings of matched parentheses. To do this, we'll use the description of these strings given in Definition 10.1.2 as the set, GoodCount, of strings of parentheses with a good count. Let c_n be the number of strings in GoodCount with exactly n left parentheses, and let $C(x)$ be the generating function for these numbers:

$$C(x) ::= c_0 + c_1x + c_2x^2 + \cdots .$$

(a) The *wrap* of a string, s, is the string, (s), that starts with a left parenthesis followed by the characters of s, and then ends with a right parenthesis. Explain why the generating function for the wraps of strings with a good count is $xC(x)$.

Hint: The wrap of a string with good count also has a good count that starts and ends with 0 and remains *positive* everywhere else.

(b) Explain why, for every string, s, with a good count, there is a unique sequence of strings $s_1, \ldots, s_k$ that are wraps of strings with good counts and $s = s_1 \cdots s_k$. For example, the string $r ::= (())()(()()) \in \text{GoodCount}$ equals $s_1 s_2 s_3$ where $s_1 = (()), s_2 = (), s_3 = (()())$, and this is the only way to express r as a sequence of wraps of strings with good counts.

(c) Conclude that

$$C = 1 + xC + (xC)^2 + \cdots + (xC)^n + \cdots , \tag{15.6}$$

so

$$C = \frac{1}{1 - xC} , \tag{15.7}$$

and hence

$$C = \frac{1 \pm \sqrt{1 - 4x}}{2x} . \tag{15.8}$$

Let $D(x) ::= 2xC(x)$. Expressing D as a power series

$$D(x) = d_0 + d_1 x + d_2 x^2 + \cdots ,$$

we have

$$c_n = \frac{d_{n+1}}{2}. \tag{15.9}$$

(d) Use (15.12), (15.13), and the value of c_0 to conclude that

$$D(x) = 1 - \sqrt{1 - 4x}.$$

(e) Prove that

$$d_n = \frac{(2n-3) \cdot (2n-5) \cdots 5 \cdot 3 \cdot 1 \cdot 2^n}{n!}.$$

Hint: $d_n = D^{(n)}(0)/n!$

(f) Conclude that

$$c_n = \frac{1}{n+1} \binom{2n}{n}.$$

Exam Problems

Problem 15.5.
Define the sequence $r_0, r_1, r_2, \ldots$ recursively by the rule that $r_0 = r_1 = 0$ and

$$r_n = 7r_{n-1} + 4r_{n-2} + (n+1),$$

for $n \geq 2$. Express the generating function of this sequence as a quotient of polynomials or products of polynomials. You do *not* have to find a closed form for r_n.

15.3 Counting with Generating Functions

Generating functions are particularly useful for solving counting problems. In particular, problems involving choosing items from a set often lead to nice generating functions by letting the coefficient of x^n be the number of ways to choose n items.

15.3.1 Choosing Distinct Items from a Set

The generating function for binomial coefficients follows directly from the Binomial Theorem:

$$\begin{aligned} \left\langle \binom{k}{0}, \binom{k}{1}, \binom{k}{2}, \ldots, \binom{k}{k}, 0, 0, 0, \ldots \right\rangle \quad &\longleftrightarrow \quad \binom{k}{0} + \binom{k}{1}x + \binom{k}{2}x^2 + \cdots + \binom{k}{k}x^k \\ &= \quad (1+x)^k \end{aligned}$$

Thus, the coefficient of x^n in $(1+x)^k$ is $\binom{k}{n}$, the number of ways to choose n distinct items from a set of size k. For example, the coefficient of x^2 is $\binom{k}{2}$, the number of ways to choose 2 items from a set with k elements. Similarly, the coefficient of x^{k+1} is the number of ways to choose $k+1$ items from a size k set, which is zero. (Watch out for this reversal of the roles that k and n played in earlier examples; we're led to this reversal because we've been using n to refer to the power of x in a power series.)

15.3.2 Building Generating Functions that Count

Often we can translate the description of a counting problem directly into a generating function for the solution. For example, we could figure out that $(1+x)^k$ generates the number of ways to select n distinct items from a k-element set without resorting to the Binomial Theorem or even fussing with binomial coefficients!

Here is how. First, consider a single-element set $\{a_1\}$. The generating function for the number of ways to select n elements from this set is simply $1+x$: we have 1 way to select zero elements, 1 way to select one element, and 0 ways to select more than one element. Similarly, the number of ways to select n elements from the set $\{a_2\}$ is also given by the generating function $1+x$. The fact that the elements differ in the two cases is irrelevant.

Now here is the main trick: *the generating function for choosing elements from a union of disjoint sets is the product of the generating functions for choosing from each set.* We'll justify this in a moment, but let's first look at an example. According to this principle, the generating function for the number of ways to select n elements from the $\{a_1, a_2\}$ is:

$$\underbrace{(1+x)}_{\substack{\text{gen func for}\\ \text{selecting an } a_1}} \cdot \underbrace{(1+x)}_{\substack{\text{gen func for}\\ \text{selecting an } a_2}} = \underbrace{(1+x)^2}_{\substack{\text{gen func for}\\ \text{selecting from}\\ \{a_1, a_2\}}} = 1 + 2x + x^2$$

Sure enough, for the set $\{a_1, a_2\}$, we have 1 way to select zero elements, 2 ways to select one element, 1 way to select two elements, and 0 ways to select more than two elements.

Repeated application of this rule gives the generating function for selecting n items from a k-element set $\{a_1, a_2, \ldots, a_k\}$:

$$\underbrace{(1+x)}_{\substack{\text{gen func for}\\ \text{selecting an } a_1}} \cdot \underbrace{(1+x)}_{\substack{\text{gen func for}\\ \text{selecting an } a_2}} \cdots \underbrace{(1+x)}_{\substack{\text{gen func for}\\ \text{selecting an } a_k}} = \underbrace{(1+x)^k}_{\substack{\text{gen func for}\\ \text{selecting from}\\ \{a_1, a_2, \ldots, a_k\}}}$$

This is the same generating function that we obtained by using the Binomial Theorem. But this time around we translated directly from the counting problem to the generating function.

We can extend these ideas to a general principle:

Rule 16 (Convolution Rule)**.** *Let $A(x)$ be the generating function for selecting items from set $\mathcal{A}$, and let $B(x)$ be the generating function for selecting items from set $\mathcal{B}$. If $\mathcal{A}$ and $\mathcal{B}$ are disjoint, then the generating function for selecting items from the union $\mathcal{A} \cup \mathcal{B}$ is the product $A(x) \cdot B(x)$.*

This rule is rather ambiguous: what exactly are the rules governing the selection of items from a set? Remarkably, the Convolution Rule remains valid under *many* interpretations of selection. For example, we could insist that distinct items be selected or we might allow the same item to be picked a limited number of times or any number of times. Informally, the only restrictions are that (1) the order in which items are selected is disregarded and (2) restrictions on the selection of items from sets $\mathcal{A}$ and $\mathcal{B}$ also apply in selecting items from $\mathcal{A} \cup \mathcal{B}$. (Formally, there must be a bijection between n-element selections from $\mathcal{A} \cup \mathcal{B}$ and ordered pairs of selections from $\mathcal{A}$ and $\mathcal{B}$ containing a total of n elements.)

To count the number of ways to select n items from $\mathcal{A} \cup \mathcal{B}$, we observe that we can select n items by choosing j items from $\mathcal{A}$ and $n - j$ items from $\mathcal{B}$, where j is any number from 0 to n. This can be done in $a_j b_{n-j}$ ways. Summing over all the possible values of j gives a total of

$$a_0 b_n + a_1 b_{n-1} + a_2 b_{n-2} + \cdots + a_n b_0$$

ways to select n items from $\mathcal{A} \cup \mathcal{B}$. By the Product Rule, this is precisely the coefficient of x^n in the series for $A(x)B(x)$.

15.3.3 Choosing Items with Repetition

The first counting problem we considered was the number of ways to select a dozen doughnuts when five flavors were available. We can generalize this question as follows: in how many ways can we select n items from a k-element set if we're allowed to pick the same item multiple times? In these terms, the doughnut problem asks in how many ways we can select $n = 12$ doughnuts from the set of $k = 5$ flavors

$$\{\text{chocolate}, \text{lemon-filled}, \text{sugar}, \text{glazed}, \text{plain}\}$$

where, of course, we're allowed to pick several doughnuts of the same flavor. Let's approach this question from a generating functions perspective.

Suppose we make n choices (with repetition allowed) of items from a set containing a single item. Then there is one way to choose zero items, one way to choose one item, one way to choose two items, etc. Thus, the generating function for choosing n elements with repetition from a 1-element set is:

$$\begin{aligned} \langle 1, 1, 1, 1, \ldots \rangle \quad &\longleftrightarrow \quad 1 + x + x^2 + x^3 + \cdots \\ &= \quad \frac{1}{1-x} \end{aligned}$$

The Convolution Rule says that the generating function for selecting items from a union of disjoint sets is the product of the generating functions for selecting items from each set:

$$\underbrace{\frac{1}{1-x}}_{\substack{\text{gen func for} \\ \text{choosing } a_1\text{'s}}} \cdot \underbrace{\frac{1}{1-x}}_{\substack{\text{gen func for} \\ \text{choosing } a_2\text{'s}}} \cdots \underbrace{\frac{1}{1-x}}_{\substack{\text{gen func for} \\ \text{choosing } a_k\text{'s}}} = \underbrace{\frac{1}{(1-x)^k}}_{\substack{\text{gen func for} \\ \text{repeated choice from} \\ \{a_1, a_2, \ldots, a_k\}}}$$

Therefore, the generating function for choosing items from a k-element set with repetition allowed is $1/(1-x)^k$.

Now the Bookkeeper Rule tells us that the number of ways to choose n items with repetition from an k element set is

$$\binom{n+k-1}{n},$$

so this is the coefficient of x^n in the series expansion of $1/(1-x)^k$.

On the other hand, it's instructive to derive this coefficient algebraically, which we can do using Taylor's Theorem:

Theorem 15.3.1 (Taylor's Theorem)**.**

$$f(x) = f(0) + f'(0)x + \frac{f''(0)}{2!}x^2 + \frac{f'''(0)}{3!}x^3 + \cdots + \frac{f^{(n)}(0)}{n!}x^n + \cdots .$$

This theorem says that the nth coefficient of $1/(1-x)^k$ is equal to its nth derivative evaluated at 0 and divided by $n!$. Computing the nth derivative turns out not to be very difficult (Problem 15.7).

15.3.4 Problems

Practice Problems

Problem 15.6.
You would like to buy a bouquet of flowers. You find an online service that will make bouquets of **lilies**, **roses** and **tulips**, subject to the following constraints:

- there must be at most 3 lilies,
- there must be an odd number of tulips,
- there can be any number of roses.

Example: A bouquet of 3 tulips, 5 roses and no lilies satisfies the constraints.

Let f_n be the number of possible bouquets with n flowers that fit the service's constraints. Express $F(x)$, the generating function corresponding to $\langle f_0, f_1, f_2, \dots \rangle$, as a quotient of polynomials (or products of polynomials). You do not need to simplify this expression.

Class Problems

Problem 15.7.
Let $A(x) = \sum_{n=0}^{\infty} a_n x^n$. Then it's easy to check that

$$a_n = \frac{A^{(n)}(0)}{n!},$$

where $A^{(n)}$ is the nth derivative of A. Use this fact (which you may assume) instead of the Convolution Counting Principle, to prove that

$$\frac{1}{(1-x)^k} = \sum_{n=0}^{\infty} \binom{n+k-1}{k-1} x^n.$$

So if we didn't already know the Bookkeeper Rule, we could have proved it from this calculation and the Convolution Rule for generating functions.

Problem 15.8.
We are interested in generating functions for the number of different ways to compose a bag of n donuts subject to various restrictions. For each of the restrictions in (a)-(e) below, find a closed form for the corresponding generating function.

(a) All the donuts are chocolate and there are at least 3.

(b) All the donuts are glazed and there are at most 2.

(c) All the donuts are coconut and there are exactly 2 or there are none.

(d) All the donuts are plain and their number is a multiple of 4.

(e) The donuts must be chocolate, glazed, coconut, or plain and:

- there must be at least 3 chocolate donuts, and
- there must be at most 2 glazed, and
- there must be exactly 0 or 2 coconut, and
- there must be a multiple of 4 plain.

(f) Find a closed form for the number of ways to select n donuts subject to the constraints of the previous part.

Problem 15.9. **(a)** Let

$$S(x) ::= \frac{x^2 + x}{(1-x)^3}.$$

What is the coefficient of x^n in the generating function series for $S(x)$?

(b) Explain why $S(x)/(1-x)$ is the generating function for the sums of squares. That is, the coefficient of x^n in the series for $S(x)/(1-x)$ is $\sum_{k=1}^{n} k^2$.

(c) Use the previous parts to prove that

$$\sum_{k=1}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}.$$

Homework Problems

Problem 15.10.
We will use generating functions to determine how many ways there are to use pennies, nickels, dimes, quarters, and half-dollars to give n cents change.

(a) Write the sequence P_n for the number of ways to use only pennies to change n cents. Write the generating function for that sequence.

(b) Write the sequence N_n for the number of ways to use only nickels to change n cents. Write the generating function for that sequence.

(c) Write the generating function for the number of ways to use only nickels and pennies to change n cents.

(d) Write the generating function for the number of ways to use pennies, nickels, dimes, quarters, and half-dollars to give n cents change.

(e) Explain how to use this function to find out how many ways are there to change 50 cents; you do *not* have to provide the answer or actually carry out the process.

Exam Problems

Problem 15.11.
The working days in the next year can be numbered 1, 2, 3, ..., 300. I'd like to avoid as many as possible.

- On even-numbered days, I'll say I'm sick.
- On days that are a multiple of 3, I'll say I was stuck in traffic.
- On days that are a multiple of 5, I'll refuse to come out from under the blankets.

In total, how many work days will I *avoid* in the coming year?

Problem 15.12.
Define the sequence $r_0, r_1, r_2, \ldots$ recursively by the rule that $r_0 = r_1 = 0$ and

$$r_n = 7r_{n-1} + 4r_{n-2} + (n+1),$$

for $n \geq 2$. Express the generating function of this sequence as a quotient of polynomials or products of polynomials. You do *not* have to find a closed form for r_n.

Problem 15.13.
Find the coefficients of $x^{10}y^5$ in $(19x + 4y)^{15}$

15.3.5 An "Impossible" Counting Problem

So far everything we've done with generating functions we could have done another way. But here is an absurd counting problem —really over the top! In how many ways can we fill a bag with n fruits subject to the following constraints?

- The number of apples must be even.

- The number of bananas must be a multiple of 5.
- There can be at most four oranges.
- There can be at most one pear.

For example, there are 7 ways to form a bag with 6 fruits:

Apples	6	4	4	2	2	0	0
Bananas	0	0	0	0	0	5	5
Oranges	0	2	1	4	3	1	0
Pears	0	0	1	0	1	0	1

These constraints are so complicated that the problem seems hopeless! But let's see what generating functions reveal.

Let's first construct a generating function for choosing apples. We can choose a set of 0 apples in one way, a set of 1 apple in zero ways (since the number of apples must be even), a set of 2 apples in one way, a set of 3 apples in zero ways, and so forth. So we have:

$$A(x) = 1 + x^2 + x^4 + x^6 + \cdots = \frac{1}{1 - x^2}$$

Similarly, the generating function for choosing bananas is:

$$B(x) = 1 + x^5 + x^{10} + x^{15} + \cdots = \frac{1}{1 - x^5}$$

Now, we can choose a set of 0 oranges in one way, a set of 1 orange in one way, and so on. However, we can not choose more than four oranges, so we have the generating function:

$$O(x) = 1 + x + x^2 + x^3 + x^4 = \frac{1 - x^5}{1 - x}$$

Here we're using the geometric sum formula. Finally, we can choose only zero or one pear, so we have:

$$P(x) = 1 + x$$

The Convolution Rule says that the generating function for choosing from among all four kinds of fruit is:

$$\begin{aligned}
A(x)B(x)O(x)P(x) &= \frac{1}{1 - x^2}\,\frac{1}{1 - x^5}\,\frac{1 - x^5}{1 - x}(1 + x) \\
&= \frac{1}{(1 - x)^2} \\
&= 1 + 2x + 3x^2 + 4x^3 + \cdots
\end{aligned}$$

Almost everything cancels! We're left with $1/(1 - x)^2$, which we found a power series for earlier: the coefficient of x^n is simply $n + 1$. Thus, the number of ways to form a bag of n fruits is just $n + 1$. This is consistent with the example we worked out, since there were 7 different fruit bags containing 6 fruits. *Amazing!*

15.3.6 Problems

Practice Problems

Problem 15.14 (Counting with Generating Functions)**.**
Let a_n be the number of ways to make change for \$$n$ using \$2 and \$3 coins. For example, $a_5 = 1$ because the only way to make change for \$5 is with one \$2 coin and one \$3 coin, but $a_6 = 2$ because there are two ways to make change for \$6, namely using three \$2 coins or using two \$3 coins.

Identify the generating function for the sequence of a_n's.

Problem 15.15.
Generating Functions and Sequences

Write a formula for the generating function of each of the following sequences.

(a) 0, 0, 1, 1, 1,...

(b) 1, 1, 0, 0, 0,...

(c) 1, 0, 1, 0, 1, 0, 1,...

(d) 1, 4, 6, 4, 1, 0, 0, 0,...

(e) 1, 1, 1/2, 1/6, 1/24, 1/120,...

(f) 1, 2, 3, 4, 5,...

(g) 1, 4, 9, 16, 25,...

Homework Problems

Problem 15.16.
Miss McGillicuddy never goes outside without a collection of pets. In particular:

- She brings a positive number of songbirds, which always come in pairs.
- She may or may not bring her alligator, Freddy.
- She brings at least 2 cats.
- She brings two or more chihuahuas and labradors leashed together in a line.

Let P_n denote the number of different collections of n pets that can accompany her, where we regard chihuahuas and labradors leashed up in different orders as different collections, even if there are the same number chihuahuas and labradors leashed in the line.

For example, $P_6 = 4$ since there are 4 possible collections of 6 pets:

- 2 songbirds, 2 cats, 2 chihuahuas leashed in line
- 2 songbirds, 2 cats, 2 labradors leashed in line
- 2 songbirds, 2 cats, a labrador leashed behind a chihuahua
- 2 songbirds, 2 cats, a chihuahua leashed behind a labrador

And $P_7 = 16$ since there are 16 possible collections of 7 pets:

- 2 songbirds, 3 cats, 2 chihuahuas leashed in line
- 2 songbirds, 3 cats, 2 labradors leashed in line
- 2 songbirds, 3 cats, a labrador leashed behind a chihuahua
- 2 songbirds, 3 cats, a chihuahua leashed behind a labrador
- 4 collections consisting of 2 songbirds, 2 cats, 1 alligator, and a line of 2 dogs
- 8 collections consisting of 2 songbirds, 2 cats, and a line of 3 dogs.

(a) Let

$$P(x) ::= P_0 + P_1 x + P_2 x^2 + P_3 x^3 + \cdots$$

be the generating function for the number of Miss McGillicuddy's pet collections. Verify that

$$P(x) = \frac{4x^6}{(1-x)^2(1-2x)}.$$

(b) Find a simple formula for P_n.

Problem 15.17.
Generating functions provide an interesting way to count the number of strings of matched parentheses. To do this, we'll use the description of these strings given in Definition 10.1.2 as the set, GoodCount, of strings of parentheses with a good count. Let c_n be the number of strings in GoodCount with exactly n left parentheses, and let $C(x)$ be the generating function for these numbers:

$$C(x) ::= c_0 + c_1 x + c_2 x^2 + \cdots .$$

(a) The *wrap* of a string, s, is the string, (s), that starts with a left parenthesis followed by the characters of s, and then ends with a right parenthesis. Explain why the generating function for the wraps of strings with a good count is $xC(x)$.

Hint: The wrap of a string with good count also has a good count that starts and ends with 0 and remains *positive* everywhere else.

(b) Explain why, for every string, s, with a good count, there is a unique sequence of strings $s_1, \ldots, s_k$ that are wraps of strings with good counts and $s = s_1 \cdots s_k$. For example, the string $r ::= (())()((())) \in \text{GoodCount}$ equals $s_1 s_2 s_3$ where $s_1 = (()), s_2 = (), s_3 = ((())$, and this is the only way to express r as a sequence of wraps of strings with good counts.

(c) Conclude that

$$C = 1 + xC + (xC)^2 + \cdots + (xC)^n + \cdots, \tag{15.10}$$

so

$$C = \frac{1}{1 - xC}, \tag{15.11}$$

and hence

$$C = \frac{1 \pm \sqrt{1-4x}}{2x}. \tag{15.12}$$

Let $D(x) ::= 2xC(x)$. Expressing D as a power series

$$D(x) = d_0 + d_1 x + d_2 x^2 + \cdots,$$

we have

$$c_n = \frac{d_{n+1}}{2}. \tag{15.13}$$

(d) Use (15.12), (15.13), and the value of c_0 to conclude that

$$D(x) = 1 - \sqrt{1-4x}.$$

(e) Prove that

$$d_n = \frac{(2n-3) \cdot (2n-5) \cdots 5 \cdot 3 \cdot 1 \cdot 2^n}{n!}.$$

Hint: $d_n = D^{(n)}(0)/n!$

(f) Conclude that

$$c_n = \frac{1}{n+1}\binom{2n}{n}.$$

Exam Problems

Problem 15.18.
T-Pain is planning an epic boat trip and he needs to decide what to bring with him.

- He definitely wants to bring burgers, but they only come in packs of 6.
- He and his two friends can't decide whether they want to dress formally or casually. He'll either bring 0 pairs of flip flops or 3 pairs.

- He doesn't have very much room in his suitcase for towels, so he can bring at most 2.
- In order for the boat trip to be truly epic, he has to bring at least 1 nautical-themed pashmina afghan.

(a) Let g_n be the the number of different ways for T-Pain to bring n items (burgers, pairs of flip flops, towels, and/or afghans) on his boat trip. Express the generating function $G(x) ::= \sum_{n=0}^{\infty} g_n x^n$ as a quotient of polynomials.

(b) Find a closed formula in n for the number of ways T-Pain can bring exactly n items with him.

Part IV

Probability

Chapter 16

Introduction to Probability

Probability plays a key role in the sciences —"hard" and social —including computer science. Many algorithms rely on randomization. Investigating their correctness and performance requires probability theory. Moreover, computer systems designs, such as memory management, branch prediction, packet routing, and load balancing are based on probabilistic assumptions and analyses. Probability is central as well in related subjects such as information theory, cryptography, artificial intelligence, and game theory. But we'll start with a more down-to-earth application: getting a prize in a game show.

16.1 Monty Hall

In the September 9, 1990 issue of *Parade* magazine, the columnist Marilyn vos Savant responded to this letter:

> *Suppose you're on a game show, and you're given the choice of three doors. Behind one door is a car, behind the others, goats. You pick a door, say number 1, and the host, who knows what's behind the doors, opens another door, say number 3, which has a goat. He says to you, "Do you want to pick door number 2?" Is it to your advantage to switch your choice of doors?*
>
> Craig. F. Whitaker
> Columbia, MD

The letter describes a situation like one faced by contestants on the 1970's game show *Let's Make a Deal*, hosted by Monty Hall and Carol Merrill. Marilyn replied that the contestant should indeed switch. She explained that if the car was behind either of the two unpicked doors —which is twice as likely as the the car being behind the picked door —the contestant wins by switching. But she soon received a torrent of letters, many from mathematicians, telling her that she was wrong. The problem generated thousands of hours of heated debate.

This incident highlights a fact about probability: the subject uncovers lots of examples where ordinary intuition leads to completely wrong conclusions. So until you've studied probabilities enough to have refined your intuition, a way to avoid errors is to fall back on a rigorous, systematic approach such as the Four Step Method.

16.1.1 The Four Step Method

Every probability problem involves some sort of randomized experiment, process, or game. And each such problem involves two distinct challenges:

1. How do we model the situation mathematically?

2. How do we solve the resulting mathematical problem?

In this section, we introduce a four step approach to questions of the form, "What is the probability that —— ?" In this approach, we build a probabilistic model step-by-step, formalizing the original question in terms of that model. Remarkably, the structured thinking that this approach imposes provides simple solutions to many famously-confusing problems. For example, as you'll see, the four step method cuts through the confusion surrounding the Monty Hall problem like a Ginsu knife. However, more complex probability questions may spin off challenging counting, summing, and approximation problems— which, fortunately, you've already spent weeks learning how to solve.

16.1.2 Clarifying the Problem

Craig's original letter to Marilyn vos Savant is a bit vague, so we must make some assumptions in order to have any hope of modeling the game formally:

1. The car is equally likely to be hidden behind each of the three doors.

2. The player is equally likely to pick each of the three doors, regardless of the car's location.

3. After the player picks a door, the host *must* open a different door with a goat behind it and offer the player the choice of staying with the original door or switching.

4. If the host has a choice of which door to open, then he is equally likely to select each of them.

In making these assumptions, we're reading a lot into Craig Whitaker's letter. Other interpretations are at least as defensible, and some actually lead to different answers. But let's accept these assumptions for now and address the question, "What is the probability that a player who switches wins the car?"

16.1.3 Step 1: Find the Sample Space

Our first objective is to identify all the possible outcomes of the experiment. A typical experiment involves several randomly-determined quantities. For example, the Monty Hall game involves three such quantities:

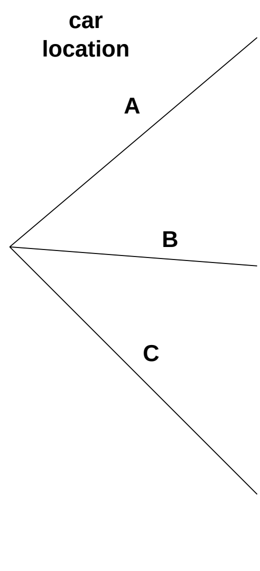
1. The door concealing the car.

2. The door initially chosen by the player.

3. The door that the host opens to reveal a goat.

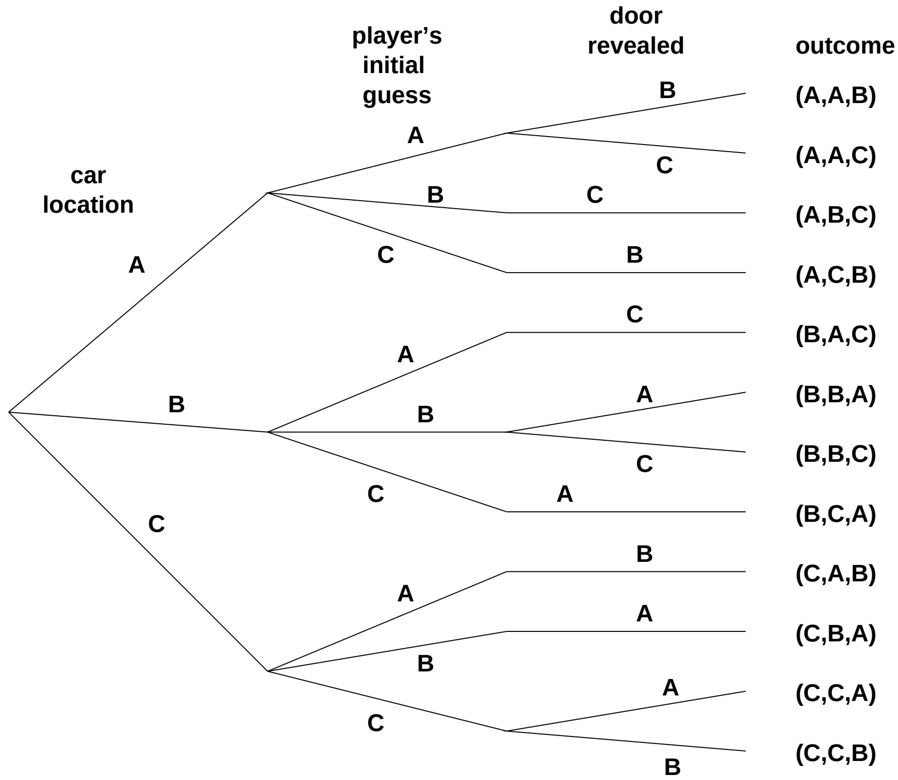
Every possible combination of these randomly-determined quantities is called an *outcome*. The set of all possible outcomes is called the *sample space* for the experiment.

A *tree diagram* is a graphical tool that can help us work through the four step approach when the number of outcomes is not too large or the problem is nicely structured. In particular, we can use a tree diagram to help understand the sample space of an experiment. The first randomly-determined quantity in our experiment is the door concealing the prize. We represent this as a tree with three branches:



In this diagram, the doors are called A, B, and C instead of 1, 2, and 3 because we'll be adding a lot of other numbers to the picture later.

Now, for each possible location of the prize, the player could initially choose any of the three doors. We represent this in a second layer added to the tree. Then a third layer represents the possibilities of the final step when the host opens a door to reveal a goat:



Notice that the third layer reflects the fact that the host has either one choice or two, depending on the position of the car and the door initially selected by the player. For example, if the prize is behind door A and the player picks door B, then the host must open door C. However, if the prize is behind door A and the player picks door A, then the host could open either door B or door C.

Now let's relate this picture to the terms we introduced earlier: the leaves of the tree represent *outcomes* of the experiment, and the set of all leaves represents the *sample space*. Thus, for this experiment, the sample space consists of 12 outcomes. For reference, we've labeled each outcome with a triple of doors indicating:

$$(\text{door concealing prize},\ \text{door initially chosen},\ \text{door opened to reveal a goat})$$

In these terms, the sample space is the set:

$$\left\{ \begin{array}{cccccc} (A,A,B), & (A,A,C), & (A,B,C), & (A,C,B), & (B,A,C), & (B,B,A), \\ (B,B,C), & (B,C,A), & (C,A,B), & (C,B,A), & (C,C,A), & (C,C,B) \end{array} \right\}$$

The tree diagram has a broader interpretation as well: we can regard the whole experiment as following a path from the root to a leaf, where the branch taken at each stage is "randomly" determined. Keep this interpretation in mind; we'll use it again later.

16.1.4 Step 2: Define Events of Interest

Our objective is to answer questions of the form "What is the probability that ...?", where the missing phrase might be "the player wins by switching", "the player initially picked the door concealing the prize", or "the prize is behind door C", for example. Each of these phrases characterizes a set of outcomes: the outcomes specified by "the prize is behind door C" is:

$$\{(C, A, B), (C, B, A), (C, C, A), (C, C, B)\}$$

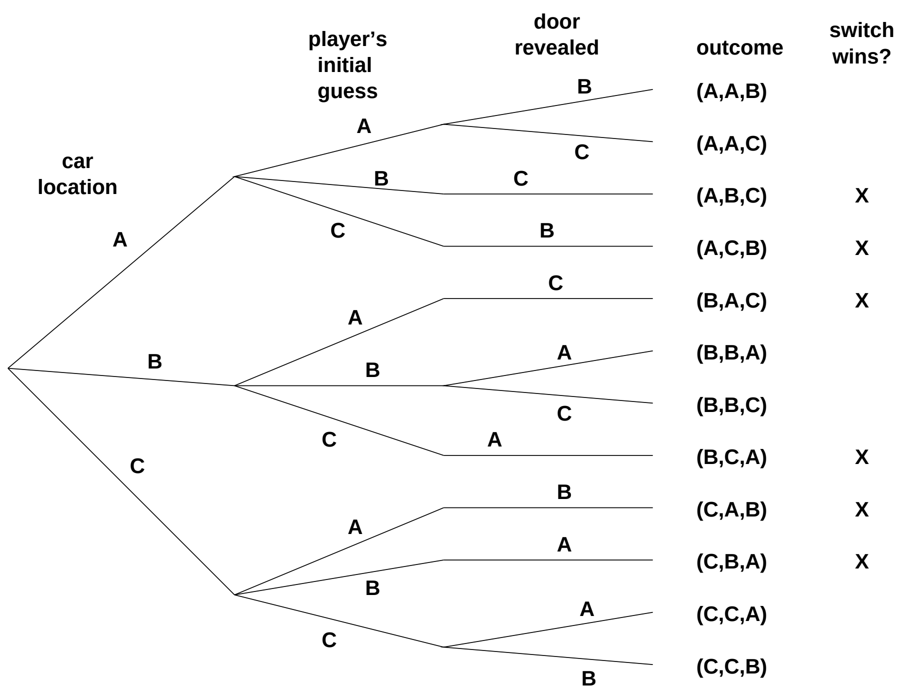
A set of outcomes is called an *event*. So the event that the player initially picked the door concealing the prize is the set:

$$\{(A, A, B), (A, A, C), (B, B, A), (B, B, C), (C, C, A), (C, C, B)\}$$

And what we're really after, the event that the player wins by switching, is the set of outcomes:

$$\{(A, B, C), (A, C, B), (B, A, C), (B, C, A), (C, A, B), (C, B, A)\}$$

Let's annotate our tree diagram to indicate the outcomes in this event.



Notice that exactly half of the outcomes are marked, meaning that the player wins by switching in half of all outcomes. You might be tempted to conclude that a player who switches wins with probability $1/2$. *This is wrong.* The reason is that these outcomes are not all equally likely, as we'll see shortly.

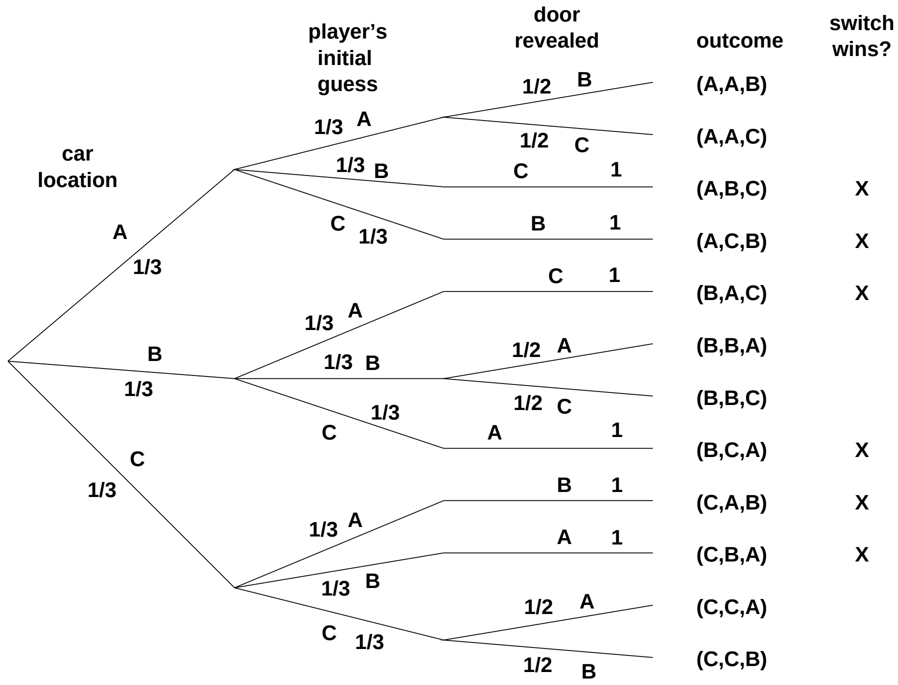
16.1.5 Step 3: Determine Outcome Probabilities

So far we've enumerated all the possible outcomes of the experiment. Now we must start assessing the likelihood of those outcomes. In particular, the goal of this step is to assign each outcome a probability, indicating the fraction of the time this outcome is expected to occur. The sum of all outcome probabilities must be one, reflecting the fact that there always is an outcome.

Ultimately, outcome probabilities are determined by the phenomenon we're modeling and thus are not quantities that we can derive mathematically. However, mathematics can help us compute the probability of every outcome *based on fewer and more elementary modeling decisions.* In particular, we'll break the task of determining outcome probabilities into two stages.

Step 3a: Assign Edge Probabilities

First, we record a probability on each *edge* of the tree diagram. These edge-probabilities are determined by the assumptions we made at the outset: that the prize is equally likely to be behind each door, that the player is equally likely to pick each door, and that the host is equally likely to reveal each goat, if he has a choice. Notice that when the host has no choice regarding which door to open, the single branch is assigned probability 1.



Step 3b: Compute Outcome Probabilities

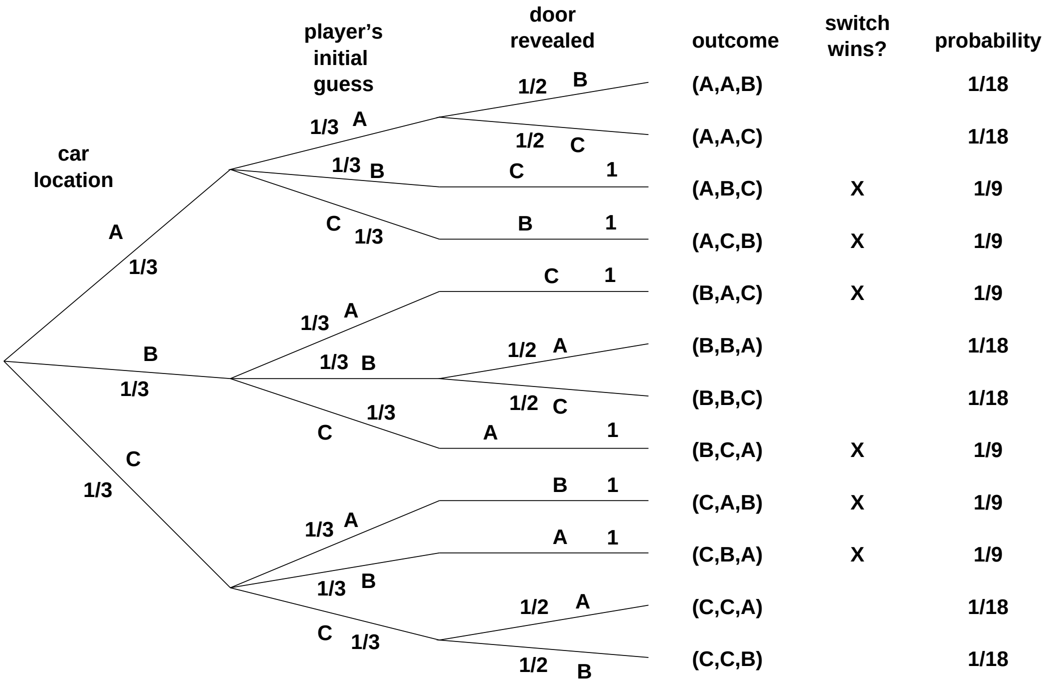
Our next job is to convert edge probabilities into outcome probabilities. This is a purely mechanical process: *the probability of an outcome is equal to the product of the edge-probabilities on the path from the root to that outcome*. For example, the probability of the topmost outcome, (A, A, B) is

$$\frac{1}{3} \cdot \frac{1}{3} \cdot \frac{1}{2} = \frac{1}{18}.$$

There's an easy, intuitive justification for this rule. As the steps in an experiment progress randomly along a path from the root of the tree to a leaf, the probabilities on the edges indicate how likely the walk is to proceed along each branch. For example, a path starting at the root in our example is equally likely to go down each of the three top-level branches.

Now, how likely is such a walk to arrive at the topmost outcome, (A, A, B)? Well, there is a 1-in-3 chance that a walk would follow the A-branch at the top level, a 1-in-3 chance it would continue along the A-branch at the second level, and 1-in-2 chance it would follow the B-branch at the third level. Thus, it seems that about 1 walk in 18 should arrive at the (A, A, B) leaf, which is precisely the probability we assign it.

Anyway, let's record all the outcome probabilities in our tree diagram.



Specifying the probability of each outcome amounts to defining a function that maps each outcome to a probability. This function is usually called **Pr**. In these terms, we've just determined that:

$$\Pr\{(A, A, B)\} = \frac{1}{18}$$
$$\Pr\{(A, A, C)\} = \frac{1}{18}$$
$$\Pr\{(A, B, C)\} = \frac{1}{9}$$
etc.

16.1.6 Step 4: Compute Event Probabilities

We now have a probability for each *outcome*, but we want to determine the probability of an *event* which will be the sum of the probabilities of the outcomes in it. The probability of an event, E, is written $\Pr\{E\}$. For example, the probability of the event that the player wins by switching is:

$$\begin{aligned}
\Pr\{\text{switching wins}\} &= \Pr\{(A,B,C)\} + \Pr\{(A,C,B)\} + \Pr\{(B,A,C)\} + \\
&\qquad \Pr\{(B,C,A)\} + \Pr\{(C,A,B)\} + \Pr\{(C,B,A)\} \\
&= \frac{1}{9} + \frac{1}{9} + \frac{1}{9} + \frac{1}{9} + \frac{1}{9} + \frac{1}{9} \\
&= \frac{2}{3}
\end{aligned}$$

It seems Marilyn's answer is correct; a player who switches doors wins the car with probability $2/3$! In contrast, a player who stays with his or her original door wins with probability $1/3$, since staying wins if and only if switching loses.

We're done with the problem! We didn't need any appeals to intuition or ingenious analogies. In fact, no mathematics more difficult than adding and multiplying fractions was required. The only hard part was resisting the temptation to leap to an "intuitively obvious" answer.

16.1.7 An Alternative Interpretation of the Monty Hall Problem

Was Marilyn really right? Our analysis suggests she was. But a more accurate conclusion is that her answer is correct *provided we accept her interpretation of the question*. There is an equally plausible interpretation in which Marilyn's answer is wrong. Notice that Craig Whitaker's original letter does not say that the host is *required* to reveal a goat and offer the player the option to switch, merely that he *did* these things. In fact, on the *Let's Make a Deal* show, Monty Hall sometimes simply opened the door that the contestant picked initially. Therefore, if he wanted to, Monty could give the option of switching only to contestants who picked the correct door initially. In this case, switching never works!

16.1.8 Problems

Class Problems

Problem 16.1.
[A Baseball Series]
The New York Yankees and the Boston Red Sox are playing a two-out-of-three series. (In other words, they play until one team has won two games. Then that team is declared the overall winner and the series ends.) Assume that the Red Sox win each game with probability $3/5$, regardless of the outcomes of previous games.

Answer the questions below using the four step method. You can use the same tree diagram for all three problems.

(a) What is the probability that a total of 3 games are played?

(b) What is the probability that the winner of the series loses the first game?

(c) What is the probability that the *correct* team wins the series?

Problem 16.2.
To determine which of two people gets a prize, a coin is flipped twice. If the flips are a Head and then a Tail, the first player wins. If the flips are a Tail and then a Head, the second player wins. However, if both coins land the same way, the flips don't count and whole the process starts over.

Assume that on each flip, a Head comes up with probability p, regardless of what happened on other flips. Use the four step method to find a simple formula for the probability that the first player wins. What is the probability that neither player wins?

Suggestions: The tree diagram and sample space are infinite, so you're not going to finish drawing the tree. Try drawing only enough to see a pattern. Summing all the winning outcome probabilities directly is difficult. However, a neat trick solves this problem and many others. Let s be the sum of all winning outcome probabilities in the whole tree. Notice that *you can write the sum of all the winning probabilities in certain subtrees as a function of* s. Use this observation to write an equation in s and then solve.

Problem 16.3.
[The Four-Door Deal]
Let's see what happens when *Let's Make a Deal* is played with **four** doors. A prize is hidden behind one of the four doors. Then the contestant picks a door. Next, the host opens an unpicked door that has no prize behind it. The contestant is allowed to stick with their original door or to switch to one of the two unopened, unpicked doors. The contestant wins if their final choice is the door hiding the prize.

Use The Four Step Method of Section 16.1 to find the following probabilities. The tree diagram may become awkwardly large, in which case just draw enough of it to make its structure clear.

(a) Contestant Stu, a sanitation engineer from Trenton, New Jersey, stays with his original door. What is the probability that Stu wins the prize?

(b) Contestant Zelda, an alien abduction researcher from Helena, Montana, switches to one of the remaining two doors with equal probability. What is the probability that Zelda wins the prize?

Problem 16.4.
[Simulating a fair coin] Suppose you need a fair coin to decide which door to choose in the 6.042 Monty Hall game. After making everyone in your group empty their pockets, all you managed to turn up is some crumpled bubble gum wrappers, a few used tissues, and one penny. However, the penny was from Prof. Meyer's pocket, so it is **not** safe to assume that it is a fair coin.

How can we use a coin of unknown bias to get the same effect as a fair coin of bias $1/2$? Draw the tree diagram for your solution, but since it is infinite, draw only enough to see a pattern.

Suggestion: A neat trick allows you to sum all the outcome probabilities that cause you to say "Heads": Let s be the sum of all "Heads" outcome probabilities in the whole tree. Notice that *you can write the sum of all the "Heads" outcome probabilities in certain subtrees as a function of s*. Use this observation to write an equation in s and then solve.

Homework Problems

Problem 16.5.
I have a deck of 52 regular playing cards, 26 red, 26 black, randomly shuffled. They all lie face down in the deck so that you can't see them. I will draw a card off the top of the deck and turn it face up so that you can see it and then put it aside. I will continue to turn up cards like this but at some point while there are still cards left in the deck, you have to declare that you want the next card in the deck to be turned up. If that next card turns up black you win and otherwise you lose. Either way, the game is then over.

(a) Show that if you take the first card before you have seen any cards, you then have probability $1/2$ of winning the game.

(b) Suppose you don't take the first card and it turns up red. Show that you have then have a probability of winning the game that is greater than $1/2$.

(c) If there are r red cards left in the deck and b black cards, show that the probability of winning in you take the next card is $b/(r + b)$.

(d) Either,

1. come up with a strategy for this game that gives you a probability of winning strictly greater than $1/2$ and prove that the strategy works, or,
2. come up with a proof that no such strategy can exist.

16.2 Set Theory and Probability

Let's abstract what we've just done in this Monty Hall example into a general mathematical definition of probability. In the Monty Hall example, there were only finitely many possible outcomes. Other examples in this course will have a countably infinite number of outcomes.

General probability theory deals with uncountable sets like the set of real numbers, but we won't need these, and sticking to countable sets lets us define the probability of events using sums instead of integrals. It also lets us avoid some distracting technical problems in set theory like the Banach-Tarski "paradox" mentioned in Chapter 5.1.6.

16.2.1 Probability Spaces

Definition 16.2.1. A countable *sample space*, $\mathcal{S}$, is a nonempty countable set. An element $w \in \mathcal{S}$ is called an *outcome*. A subset of $\mathcal{S}$ is called an *event*.

Definition 16.2.2. A *probability function* on a sample space, $\mathcal{S}$, is a total function $\Pr\{\} : \mathcal{S} \to \mathbb{R}$ such that

- $\Pr\{w\} \geq 0$ for all $w \in \mathcal{S}$, and
- $\sum_{w \in \mathcal{S}} \Pr\{w\} = 1$.

The sample space together with a probability function is called a *probability space*.

For any event, $E \subseteq \mathcal{S}$, the *probability of* E is defined to be the sum of the probabilities of the outcomes in E:

$$\Pr\{E\} ::= \sum_{w \in E} \Pr\{w\}.$$

An immediate consequence of the definition of event probability is that for *disjoint* events, E, F,

$$\Pr\{E \cup F\} = \Pr\{E\} + \Pr\{F\}.$$

This generalizes to a countable number of events. Namely, a collection of sets is *pairwise disjoint* when no element is in more than one of them —formally, $A \cap B = \emptyset$ for all sets $A \neq B$ in the collection.

Rule (Sum Rule). *If* $\{E_0, E_1, \ldots\}$ *is collection of pairwise disjoint events, then*

$$\Pr\left\{\bigcup_{n \in \mathbb{N}} E_n\right\} = \sum_{n \in \mathbb{N}} \Pr\{E_n\}.$$

The Sum Rule[1] lets us analyze a complicated event by breaking it down into simpler cases. For example, if the probability that a randomly chosen MIT student is native to the United States is 60%, to Canada is 5%, and to Mexico is 5%, then the probability that a random MIT student is native to North America is 70%.

Another consequence of the Sum Rule is that $\Pr\{A\} + \Pr\left\{\overline{A}\right\} = 1$, which follows because $\Pr\{\mathcal{S}\} = 1$ and $\mathcal{S}$ is the union of the disjoint sets A and $\overline{A}$. This equation often comes up in the form

[1] If you think like a mathematician, you should be wondering if the infinite sum is really necessary. Namely, suppose we had only used finite sums in Definition 16.2.2 instead of sums over all natural numbers. Would this imply the result for infinite sums? It's hard to find counterexamples, but there are some: it is possible to find a pathological "probability" measure on a sample space satisfying the Sum Rule for finite unions, in which the outcomes $w_0, w_1, \ldots$ each have probability zero, and the probability assigned to any event is either zero or one! So the infinite Sum Rule fails dramatically, since the whole space is of measure one, but it is a union of the outcomes of measure zero.

The construction of such weird examples is beyond the scope of this text. You can learn more about this by taking a course in Set Theory and Logic that covers the topic of "ultrafilters."

Rule (Complement Rule)**.**

$$\Pr\left\{\overline{A}\right\} = 1 - \Pr\{A\}.$$

Sometimes the easiest way to compute the probability of an event is to compute the probability of its complement and then apply this formula.

Some further basic facts about probability parallel facts about cardinalities of finite sets. In particular:

$$\Pr\{B - A\} = \Pr\{B\} - \Pr\{A \cap B\}, \qquad \text{(Difference Rule)}$$
$$\Pr\{A \cup B\} = \Pr\{A\} + \Pr\{B\} - \Pr\{A \cap B\}, \qquad \text{(Inclusion-Exclusion)}$$
$$\Pr\{A \cup B\} \leq \Pr\{A\} + \Pr\{B\}. \qquad \text{(Boole's Inequality)}$$

The Difference Rule follows from the Sum Rule because B is the union of the disjoint sets $B - A$ and $A \cap B$. Inclusion-Exclusion then follows from the Sum and Difference Rules, because $A \cup B$ is the union of the disjoint sets A and $B - A$. Boole's inequality is an immediate consequence of Inclusion-Exclusion since probabilities are nonnegative.

The two event Inclusion-Exclusion equation above generalizes to n events in the same way as the corresponding Inclusion-Exclusion rule for n sets. Boole's inequality also generalizes to

$$\Pr\{E_1 \cup \cdots \cup E_n\} \leq \Pr\{E_1\} + \cdots + \Pr\{E_n\}. \qquad \text{(Union Bound)}$$

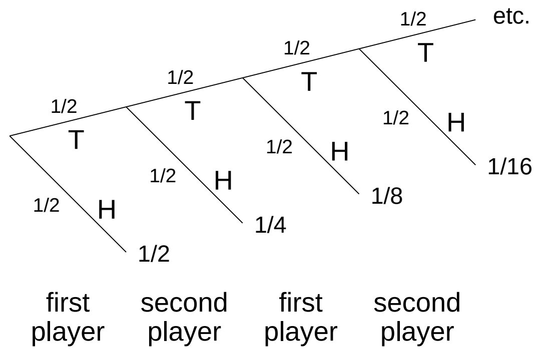
This simple Union Bound is actually useful in many calculations. For example, suppose that E_i is the event that the i-th critical component in a spacecraft fails. Then $E_1 \cup \cdots \cup E_n$ is the event that *some* critical component fails. The Union Bound can give an adequate upper bound on this vital probability.

Similarly, the Difference Rule implies that

$$\text{If } A \subseteq B, \text{ then } \Pr\{A\} \leq \Pr\{B\}. \qquad \text{(Monotonicity)}$$

16.2.2 An Infinite Sample Space

Suppose two players take turns flipping a fair coin. Whoever flips heads first is declared the winner. What is the probability that the first player wins? A tree diagram for this problem is shown below:



The event that the first player wins contains an infinite number of outcomes, but we can still sum their probabilities:

$$\begin{aligned}\Pr\{\text{first player wins}\} &= \frac{1}{2} + \frac{1}{8} + \frac{1}{32} + \frac{1}{128} + \cdots \\ &= \frac{1}{2}\sum_{n=0}^{\infty}\left(\frac{1}{4}\right)^n \\ &= \frac{1}{2}\left(\frac{1}{1-1/4}\right) = \frac{2}{3}.\end{aligned}$$

Similarly, we can compute the probability that the second player wins:

$$\begin{aligned}\Pr\{\text{second player wins}\} &= \frac{1}{4} + \frac{1}{16} + \frac{1}{64} + \frac{1}{256} + \cdots \\ &= \frac{1}{3}.\end{aligned}$$

To be formal about this, sample space is the infinite set

$$\mathcal{S} ::= \{\mathtt{T}^n\mathtt{H} \mid n \in \mathbb{N}\}$$

where $\mathtt{T}^n$ stands for a length n string of $\mathtt{T}$'s. The probability function is

$$\Pr\{\mathtt{T}^n\mathtt{H}\} ::= \frac{1}{2^{n+1}}.$$

Since this function is obviously nonnegative, To verify that this is a probability space, we just have to check that all the probabilities sum to 1. But this follows directly from the formula for the sum of a geometric series:

$$\sum_{\mathtt{T}^n\mathtt{H}\in\mathcal{S}} \Pr\{\mathtt{T}^n\mathtt{H}\} = \sum_{n\in\mathbb{N}} \frac{1}{2^{n+1}} = \frac{1}{2}\sum_{n\in\mathbb{N}}\frac{1}{2^n} = 1.$$

Notice that this model does not have an outcome corresponding to the possibility that both players keep flipping tails forever —in the diagram, flipping forever corresponds to following the infinite path in the tree without ever reaching a leaf/outcome. If leaving this possibility out of the model bothers you, you're welcome to fix it by adding another outcome, w_{forever}, to indicate that that's what happened. Of course since the probabililities of the other outcomes already sum to 1, you have to define the probability of w_{forever} to be 0. Now outcomes with probability zero will have no impact on our calculations, so there's no harm in adding it in if it makes you happier. On the other hand, there's also no harm in simply leaving it out as we did, since it has no impact.

The mathematical machinery we've developed is adequate to model and analyze many interesting probability problems with infinite sample spaces. However, some intricate infinite processes require uncountable sample spaces along with more powerful (and more complex) measure-theoretic notions of probability. For example, if we generate an infinite sequence of random bits $b_1, b_2, b_3, \ldots$, then what is the probability that

$$\frac{b_1}{2^1} + \frac{b_2}{2^2} + \frac{b_3}{2^3} + \cdots$$

is a rational number? Fortunately, we won't have any need to worry about such things.

16.2.3 Problems

Class Problems

Problem 16.6.
Suppose there is a system with n components, and we know from past experience that any particular component will fail in a given year with probability p. That is, letting F_i be the event that the ith component fails within one year, we have

$$\Pr\{F_i\} = p$$

for $1 \le i \le n$. The *system* will fail if *any one* of its components fails. What can we say about the probability that the system will fail within one year?

Let F be the event that the system fails within one year. Without any additional assumptions, we can't get an exact answer for $\Pr\{F\}$. However, we can give useful upper and lower bounds, namely,

$$p \le \Pr\{F\} \le np. \tag{16.1}$$

We may as well assume $p < 1/n$, since the upper bound is trivial otherwise. For example, if $n = 100$ and $p = 10^{-5}$, we conclude that there is at most one chance in 1000 of system failure within a year and at least one chance in 100,000.

Let's model this situation with the sample space $\mathcal{S} ::= \mathcal{P}(\{1, \ldots, n\})$ whose outcomes are subsets of positive integers $\le n$, where $s \in \mathcal{S}$ corresponds to the indices of exactly those components that fail within one year. For example, $\{2, 5\}$ is the outcome that the second and fifth components failed within a year and none of the other components failed. So the outcome that the system did not fail corresponds to the emptyset, $\emptyset$.

(a) Show that the probability that the system fails could be as small as p by describing appropriate probabilities for the outcomes. Make sure to verify that the sum of your outcome probabilities is 1.

(b) Show that the probability that the system fails could actually be as large as np by describing appropriate probabilities for the outcomes. Make sure to verify that the sum of your outcome probabilities is 1.

(c) Prove inequality (16.1).

Problem 16.7.
Here are some handy rules for reasoning about probabilities that all follow directly from the Disjoint Sum Rule in the Appendix. Prove them.

$$\Pr\{A - B\} = \Pr\{A\} - \Pr\{A \cap B\} \qquad \text{(Difference Rule)}$$

$$\Pr\{\overline{A}\} = 1 - \Pr\{A\} \qquad \text{(Complement Rule)}$$

$$\Pr\{A \cup B\} = \Pr\{A\} + \Pr\{B\} - \Pr\{A \cap B\} \qquad \text{(Inclusion-Exclusion)}$$

$$\Pr\{A \cup B\} \le \Pr\{A\} + \Pr\{B\}. \qquad \text{(2-event Union Bound)}$$

$$\text{If } A \subseteq B, \text{ then } \Pr\{A\} \le \Pr\{B\}. \qquad \text{(Monotonicity)}$$

Problem 16.8.
Suppose $\Pr\{\} : \mathcal{S} \to [0,1]$ is a probability function on a sample space, $\mathcal{S}$, and let B be an event such that $\Pr\{B\} > 0$. Define a function $\Pr_B\{\cdot\}$ on events outcomes $w \in \mathcal{S}$ by the rule:

$$\Pr_B\{w\} ::= \begin{cases} \Pr\{w\}/\Pr\{B\} & \text{if } w \in B, \\ 0 & \text{if } w \notin B. \end{cases} \qquad (16.2)$$

(a) Prove that $\Pr_B\{\cdot\}$ is also a probability function on $\mathcal{S}$ according to Definition 16.2.2.

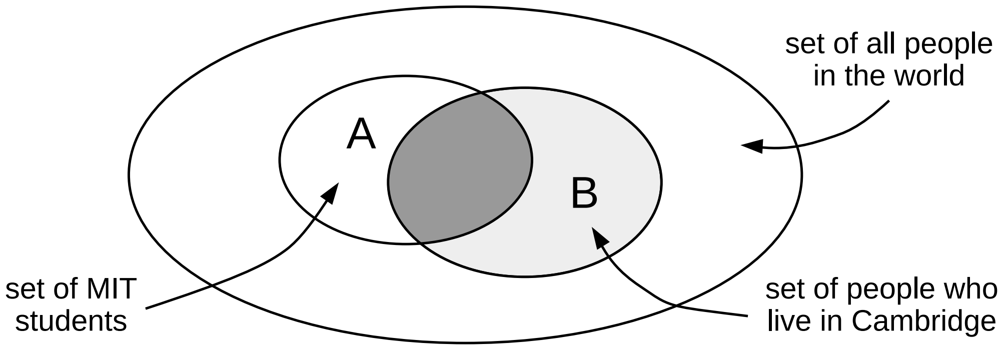
(b) Prove that

$$\Pr_B\{A\} = \frac{\Pr\{A \cap B\}}{\Pr\{B\}}$$

for all $A \subseteq \mathcal{S}$.

16.3 Conditional Probability

Suppose that we pick a random person in the world. Everyone has an equal chance of being selected. Let A be the event that the person is an MIT student, and let B be the event that the person lives in Cambridge. What are the probabilities of these events? Intuitively, we're picking a random point in the big ellipse shown below and asking how likely that point is to fall into region A or B:



The vast majority of people in the world neither live in Cambridge nor are MIT students, so events A and B both have low probability. But what is the probability that a person is an MIT student, *given* that the person lives in Cambridge? This should be much greater— but what is it exactly?

What we're asking for is called a *conditional probability*; that is, the probability that one event happens, given that some other event definitely happens. Questions about conditional probabilities come up all the time:

- What is the probability that it will rain this afternoon, given that it is cloudy this morning?
- What is the probability that two rolled dice sum to 10, given that both are odd?
- What is the probability that I'll get four-of-a-kind in Texas No Limit Hold 'Em Poker, given that I'm initially dealt two queens?

There is a special notation for conditional probabilities. In general, $\Pr\{A \mid B\}$ denotes the probability of event A, given that event B happens. So, in our example, $\Pr\{A \mid B\}$ is the probability that a random person is an MIT student, given that he or she is a Cambridge resident.

How do we compute $\Pr\{A \mid B\}$? Since we are *given* that the person lives in Cambridge, we can forget about everyone in the world who does not. Thus, all outcomes outside event B are irrelevant. So, intuitively, $\Pr\{A \mid B\}$ should be the fraction of Cambridge residents that are also MIT students; that is, the answer should be the probability that the person is in set $A \cap B$ (darkly shaded) divided by the probability that the person is in set B (lightly shaded). This motivates the definition of conditional probability:

Definition 16.3.1.

$$\Pr\{A \mid B\} ::= \frac{\Pr\{A \cap B\}}{\Pr\{B\}}$$

If $\Pr\{B\} = 0$, then the conditional probability $\Pr\{A \mid B\}$ is undefined.

Pure probability is often counterintuitive, but conditional probability is worse! Conditioning can subtly alter probabilities and produce unexpected results in randomized algorithms and computer systems as well as in betting games. Yet, the mathematical definition of conditional probability given above is very simple and should give you no trouble— provided you rely on formal reasoning and not intuition.

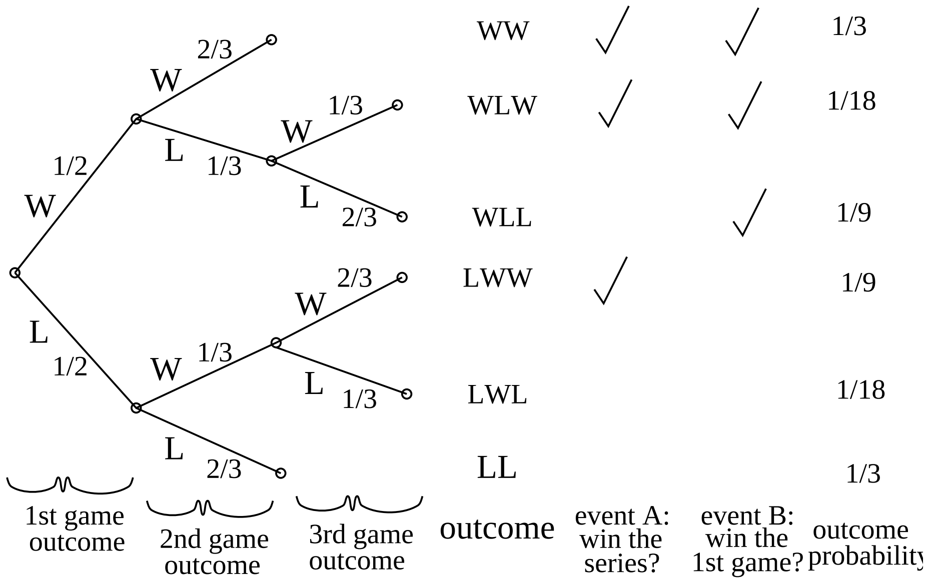
16.3.1 The "Halting Problem"

The *Halting Problem* was the first example of a property that could not be tested by any program. It was introduced by Alan Turing in his seminal 1936 paper. The problem is to determine whether a Turing machine halts on a given ... yadda yadda yadda ... what's much *more important*, it was the name of the MIT EECS department's famed C-league hockey team.

In a best-of-three tournament, the Halting Problem wins the first game with probability $1/2$. In subsequent games, their probability of winning is determined by the outcome of the previous game. If the Halting Problem won the previous game, then they are invigorated by victory and win the current game with probability $2/3$. If they lost the previous game, then they are demoralized by defeat and win the current game with probablity only $1/3$. What is the probability that the Halting Problem wins the tournament, given that they win the first game?

This is a question about a conditional probability. Let A be the event that the Halting Problem wins the tournament, and let B be the event that they win the first game. Our goal is then to determine the conditional probability $\Pr\{A \mid B\}$.

We can tackle conditional probability questions just like ordinary probability problems: using a tree diagram and the four step method. A complete tree diagram is shown below, followed by an explanation of its construction and use.



Step 1: Find the Sample Space

Each internal vertex in the tree diagram has two children, one corresponding to a win for the Halting Problem (labeled W) and one corresponding to a loss (labeled L). The complete sample space is:

$$\mathcal{S} = \{WW,\ WLW,\ WLL,\ LWW,\ LWL,\ LL\}$$

Step 2: Define Events of Interest

The event that the Halting Problem wins the whole tournament is:

$$T = \{WW,\ WLW,\ LWW\}$$

And the event that the Halting Problem wins the first game is:

$$F = \{WW, WLW, WLL\}$$

The outcomes in these events are indicated with checkmarks in the tree diagram.

Step 3: Determine Outcome Probabilities

Next, we must assign a probability to each outcome. We begin by labeling edges as specified in the problem statement. Specifically, The Halting Problem has a $1/2$ chance of winning the first game, so the two edges leaving the root are each assigned probability $1/2$. Other edges are labeled $1/3$ or $2/3$ based on the outcome of the preceding game. We then find the probability of each outcome by multiplying all probabilities along the corresponding root-to-leaf path. For example, the probability of outcome WLL is:

$$\frac{1}{2} \cdot \frac{1}{3} \cdot \frac{2}{3} = \frac{1}{9}$$

Step 4: Compute Event Probabilities

We can now compute the probability that The Halting Problem wins the tournament, given that they win the first game:

$$\begin{aligned}
\Pr\{A \mid B\} &= \frac{\Pr\{A \cap B\}}{\Pr\{B\}} \\
&= \frac{\Pr\{\{WW, WLW\}\}}{\Pr\{\{WW, WLW, WLL\}\}} \\
&= \frac{1/3 + 1/18}{1/3 + 1/18 + 1/9} \\
&= \frac{7}{9}
\end{aligned}$$

We're done! If the Halting Problem wins the first game, then they win the whole tournament with probability $7/9$.

16.3.2 Why Tree Diagrams Work

We've now settled into a routine of solving probability problems using tree diagrams. But we've left a big question unaddressed: what is the mathematical justification behind those funny little pictures? Why do they work?

The answer involves conditional probabilities. In fact, the probabilities that we've been recording on the edges of tree diagrams *are* conditional probabilities. For example, consider the uppermost path in the tree diagram for the Halting Problem, which corresponds to the outcome WW. The first edge is labeled $1/2$, which is the probability that the Halting Problem wins the first game. The second edge is labeled $2/3$, which is the probability that the Halting Problem wins the second game, *given* that they won the first— that's a conditional probability! More generally, on each edge of a tree diagram, we record the probability that the experiment proceeds along that path, given that it reaches the parent vertex.

So we've been using conditional probabilities all along. But why can we multiply edge probabilities to get outcome probabilities? For example, we concluded that:

$$\begin{aligned}
\Pr\{WW\} &= \frac{1}{2} \cdot \frac{2}{3} \\
&= \frac{1}{3}
\end{aligned}$$

Why is this correct?

The answer goes back to Definition 16.3.1 of conditional probability which could be written in a form called the *Product Rule* for probabilities:

Rule (Product Rule for 2 Events)**.** *If* $\Pr\{E_1\} \neq 0$*, then:*

$$\Pr\{E_1 \cap E_2\} = \Pr\{E_1\} \cdot \Pr\{E_2 \mid E_1\}$$

Multiplying edge probabilities in a tree diagram amounts to evaluating the right side of this equation. For example:

$$\begin{aligned} &\Pr\{\text{win first game} \cap \text{win second game}\} \\ &\qquad = \Pr\{\text{win first game}\} \cdot \Pr\{\text{win second game} \mid \text{win first game}\} \\ &\qquad = \frac{1}{2} \cdot \frac{2}{3} \end{aligned}$$

So the Product Rule is the formal justification for multiplying edge probabilities to get outcome probabilities! Of course to justify multiplying edge probabilities along longer paths, we need a Product Rule for n events. The pattern of the n event rule should be apparent from

Rule (Product Rule for 3 Events)**.**

$$\Pr\{E_1 \cap E_2 \cap E_3\} = \Pr\{E_1\} \cdot \Pr\{E_2 \mid E_1\} \cdot \Pr\{E_3 \mid E_2 \cap E_1\}$$

providing $\Pr\{E_1 \cap E_2\} \neq 0$*.*

This rule follows from the definition of conditional probability and the trivial identity

$$\Pr\{E_1 \cap E_2 \cap E_3\} = \Pr\{E_1\} \cdot \frac{\Pr\{E_2 \cap E_1\}}{\Pr\{E_1\}} \cdot \frac{\Pr\{E_3 \cap E_2 \cap E_1\}}{\Pr\{E_2 \cap E_1\}}$$

16.3.3 The Law of Total Probability

Breaking a probability calculation into cases simplifies many problems. The idea is to calculate the probability of an event A by splitting into two cases based on whether or not another event E occurs. That is, calculate the probability of $A \cap E$ and $A \cap \overline{E}$. By the Sum Rule, the sum of these probabilities equals $\Pr\{A\}$. Expressing the intersection probabilities as conditional probabilities yields

Rule (Total Probability)**.**

$$\Pr\{A\} = \Pr\{A \mid E\} \cdot \Pr\{E\} + \Pr\left\{A \mid \overline{E}\right\} \cdot \Pr\left\{\overline{E}\right\}.$$

For example, suppose we conduct the following experiment. First, we flip a coin. If heads comes up, then we roll one die and take the result. If tails comes up, then we roll two dice and take the sum of the two results. What is the probability that this process yields a 2? Let E be the event that the coin comes up heads, and let A be the event that we get a 2 overall. Assuming that the coin is fair, $\Pr\{E\} = \Pr\{\overline{E}\} = 1/2$. There are now two cases. If we flip heads, then we roll a 2 on a single die with probabilty $\Pr\{A \mid E\} = 1/6$. On the other hand, if we flip tails, then we get a sum of 2 on two dice with probability $\Pr\{A \mid \overline{E}\} = 1/36$. Therefore, the probability that the whole process yields a 2 is

$$\Pr\{A\} = \frac{1}{2} \cdot \frac{1}{6} + \frac{1}{2} \cdot \frac{1}{36} = \frac{7}{72}.$$

There is also a form of the rule to handle more than two cases.

Rule (Multicase Total Probability). *If $E_1, \ldots, E_n$ are pairwise disjoint events whose union is the whole sample space, then:*

$$\Pr\{A\} = \sum_{i=1}^{n} \Pr\{A \mid E_i\} \cdot \Pr\{E_i\}.$$

16.3.4 Medical Testing

There is an unpleasant condition called *BO* suffered by 10% of the population. There are no prior symptoms; victims just suddenly start to stink. Fortunately, there is a test for latent *BO* before things start to smell. The test is not perfect, however:

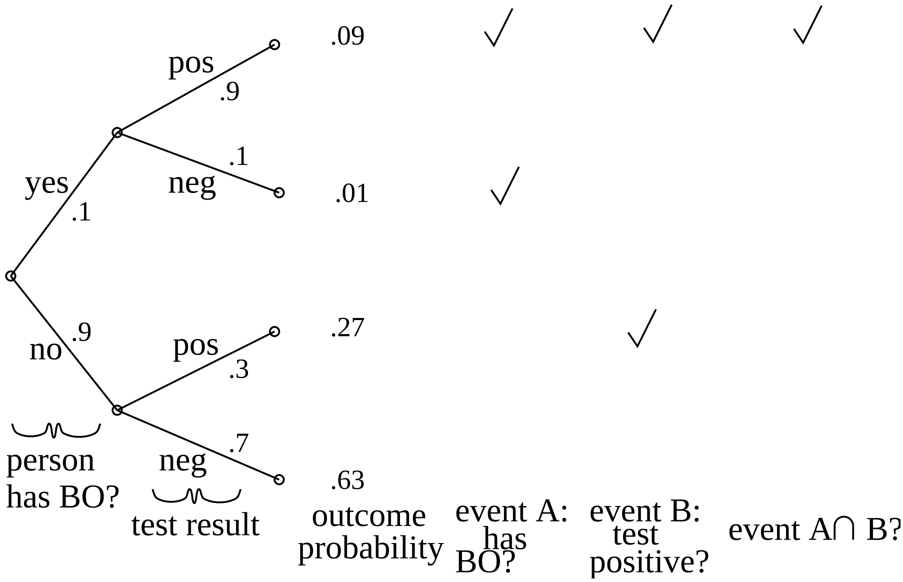
- If you have the condition, there is a 10% chance that the test will say you do not. (These are called "false negatives".)

- If you do not have the condition, there is a 30% chance that the test will say you do. (These are "false positives".)

Suppose a random person is tested for latent *BO*. If the test is positive, then what is the probability that the person has the condition?

Step 1: Find the Sample Space

The sample space is found with the tree diagram below.



Step 2: Define Events of Interest

Let A be the event that the person has *BO*. Let B be the event that the test was positive. The outcomes in each event are marked in the tree diagram. We want to find $\Pr\{A \mid B\}$, the probability that a person has *BO*, given that the test was positive.

Step 3: Find Outcome Probabilities

First, we assign probabilities to edges. These probabilities are drawn directly from the problem statement. By the Product Rule, the probability of an outcome is the product of the probabilities on the corresponding root-to-leaf path. All probabilities are shown in the figure.

Step 4: Compute Event Probabilities

p

$$\begin{aligned}\Pr\{A \mid B\} &= \frac{\Pr\{A \cap B\}}{\Pr\{B\}} \\ &= \frac{0.09}{0.09 + 0.27} \\ &= \frac{1}{4}\end{aligned}$$

If you test positive, then there is only a 25% chance that you have the condition!

This answer is initially surprising, but makes sense on reflection. There are two ways you could test positive. First, it could be that you are sick and the test is correct. Second, it could be that you are healthy and the test is incorrect. The problem is that almost everyone is healthy; therefore, most of the positive results arise from incorrect tests of healthy people!

We can also compute the probability that the test is correct for a random person. This event consists of two outcomes. The person could be sick and the test positive (probability 0.09), or the person could be healthy and the test negative (probability 0.63). Therefore, the test is correct with probability $0.09 + 0.63 = 0.72$. This is a relief; the test is correct almost three-quarters of the time.

But wait! There is a simple way to make the test correct 90% of the time: always return a negative result! This "test" gives the right answer for all healthy people and the wrong answer only for the 10% that actually have the condition. The best strategy is to completely ignore the test result!

There is a similar paradox in weather forecasting. During winter, almost all days in Boston are wet and overcast. Predicting miserable weather every day may be more accurate than really trying to get it right!

16.3.5 Conditional Identities

The probability rules above extend to probabilities conditioned on the same event. For example, the Inclusion-Exclusion formula for two sets holds when all probabilities are conditioned on an event C:

$$\Pr\{A \cup B \mid C\} = \Pr\{A \mid C\} + \Pr\{B \mid C\} - \Pr\{A \cap B \mid C\}.$$

This follows from the fact that if $\Pr\{C\} \neq 0$ and we define

$$\Pr_C\{A\} ::= \Pr\{A \mid C\}$$

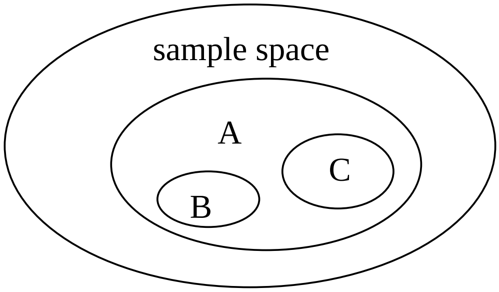
then $\Pr_C\{\}$ satisfies the definition of being probability function.

It is important not to mix up events before and after the conditioning bar. For example, the following is *not* a valid identity:

False Claim.

$$\Pr\{A \mid B \cup C\} = \Pr\{A \mid B\} + \Pr\{A \mid C\} - \Pr\{A \mid B \cap C\}. \tag{16.3}$$

A counterexample is shown below. In this case, $\Pr\{A \mid B\} = 1$, $\Pr\{A \mid C\} = 1$, and $\Pr\{A \mid B \cup C\} = 1$. However, since $1 \neq 1 + 1$, the equation above does not hold.



So you're convinced that this equation is false in general, right? Let's see if you *really* believe that.

16.3.6 Discrimination Lawsuit

Several years ago there was a sex discrimination lawsuit against Berkeley. A female professor was denied tenure, allegedly because she was a woman. She argued that in every one of Berkeley's 22 departments, the percentage of male applicants accepted was greater than the percentage of female applicants accepted. This sounds very suspicious!

However, Berkeley's lawyers argued that across the whole university the percentage of male tenure applicants accepted was actually *lower* than the percentage of female applicants accepted. This suggests that if there was any sex discrimination, then it was against men! Surely, at least one party in the dispute must be lying.

Let's simplify the problem and express both arguments in terms of conditional probabilities. Suppose that there are only two departments, EE and CS, and consider the experiment where we pick a random applicant. Define the following events:

- Let A be the event that the applicant is accepted.
- Let F_{EE} the event that the applicant is a female applying to EE.
- Let F_{CS} the event that the applicant is a female applying to CS.
- Let M_{EE} the event that the applicant is a male applying to EE.
- Let M_{CS} the event that the applicant is a male applying to CS.

Assume that all applicants are either male or female, and that no applicant applied to both departments. That is, the events F_{EE}, F_{CS}, M_{EE}, and M_{CS} are all disjoint.

In these terms, the plaintiff is make the following argument:

$$\Pr\{A \mid F_{EE}\} < \Pr\{A \mid M_{EE}\}$$
$$\Pr\{A \mid F_{CS}\} < \Pr\{A \mid M_{CS}\}$$

That is, in both departments, the probability that a woman is accepted for tenure is less than the probability that a man is accepted. The university retorts that overall a woman applicant is *more* likely to be accepted than a man:

$$\Pr\{A \mid F_{EE} \cup F_{CS}\} > \Pr\{A \mid M_{EE} \cup M_{CS}\}$$

It is easy to believe that these two positions are contradictory. In fact, we might even try to prove this by adding the plaintiff's two inequalities and then arguing as follows:

$$\begin{aligned} & \Pr\{A \mid F_{EE}\} + \Pr\{A \mid F_{CS}\} < \Pr\{A \mid M_{EE}\} + \Pr\{A \mid M_{CS}\} \\ \Rightarrow \quad & \Pr\{A \mid F_{EE} \cup F_{CS}\} < \Pr\{A \mid M_{EE} \cup M_{CS}\} \end{aligned}$$

The second line exactly contradicts the university's position! But there is a big problem with this argument; the second inequality follows from the first only if we accept the false identity (16.3). This argument is bogus! Maybe the two parties do not hold contradictory positions after all!

In fact, the table below shows a set of application statistics for which the assertions of both the plaintiff and the university hold:

CS	0 females accepted, 1 applied	0%
	50 males accepted, 100 applied	50%
EE	70 females accepted, 100 applied	70%
	1 male accepted, 1 applied	100%
Overall	70 females accepted, 101 applied	$\approx 70\%$
	51 males accepted, 101 applied	$\approx 51\%$

In this case, a higher percentage of males were accepted in both departments, but overall a higher percentage of females were accepted! Bizarre!

16.3.7 *A Posteriori* Probabilities

Suppose that we turn the hockey question around: what is the probability that the Halting Problem won their first game, given that they won the series?

This seems like an absurd question! After all, if the Halting Problem won the series, then the winner of the first game has already been determined. Therefore, who won the first game is a question of fact, not a question of probability. However, our mathematical theory of probability contains no notion of one event preceding another— there is no notion of time at all. Therefore, from a mathematical perspective, this is a perfectly valid question. And this is also a meaningful question from a practical perspective. Suppose that you're told that the Halting Problem won the series, but not told the results of individual games. Then, from your perspective, it makes perfect sense to wonder how likely it is that The Halting Problem won the first game.

A conditional probability $\Pr\{B \mid A\}$ is called *a posteriori* if event B precedes event A in time. Here are some other examples of a posteriori probabilities:

- The probability it was cloudy this morning, given that it rained in the afternoon.
- The probability that I was initially dealt two queens in Texas No Limit Hold 'Em poker, given that I eventually got four-of-a-kind.

Mathematically, a posteriori probabilities are *no different* from ordinary probabilities; the distinction is only at a higher, philosophical level. Our only reason for drawing attention to them is to say, "Don't let them rattle you."

Let's return to the original problem. The probability that the Halting Problem won their first game, given that they won the series is $\Pr\{B \mid A\}$. We can compute this using the definition of conditional probability and our earlier tree diagram:

$$\begin{aligned}
\Pr\{B \mid A\} &= \frac{\Pr\{B \cap A\}}{\Pr\{A\}} \\
&= \frac{1/3 + 1/18}{1/3 + 1/18 + 1/9} \\
&= \frac{7}{9}
\end{aligned}$$

This answer is suspicious! In the preceding section, we showed that $\Pr\{A \mid B\}$ was also $7/9$. Could it be true that $\Pr\{A \mid B\} = \Pr\{B \mid A\}$ in general? Some reflection suggests this is unlikely. For example, the probability that I feel uneasy, given that I was abducted by aliens, is pretty large. But the probability that I was abducted by aliens, given that I feel uneasy, is rather small.

Let's work out the general conditions under which $\Pr\{A \mid B\} = \Pr\{B \mid A\}$. By the definition of conditional probability, this equation holds if an only if:

$$\frac{\Pr\{A \cap B\}}{\Pr\{B\}} = \frac{\Pr\{A \cap B\}}{\Pr\{A\}}$$

This equation, in turn, holds only if the denominators are equal or the numerator is 0:

$$\Pr\{B\} = \Pr\{A\} \qquad \text{or} \qquad \Pr\{A \cap B\} = 0$$

The former condition holds in the hockey example; the probability that the Halting Problem wins the series (event A) is equal to the probability that it wins the first game (event B). In fact, both probabilities are $1/2$.

Such pairs of probabilities are related by Bayes' Rule:

Theorem 16.3.2 (Bayes' Rule)**.** *If* $\Pr\{A\}$ *and* $\Pr\{B\}$ *are nonzero, then:*

$$\frac{\Pr\{A \mid B\} \cdot \Pr\{B\}}{\Pr\{A\}} = \Pr\{B \mid A\} \tag{16.4}$$

Proof. When $\Pr\{A\}$ and $\Pr\{B\}$ are nonzero, we have

$$\Pr\{A \mid B\} \cdot \Pr\{B\} = \Pr\{A \cap B\} = \Pr\{B \mid A\} \cdot \Pr\{A\}$$

by definition of conditional probability. Dividing by $\Pr\{A\}$ gives (16.4).

■

In the hockey problem, the probability that the Halting Problem wins the first game is $1/2$ and so is the probability that the Halting Problem wins the series. Therefore, $\Pr\{A\} = \Pr\{B\} = 1/2$. This, together with Bayes' Rule, explains why $\Pr\{A \mid B\}$ and $\Pr\{B \mid A\}$ turned out to be equal in the hockey example.

16.3.8 Problems

Practice Problems

Problem 16.9.
Dirty Harry places two bullets in the six-shell cylinder of his revolver. He gives the cylinder a random spin and says "Feeling lucky?" as he holds the gun against your heart.

(a) What is the probability that you will get shot if he pulls the trigger?

(b) Suppose he pulls the trigger and you don't get shot. What is the probability that you will get shot if he pulls the trigger a second time?

(c) Suppose you noticed that he placed the two shells next to each other in the cylinder. How does this change the answers to the previous two questions?

Class Problems

Problem 16.10.
There are two decks of cards. One is complete, but the other is missing the ace of spades. Suppose you pick one of the two decks with equal probability and then select a card from that deck uniformly at random. What is the probability that you picked the complete deck, given that you selected the eight of hearts? Use the four-step method and a tree diagram.

Problem 16.11.
There are three prisoners in a maximum-security prison for fictional villains: the Evil Wizard Voldemort, the Dark Lord Sauron, and Little Bunny Foo-Foo. The parole board has declared that it will release two of the three, chosen uniformly at random, but has not yet released their names. Naturally, Sauron figures that he will be released to his home in Mordor, where the shadows lie, with probability $2/3$.

A guard offers to tell Sauron the name of one of the other prisoners who will be released (either Voldemort or Foo-Foo). Sauron knows the guard to be a truthful fellow. However, Sauron declines this offer. He reasons that if the guard says, for example, "Little Bunny Foo-Foo will be released", then his own probability of release will drop to $1/2$. This is because he will then know that either he or Voldemort will also be released, and these two events are equally likely.

Using a tree diagram and the four-step method, either prove that the Dark Lord Sauron has reasoned correctly or prove that he is wrong. Assume that if the guard has a choice of naming either Voldemort or Foo-Foo (because both are to be released), then he names one of the two uniformly at random.

Homework Problems

Problem 16.12.
There is a course —not 6.042, naturally —in which 10% of the assigned problems contain errors. If you ask a TA whether a problem has an error, then he or she will answer correctly 80% of the time. This 80% accuracy holds regardless of whether or not a problem has an error. Likewise when you ask a lecturer, but with only 75% accuracy.

We formulate this as an experiment of choosing one problem randomly and asking a particular TA and Lecturer about it. Define the following events:

$$\begin{aligned} E &::= \text{"the problem has an error,"} \\ T &::= \text{"the TA says the problem has an error,"} \\ L &::= \text{"the lecturer says the problem has an error."} \end{aligned}$$

(a) Translate the description above into a precise set of equations involving conditional probabilities among the events E, T, and L

(b) Suppose you have doubts about a problem and ask a TA about it, and she tells you that the problem is correct. To double-check, you ask a lecturer, who says that the problem has an error. Assuming that *the correctness of the lecturers' answer and the TA's answer are independent of each other, regardless of whether there is an error*[2], what is the probability that there is an error in the problem?

(c) Is the event that "the TA says that there is an error", independent of the event that "the lecturer says that there is an error"?

Problem 16.13. (a) Suppose you repeatedly flip a fair coin until you see the sequence `HHT` or the sequence `TTH`. What is the probability you will see `HHT` first? *Hint:* Symmetry between Heads and Tails.

(b) What is the probability you see the sequence `HTT` before you see the sequence `HHT`? *Hint:* Try to find the probability that `HHT` comes before `HTT` conditioning on whether you first toss an `H` or a `T`. The answer is not $1/2$.

Problem 16.14.
A 52-card deck is thoroughly shuffled and you are dealt a hand of 13 cards.

(a) If you have one ace, what is the probability that you have a second ace?

[2]This assumption is questionable: by and large, we would expect the lecturer and the TA's to spot the same glaring errors and to be fooled by the same subtle ones.

(b) If you have the ace of spades, what is the probability that you have a second ace?

Remarkably, the two answers are different. This problem will test your counting ability!

Problem 16.15.
You are organizing a neighborhood census and instruct your census takers to knock on doors and note the sex of any child that answers the knock. Assume that there are two children in a household and that girls and boys are equally likely to be children and to open the door.

A sample space for this experiment has outcomes that are triples whose first element is either `B` or `G` for the sex of the elder child, likewise for the second element and the sex of the younger child, and whose third coordinate is `E` or `Y` indicating whether the *e*lder child or *y*ounger child opened the door. For example, $(\mathtt{B}, \mathtt{G}, \mathtt{Y})$ is the outcome that the elder child is a boy, the younger child is a girl, and the girl opened the door.

(a) Let T be the event that the household has two girls, and O be the event that a girl opened the door. List the outcomes in T and O.

(b) What is the probability $\Pr\{T \mid O\}$, that both children are girls, given that a girl opened the door?

(c) Where is the mistake in the following argument?

> If a girl opens the door, then we know that there is at least one girl in the household. The probability that there is at least one girl is
>
> $$1 - \Pr\{\text{both children are boys}\} = 1 - (1/2 \times 1/2) = 3/4. \tag{16.5}$$
>
> So,
>
> $$\begin{aligned} &\Pr\{T \mid \text{there is at least one girl in the household}\} && (16.6)\\ &= \frac{\Pr\{T \cap \text{there is at least one girl in the household}\}}{\Pr\{\text{there is at least one girl in the household}\}} && (16.7)\\ &= \frac{\Pr\{T\}}{\Pr\{\text{there is at least one girl in the household}\}} && (16.8)\\ &= (1/4)/(3/4) = 1/3. && (16.9)\end{aligned}$$
>
> Therefore, given that a girl opened the door, the probability that there are two girls in the household is $1/3$.

16.4 Independence

Suppose that we flip two fair coins simultaneously on opposite sides of a room. Intuitively, the way one coin lands does not affect the way the other coin lands.

The mathematical concept that captures this intuition is called *independence*:

Definition. Events A and B are independent if and only if:

$$\Pr\{A \cap B\} = \Pr\{A\} \cdot \Pr\{B\}$$

Generally, independence is something you *assume* in modeling a phenomenon—or wish you could realistically assume. Many useful probability formulas only hold if certain events are independent, so a dash of independence can greatly simplify the analysis of a system.

16.4.1 Examples

Let's return to the experiment of flipping two fair coins. Let A be the event that the first coin comes up heads, and let B be the event that the second coin is heads. If we assume that A and B are independent, then the probability that both coins come up heads is:

$$\begin{aligned}\Pr\{A \cap B\} &= \Pr\{A\} \cdot \Pr\{B\} \\ &= \frac{1}{2} \cdot \frac{1}{2} \\ &= \frac{1}{4}\end{aligned}$$

On the other hand, let C be the event that tomorrow is cloudy and R be the event that tomorrow is rainy. Perhaps $\Pr\{C\} = 1/5$ and $\Pr\{R\} = 1/10$ around here. If these events were independent, then we could conclude that the probability of a rainy, cloudy day was quite small:

$$\begin{aligned}\Pr\{R \cap C\} &= \Pr\{R\} \cdot \Pr\{C\} \\ &= \frac{1}{5} \cdot \frac{1}{10} \\ &= \frac{1}{50}\end{aligned}$$

Unfortunately, these events are definitely not independent; in particular, every rainy day is cloudy. Thus, the probability of a rainy, cloudy day is actually $1/10$.

16.4.2 Working with Independence

There is another way to think about independence that you may find more intuitive. According to the definition, events A and B are independent if and only if $\Pr\{A \cap B\} = \Pr\{A\} \cdot \Pr\{B\}$. This equation holds even if $\Pr\{B\} = 0$, but assuming it is not, we can divide both sides by $\Pr\{B\}$ and use the definition of conditional probability to obtain an alternative formulation of independence:

Proposition. *If $\Pr\{B\} \neq 0$, then events A and B are independent if and only if*

$$\Pr\{A \mid B\} = \Pr\{A\}. \tag{16.10}$$

Equation (16.10) says that events A and B are independent if the probability of A is unaffected by the fact that B happens. In these terms, the two coin tosses of the previous section were independent, because the probability that one coin comes up heads is unaffected by the fact that the other came up heads. Turning to our other example, the probability of clouds in the sky is strongly affected by the fact that it is raining. So, as we noted before, these events are not independent.

Warning: Students sometimes get the idea that disjoint events are independent. The *opposite* is true: if $A \cap B = \emptyset$, then knowing that A happens means you know that B does not happen. So disjoint events are *never* independent —unless one of them has probability zero.

16.4.3 Mutual Independence

We have defined what it means for two events to be independent. But how can we talk about independence when there are more than two events? For example, how can we say that the orientations of n coins are all independent of one another?

Events $E_1, \ldots, E_n$ are *mutually independent* if and only if *for every subset* of the events, the probability of the intersection is the product of the probabilities. In other words, all of the following equations must hold:

$$
\begin{aligned}
\Pr\{E_i \cap E_j\} &= \Pr\{E_i\} \cdot \Pr\{E_j\} && \text{for all distinct } i, j \\
\Pr\{E_i \cap E_j \cap E_k\} &= \Pr\{E_i\} \cdot \Pr\{E_j\} \cdot \Pr\{E_k\} && \text{for all distinct } i, j, k \\
\Pr\{E_i \cap E_j \cap E_k \cap E_l\} &= \Pr\{E_i\} \cdot \Pr\{E_j\} \cdot \Pr\{E_k\} \cdot \Pr\{E_l\} && \text{for all distinct } i, j, k, l \\
&\cdots \\
\Pr\{E_1 \cap \cdots \cap E_n\} &= \Pr\{E_1\} \cdots \Pr\{E_n\}
\end{aligned}
$$

As an example, if we toss 100 fair coins and let E_i be the event that the ith coin lands heads, then we might reasonably assume that $E_1, \ldots, E_{100}$ are mutually independent.

16.4.4 Pairwise Independence

The definition of mutual independence seems awfully complicated— there are so many conditions! Here's an example that illustrates the subtlety of independence when more than two events are involved and the need for all those conditions. Suppose that we flip three fair, mutually-independent coins. Define the following events:

- A_1 is the event that coin 1 matches coin 2.
- A_2 is the event that coin 2 matches coin 3.
- A_3 is the event that coin 3 matches coin 1.

Are A_1, A_2, A_3 mutually independent?

The sample space for this experiment is:

$$\{HHH,\ HHT,\ HTH,\ HTT,\ THH,\ THT,\ TTH,\ TTT\}$$

Every outcome has probability $(1/2)^3 = 1/8$ by our assumption that the coins are mutually independent.

To see if events A_1, A_2, and A_3 are mutually independent, we must check a sequence of equalities. It will be helpful first to compute the probability of each event A_i:

$$\begin{aligned}
\Pr\{A_1\} &= \Pr\{HHH\} + \Pr\{HHT\} + \Pr\{TTH\} + \Pr\{TTT\} \\
&= \frac{1}{8} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8} \\
&= \frac{1}{2}
\end{aligned}$$

By symmetry, $\Pr\{A_2\} = \Pr\{A_3\} = 1/2$ as well. Now we can begin checking all the equalities required for mutual independence.

$$\begin{aligned}
\Pr\{A_1 \cap A_2\} &= \Pr\{HHH\} + \Pr\{TTT\} \\
&= \frac{1}{8} + \frac{1}{8} \\
&= \frac{1}{4} \\
&= \frac{1}{2} \cdot \frac{1}{2} \\
&= \Pr\{A_1\} \Pr\{A_2\}
\end{aligned}$$

By symmetry, $\Pr\{A_1 \cap A_3\} = \Pr\{A_1\} \cdot \Pr\{A_3\}$ and $\Pr\{A_2 \cap A_3\} = \Pr\{A_2\} \cdot \Pr\{A_3\}$ must hold also. Finally, we must check one last condition:

$$\begin{aligned}
\Pr\{A_1 \cap A_2 \cap A_3\} &= \Pr\{HHH\} + \Pr\{TTT\} \\
&= \frac{1}{8} + \frac{1}{8} \\
&= \frac{1}{4} \\
&\neq \Pr\{A_1\} \Pr\{A_2\} \Pr\{A_3\} = \frac{1}{8}
\end{aligned}$$

The three events A_1, A_2, and A_3 are not mutually independent even though any two of them are independent! This not-quite mutual independence seems weird at first, but it happens. It even generalizes:

Definition 16.4.1. A set $A_0, A_1, \ldots$ of events is *k-way independent* iff every set of k of these events is mutually independent. The set is *pairwise independent* iff it is 2-way independent.

So the sets A_1, A_2, A_3 above are pairwise independent, but not mutually independent. Pairwise independence is a much weaker property than mutual independence, but it's all that's needed to justify a standard approach to making probabilistic estimates that will come up later.

16.4.5 Problems

Class Problems

Problem 16.16.
Suppose that you flip three fair, mutually independent coins. Define the following events:

- Let A be the event that *the first* coin is heads.
- Let B be the event that *the second* coin is heads.
- Let C be the event that *the third* coin is heads.
- Let D be the event that *an even number of* coins are heads.

(a) Use the four step method to determine the probability space for this experiment and the probability of each of A, B, C, D.

(b) Show that these events are not mutually independent.

(c) Show that they are 3-way independent.

16.5 The Birthday Principle

There are 85 students in a class. What is the probability that some birthday is shared by two people? Comparing 85 students to the 365 possible birthdays, you might guess the probability lies somewhere around $1/4$ —but you'd be wrong: the probability that there will be two people in the class with matching birthdays is actually more than 0.9999.

To work this out, we'll assume that the probability that a randomly chosen student has a given birthday is $1/d$, where $d = 365$ in this case. We'll also assume that a class is composed of n randomly and independently selected students, with $n = 85$ in this case. These randomness assumptions are not really true, since more babies are born at certain times of year, and students' class selections are typically not independent of each other, but simplifying in this way gives us a start on analyzing the problem. More importantly, these assumptions are justifiable in important computer science applications of birthday matching. For example, the birthday matching is a good model for collisions between items randomly inserted into a hash table. So we won't worry about things like Spring procreation preferences that make January birthdays more common, or about twins' preferences to take classes together (or not).

Selecting a sequence of n students for a class yields a sequence of n birthdays. Under the assumptions above, the d^n possible birthday sequences are equally likely outcomes. Let's examine the consequences of this probability model by focussing on the ith and jth elements in a birthday sequence, where $1 \leq i \neq j \leq n$. It makes for a better story if we refer to the ith birthday as "Alice's" and the jth as "Bob's."

Now since Bob's birthday is assumed to be independent of Alice's, it follows that whichever of the d birthdays Alice's happens to be, the probability that Bob has the same birthday $1/d$. Next, If we look at two other birthdays —call them "Carol's" and "Don's" —then whether Alice and Bob have matching birthdays has nothing to do with whether Carol and Don have matching birthdays. That is, the event that Alice and Bob have matching birthdays is independent of the event that Carol and Don have matching birthdays. In fact, for any set of non-overlapping couples, the events that a couple has matching birthdays are mutually independent.

In fact, it's pretty clear that the probability that Alice and Bob have matching birthdays remains $1/d$ whether or not Carol and *Alice* have matching birthdays. That is, the event that Alice and Bob match is also independent of Alice and Carol matching. In short, the set of all events in which a couple has macthing birthdays is *pairwise* independent, despite the overlapping couples. This will be important in Chapter 19 because pairwise independence will be enough to justify some conclusions about the expected number of matches. However, it's obvious that these matching birthday events are *not* mutually independent, not even 3-way independent: if Alice and Bob match and also Alice and Carol match, then Bob and Carol will match.

We could justify all these assertions of independence routinely using the four step method, but it's pretty boring, and we'll skip it.

It turns out that as long as the number of students is noticeably smaller than the number of possible birthdays, we can get a pretty good estimate of the birthday matching probabilities by *pretending* that the matching events are mutually independent. (An intuitive justification for this is that with only a small number of matching pairs, it's likely that none of the pairs overlap.) Then the probability of *no* matching birthdays would be the same as rth power of the probability that a couple does *not* have matching birthdays, where $r ::= \binom{n}{2}$ is the number of couples. That is, the probability of no matching birthdays would be

$$(1 - 1/d)^{\binom{n}{2}}. \tag{16.11}$$

Using the fact that $e^x > 1 + x$ for all x,[3] we would conclude that the probability of no matching birthdays is at most

$$e^{-\frac{\binom{n}{2}}{d}}. \tag{16.12}$$

[3]This approximation is obtained by truncating the Taylor series $e^{-x} = 1 - x + x^2/2! - x^3/3! + \cdots$. The approximation $e^{-x} \approx 1 - x$ is pretty accurate when x is small.

The matching birthday problem fits in here so far as a nice example illustrating pairwise and mutual independence. But it's actually not hard to justify the bound (16.12) without any pretence or any explicit consideration of independence. Namely, there are $d(d-1)(d-2)\cdots(d-(n-1))$ length n sequences of distinct birthdays. So the probability that everyone has a different birthday is:

$$
\begin{aligned}
&\frac{d(d-1)(d-2)\cdots(d-(n-1))}{d^n} \\
&= \frac{d}{d}\cdot\frac{d-1}{d}\cdot\frac{d-2}{d}\cdots\frac{d-(n-1)}{d} \\
&= \left(1-\frac{0}{d}\right)\left(1-\frac{1}{d}\right)\left(1-\frac{2}{d}\right)\cdots\left(1-\frac{n-1}{d}\right) \\
&< e^0\cdot e^{-1/d}\cdot e^{-2/d}\cdots e^{-(n-1)/d} && (\text{since } 1+x<e^x) \\
&= e^{-\left(\sum_{i=1}^{n-1} i/d\right)} \\
&= e^{-(n(n-1)/2d)} \\
&= \text{the bound (16.12).}
\end{aligned}
$$

For $n = 85$ and $d = 365$, (16.12) is less than $1/17,000$, which means the probability of having some pair of matching birthdays actually is more than $1-1/17,000 > 0.9999$. So it would be pretty astonishing if there were no pair of students in the class with matching birthdays.

For $d \le n^2/2$, the probability of no match turns out to be asymptotically equal to the upper bound (16.12). For $d = n^2/2$ in particular, the probability of no match is asymptotically equal to $1/e$. This leads to a rule of thumb which is useful in many contexts in computer science:

The Birthday Principle

If there are d days in a year and $\sqrt{2d}$ people in a room, then the probability that two share a birthday is about $1-1/e \approx 0.632$.

For example, the Birthday Principle says that if you have $\sqrt{2\cdot 365} \approx 27$ people in a room, then the probability that two share a birthday is about 0.632. The actual probability is about 0.626, so the approximation is quite good.

Among other applications, the Birthday Principle famously comes into play as the basis of "birthday attacks" that crack certain cryptographic systems.

Chapter 17

Random Processes

Random Walks are used to model situations in which an object moves in a sequence of steps in randomly chosen directions. For example in Physics, three-dimensional random walks are used to model Brownian motion and gas diffusion. In this chapter we'll examine two examples of random walks. First, we'll model gambling as a simple 1-dimensional random walk —a walk along a straight line. Then we'll explain how the Google search engine used random walks through the graph of world-wide web links to determine the relative importance of websites.

17.1 Gamblers' Ruin

a Suppose a gambler starts with an initial stake of n dollars and makes a sequence of \$1 bets. If he wins an individual bet, he gets his money back plus another \$1. If he loses, he loses the \$1.

We can model this scenario as a random walk between integer points on the reall line. The position on the line at any time corresponds to the gambler's cash-on-hand or *capital*. Walking one step to the right (left) corresponds to winning (losing) a \$1 bet and thereby increasing (decreasing) his capital by \$1. The gambler plays until either he is bankrupt or increases his capital to a target amount of T dollars. If he reaches his target, then he is called an overall *winner*, and his *profit*, m, will be $T - n$ dollars. If his capital reaches zero dollars before reaching his target, then we say that he is "ruined" or *goes broke*. We'll assume that the gambler has the same probability, p, of winning each individual \$1 bet and that the bets are mutually independent. We'd like to find the probability that the gambler wins.

The gambler's situation as he proceeds with his \$1 bets is illustrated in Figure 17.1. The random walk has boundaries at 0 and T. If the random walk ever reaches either of these boundary values, then it terminates.

In a *fair game*, the gambler is equally likely to win or lose each bet, that is $p = 1/2$. The corresponding random walk is called *unbiased*. The gambler is more likely to win if $p > 1/2$ and less likely to win if $p < 1/2$; these random walks are called

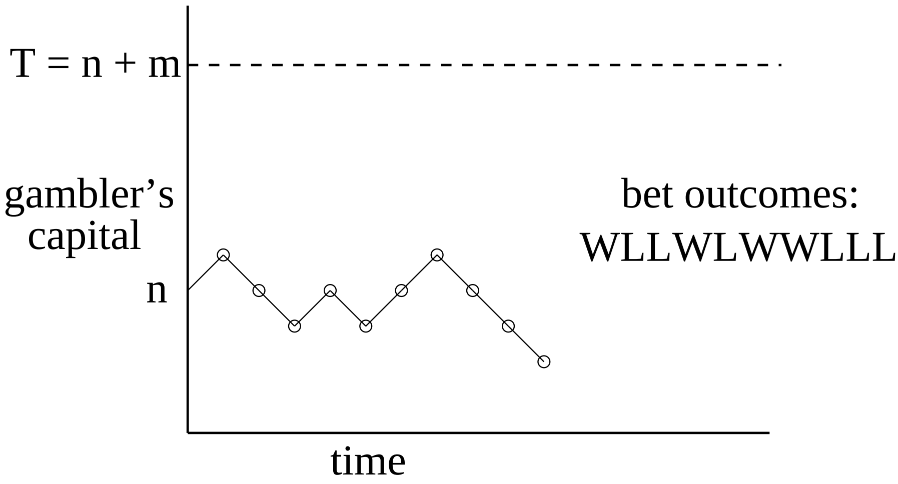


Figure 17.1: *This is a graph of the gambler's capital versus time for one possible sequence of bet outcomes. At each time step, the graph goes up with probability p and down with probability $1 - p$. The gambler continues betting until the graph reaches either 0 or T.*

biased. We want to determine the probability that the walk terminates at boundary T, namely, the probability that the gambler is a winner. We'll do this by showing that the probability satisfies a simple linear recurrence and solving the recurrence, but before we derive the probability, let's just look at what it turns out to be.

Let's begin by supposing the coin is fair, the gambler starts with 100 dollars, and he wants to double his money. That is, he plays until he goes broke or reaches a target of 200 dollars. Since he starts equidistant from his target and bankruptcy, it's clear by symmetry that his probability of winning in this case is 1/2.

We'll show below that starting with n dollars and aiming for a target of $T \geq n$ dollars, the probability the gambler reaches his target before going broke is n/T. For example, suppose he want to win the same \$100, but instead starts out with \$500. Now his chances are pretty good: the probability of his making the 100 dollars is $5/6$. And if he started with one million dollars still aiming to win \$100 dollars he almost certain to win: the probability is $1M/(1M + 100) > .9999$.

So in the fair game, the larger the initial stake relative to the target, the higher the probability the gambler will win, which makes some intuitive sense. But note that although the gambler now wins nearly all the time, the game is still fair. When he wins, he only wins \$100; when he loses, he loses big: \$1M. So the gambler's average win is actually zero dollars.

Now suppose instead that the gambler chooses to play roulette in an American casino, always betting \$1 on red. A roulette wheel has 18 black numbers, 18 red numbers, and 2 green numbers, designed so that each number is equally likely to appear. So this game is slightly biased against the gambler: the probability of winning a single bet is $p = 18/38 \approx 0.47$. It's the two green numbers that slightly bias the bets and give the casino an edge. Still, the bets are almost fair, and you might expect that starting with \$500, the gambler has a reasonable chance of winning \$100 —the 5/6 probability of winning in the unbiased game surely gets reduced, but perhaps not too drastically.

Not so! The gambler's odds of winning \$100 making one dollar bets against the "slightly" unfair roulette wheel are less than 1 in 37,000. If that seems surprising, listen to this: *no matter how much money* the gambler has to start —\$5000, \$50,000, $\$5 \cdot 10^{12}$ —his odds are still less than 1 in 37,000 of winning a mere 100 dollars!

Moral: Don't play!

The theory of random walks is filled with such fascinating and counter-intuitive conclusions.

17.1.1 A Recurrence for the Probability of Winning

The probability the gambler wins is a function of his initial capital, n, his target, $T \geq n$, and the probability, p, that he wins an individual one dollar bet. Let's let p and T be fixed, and let w_n be the gambler's probabiliity of winning when his initial capital is n dollars. For example, w_0 is the probability that the gambler will win given that he starts off broke and w_T is the probability he will win if he starts off with his target amount, so clearly

$$w_0 = 0, \tag{17.1}$$
$$w_T = 1. \tag{17.2}$$

Otherwise, the gambler starts with n dollars, where $0 < n < T$. Consider the outcome of his first bet. The gambler wins the first bet with probability p. In this case, he is left with $n+1$ dollars and becomes a winner with probability w_{n+1}. On the other hand, he loses the first bet with probability $q ::= 1-p$. Now he is left with $n-1$ dollars and becomes a winner with probability w_{n-1}. By the Total Probability Rule, he wins with probability $w_n = pw_{n+1} + qw_{n-1}$. Solving for w_{n+1} we have

$$w_{n+1} = \frac{w_n}{p} - rw_{n-1} \tag{17.3}$$

where

$$r ::= \frac{q}{p}.$$

This recurrence holds only for $n+1 \leq T$, but there's no harm in using (17.3) to define w_{n+1} for all $n+1 > 1$. Now, letting

$$W(x) ::= w_0 + w_1 x + w_2 x^2 + \cdots$$

be the generating function for the w_n, we derive from (17.3) and (17.1) using our generating function methods that

$$xW(x) = \frac{w_1 x}{(1-x)(1-rx)}, \tag{17.4}$$

so if $p \neq q$, then using partial fractions we can calculate that

$$W(x) = \frac{w_1}{r-1}\left(\frac{1}{1-rx} - \frac{1}{1-x}\right),$$

which implies

$$w_n = w_1 \frac{r^n - 1}{r-1}. \tag{17.5}$$

Now we can use (17.5) to solve for w_1 by letting $n = T$ to get

$$w_1 = \frac{r-1}{r^T - 1}.$$

Plugging this value of w_1 into (17.5), we finally arrive at the solution:

$$w_n = \frac{r^n - 1}{r^T - 1}. \tag{17.6}$$

The expression (17.6) for the probability that the Gambler wins in the biased game is a little hard to interpret. There is a simpler upper bound which is nearly tight when the gambler's starting capital is large and the game is biased *against* the gambler. Then both the numerator and denominator in the quotient in (17.6) are positive, and the quotient is less than one. This implies that

$$w_n < \frac{r^n}{r^T} = r^{T-n},$$

which proves:

Corollary 17.1.1. *In the Gambler's Ruin game with probability $p < 1/2$ of winning each individual bet, with initial capital, n, and target, T,*

$$\Pr\{\textit{the gambler is a winner}\} < \left(\frac{p}{q}\right)^{T-n} \tag{17.7}$$

The amount $T - n$ is called the Gambler's *intended profit*. So the gambler gains his intended profit before going broke with probability at most p/q raised to the intended-profit power. Notice that this upper bound does not depend on the gambler's starting capital, but only on his intended profit. This has the amazing consequence we announced above: *no matter how much money he starts with*, if he makes \$1 bets on red in roulette aiming to win \$100, the probability that he wins is less than

$$\left(\frac{18/38}{20/38}\right)^{100} = \left(\frac{9}{10}\right)^{100} < \frac{1}{37,648}.$$

The bound (17.7) is exponential in the intended profit. So, for example, doubling his intended profit will square his probability of winning. In particular, the probability that the gambler's stake goes up 200 dollars before he goes broke playing roulette is at most

$$(9/10)^{200} = ((9/10)^{100})^2 = \left(\frac{1}{37,648}\right)^2,$$

which is about 1 in 70 billion.

The solution (17.6) only applies to biased walks, but the method above works just as well in getting a formula for the unbiased case (except that the partial fractions involve a repeated root). But it's simpler settle the fair case simply by taking the limit as r approaches 1 of (17.6). By L'Hopital's Rule this limit is n/T, as we claimed above.

17.1.2 Intuition

Why is the gambler so unlikely to make money when the game is slightly biased against him? Intuitively, there are two forces at work. First, the gambler's capital has random upward and downward *swings* due to runs of good and bad luck. Second, the gambler's capital will have a steady, downward *drift*, because the negative bias means an average loss of a few cents on each \$1 bet. The situation is shown in Figure 17.2.

Our intuition is that if the gambler starts with, say, a billion dollars, then he is sure to play for a very long time, so at some point there should be a lucky, upward swing that puts him \$100 ahead. The problem is that his capital is steadily drifting downward. If the gambler does not have a lucky, upward swing early on, then he is doomed. After his capital drifts downward a few hundred dollars, he needs a huge upward swing to save himself. And such a huge swing is extremely improbable. As a rule of thumb, *drift dominates swings* in the long term.

17.1.3 Problems

Class Problems

Problem 17.1.
A gambler is placing \$1 bets on the "1st dozen" in roulette. This bet wins when a number from one to twelve comes in, and then the gambler gets his \$1 back plus \$3 more. Recall that there are 38 numbers on the roulette wheel.

The gambler's initial stake in $\$n$ and his target is $\$T$. He will keep betting until he runs out of money ("goes broke") or reachs his target. Let w_n be the probability of the gambler winning, that is, reaching target $\$T$ before going broke.

(a) Write a linear recurrence for w_n; you need *not* solve the recurrence.

(b) Let e_n be the expected number of bets until the game ends. Write a linear recurrence for e_n; you need *not* solve the recurrence.

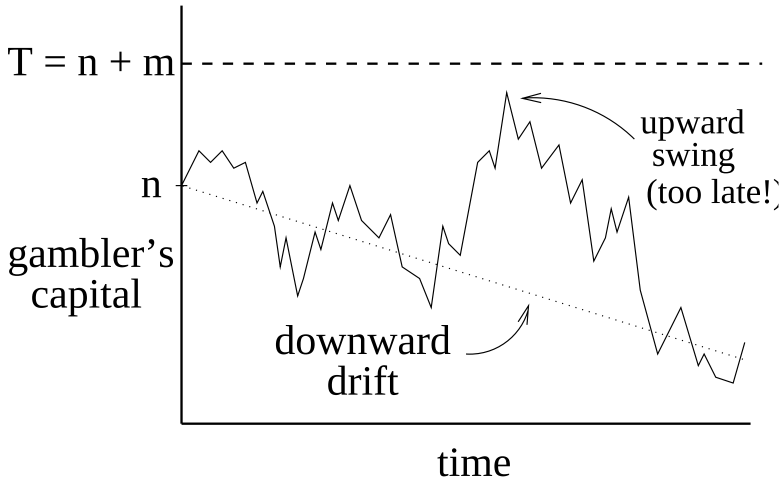


Figure 17.2: *In an unfair game, the gambler's capital swings randomly up and down, but steadily drifts downward. If the gambler does not have a winning swing early on, then his capital drifts downward, and later upward swings are insufficient to make him a winner.*

Homework Problems

Problem 17.2.
A drunken sailor wanders along main street, which conveniently consists of the points along the x axis with integral coordinates. In each step, the sailor moves one unit left or right along the x axis. A particular *path* taken by the sailor can be described by a sequence of "left" and "right" steps. For example, ⟨left,left,right⟩ describes the walk that goes left twice then goes right.

We model this scenario with a random walk graph whose vertices are the integers and with edges going in each direction between consecutive integers. All edges are labelled $1/2$.

The sailor begins his random walk at the origin. This is described by an initial distribution which labels the origin with probability 1 and all other vertices with probability 0. After one step, the sailor is equally likely to be at location 1 or -1, so the distribution after one step gives label $1/2$ to the vertices 1 and -1 and labels all other vertices with probability 0.

(a) Give the distributions after the 2nd, 3rd, and 4th step by filling in the table of probabilities below, where omitted entries are 0. For each row, write all the nonzero entries so they have the same denominator.

	location								
	-4	-3	-2	-1	0	1	2	3	4
initially					1				
after 1 step				1/2	0	1/2			
after 2 steps			?	?	?	?	?		
after 3 steps		?	?	?	?	?	?	?	
after 4 steps	?	?	?	?	?	?	?	?	?

(b)

1. What is the final location of a t-step path that moves right exactly i times?
2. How many different paths are there that end at that location?
3. What is the probability that the sailor ends at this location?

(c) Let L be the random variable giving the sailor's location after t steps, and let $B ::= (L+t)/2$. Use the answer to part (b) to show that B has an unbiased binomial density function.

(d) Again let L be the random variable giving the sailor's location after t steps, where t is even. Show that

$$\Pr\left\{|L| < \frac{\sqrt{t}}{2}\right\} < \frac{1}{2}.$$

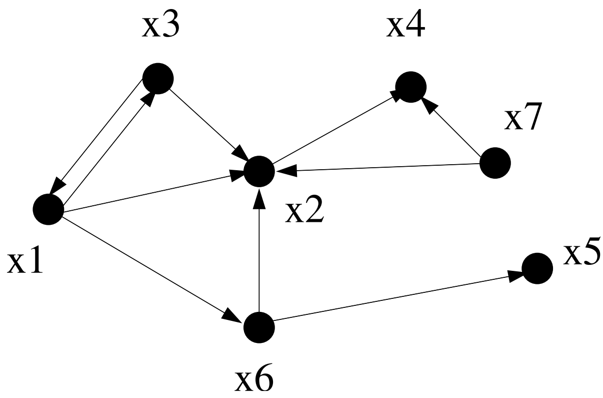
So there is a better than even chance that the sailor ends up at least $\sqrt{t}/2$ steps from where he started.

Hint: Work in terms of B. Then you can use an estimate that bounds the binomial distribution. Alternatively, observe that the origin is the most likely final location and then use the asymptotic estimate

$$\Pr\{L = 0\} = \Pr\{B = t/2\} \sim \sqrt{\frac{2}{\pi t}}.$$

17.2 Random Walks on Graphs

The hyperlink structure of the World Wide Web can be described as a digraph. The vertices are the web pages with a directed edge from vertex x to vertex y if x has a link to y. For example, in the following graph the vertices $x_1, \ldots, x_n$ correspond to web pages and $x_i \to x_j$ is a directed edge when page x_i contains a hyperlink to page x_j.



The web graph is an enormous graph with many billions and probably even trillions of vertices. At first glance, this graph wouldn't seem to be very interesting. But in 1995, two students at Stanford, Larry Page and indexBrin, Sergey Sergey Brin realized that the structure of this graph could be very useful in building a search engine. Traditional document searching programs had been around for a long time and they worked in a fairly straightforward way. Basically, you would enter some search terms and the searching program would return all documents containing those terms. A relevance score might also be returned for each document based on the frequency or position that the search terms appeared in the document. For example, if the search term appeared in the title or appeared 100 times in a document, that document would get a higher score. So if an author wanted a document to get a higher score for certain keywords, he would put the keywords in the title and make it appear in lots of places. You can even see this today with some bogus web sites.

This approach works fine if you only have a few documents that match a search term. But on the web, there are billions of documents and millions of matches to a typical search.

For example, a few years ago a search on Google for "Math for Computer Science notes" gave 378,000 hits! How does Google decide which 10 or 20 to show first? It wouldn't be smart to pick a page that gets a high keyword score because it has "Math Math . . . Math" across the front of the document.

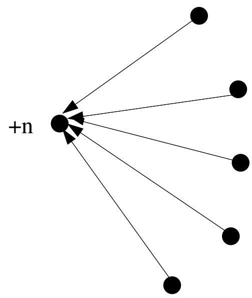
One way to get placed high on the list is to pay Google an advertising fees —and Google gets an enormous revenue stream from these fees. Of course an early listing is worth a fee only if an advertiser's target audience is attracted to the listing. But an audience does get attracted to Google listings because its ranking method is really good at determining the most relevant web pages. For example, Google demonstrated its accuracy in our case by giving first rank to our Fall 2002 Math for Computer Science notes on the MIT Open Courseware webpage $:-)$. So how did Google know to pick our class webage to be first out of $378,000$?

Well back in 1995, Larry and Sergey got the idea to allow the digraph structure of the web to determine which pages are likely to be the most important.

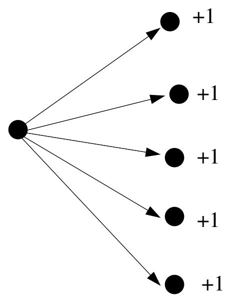
17.2.1 A First Crack at Page Rank

Looking at the web graph, any idea which vertex/page might be the best to rank 1st? Assume that all the pages match the search terms for now. Well, intuitively, we should choose x_2, since lots of other pages point to it. This leads us to their first idea: try defining the *page rank* of x to be the number of links pointing to x, that is, indegree(x). The idea is to think of web pages as voting for the most important page —the more votes, the better rank.

Of course, there are some problems with this idea. Suppose you wanted to have your page get a high ranking. One thing you could do is to create lots of dummy pages with links to your page.



There is another problem —a page could become unfairly influential by having lots of links to other pages it wanted to hype.



So this strategy for high ranking would amount to, "vote early, vote often," which is no good if you want to build a search engine that's worth paying fees for. So, admittedly, their original idea was not so great. It was better than nothing, but certainly not worth billions of dollars.

17.2.2 Random Walk on the Web Graph

But then Sergey and Larry thought some more and came up with a couple of improvements. Instead of just counting the indegree of a vertex, they considered the probability of being at each page after a long random walk on the web graph. In particular, they decided to model a user's web experience as following each link on a page with uniform probability. That is, they assigned each edge $x \to y$ of the web graph with a probability conditioned on being on page x:

$$\Pr\{\text{follow link } x \to y \mid \text{at page } x\} ::= \frac{1}{\text{outdegree}(x)}.$$

The user experience is then just a random walk on the web graph.

For example, if the user is at page x, and there are three links from page x, then each link is followed with probability $1/3$.

We can also compute the probability of arriving at a particular page, y, by summing over all edges pointing to y. We thus have

$$\begin{aligned} \Pr\{\text{go to } y\} &= \sum_{\text{edges } x \to y} \Pr\{\text{follow link } x \to y \mid \text{at page } x\} \cdot \Pr\{\text{at page } x\} \\ &= \sum_{\text{edges } x \to y} \frac{\Pr\{\text{at } x\}}{\text{outdegree}(x)} \end{aligned} \tag{17.8}$$

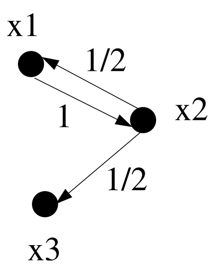
For example, in our web graph, we have

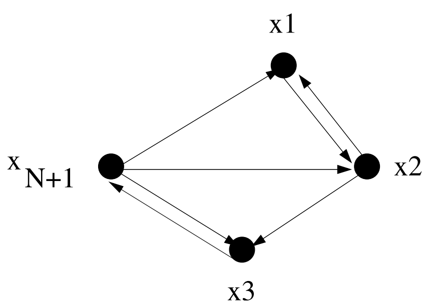
$$\Pr\{\text{go to } x_4\} = \frac{\Pr\{\text{at } x_7\}}{2} + \frac{\Pr\{\text{at } x_2\}}{1}.$$

One can think of this equation as x_7 sending half its probability to x_2 and the other half to x_4. The page x_2 sends all of its probability to x_4.

There's one aspect of the web graph described thus far that doesn't mesh with the user experience —some pages have no hyperlinks out. Under the current model, the user cannot escape these pages. In reality, however, the user doesn't fall off the end of the web into a void of nothingness. Instead, he restarts his web journey.

To model this aspect of the web, Sergey and Larry added a supervertex to the web graph and had every page with no hyperlinks point to it. Moreover, the supervertex points to every other vertex in the graph, allowing you to restart the walk from a random place. For example, below left is a graph and below right is the same graph after adding the supervertex x_{N+1}.





The addition of the supervertex also removes the possibility that the value $1/\text{outdegree}(x)$ might involve a division by zero.

17.2.3 Stationary Distribution & Page Rank

The basic idea of page rank is just a stationary distribution over the web graph, so let's define a stationary distribution.

Suppose each vertex is assigned a probability that corresponds, intuitively, to the likelihood that a random walker is at that vertex at a randomly chosen time. We assume that the walk never leaves the vertices in the graph, so we require that

$$\sum_{\text{vertices } x} \Pr\{\text{at } x\} = 1. \tag{17.9}$$

Definition 17.2.1. An assignment of probabililties to vertices in a digraph is a *stationary distribution* if for all vertices x

$$\Pr\{\text{at } x\} = \Pr\{\text{go to } x \text{ at next step}\}$$

Sergey and Larry defined their page ranks to be a stationary distribution. They did this by solving the following system of linear equations: find a nonnegative number, $\text{PR}(x)$, for each vertex, x, such that

$$\text{PR}(x) = \sum_{\text{edges } y \to x} \frac{\text{PR}(y)}{\text{outdegree}(y)}, \tag{17.10}$$

corresponding to the intuitive equations given in (17.8). These numbers must also satisfy the additional constraint corresponding to (17.9):

$$\sum_{\text{vertices } x} \text{PR}(x) = 1. \tag{17.11}$$

So if there are n vertices, then equations (17.10) and (17.11) provide a system of $n + 1$ linear equations in the n variables, $\text{PR}(x)$. Note that constraint (17.11) is needed because the remaining constraints (17.10) could be satisfied by letting $\text{PR}(x) ::= 0$ for all x, which is useless.

Sergey and Larry were smart fellows, and they set up their page rank algorithm so it would always have a meaningful solution. Their addition of a supervertex ensures there is always a *unique* stationary distribution. Moreover, starting from *any* vertex and taking a sufficiently long random walk on the graph, the probability of being at each page will get closer and closer to the stationary distribution. Note that general digraphs without supervertices may have neither of these properties: there may not be a unique stationary distribution, and even when there is, there may be starting points from which the probabilities of positions during a random walk do not converge to the stationary distribution.

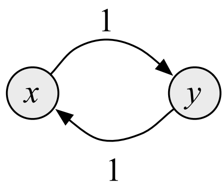
Now just keeping track of the digraph whose vertices are billions of web pages is a daunting task. That's why Google is building power plants. Indeed, Larry and Sergey named their system Google after the number 10^{100} —which called a "googol" —to reflect the fact that the web graph is so enormous.

Anyway, now you can see how our Math for Computer Science notes ranked first out of 378,000 matches. Lots of other universities used our notes and presumably have links to the notes on the Open Courseware site, and the university sites themselves are legitimate, which ultimately leads to our notes getting a high page rank in the web graph.

17.2.4 Problems

Class Problems

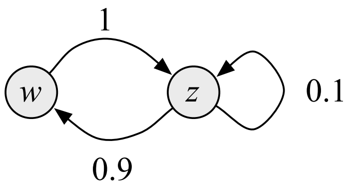
Problem 17.3.
Consider the following random-walk graph:



(a) Find a stationary distribution.

(b) If you start at node x and take a (long) random walk, does the distribution over nodes ever get close to the stationary distribution? Explain.

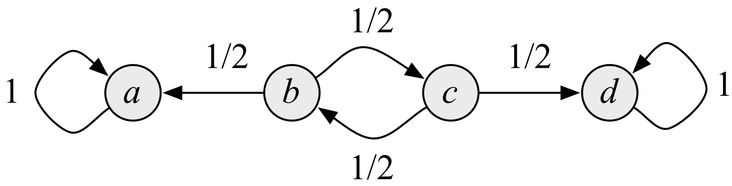
Consider the following random-walk graph:



(c) Find a stationary distribution.

(d) If you start at node w and take a (long) random walk, does the distribution over nodes ever get close to the stationary distribution? We don't want you to prove anything here, just write out a few steps and see what's happening.

Consider the following random-walk graph:



(e) Describe the stationary distributions for this graph.

(f) If you start at node b and take a long random walk, the probability you are at node d will be close to what fraction? Explain.

Problem 17.4.
A Google-graph is a random-walk graph such that every edge leaving any given vertex has the same probability. That is, the probability of each edge $v \to w$ is $1/\text{out-degree}(v)$.

A directed graph is *symmetric* if, whenever $v \to w$ is an edge, so is $w \to v$.

Given any finite, symmetric Google-graph, let

$$d(v) ::= \frac{\text{out-degree}(v)}{e},$$

where e is the total number of edges in the graph. Show that d is a stationary distribution.

Homework Problems

Problem 17.5.
A digraph is *strongly connected* iff there is a directed path between every pair of distinct vertices. In this problem we consider a finite random walk graph that is strongly connected.

(a) Let d_1 and d_2 be distinct distributions for the graph, and define the *maximum dilation*, γ, of d_1 over d_2 to be

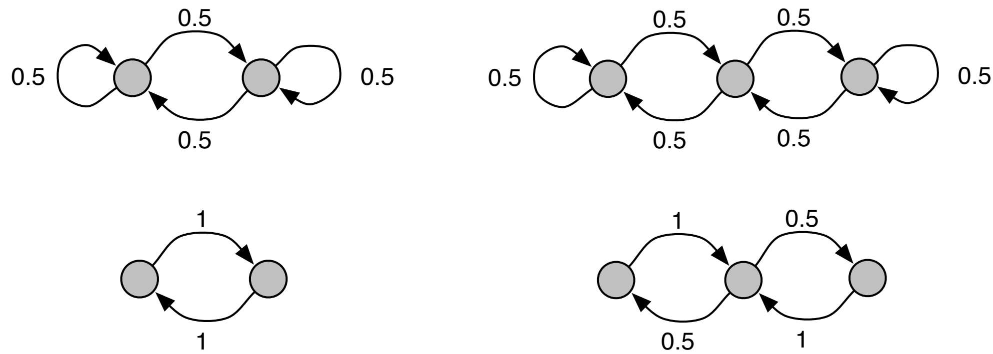
$$\gamma ::= \max_{x \in V} \frac{d_1(x)}{d_2(x)} .$$

Call a vertex, x, *dilated* if $d_1(x)/d_2(x) = \gamma$. Show that there is an edge, $y \to z$, from an undilated vertex y to a dilated vertex, z. *Hint:* Choose any dilated vertex, x, and consider the set, D, of dilated vertices connected to x by a directed path (going to x) that only uses dilated vertices. Explain why $D \neq V$, and then use the fact that the graph is strongly connected.

(b) Prove that the graph has *at most one* stationary distribution. (There always *is* a stationary distribution, but we're not asking you prove this.) *Hint:* Let d_1 be a stationary distribution and d_2 be a different distribution. Let z be the vertex from part (a). Show that starting from d_2, the probability of z changes at the next step. That is, $\widehat{d_2}(z) \neq d_2(z)$.

Exam Problems

Problem 17.6.
For which of the following graphs is the uniform distribution over nodes a stationary distribution? The edges are labeled with transition probabilities. Explain your reasoning.


0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
1
1
1
0.5
0.5
1

Chapter 18

Random Variables

So far we focused on probabilities of events —that you win the Monty Hall game; that you have a rare medical condition, given that you tested positive; .... Now we focus on quantitative questions: *How many* contestants must play the Monty Hall game until one of them finally wins? ...*How long* will this condition last? *How much* will I lose playing silly Math games all day? Random variables are the mathematical tool for addressing such questions, and in this chapter we work out their basic properties, especially properties of their *mean* or *expected value*.

18.1 Random Variable Examples

Definition 18.1.1. A *random variable*, R, on a probability space is a total function whose domain is the sample space.

The codomain of R can be anything, but will usually be a subset of the real numbers. Notice that the name "random variable" is a misnomer; random variables are actually functions!

For example, suppose we toss three independent, unbiased coins. Let C be the number of heads that appear. Let $M = 1$ if the three coins come up all heads or all tails, and let $M = 0$ otherwise. Now every outcome of the three coin flips uniquely determines the values of C and M. For example, if we flip heads, tails, heads, then $C = 2$ and $M = 0$. If we flip tails, tails, tails, then $C = 0$ and $M = 1$. In effect, C counts the number of heads, and M indicates whether all the coins match.

Since each outcome uniquely determines C and M, we can regard them as functions mapping outcomes to numbers. For this experiment, the sample space is:

$$\mathcal{S} = \{HHH, HHT, HTH, HTT, THH, THT, TTH, TTT\}.$$

Now C is a function that maps each outcome in the sample space to a number as follows:

$$\begin{array}{rclcrcl} C(HHH) & = & 3 & \qquad & C(THH) & = & 2 \\ C(HHT) & = & 2 & & C(THT) & = & 1 \\ C(HTH) & = & 2 & & C(TTH) & = & 1 \\ C(HTT) & = & 1 & & C(TTT) & = & 0. \end{array}$$

Similarly, M is a function mapping each outcome another way:

$$\begin{array}{rclcrcl} M(HHH) & = & 1 & \qquad & M(THH) & = & 0 \\ M(HHT) & = & 0 & & M(THT) & = & 0 \\ M(HTH) & = & 0 & & M(TTH) & = & 0 \\ M(HTT) & = & 0 & & M(TTT) & = & 1. \end{array}$$

So C and M are random variables.

18.1.1 Indicator Random Variables

An *indicator random variable* is a random variable that maps every outcome to either 0 or 1. These are also called *Bernoulli variables*. The random variable M is an example. If all three coins match, then $M = 1$; otherwise, $M = 0$.

Indicator random variables are closely related to events. In particular, an indicator partitions the sample space into those outcomes mapped to 1 and those outcomes mapped to 0. For example, the indicator M partitions the sample space into two blocks as follows:

$$\underbrace{HHH \quad TTT}_{M=1} \quad \underbrace{HHT \quad HTH \quad HTT \quad THH \quad THT \quad TTH}_{M=0}.$$

In the same way, an event, E, partitions the sample space into those outcomes in E and those not in E. So E is naturally associated with an indicator random variable, I_E, where $I_E(p) = 1$ for outcomes $p \in E$ and $I_E(p) = 0$ for outcomes $p \notin E$. Thus, $M = I_F$ where F is the event that all three coins match.

18.1.2 Random Variables and Events

There is a strong relationship between events and more general random variables as well. A random variable that takes on several values partitions the sample space into several blocks. For example, C partitions the sample space as follows:

$$\underbrace{TTT}_{C=0} \quad \underbrace{TTH \quad THT \quad HTT}_{C=1} \quad \underbrace{THH \quad HTH \quad HHT}_{C=2} \quad \underbrace{HHH}_{C=3}.$$

Each block is a subset of the sample space and is therefore an event. Thus, we can regard an equation or inequality involving a random variable as an event. For example, the event that $C = 2$ consists of the outcomes THH, HTH, and HHT. The event $C \leq 1$ consists of the outcomes TTT, TTH, THT, and HTT.

Naturally enough, we can talk about the probability of events defined by properties of random variables. For example,

$$\begin{aligned}\Pr\{C=2\} &= \Pr\{THH\}+\Pr\{HTH\}+\Pr\{HHT\}\\ &= \frac{1}{8}+\frac{1}{8}+\frac{1}{8}=\frac{3}{8}.\end{aligned}$$

18.1.3 Independence

The notion of independence carries over from events to random variables as well. Random variables R_1 and R_2 are *independent* iff for all x_1 in the codomain of R_1, and x_2 in the codomain of R_2, we have:

$$\Pr\{R_1=x_1 \text{ AND } R_2=x_2\}=\Pr\{R_1=x_1\}\cdot\Pr\{R_2=x_2\}.$$

As with events, we can formulate independence for random variables in an equivalent and perhaps more intuitive way: random variables R_1 and R_2 are independent if for all x_1 and x_2

$$\Pr\{R_1=x_1 \mid R_2=x_2\}=\Pr\{R_1=x_1\}.$$

whenever the lefthand conditional probability is defined, that is, whenever $\Pr\{R_2=x_2\}>0$.

As an example, are C and M independent? Intuitively, the answer should be "no". The number of heads, C, completely determines whether all three coins match; that is, whether $M=1$. But, to verify this intuition, we must find some $x_1,x_2\in\mathbb{R}$ such that:

$$\Pr\{C=x_1 \text{ AND } M=x_2\}\neq\Pr\{C=x_1\}\cdot\Pr\{M=x_2\}.$$

One appropriate choice of values is $x_1=2$ and $x_2=1$. In this case, we have:

$$\Pr\{C=2 \text{ AND } M=1\}=0\neq\frac{1}{4}\cdot\frac{3}{8}=\Pr\{M=1\}\cdot\Pr\{C=2\}.$$

The first probability is zero because we never have exactly two heads ($C=2$) when all three coins match ($M=1$). The other two probabilities were computed earlier.

On the other hand, let H_1 be the indicator variable for event that the first flip is a Head, so

$$[H_1=1]=\{HHH,HTH,HHT,HTT\}.$$

Then H_1 is independent of M, since

$$\begin{aligned}\Pr\{M=1\}=1/4&=\Pr\{M=1 \mid H_1=1\}=\Pr\{M=1 \mid H_1=0\}\\ \Pr\{M=0\}=3/4&=\Pr\{M=0 \mid H_1=1\}=\Pr\{M=0 \mid H_1=0\}\end{aligned}$$

This example is an instance of a simple lemma:

Lemma 18.1.2. *Two events are independent iff their indicator variables are independent.*

As with events, the notion of independence generalizes to more than two random variables.

Definition 18.1.3. Random variables $R_1, R_2, \dots, R_n$ are *mutually independent* iff

$$\begin{aligned} &\Pr\{R_1 = x_1 \text{ AND } R_2 = x_2 \text{ AND } \cdots \text{ AND } R_n = x_n\} \\ &= \quad \Pr\{R_1 = x_1\} \cdot \Pr\{R_2 = x_2\} \cdots \Pr\{R_n = x_n\}. \end{aligned}$$

for all $x_1, x_2, \dots, x_n$.

It is a simple exercise to show that the probability that any *subset* of the variables takes a particular set of values is equal to the product of the probabilities that the individual variables take their values. Thus, for example, if $R_1, R_2, \dots, R_{100}$ are mutually independent random variables, then it follows that:

$$\Pr\{R_1 = 7 \text{ AND } R_7 = 9.1 \text{ AND } R_{23} = \pi\} = \Pr\{R_1 = 7\} \cdot \Pr\{R_7 = 9.1\} \cdot \Pr\{R_{23} = \pi\}.$$

18.2 Probability Distributions

A random variable maps outcomes to values, but random variables that show up for different spaces of outcomes wind up behaving in much the same way because they have the same probability of taking any given value. Namely, random variables on different probability spaces may wind up having the same probability density function.

Definition 18.2.1. Let R be a random variable with codomain V. The *probability density function (pdf)* of R is a function $\text{PDF}_R : V \to [0, 1]$ defined by:

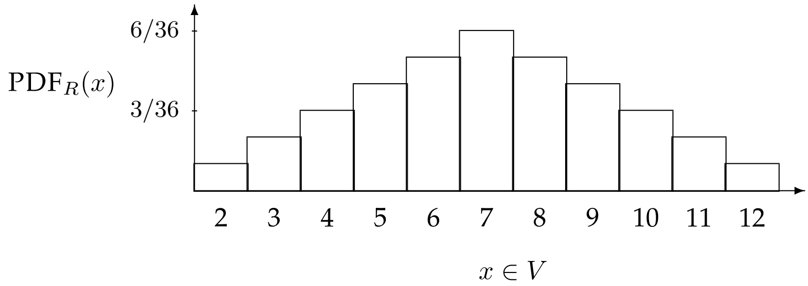
$$\text{PDF}_R(x) ::= \begin{cases} \Pr\{R = x\} & \text{if } x \in \text{range}(R), \\ 0 & \text{if } x \notin \text{range}(R). \end{cases}$$

A consequence of this definition is that

$$\sum_{x \in \text{range}(R)} \text{PDF}_R(x) = 1.$$

This follows because R has a value for each outcome, so summing the probabilities over all outcomes is the same as summing over the probabilities of each value in the range of R.

As an example, let's return to the experiment of rolling two fair, independent dice. As before, let T be the total of the two rolls. This random variable takes on values in the set $V = \{2, 3, \dots, 12\}$. A plot of the probability density function is shown below:

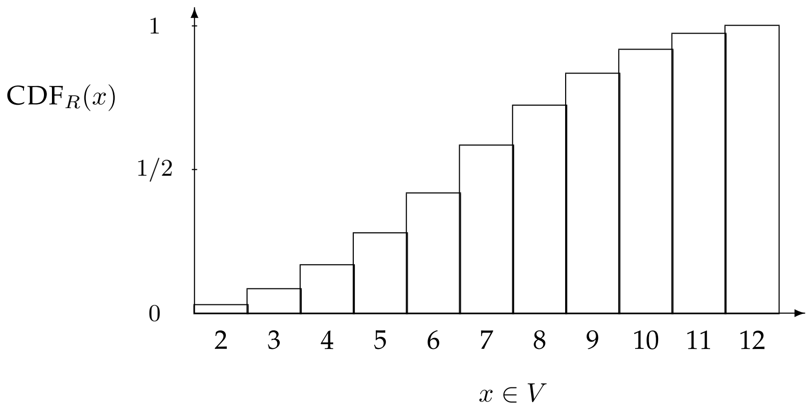


The lump in the middle indicates that sums close to 7 are the most likely. The total area of all the rectangles is 1 since the dice must take on exactly one of the sums in $V = \{2, 3, \ldots, 12\}$.

A closely-related idea is the *cumulative distribution function (cdf)* for a random variable R whose codomain is real numbers. This is a function $\text{CDF}_R : \mathbb{R} \to [0, 1]$ defined by:

$$\text{CDF}_R(x) = \Pr\{R \leq x\}$$

As an example, the cumulative distribution function for the random variable T is shown below:



The height of the i-th bar in the cumulative distribution function is equal to the *sum* of the heights of the leftmost i bars in the probability density function. This follows from the definitions of pdf and cdf:

$$\begin{aligned}
\text{CDF}_R(x) &= \Pr\{R \leq x\} \\
&= \sum_{y \leq x} \Pr\{R = y\} \\
&= \sum_{y \leq x} \text{PDF}_R(y)
\end{aligned}$$

In summary, $\text{PDF}_R(x)$ measures the probability that $R = x$ and $\text{CDF}_R(x)$ measures the probability that $R \leq x$. Both the PDF_R and CDF_R capture the same information about the random variable R— you can derive one from the other —but sometimes one is more convenient. The key point here is that neither the probability density function nor the cumulative distribution function involves the sample space of an experiment.

We'll now look at three important distributions and some applications.

18.2.1 Bernoulli Distribution

Indicator random variables are perhaps the most common type because of their close association with events. The probability density function of an indicator random variable, B, is always

$$\begin{aligned}\mathrm{PDF}_B(0) &= p\\ \mathrm{PDF}_B(1) &= 1-p\end{aligned}$$

where $0 \leq p \leq 1$. The corresponding cumulative distribution function is:

$$\begin{aligned}\mathrm{CDF}_B(0) &= p\\ \mathrm{CDF}_B(1) &= 1\end{aligned}$$

18.2.2 Uniform Distribution

A random variable that takes on each possible value with the same probability is called *uniform*. For example, the probability density function of a random variable U that is uniform on the set $\{1, 2, \ldots, N\}$ is:

$$\mathrm{PDF}_U(k) = \frac{1}{N}$$

And the cumulative distribution function is:

$$\mathrm{CDF}_U(k) = \frac{k}{N}$$

Uniform distributions come up all the time. For example, the number rolled on a fair die is uniform on the set $\{1, 2, \ldots, 6\}$.

18.2.3 The Numbers Game

Let's play a game! I have two envelopes. Each contains an integer in the range $0, 1, \ldots, 100$, and the numbers are distinct. To win the game, you must determine which envelope contains the larger number. To give you a fighting chance, I'll let you peek at the number in one envelope selected at random. Can you devise a strategy that gives you a better than 50% chance of winning?

For example, you could just pick an envelope at random and guess that it contains the larger number. But this strategy wins only 50% of the time. Your challenge is to do better.

So you might try to be more clever. Suppose you peek in the left envelope and see the number 12. Since 12 is a small number, you might guess that that other number is larger. But perhaps I'm sort of tricky and put small numbers in *both* envelopes. Then your guess might not be so good!

An important point here is that the numbers in the envelopes may *not* be random. I'm picking the numbers and I'm choosing them in a way that I think will defeat your guessing strategy. I'll only use randomization to choose the numbers if that serves *my* end: making you lose!

Intuition Behind the Winning Strategy

Amazingly, there is a strategy that wins more than 50% of the time, regardless of what numbers I put in the envelopes!

Suppose that you somehow knew a number x *between* my lower number and higher numbers. Now you peek in an envelope and see one or the other. If it is bigger than x, then you know you're peeking at the higher number. If it is smaller than x, then you're peeking at the lower number. In other words, if you know a number x between my lower and higher numbers, then you are certain to win the game.

The only flaw with this brilliant strategy is that you do *not* know x. Oh well.

But what if you try to *guess* x? There is some probability that you guess correctly. In this case, you win 100% of the time. On the other hand, if you guess incorrectly, then you're no worse off than before; your chance of winning is still 50%. Combining these two cases, your overall chance of winning is better than 50%!

Informal arguments about probability, like this one, often sound plausible, but do not hold up under close scrutiny. In contrast, this argument sounds completely implausible— but is actually correct!

Analysis of the Winning Strategy

For generality, suppose that I can choose numbers from the set $\{0, 1, \ldots, n\}$. Call the lower number L and the higher number H.

Your goal is to guess a number x between L and H. To avoid confusing equality cases, you select x at random from among the half-integers:

$$\left\{\frac{1}{2},\ 1\frac{1}{2},\ 2\frac{1}{2},\ \ldots,\ n-\frac{1}{2}\right\}$$

But what probability distribution should you use?

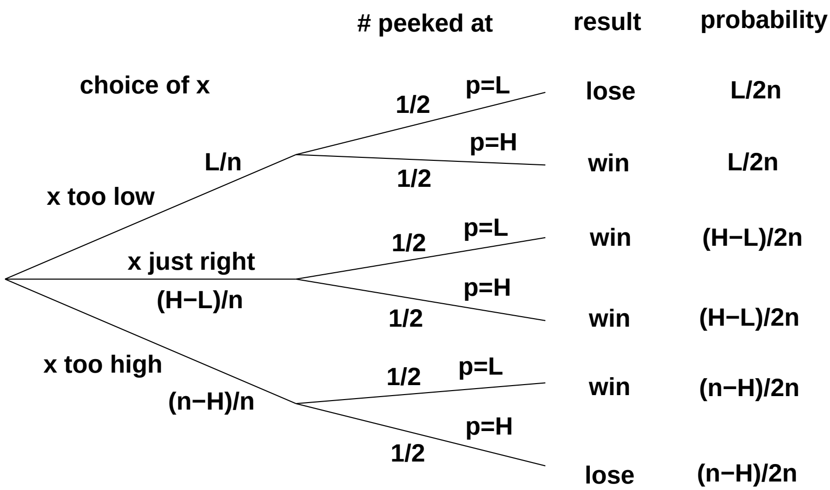
The uniform distribution turns out to be your best bet. An informal justification is that if I figured out that you were unlikely to pick some number— say $50\frac{1}{2}$— then I'd always put 50 and 51 in the evelopes. Then you'd be unlikely to pick an x between L and H and would have less chance of winning.

After you've selected the number x, you peek into an envelope and see some number p. If $p > x$, then you guess that you're looking at the larger number. If

$p < x$, then you guess that the other number is larger.

All that remains is to determine the probability that this strategy succeeds. We can do this with the usual four step method and a tree diagram.

Step 1: Find the sample space. You either choose x too low ($< L$), too high ($> H$), or just right ($L < x < H$). Then you either peek at the lower number ($p = L$) or the higher number ($p = H$). This gives a total of six possible outcomes.



Step 2: Define events of interest. The four outcomes in the event that you win are marked in the tree diagram.

Step 3: Assign outcome probabilities. First, we assign edge probabilities. Your guess x is too low with probability L/n, too high with probability $(n - H)/n$, and just right with probability $(H - L)/n$. Next, you peek at either the lower or higher number with equal probability. Multiplying along root-to-leaf paths gives the outcome probabilities.

Step 4: Compute event probabilities. The probability of the event that you win is the sum of the probabilities of the four outcomes in that event:

$$\begin{aligned}
\Pr\{\text{win}\} &= \frac{L}{2n} + \frac{H-L}{2n} + \frac{H-L}{2n} + \frac{n-H}{2n} \\
&= \frac{1}{2} + \frac{H-L}{2n} \\
&\geq \frac{1}{2} + \frac{1}{2n}
\end{aligned}$$

The final inequality relies on the fact that the higher number H is at least 1 greater than the lower number L since they are required to be distinct.

Sure enough, you win with this strategy more than half the time, regardless of the numbers in the envelopes! For example, if I choose numbers in the range $0, 1, \ldots, 100$, then you win with probability at least $\frac{1}{2} + \frac{1}{200} = 50.5\%$. Even better, if I'm allowed only numbers in the range $0, \ldots, 10$, then your probability of winning rises to 55%! By Las Vegas standards, those are great odds!

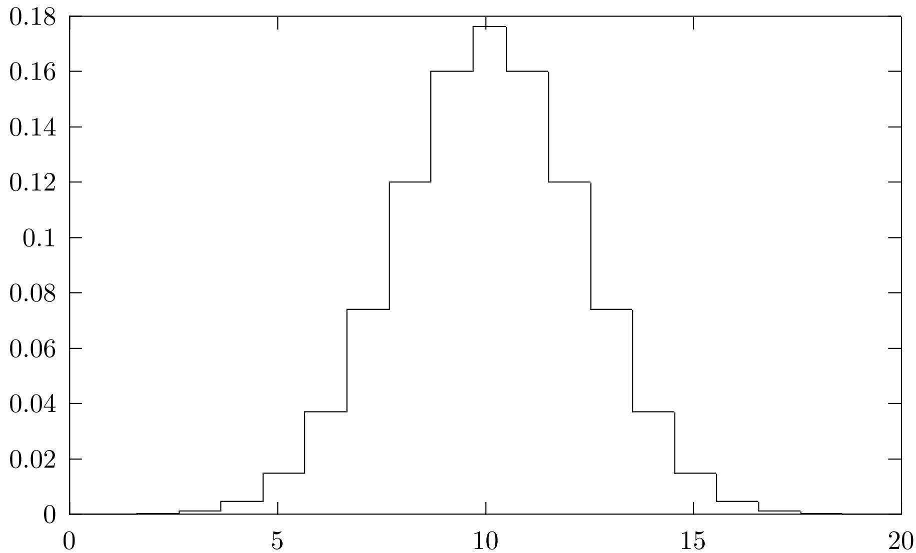
18.2.4 Binomial Distribution

The *binomial distribution* plays an important role in Computer Science as it does in most other sciences. The standard example of a random variable with a binomial distribution is the number of heads that come up in n independent flips of a coin; call this random variable H_n. If the coin is fair, then H_n has an *unbiased binomial density function*:

$$\mathrm{PDF}_{H_n}(k) = \binom{n}{k} 2^{-n}.$$

This follows because there are $\binom{n}{k}$ sequences of n coin tosses with exactly k heads, and each such sequence has probability 2^{-n}.

Here is a plot of the unbiased probability density function $\mathrm{PDF}_{H_n}(k)$ corresponding to $n = 20$ coins flips. The most likely outcome is $k = 10$ heads, and the probability falls off rapidly for larger and smaller values of k. These falloff regions to the left and right of the main hump are usually called the *tails of the distribution*.



In many fields, including Computer Science, probability analyses come down to getting small bounds on the tails of the binomial distribution. In the context of a problem, this typically means that there is very small probability that something *bad* happens, which could be a server or communication link overloading or a randomized algorithm running for an exceptionally long time or producing the wrong result.

As an example, we can calculate the probability of flipping at most 25 heads in 100 tosses of a fair coin and see that it is very small, namely, less than 1 in 3,000,000.

In fact, the tail of the distribution falls off so rapidly that the probability of flipping exactly 25 heads is nearly twice the probability of flipping fewer than 25 heads! That is, the probability of flipping exactly 25 heads —small as it is —is still nearly twice as large as the probability of flipping exactly 24 heads *plus* the probability of flipping exactly 23 heads *plus* ... the probability of flipping no heads.

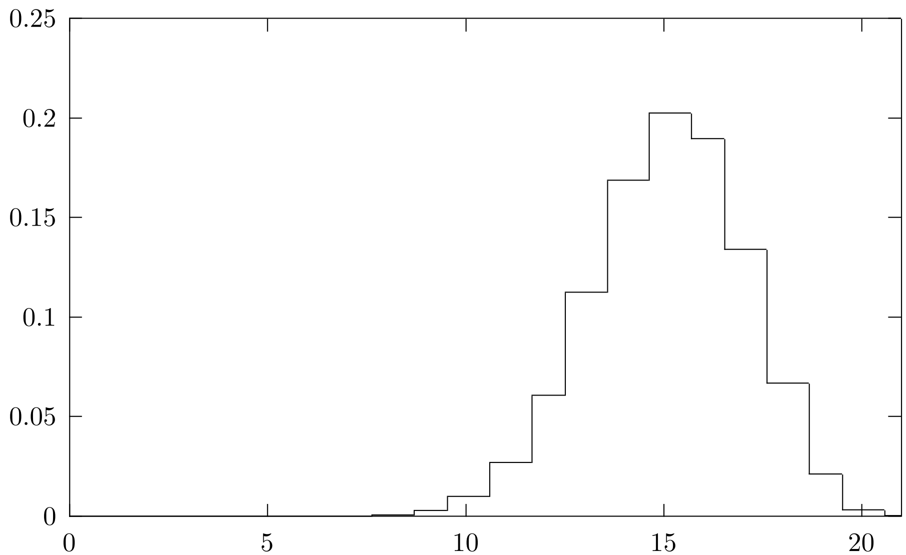
The General Binomial Distribution

Now let J be the number of heads that come up on n independent coins, each of which is heads with probability p. Then J has a *general binomial density function*:

$$\mathrm{PDF}_J(k) = \binom{n}{k} p^k (1-p)^{n-k}.$$

As before, there are $\binom{n}{k}$ sequences with k heads and $n-k$ tails, but now the probability of each such sequence is $p^k(1-p)^{n-k}$.

As an example, the plot below shows the probability density function $\mathrm{PDF}_J(k)$ corresponding to flipping $n = 20$ independent coins that are heads with probabilty $p = 0.75$. The graph shows that we are most likely to get around $k = 15$ heads, as you might expect. Once again, the probability falls off quickly for larger and smaller values of k.



18.2.5 Problems

Class Problems

Guess the Bigger Number Game
Team 1: • Write different integers between 0 and 7 on two pieces of paper. • Put the papers face down on a table. Team 2: • Turn over one paper and look at the number on it. • Either stick with this number or switch to the unseen other number. Team 2 wins if it chooses the larger number.

Problem 18.1.
The analysis in section 18.2.3 implies that Team 2 has a strategy that wins 4/7 of the time no matter how Team 1 plays. Can Team 2 do better? The answer is "no," because Team 1 has a strategy that guarantees that it wins at least 3/7 of the time, no matter how Team 2 plays. Describe such a strategy for Team 1 and explain why it works.

Problem 18.2.
Suppose X_1, X_2, and X_3 are three mutually independent random variables, each having the uniform distribution

$$\Pr\{X_i = k\} \text{ equal to } 1/3 \text{ for each of } k = 1, 2, 3.$$

Let M be another random variable giving the maximum of these three random variables. What is the density function of M?

Homework Problems

Problem 18.3.
A drunken sailor wanders along main street, which conveniently consists of the points along the x axis with integral coordinates. In each step, the sailor moves

one unit left or right along the x axis. A particular *path* taken by the sailor can be described by a sequence of "left" and "right" steps. For example, ⟨left,left,right⟩ describes the walk that goes left twice then goes right.

We model this scenario with a random walk graph whose vertices are the integers and with edges going in each direction between consecutive integers. All edges are labelled $1/2$.

The sailor begins his random walk at the origin. This is described by an initial distribution which labels the origin with probability 1 and all other vertices with probability 0. After one step, the sailor is equally likely to be at location 1 or -1, so the distribution after one step gives label $1/2$ to the vertices 1 and -1 and labels all other vertices with probability 0.

(a) Give the distributions after the 2nd, 3rd, and 4th step by filling in the table of probabilities below, where omitted entries are 0. For each row, write all the nonzero entries so they have the same denominator.

	location								
	-4	-3	-2	-1	0	1	2	3	4
initially					1				
after 1 step				1/2	0	1/2			
after 2 steps			?	?	?	?	?		
after 3 steps		?	?	?	?	?	?	?	
after 4 steps	?	?	?	?	?	?	?	?	?

(b)

1. What is the final location of a t-step path that moves right exactly i times?
2. How many different paths are there that end at that location?
3. What is the probability that the sailor ends at this location?

(c) Let L be the random variable giving the sailor's location after t steps, and let $B ::= (L+t)/2$. Use the answer to part (b) to show that B has an unbiased binomial density function.

(d) Again let L be the random variable giving the sailor's location after t steps, where t is even. Show that

$$\Pr\left\{|L| < \frac{\sqrt{t}}{2}\right\} < \frac{1}{2}.$$

So there is a better than even chance that the sailor ends up at least $\sqrt{t}/2$ steps from where he started.

Hint: Work in terms of B. Then you can use an estimate that bounds the binomial distribution. Alternatively, observe that the origin is the most likely final location and then use the asymptotic estimate

$$\Pr\{L = 0\} = \Pr\{B = t/2\} \sim \sqrt{\frac{2}{\pi t}}.$$

18.3 Average & Expected Value

The *expectation* of a random variable is its average value, where each value is weighted according to the probability that it comes up. The expectation is also called the *expected value* or the *mean* of the random variable.

For example, suppose we select a student uniformly at random from the class, and let R be the student's quiz score. Then $\mathrm{E}[R]$ is just the class average —the first thing everyone wants to know after getting their test back! For similar reasons, the first thing you usually want to know about a random variable is its expected value.

Definition 18.3.1.

$$\begin{aligned}\mathrm{E}[R] &::= \sum_{x \in \mathrm{range}(R)} x \cdot \Pr\{R = x\} \qquad (18.1)\\ &= \sum_{x \in \mathrm{range}(R)} x \cdot \mathrm{PDF}_R(x).\end{aligned}$$

Let's work through an example. Let R be the number that comes up on a fair, six-sided die. Then by (18.1), the expected value of R is:

$$\begin{aligned}\mathrm{E}[R] &= \sum_{k=1}^{6} k \cdot \frac{1}{6}\\ &= 1 \cdot \frac{1}{6} + 2 \cdot \frac{1}{6} + 3 \cdot \frac{1}{6} + 4 \cdot \frac{1}{6} + 5 \cdot \frac{1}{6} + 6 \cdot \frac{1}{6}\\ &= \frac{7}{2}\end{aligned}$$

This calculation shows that the name "expected value" is a little misleading; the random variable might *never* actually take on that value. You don't ever expect to roll a $3\frac{1}{2}$ on an ordinary die!

There is an even simpler formula for expectation:

Theorem 18.3.2. *If R is a random variable defined on a sample space, $\mathcal{S}$, then*

$$\mathrm{E}[R] = \sum_{\omega \in \mathcal{S}} R(\omega) \Pr\{\omega\} \qquad (18.2)$$

The proof of Theorem 18.3.2, like many of the elementary proofs about expectation in this chapter, follows by judicious regrouping of terms in the defining sum (18.1):

Proof.

$$
\begin{aligned}
\mathrm{E}[R] &::= \sum_{x \in \mathrm{range}(R)} x \cdot \Pr\{R = x\} && \text{(Def 18.3.1 of expectation)} \\
&= \sum_{x \in \mathrm{range}(R)} x \left(\sum_{\omega \in [R=x]} \Pr\{\omega\} \right) && \text{(def of } \Pr\{R = x\}\text{)} \\
&= \sum_{x \in \mathrm{range}(R)} \sum_{\omega \in [R=x]} x \Pr\{\omega\} && \text{(distributing } x \text{ over the inner sum)} \\
&= \sum_{x \in \mathrm{range}(R)} \sum_{\omega \in [R=x]} R(\omega) \Pr\{\omega\} && \text{(def of the event } [R = x]\text{)} \\
&= \sum_{\omega \in \mathcal{S}} R(\omega) \Pr\{\omega\}
\end{aligned}
$$

The last equality follows because the events $[R = x]$ for $x \in \mathrm{range}(R)$ partition the sample space, $\mathcal{S}$, so summing over the outcomes in $[R = x]$ for $x \in \mathrm{range}(R)$ is the same as summing over $\mathcal{S}$. ■

In general, the defining sum (18.1) is better for calculating expected values and has the advantage that it does not depend on the sample space, but only on the density function of the random variable. On the other hand, the simpler sum over all outcomes (18.2)is sometimes easier to use in proofs about expectation.

18.3.1 Expected Value of an Indicator Variable

The expected value of an indicator random variable for an event is just the probability of that event.

Lemma 18.3.3. *If I_A is the indicator random variable for event A, then*

$$\mathrm{E}[I_A] = \Pr\{A\}.$$

Proof.

$$
\begin{aligned}
\mathrm{E}[I_A] &= 1 \cdot \Pr\{I_A = 1\} + 0 \cdot \Pr\{I_A = 0\} \\
&= \Pr\{I_A = 1\} \\
&= \Pr\{A\}. && \text{(def of } I_A\text{)}
\end{aligned}
$$

■

For example, if A is the event that a coin with bias p comes up heads, $\mathrm{E}[I_A] = \Pr\{I_A = 1\} = p$.

18.3.2 Conditional Expectation

Just like event probabilities, expectations can be conditioned on some event.

Definition 18.3.4. The *conditional expectation*, $\mathrm{E}[R \mid A]$, of a random variable, R, given event, A, is:

$$\mathrm{E}[R \mid A] ::= \sum_{r \in \mathrm{range}(R)} r \cdot \Pr\{R = r \mid A\}. \tag{18.3}$$

In other words, it is the average value of the variable R when values are weighted by their conditional probabilities given A.

For example, we can compute the expected value of a roll of a fair die, *given*, for example, that the number rolled is at least 4. We do this by letting R be the outcome of a roll of the die. Then by equation (18.3),

$$\mathrm{E}[R \mid R \geq 4] = \sum_{i=1}^{6} i \cdot \Pr\{R = i \mid R \geq 4\} = 1 \cdot 0 + 2 \cdot 0 + 3 \cdot 0 + 4 \cdot \tfrac{1}{3} + 5 \cdot \tfrac{1}{3} + 6 \cdot \tfrac{1}{3} = 5.$$

The power of conditional expectation is that it lets us divide complicated expectation calculations into simpler cases. We can find the desired expectation by calculating the conditional expectation in each simple case and averaging them, weighing each case by its probability.

For example, suppose that 49.8% of the people in the world are male and the rest female —which is more or less true. Also suppose the expected height of a randomly chosen male is $5'\,11''$, while the expected height of a randomly chosen female is $5'\,5''$. What is the expected height of a randomly chosen individual? We can calculate this by averaging the heights of men and women. Namely, let H be the height (in feet) of a randomly chosen person, and let M be the event that the person is male and F the event that the person is female. We have

$$\begin{aligned}
\mathrm{E}[H] &= \mathrm{E}[H \mid M] \Pr\{M\} + \mathrm{E}[H \mid F] \Pr\{F\} \\
&= (5 + 11/12) \cdot 0.498 + (5 + 5/12) \cdot 0.502 \\
&= 5.665
\end{aligned}$$

which is a little less that $5'\,8''$.

The Law of *Total Expectation* justifies this method.

Theorem 18.3.5. *Let $A_1, A_2, \ldots$ be a partition of the sample space. Then*

Rule (Law of Total Expectation)**.**

$$\mathrm{E}[R] = \sum_i \mathrm{E}[R \mid A_i] \Pr\{A_i\}.$$

Proof.

$$
\begin{aligned}
\mathrm{E}[R] &::= \sum_{r \in \mathrm{range}(R)} r \cdot \Pr\{R = r\} && \text{(Def 18.3.1 of expectation)} \\
&= \sum_r r \cdot \sum_i \Pr\{R = r \mid A_i\} \Pr\{A_i\} && \text{(Law of Total Probability)} \\
&= \sum_r \sum_i r \cdot \Pr\{R = r \mid A_i\} \Pr\{A_i\} && \text{(distribute constant } r\text{)} \\
&= \sum_i \sum_r r \cdot \Pr\{R = r \mid A_i\} \Pr\{A_i\} && \text{(exchange order of summation)} \\
&= \sum_i \Pr\{A_i\} \sum_r r \cdot \Pr\{R = r \mid A_i\} && \text{(factor constant } \Pr\{A_i\}\text{)} \\
&= \sum_i \Pr\{A_i\} \mathrm{E}[R \mid A_i]. && \text{(Def 18.3.4 of cond. expectation)}
\end{aligned}
$$

■

Chapter 19

Deviation from the Mean

19.1 Why the Mean?

In the previous chapter we took it for granted that expectation is important, and we developed a bunch of techniques for calculating expected (mean) values. But why should we care about the mean? After all, a random variable may never take a value anywhere near its expected value.

The most important reason to care about the mean value comes from its connection to estimation by sampling. For example, suppose we want to estimate the average age, income, family size, or other measure of a population. To do this, we determine a random process for selecting people —say throwing darts at census lists. This process makes the selected person's age, income, and so on into a random variable whose *mean* equals the *actual average* age or income of the population. So we can select a random sample of people and calculate the average of people in the sample to estimate the true average in the whole population. Many fundamental results of probability theory explain exactly how the reliability of such estimates improves as the sample size increases, and in this chapter we'll examine a few such results.

In particular, when we make an estimate by repeated sampling, we need to know how much confidence we should have that our estimate is OK. Technically, this reduces to finding the probability that an estimate *deviates* a lot from its expected value. This topic of *deviation from the mean* is the focus of this final chapter.

The first technical result about deviation will be Markov's Theorem, which gives a simple, but typically coarse, upper bound on the probability that the value of a random variable is more than a certain multiple of its mean. Markov's result holds if we know nothing about a random variable except what its mean is and that its values are nonnegative. Accordingly, Markov's Theorem is very general, but also is much weaker than results which take into account more information about the distribution of the variable.

In many situations, we not only know the mean, but also another numerical quantity called the *variance* of the random variable. The second basic result is

Chebyshev's Theorem, which combines Markov's Theorem and information about the variance to give more refined bounds.

The final result we obtain about deviation is Chernoff's bound. Chernoff's bound applies to a random variable that is a sum of bounded independent random variables. Its bound is exponentially tighter than the other two.

19.2 Markov's Theorem

Markov's theorem is an easy result that gives a generally rough estimate of the probability that a random variable takes a value *much larger* than its mean.

The idea behind Markov's Theorem can be explained with a simple example of *intelligence quotient*, IQ. This quantity was devised so that the average IQ measurement would be 100. Now from this fact alone we can conclude that at most 1/3 the population can have an IQ of 300 or more, because if more than a third had an IQ of 300, then the average would have to be *more* than $(1/3)300 = 100$, contradicting the fact that the average is 100. So the probability that a randomly chosen person has an IQ of 300 or more is at most 1/3. Of course this is not a very strong conclusion; in fact no IQ of over 300 has ever been recorded. But by the same logic, we can also conclude that at most 2/3 of the population can have an IQ of 150 or more. IQ's of over 150 have certainly been recorded, though again, a much smaller fraction than 2/3 of the population actually has an IQ that high.

But although these conclusions about IQ are weak, they are actually the strongest general conclusions that can be reached about a random variable using *only* the fact that it is nonnegative and its mean is 100. For example, if we choose a random variable equal to 300 with probability 1/3, and 0 with probability 2/3, then its mean is 100, and the probability of a value of 300 or more really is 1/3. So we can't hope to get a better upper bound based solely on this limited amount of information.

Theorem 19.2.1 (Markov's Theorem)**.** *If R is a nonnegative random variable, then for all $x > 0$*

$$\Pr\{R \geq x\} \leq \frac{\mathrm{E}[R]}{x}.$$

Proof. For any $x > 0$

$$
\begin{aligned}
\mathrm{E}[R] &::= \sum_{y \in \text{range}(R)} y \Pr\{R = y\} \\
&\geq \sum_{\substack{y \geq x, \\ y \in \text{range}(R)}} y \Pr\{R = y\} && \text{(because } R \geq 0\text{)} \\
&\geq \sum_{\substack{y \geq x, \\ y \in \text{range}(R)}} x \Pr\{R = y\} \\
&= x \sum_{\substack{y \geq x, \\ y \in \text{range}(R)}} \Pr\{R = y\} \\
&= x \Pr\{R \geq x\}. && (19.1)
\end{aligned}
$$

Dividing the first and last expression (19.1) by x gives the desired result. ■

Our focus is deviation from the mean, so it's useful to rephrase Markov's Theorem this way:

Corollary 19.2.2. *If R is a nonnegative random variable, then for all $c \geq 1$*

$$\Pr\{R \geq c \cdot \mathrm{E}[R]\} \leq \frac{1}{c}. \tag{19.2}$$

This Corollary follows immediately from Markov's Theorem(19.2.1) by letting x be $c \cdot \mathrm{E}[R]$.

19.2.1 Applying Markov's Theorem

Let's consider the Hat-Check problem again. Now we ask what the probability is that x or more men get the right hat, this is, what the value of $\Pr\{G \geq x\}$ is.

We can compute an upper bound with Markov's Theorem. Since we know $\mathrm{E}[G] = 1$, Markov's Theorem implies

$$\Pr\{G \geq x\} \leq \frac{\mathrm{E}[G]}{x} = \frac{1}{x}.$$

For example, there is no better than a 20% chance that 5 men get the right hat, regardless of the number of people at the dinner party.

The Chinese Appetizer problem is similar to the Hat-Check problem. In this case, n people are eating appetizers arranged on a circular, rotating Chinese banquet tray. Someone then spins the tray so that each person receives a random appetizer. What is the probability that everyone gets the same appetizer as before?

There are n equally likely orientations for the tray after it stops spinning. Everyone gets the right appetizer in just one of these n orientations. Therefore, the correct answer is $1/n$.

But what probability do we get from Markov's Theorem? Let the random variable, R, be the number of people that get the right appetizer. Then of course $\text{E}[R] = 1$ (right?), so applying Markov's Theorem, we find:

$$\Pr\{R \geq n\} \leq \frac{\text{E}[R]}{n} = \frac{1}{n}.$$

So for the Chinese appetizer problem, Markov's Theorem is tight!

On the other hand, Markov's Theorem gives the same $1/n$ bound for the probability everyone gets their hat in the Hat-Check problem in the case that all permutations are equally likely. But the probability of this event is $1/(n!)$. So for this case, Markov's Theorem gives a probability bound that is way off.

19.2.2 Markov's Theorem for Bounded Variables

Suppose we learn that the average IQ among MIT students is 150 (which is not true, by the way). What can we say about the probability that an MIT student has an IQ of more than 200? Markov's theorem immediately tells us that no more than $150/200$ or $3/4$ of the students can have such a high IQ. Here we simply applied Markov's Theorem to the random variable, R, equal to the IQ of a random MIT student to conclude:

$$\Pr\{R > 200\} \leq \frac{\text{E}[R]}{200} = \frac{150}{200} = \frac{3}{4}.$$

But let's observe an additional fact (which may be true): no MIT student has an IQ less than 100. This means that if we let $T ::= R - 100$, then T is nonnegative and $\text{E}[T] = 50$, so we can apply Markov's Theorem to T and conclude:

$$\Pr\{R > 200\} = \Pr\{T > 100\} \leq \frac{\text{E}[T]}{100} = \frac{50}{100} = \frac{1}{2}.$$

So only half, not 3/4, of the students can be as amazing as they think they are. A bit of a relief!

More generally, we can get better bounds applying Markov's Theorem to $R - l$ instead of R for any lower bound $l > 0$ on R.

Similarly, if we have any upper bound, u, on a random variable, S, then $u - S$ will be a nonnegative random variable, and applying Markov's Theorem to $u - S$ will allow us to bound the probability that S is much *less* than its expectation.

19.2.3 Problems

Class Problems

Problem 19.1.

A herd of cows is stricken by an outbreak of *cold cow disease*. The disease lowers the normal body temperature of a cow, and a cow will die if its temperature goes below 90 degrees F. The disease epidemic is so intense that it lowered the average temperature of the herd to 85 degrees. Body temperatures as low as 70 degrees, **but no lower**, were actually found in the herd.

(a) Prove that at most 3/4 of the cows could have survived.

Hint: Let T be the temperature of a random cow. Make use of Markov's bound.

(b) Suppose there are 400 cows in the herd. Show that the bound of part (a) is best possible by giving an example set of temperatures for the cows so that the average herd temperature is 85, and with probability 3/4, a randomly chosen cow will have a high enough temperature to survive.

19.3 Chebyshev's Theorem

There's a really good trick for getting more mileage out of Markov's Theorem: instead of applying it to the variable, R, apply it to some function of R. One useful choice of functions to use turns out to be taking a power of $|R|$.

In particular, since $|R|^\alpha$ is nonnegative, Markov's inequality also applies to the event $[|R|^\alpha \geq x^\alpha]$. But this event is equivalent to the event $[|R| \geq x]$, so we have:

Lemma 19.3.1. *For any random variable R, $\alpha \in \mathbb{R}^+$, and $x > 0$,*

$$\Pr\{|R| \geq x\} \leq \frac{\mathrm{E}[|R|^\alpha]}{x^\alpha}.$$

Rephrasing (19.3.1) in terms of the random variable, $|R - \mathrm{E}[R]|$, that measures R's deviation from its mean, we get

$$\Pr\{|R - \mathrm{E}[R]| \geq x\} \leq \frac{\mathrm{E}[(R - \mathrm{E}[R])^\alpha]}{x^\alpha}. \tag{19.3}$$

The case when $\alpha = 2$ is turns out to be so important that numerator of the right hand side of (19.3) has been given a name:

Definition 19.3.2. The *variance*, $\mathrm{Var}[R]$, of a random variable, R, is:

$$\mathrm{Var}[R] ::= \mathrm{E}\left[(R - \mathrm{E}[R])^2\right].$$

The restatement of (19.3) for $\alpha = 2$ is known as *Chebyshev's Theorem*.

Theorem 19.3.3 (Chebyshev)**.** *Let R be a random variable and $x \in \mathbb{R}^+$. Then*

$$\Pr\{|R - \mathrm{E}[R]| \geq x\} \leq \frac{\mathrm{Var}[R]}{x^2}.$$

The expression $\mathrm{E}\left[(R - \mathrm{E}[R])^2\right]$ for variance is a bit cryptic; the best approach is to work through it from the inside out. The innermost expression, $R - \mathrm{E}[R]$, is precisely the deviation of R above its mean. Squaring this, we obtain, $(R - \mathrm{E}[R])^2$. This is a random variable that is near 0 when R is close to the mean and is a large positive number when R deviates far above or below the mean. So if R is always close to the mean, then the variance will be small. If R is often far from the mean, then the variance will be large.

19.3.1 Variance in Two Gambling Games

The relevance of variance is apparent when we compare the following two gambling games.

Game A: We win \$2 with probability $2/3$ and lose \$1 with probability $1/3$.

Game B: We win \$1002 with probability $2/3$ and lose \$2001 with probability $1/3$.

Which game is better financially? We have the same probability, 2/3, of winning each game, but that does not tell the whole story. What about the expected return for each game? Let random variables A and B be the payoffs for the two games. For example, A is 2 with probability 2/3 and -1 with probability 1/3. We can compute the expected payoff for each game as follows:

$$\begin{aligned} \mathrm{E}[A] &= 2 \cdot \frac{2}{3} + (-1) \cdot \frac{1}{3} = 1, \\ \mathrm{E}[B] &= 1002 \cdot \frac{2}{3} + (-2001) \cdot \frac{1}{3} = 1. \end{aligned}$$

The expected payoff is the same for both games, but they are obviously very different! This difference is not apparent in their expected value, but is captured by variance. We can compute the $\mathrm{Var}[A]$ by working "from the inside out" as follows:

$$\begin{aligned} A - \mathrm{E}[A] &= \begin{cases} 1 & \text{with probability } \frac{2}{3} \\ -2 & \text{with probability } \frac{1}{3} \end{cases} \\ (A - \mathrm{E}[A])^2 &= \begin{cases} 1 & \text{with probability } \frac{2}{3} \\ 4 & \text{with probability } \frac{1}{3} \end{cases} \\ \mathrm{E}\left[(A - \mathrm{E}[A])^2\right] &= 1 \cdot \frac{2}{3} + 4 \cdot \frac{1}{3} \\ \mathrm{Var}[A] &= 2. \end{aligned}$$

Similarly, we have for $\mathrm{Var}[B]$:

$$\begin{aligned} B - \mathrm{E}[B] &= \begin{cases} 1001 & \text{with probability } \frac{2}{3} \\ -2002 & \text{with probability } \frac{1}{3} \end{cases} \\ (B - \mathrm{E}[B])^2 &= \begin{cases} 1,002,001 & \text{with probability } \frac{2}{3} \\ 4,008,004 & \text{with probability } \frac{1}{3} \end{cases} \\ \mathrm{E}\left[(B - \mathrm{E}[B])^2\right] &= 1,002,001 \cdot \frac{2}{3} + 4,008,004 \cdot \frac{1}{3} \\ \mathrm{Var}[B] &= 2,004,002. \end{aligned}$$

The variance of Game A is 2 and the variance of Game B is more than two million! Intuitively, this means that the payoff in Game A is usually close to the expected value of \$1, but the payoff in Game B can deviate very far from this expected value.

High variance is often associated with high risk. For example, in ten rounds of Game A, we expect to make \$10, but could conceivably lose \$10 instead. On the other hand, in ten rounds of game B, we also expect to make \$10, but could actually lose more than \$20,000!

19.3.2 Standard Deviation

Because of its definition in terms of the square of a random variable, the variance of a random variable may be very far from a typical deviation from the mean. For example, in Game B above, the deviation from the mean is 1001 in one outcome and -2002 in the other. But the variance is a whopping 2,004,002. From a dimensional analysis viewpoint, the "units" of variance are wrong: if the random variable is in dollars, then the expectation is also in dollars, but the variance is in square dollars. For this reason, people often describe random variables using standard deviation instead of variance.

Definition 19.3.4. The *standard deviation*, σ_R, of a random variable, R, is the square root of the variance:

$$\sigma_R ::= \sqrt{\text{Var}[R]} = \sqrt{\text{E}[(R - \text{E}[R])^2]}.$$

So the standard deviation is the square root of the mean of the square of the deviation, or the *root mean square* for short. It has the same units —dollars in our example —as the original random variable and as the mean. Intuitively, it measures the average deviation from the mean, since we can think of the square root on the outside as canceling the square on the inside.

Example 19.3.5. The standard deviation of the payoff in Game B is:

$$\sigma_B = \sqrt{\text{Var}[B]} = \sqrt{2,004,002} \approx 1416.$$

The random variable B actually deviates from the mean by either positive 1001 or negative 2002; therefore, the standard deviation of 1416 describes this situation reasonably well.

Intuitively, the standard deviation measures the "width" of the "main part" of the distribution graph, as illustrated in Figure 19.1.

It's useful to rephrase Chebyshev's Theorem in terms of standard deviation.

Corollary 19.3.6. *Let R be a random variable, and let c be a positive real number.*

$$\Pr\{|R - \text{E}[R]| \geq c\sigma_R\} \leq \frac{1}{c^2}.$$

Here we see explicitly how the "likely" values of R are clustered in an $O(\sigma_R)$-sized region around $\text{E}[R]$, confirming that the standard deviation measures how spread out the distribution of R is around its mean.

Proof. Substituting $x = c\sigma_R$ in Chebyshev's Theorem gives:

$$\Pr\{|R - \text{E}[R]| \geq c\sigma_R\} \leq \frac{\text{Var}[R]}{(c\sigma_R)^2} = \frac{\sigma_R^2}{(c\sigma_R)^2} = \frac{1}{c^2}.$$

■

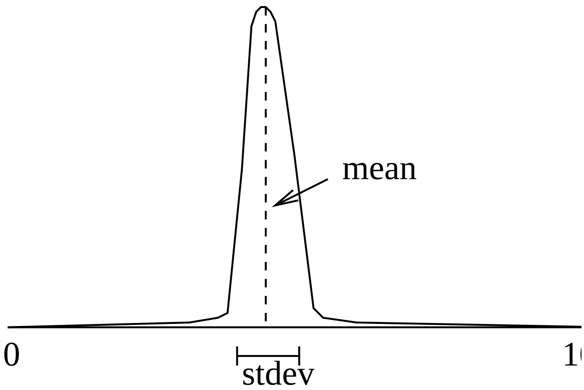


Figure 19.1: The standard deviation of a distribution indicates how wide the "main part" of it is.

The IQ Example

Suppose that, in addition to the national average IQ being 100, we also know the standard deviation of IQ's is 10. How rare is an IQ of 300 or more?

Let the random variable, R, be the IQ of a random person. So we are supposing that $\mathrm{E}[R] = 100$, $\sigma_R = 10$, and R is nonnegative. We want to compute $\Pr\{R \geq 300\}$.

We have already seen that Markov's Theorem 19.2.1 gives a coarse bound, namely,

$$\Pr\{R \geq 300\} \leq \frac{1}{3}.$$

Now we apply Chebyshev's Theorem to the same problem:

$$\Pr\{R \geq 300\} = \Pr\{|R - 100| \geq 200\} \leq \frac{\mathrm{Var}[R]}{200^2} = \frac{10^2}{200^2} = \frac{1}{400}.$$

So Chebyshev's Theorem implies that at most one person in four hundred has an IQ of 300 or more. We have gotten a much tighter bound using the additional information, namely the variance of R, than we could get knowing only the expectation.

19.4 Properties of Variance

The definition of variance of R as $\mathrm{E}\left[(R - \mathrm{E}[R])^2\right]$ may seem rather arbitrary. A direct measure of average deviation would be $\mathrm{E}[\,|R - \mathrm{E}[R]|\,]$. But the direct measure doesn't have the many useful properties that variance has, which is what this section is about.

19.4.1 A Formula for Variance

Applying linearity of expectation to the formula for variance yields a convenient alternative formula.

Lemma 19.4.1.

$$\text{Var}[R] = \text{E}\left[R^2\right] - \text{E}^2[R],$$

for any random variable, R.

Here we use the notation $\text{E}^2[R]$ as shorthand for $(\text{E}[R])^2$.

Proof. Let $\mu = \text{E}[R]$. Then

$$\begin{aligned}
\text{Var}[R] &= \text{E}\left[(R - \text{E}[R])^2\right] && \text{(Def 19.3.2 of variance)}\\
&= \text{E}\left[(R-\mu)^2\right] && \text{(def of } \mu)\\
&= \text{E}\left[R^2 - 2\mu R + \mu^2\right] \\
&= \text{E}\left[R^2\right] - 2\mu\,\text{E}[R] + \mu^2 && \text{(linearity of expectation)}\\
&= \text{E}\left[R^2\right] - 2\mu^2 + \mu^2 && \text{(def of } \mu)\\
&= \text{E}\left[R^2\right] - \mu^2 \\
&= \text{E}\left[R^2\right] - \text{E}^2[R]. && \text{(def of } \mu)
\end{aligned}$$

■

For example, if B is a Bernoulli variable where $p ::= \Pr\{B = 1\}$, then

Lemma 19.4.2.

$$\text{Var}[B] = p - p^2 = p(1-p). \tag{19.4}$$

Proof. By Lemma 18.3.3, $\text{E}[B] = p$. But since B only takes values 0 and 1, $B^2 = B$. So Lemma 19.4.2 follows immediately from Lemma 19.4.1. ■

19.4.2 Variance of Time to Failure

According to section **??**, the mean time to failure is $1/p$ for a process that fails during any given hour with probability p. What about the variance? That is, let C be the hour of the first failure, so $\Pr\{C = i\} = (1-p)^{i-1}p$. We'd like to find a formula for $\text{Var}[C]$.

By Lemma 19.4.1,

$$\text{Var}[C] = \text{E}\left[C^2\right] - (1/p)^2 \tag{19.5}$$

so all we need is a formula for $\text{E}\left[C^2\right]$:

$$\begin{aligned}
\text{E}\left[C^2\right] &::= \sum_{i \geq 1} i^2 (1-p)^{i-1} p \\
&= p \sum_{i \geq 1} i^2 x^{i-1} && \text{(where } x = 1-p). \qquad (19.6)
\end{aligned}$$

But (15.2) gives the generating function $x(1+x)/(1-x)^3$ for the nonnegative integer squares, and this implies that the generating function for the sum in (19.6) is $(1+x)/(1-x)^3$. So,

$$\begin{aligned} \mathrm{E}\left[C^2\right] &= p\,\frac{(1+x)}{(1-x)^3} && \text{(where } x = 1-p\text{)} \\ &= p\,\frac{2+p}{p^3} \\ &= \frac{1-p}{p^2} + \frac{1}{p^2}, \end{aligned} \tag{19.7}$$

Combining (19.5) and (19.7) gives a simple answer:

$$\mathrm{Var}\left[C\right] = \frac{1-p}{p^2}. \tag{19.8}$$

It's great to be able to apply generating function expertise to knock off equation (19.8) mechanically just from the definition of variance, but there's a more elementary, and memorable, alternative. In section **??** we used conditional expectation to find the mean time to failure, and a similar approach works for the variance. Namely, the expected value of C^2 is the probability, p, of failure in the first hour times 1^2, plus $(1-p)$ times the expected value of $(C+1)^2$. So

$$\begin{aligned} \mathrm{E}\left[C^2\right] &= p \cdot 1^2 + (1-p)\,\mathrm{E}\left[(C+1)^2\right] \\ &= p + (1-p)\left(\mathrm{E}\left[C^2\right] + \frac{2}{p} + 1\right), \end{aligned}$$

which directly simplifies to (19.7).

19.4.3 Dealing with Constants

It helps to know how to calculate the variance of $aR + b$:

Theorem 19.4.3. *Let R be a random variable, and a a constant. Then*

$$\mathrm{Var}\left[aR\right] = a^2\,\mathrm{Var}\left[R\right]. \tag{19.9}$$

Proof. Beginning with the definition of variance and repeatedly applying linearity of expectation, we have:

$$\begin{aligned} \mathrm{Var}\left[aR\right] &::= \mathrm{E}\left[(aR - \mathrm{E}\left[aR\right])^2\right] \\ &= \mathrm{E}\left[(aR)^2 - 2aR\,\mathrm{E}\left[aR\right] + \mathrm{E}^2\left[aR\right]\right] \\ &= \mathrm{E}\left[(aR)^2\right] - \mathrm{E}\left[2aR\,\mathrm{E}\left[aR\right]\right] + \mathrm{E}^2\left[aR\right] \\ &= a^2\,\mathrm{E}\left[R^2\right] - 2\,\mathrm{E}\left[aR\right]\mathrm{E}\left[aR\right] + \mathrm{E}^2\left[aR\right] \\ &= a^2\,\mathrm{E}\left[R^2\right] - a^2\mathrm{E}^2\left[R\right] \\ &= a^2\left(\mathrm{E}\left[R^2\right] - \mathrm{E}^2\left[R\right]\right) \\ &= a^2\,\mathrm{Var}\left[R\right] && \text{(by Lemma 19.4.1)} \end{aligned}$$

■

It's even simpler to prove that adding a constant does not change the variance, as the reader can verify:

Theorem 19.4.4. *Let R be a random variable, and b a constant. Then*

$$\text{Var}[R+b] = \text{Var}[R]. \tag{19.10}$$

Recalling that the standard deviation is the square root of variance, this implies that the standard deviation of $aR+b$ is simply $|a|$ times the standard deviation of R:

Corollary 19.4.5.

$$\sigma_{aR+b} = |a|\,\sigma_R.$$

19.4.4 Variance of a Sum

In general, the variance of a sum is not equal to the sum of the variances, but variances do add for *independent* variables. In fact, *mutual* independence is not necessary: *pairwise* independence will do. This is useful to know because there are some important situations involving variables that are pairwise independent but not mutually independent.

Theorem 19.4.6. *If R_1 and R_2 are independent random variables, then*

$$\text{Var}[R_1+R_2] = \text{Var}[R_1] + \text{Var}[R_2]. \tag{19.11}$$

Proof. We may assume that $\text{E}[R_i] = 0$ for $i = 1, 2$, since we could always replace R_i by $R_i - \text{E}[R_i]$ in equation (19.11). This substitution preserves the independence of the variables, and by Theorem 19.4.4, does not change the variances.

Now by Lemma 19.4.1, $\text{Var}[R_i] = \text{E}\left[R_i^2\right]$ and $\text{Var}[R_1+R_2] = \text{E}\left[(R_1+R_2)^2\right]$, so we need only prove

$$\text{E}\left[(R_1+R_2)^2\right] = \text{E}\left[R_1^2\right] + \text{E}\left[R_2^2\right]. \tag{19.12}$$

But (19.12) follows from linearity of expectation and the fact that

$$\text{E}[R_1R_2] = \text{E}[R_1]\,\text{E}[R_2] \tag{19.13}$$

since R_1 and R_2 are independent:

$$\begin{aligned}
\text{E}\left[(R_1+R_2)^2\right] &= \text{E}\left[R_1^2 + 2R_1R_2 + R_2^2\right] \\
&= \text{E}\left[R_1^2\right] + 2\,\text{E}[R_1R_2] + \text{E}\left[R_2^2\right] \\
&= \text{E}\left[R_1^2\right] + 2\,\text{E}[R_1]\,\text{E}[R_2] + \text{E}\left[R_2^2\right] && \text{(by (19.13))} \\
&= \text{E}\left[R_1^2\right] + 2\cdot 0\cdot 0 + \text{E}\left[R_2^2\right] \\
&= \text{E}\left[R_1^2\right] + \text{E}\left[R_2^2\right]
\end{aligned}$$

■

An independence condition is necessary. If we ignored independence, then we would conclude that $\text{Var}[R + R] = \text{Var}[R] + \text{Var}[R]$. However, by Theorem 19.4.3, the left side is equal to $4\,\text{Var}[R]$, whereas the right side is $2\,\text{Var}[R]$. This implies that $\text{Var}[R] = 0$, which, by the Lemma above, essentially only holds if R is constant.

The proof of Theorem 19.4.6 carries over straightforwardly to the sum of any finite number of variables. So we have:

Theorem 19.4.7. *[Pairwise Independent Additivity of Variance] If $R_1, R_2, \ldots, R_n$ are* pairwise *independent random variables, then*

$$\text{Var}[R_1 + R_2 + \cdots + R_n] = \text{Var}[R_1] + \text{Var}[R_2] + \cdots + \text{Var}[R_n]. \tag{19.14}$$

Now we have a simple way of computing the variance of a variable, J, that has an (n, p)-binomial distribution. We know that $J = \sum_{k=1}^{n} I_k$ where the I_k are mutually independent indicator variables with $\Pr\{I_k = 1\} = p$. The variance of each I_k is $p(1-p)$ by Lemma 19.4.2, so by linearity of variance, we have

Lemma (Variance of the Binomial Distribution)**.** *If J has the (n, p)-binomial distribution, then*

$$\text{Var}[J] = n\,\text{Var}[I_k] = np(1-p). \tag{19.15}$$

19.4.5 Problems

Practice Problems

Problem 19.2.
A gambler plays 120 hands of draw poker, 60 hands of black jack, and 20 hands of stud poker per day. He wins a hand of draw poker with probability 1/6, a hand of black jack with probability 1/2, and a hand of stud poker with probability 1/5.

(a) What is the expected number of hands the gambler wins in a day?

(b) What would the Markov bound be on the probability that the gambler will win at least 108 hands on a given day?

(c) Assume the outcomes of the card games are pairwise independent. What is the variance in the number of hands won per day?

(d) What would the Chebyshev bound be on the probability that the gambler will win at least 108 hands on a given day? You may answer with a numerical expression that is not completely evaluated.

Class Problems

Problem 19.3.
The hat-check staff has had a long day serving at a party, and at the end of the party they simply return people's hats at random. Assume that n people checked hats at the party.

Let X_i be the indicator variable for the ith person getting their own hat back. Let S_n be the total number of people who get their own hat back.

(a) What is the expected number of people who get their own hat back?

(b) Write a simple formula for $\mathrm{E}[X_i X_j]$ for $i \neq j$. *Hint:* What is $\Pr\{X_j = 1 \mid X_i = 1\}$?

(c) Show that $\mathrm{E}\left[S_n^2\right] = 2$. *Hint:* $X_i^2 = X_i$.

(d) What is the variance of S_n?

(e) Use the Chebyshev bound to show that the probability that 11 or more people get their own hat back is at most 0.01.

Problem 19.4.
For any random variable, R, with mean, μ, and standard deviation, σ, the Chebyshev Bound says that for any real number $x > 0$,

$$\Pr\{|R - \mu| \geq x\} \leq \left(\frac{\sigma}{x}\right)^2.$$

Show that for any real number, μ, and real numbers $x \geq \sigma > 0$, there is an R for which the Chebyshev Bound is tight, that is,

$$\Pr\{|R| \geq x\} = \left(\frac{\sigma}{x}\right)^2. \tag{19.16}$$

Hint: First assume $\mu = 0$ and let R only take values $0, -x$, and x.

Problem 19.5.
Let R be a positive integer valued random variable such that

$$f_R(n) = \frac{1}{cn^3},$$

where

$$c ::= \sum_{n=1}^{\infty} \frac{1}{n^3}.$$

(a) Prove that $\mathrm{E}[R]$ is finite.

(b) Prove that $\mathrm{Var}[R]$ is infinite.

Homework Problems

Problem 19.6.
There is a "one-sided" version of Chebyshev's bound for deviation above the mean:

Lemma (One-sided Chebyshev bound)**.**

$$\Pr\{R - \mathrm{E}[R] \geq x\} \leq \frac{\mathrm{Var}[R]}{x^2 + \mathrm{Var}[R]}.$$

Hint: Let $S_a ::= (R - \mathrm{E}[R] + a)^2$, for $0 \leq a \in \mathbb{R}$. So $R - \mathrm{E}[R] \geq x$ implies $S_a \geq (x+a)^2$. Apply Markov's bound to $\Pr\{S_a \geq (x+a)^2\}$. Choose a to minimize this last bound.

Problem 19.7.
A man has a set of n keys, one of which fits the door to his apartment. He tries the keys until he finds the correct one. Give the expectation and variance for the number of trials until success if

(a) he tries the keys at random (possibly repeating a key tried earlier)

(b) he chooses keys randomly from among those he has not yet tried.

19.5 Estimation by Random Sampling

Polling again

Suppose we had wanted an advance estimate of the fraction of the Massachusetts voters who favored Scott Brown over everyone else in the recent Democratic primary election to fill Senator Edward Kennedy's seat.

Let p be this unknown fraction, and let's suppose we have some random process —say throwing darts at voter registration lists —which will select each voter with equal probability. We can define a Bernoulli variable, K, by the rule that $K = 1$ if the random voter most prefers Brown, and $K = 0$ otherwise.

Now to estimate p, we take a large number, n, of random choices of voters[1] and count the fraction who favor Brown. That is, we define variables $K_1, K_2, \dots,$ where K_i is interpreted to be the indicator variable for the event that the ith chosen voter prefers Brown. Since our choices are made independently, the K_i's are independent. So formally, we model our estimation process by simply assuming we have mutually independent Bernoulli variables $K_1, K_2, \dots,$ each with the same

[1]We're choosing a random voter n times *with replacement*. That is, we don't remove a chosen voter from the set of voters eligible to be chosen later; so we might choose the same voter more than once in n tries! We would get a slightly better estimate if we required n *different* people to be chosen, but doing so complicates both the selection process and its analysis, with little gain in accuracy.

probability, p, of being equal to 1. Now let S_n be their sum, that is,

$$S_n ::= \sum_{i=1}^{n} K_i. \tag{19.17}$$

So S_n has the binomial distribution with parameter n, which we can choose, and unknown parameter p.

The variable S_n/n describes the fraction of voters we will sample who favor Scott Brown. Most people intuitively expect this sample fraction to give a useful approximation to the unknown fraction, p —and they would be right. So we will use the sample value, S_n/n, as our *statistical estimate* of p and use the Pairwise Independent Sampling Theorem 19.5.1 to work out how good an estinate this is.

19.5.1 Sampling

Suppose we want our estimate to be within 0.04 of the Brown favoring fraction, p, at least 95% of the time. This means we want

$$\Pr\left\{\left|\frac{S_n}{n} - p\right| \leq 0.04\right\} \geq 0.95\ . \tag{19.18}$$

So we better determine the number, n, of times we must poll voters so that inequality (19.18) will hold.

Now S_n is binomially distributed, so from (19.15) we have

$$\text{Var}\left[S_n\right] = n(p(1-p)) \leq n \cdot \frac{1}{4} = \frac{n}{4}$$

The bound of $1/4$ follows from the fact that $p(1-p)$ is maximized when $p = 1-p$, that is, when $p = 1/2$ (check this yourself!).

Next, we bound the variance of S_n/n:

$$\begin{aligned}
\text{Var}\left[\frac{S_n}{n}\right] &= \left(\frac{1}{n}\right)^2 \text{Var}\left[S_n\right] && \text{(by (19.9))} \\
&\leq \left(\frac{1}{n}\right)^2 \frac{n}{4} && \text{(by (19.5.1))} \\
&= \frac{1}{4n} && (19.19)
\end{aligned}$$

Now from Chebyshev and (19.19) we have:

$$\Pr\left\{\left|\frac{S_n}{n} - p\right| \geq 0.04\right\} \leq \frac{\text{Var}\left[S_n/n\right]}{(0.04)^2} = \frac{1}{4n(0.04)^2} = \frac{156.25}{n} \tag{19.20}$$

To make our our estimate with 95% confidence, we want the righthand side of (19.20) to be at most $1/20$. So we choose n so that

$$\frac{156.25}{n} \leq \frac{1}{20},$$

that is,

$$n \geq 3,125.$$

A more exact calculation of the tail of this binomial distribution shows that the above sample size is about four times larger than necessary, but it is still a feasible size to sample. The fact that the sample size derived using Chebyshev's Theorem was unduly pessimistic should not be surprising. After all, in applying the Chebyshev Theorem, we only used the variance of S_n. It makes sense that more detailed information about the distribution leads to better bounds. But working through this example using only the variance has the virtue of illustrating an approach to estimation that is applicable to arbitrary random variables, not just binomial variables.

19.5.2 Matching Birthdays

There are important cases where the relevant distributions are not binomial because the mutual independence properties of the voter preference example do not hold. In these cases, estimation methods based on the Chebyshev bound may be the best approach. Birthday Matching is an example. We already saw in Section 16.5 that in a class of 85 students it is virtually certain that two or more students will have the same birthday. This suggests that quite a few pairs of students are likely to have the same birthday. How many?

So as before, suppose there are n students and d days in the year, and let D be the number of pairs of students with the same birthday. Now it will be easy to calculate the expected number of pairs of students with matching birthdays. Then we can take the same approach as we did in estimating voter preferences to get an estimate of the probability of getting a number of pairs close to the expected number.

Unlike the situation with voter preferences, having matching birthdays for different pairs of students are not mutually independent events, but the matchings are *pairwise independent*, as explained in Section 16.5. as we did for voter preference. Namely, let $B_1, B_2, \ldots, B_n$ be the birthdays of n independently chosen people, and let $E_{i,j}$ be the indicator variable for the event that the ith and jth people chosen have the same birthdays, that is, the event $[B_i = B_j]$. So our probability model, the B_i's are mutually independent variables, the $E_{i,j}$'s are pairwise independent. Also, the expectations of $E_{i,j}$ for $i \neq j$ equals the probability that $B_i = B_j$, namely, $1/d$.

Now, D, the number of matching pairs of birthdays among the n choices is simply the sum of the $E_{i,j}$'s:

$$D ::= \sum_{1 \leq i < j \leq n} E_{i,j}. \tag{19.21}$$

So by linearity of expectation

$$\mathrm{E}[D] = \mathrm{E}\left[\sum_{1 \le i < j \le n} E_{i,j}\right] = \sum_{1 \le i < j \le n} \mathrm{E}[E_{i,j}] = \binom{n}{2} \cdot \frac{1}{d}.$$

Similarly,

$$\begin{aligned}
\mathrm{Var}[D] &= \mathrm{Var}\left[\sum_{1 \le i < j \le n} E_{i,j}\right] \\
&= \sum_{1 \le i < j \le n} \mathrm{Var}[E_{i,j}] && \text{(by Theorem 19.4.7)} \\
&= \binom{n}{2} \cdot \frac{1}{d}\left(1 - \frac{1}{d}\right). && (by Lemma\ 19.4.2)
\end{aligned}$$

In particular, for a class of $n = 85$ students with $d = 365$ possible birthdays, we have $\mathrm{E}[D] \approx 9.7$ and $\mathrm{Var}[D] < 9.7(1 - 1/365) < 9.7$. So by Chebyshev's Theorem

$$\Pr\{|D - 9.7| \ge x\} < \frac{9.7}{x^2}.$$

Letting $x = 5$, we conclude that there is a better than 50% chance that in a class of 85 students, the number of pairs of students with the same birthday will be between 5 and 14.

19.5.3 Pairwise Independent Sampling

The reasoning we used above to analyze voter polling and matching birthdays is very similar. We summarize it in slightly more general form with a basic result we call the Pairwise Independent Sampling Theorem. In particular, we do not need to restrict ourselves to sums of zero-one valued variables, or to variables with the same distribution. For simplicity, we state the Theorem for pairwise independent variables with possibly different distributions but with the same mean and variance.

Theorem 19.5.1 (Pairwise Independent Sampling)**.** *Let $G_1, \ldots, G_n$ be pairwise independent variables with the same mean, μ, and deviation, σ. Define*

$$S_n ::= \sum_{i=1}^{n} G_i. \tag{19.22}$$

Then

$$\Pr\left\{\left|\frac{S_n}{n} - \mu\right| \ge x\right\} \le \frac{1}{n}\left(\frac{\sigma}{x}\right)^2.$$

Proof. We observe first that the expectation of S_n/n is μ:

$$\begin{aligned}
\mathrm{E}\left[\frac{S_n}{n}\right] &= \mathrm{E}\left[\frac{\sum_{i=1}^n G_i}{n}\right] && \text{(def of } S_n\text{)}\\
&= \frac{\sum_{i=1}^n \mathrm{E}\left[G_i\right]}{n} && \text{(linearity of expectation)}\\
&= \frac{\sum_{i=1}^n \mu}{n}\\
&= \frac{n\mu}{n} = \mu.
\end{aligned}$$

The second important property of S_n/n is that its variance is the variance of G_i divided by n:

$$\begin{aligned}
\mathrm{Var}\left[\frac{S_n}{n}\right] &= \left(\frac{1}{n}\right)^2 \mathrm{Var}\left[S_n\right] && \text{(by (19.9))}\\
&= \frac{1}{n^2}\mathrm{Var}\left[\sum_{i=1}^n G_i\right] && \text{(def of } S_n\text{)}\\
&= \frac{1}{n^2}\sum_{i=1}^n \mathrm{Var}\left[G_i\right] && \text{(pairwise independent additivity)}\\
&= \frac{1}{n^2}\cdot n\sigma^2 = \frac{\sigma^2}{n}. && (19.23)
\end{aligned}$$

This is enough to apply Chebyshev's Theorem and conclude:

$$\begin{aligned}
\Pr\left\{\left|\frac{S_n}{n}-\mu\right| \geq x\right\} &\leq \frac{\mathrm{Var}\left[S_n/n\right]}{x^2}. && \text{(Chebyshev's bound)}\\
&= \frac{\sigma^2/n}{x^2} && \text{(by (19.23))}\\
&= \frac{1}{n}\left(\frac{\sigma}{x}\right)^2.
\end{aligned}$$

■

The Pairwise Independent Sampling Theorem provides a precise general statement about how the average of independent samples of a random variable approaches the mean. In particular, it proves what is known as the Law of Large Numbers[2] : by choosing a large enough sample size, we can get arbitrarily accurate estimates of the mean with confidence arbitrarily close to 100%.

Corollary 19.5.2. *[Weak Law of Large Numbers] Let $G_1, \ldots, G_n$ be pairwise independent variables with the same mean, μ, and the same finite deviation, and let*

$$S_n ::= \frac{\sum_{i=1}^n G_i}{n}.$$

[2]This is the *Weak* Law of Large Numbers. As you might suppose, there is also a Strong Law, but it's outside the scope of this text.

Then for every $\epsilon > 0$,

$$\lim_{n \to \infty} \Pr\{|S_n - \mu| \le \epsilon\} = 1.$$

19.6 Confidence versus Probability

So Chebyshev's Bound implies that sampling 3,125 voters will yield a fraction that, 95% of the time, is within 0.04 of the actual fraction of the voting population who prefer Brown.

Notice that the actual size of the voting population was never considered because *it did not matter*. People who have not studied probability theory often insist that the population size should matter. But our analysis shows that polling a little over 3000 people people is always sufficient, whether there are ten thousand, or million, or billion ... voters. You should think about an intuitive explanation that might persuade someone who thinks population size matters.

Now suppose a pollster actually takes a sample of 3,125 random voters to estimate the fraction of voters who prefer Brown, and the pollster finds that 1250 of them prefer Brown. It's tempting, **but sloppy**, to say that this means:

False Claim. *With probability 0.95, the fraction, p, of voters who prefer Brown is* $1250/3125 \pm 0.04$. *Since* $1250/3125 - 0.04 > 1/3$, *there is a 95% chance that more than a third of the voters prefer Brown to all other candidates.*

What's objectionable about this statement is that it talks about the probability or "chance" that a real world fact is true, namely that the actual fraction, p, of voters favoring Brown is more than 1/3. But p is what it is, and it simply makes no sense to talk about the probability that it is something else. For example, suppose p is actually 0.3; then it's nonsense to ask about the probability that it is within 0.04 of 1250/3125 —it simply isn't.

This example of voter preference is typical: we want to estimate a fixed, unknown real-world quantity. But *being unknown does not make this quantity a random variable*, so it makes no sense to talk about the probability that it has some property.

A more careful summary of what we have accomplished goes this way:

> We have described a probabilistic procedure for estimating the value of the actual fraction, p. The probability that *our estimation procedure* will yield a value within 0.04 of p is 0.95.

This is a bit of a mouthful, so special phrasing closer to the sloppy language is commonly used. The pollster would describe his conclusion by saying that

> At the 95% *confidence level*, the fraction of voters who prefer Brown is $1250/3125 \pm 0.04$.

So confidence levels refer to the results of estimation procedures for real-world quantities. The phrase "confidence level" should be heard as a reminder that some statistical procedure was used to obtain an estimate, and in judging the credibility of the estimate, it may be important to learn just what this procedure was.

19.6.1 Problems

Practice Problems

Problem 19.8.
You work for the president and you want to estimate the fraction p of voters in the entire nation that will prefer him in the upcoming elections. You do this by random sampling. Specifically, you select n voters independently and randomly, ask them who they are going to vote for, and use the fraction P of those that say they will vote for the President as an estimate for p.

(a) Our theorems about sampling and distributions allow us to calculate how confident we can be that the random variable, P, takes a value near the constant, p. This calculation uses some facts about voters and the way they are chosen. Which of the following facts are true?

1. Given a particular voter, the probability of that voter preferring the President is p.
2. Given a particular voter, the probability of that voter preferring the President is 1 or 0.
3. The probability that some voter is chosen more than once in the sequence goes to zero as n increases.
4. All voters are equally likely to be selected as the third in our sequence of n choices of voters (assuming $n \geq 3$).
5. The probability that the second voter chosen will favor the President, given that the first voter chosen prefers the President, is greater than p.
6. The probability that the second voter chosen will favor the President, given that the second voter chosen is from the same state as the first, may not equal p.

(b) Suppose that according to your calculations, the following is true about your polling:

$$\Pr\{|P - p| \leq 0.04\} \geq 0.95.$$

You do the asking, you count how many said they will vote for the President, you divide by n, and find the fraction is 0.53. You call the President, and ...what do you say?

1. Mr. President, $p = 0.53$!
2. Mr. President, with probability at least 95 percent, p is within 0.04 of 0.53.
3. Mr. President, either p is within 0.04 of 0.53 or something very strange (5-in-100) has happened.
4. Mr. President, we can be 95% confident that p is within 0.04 of 0.53.

Class Problems

Problem 19.9.
A recent Gallup poll found that 35% of the adult population of the United States believes that the theory of evolution is "well-supported by the evidence." Gallup polled 1928 Americans selected uniformly and independently at random. Of these, 675 asserted belief in evolution, leading to Gallup's estimate that the fraction of Americans who believe in evolution is $675/1928 \approx 0.350$. Gallup claims a margin of error of 3 percentage points, that is, he claims to be confident that his estimate is within 0.03 of the actual percentage.

(a) What is the largest variance an indicator variable can have?

(b) Use the Pairwise Independent Sampling Theorem to determine a confidence level with which Gallup can make his claim.

(c) Gallup actually claims greater than 99% confidence in his estimate. How might he have arrived at this conclusion? (Just explain what quantity he could calculate; you do not need to carry out a calculation.)

(d) Accepting the accuracy of all of Gallup's polling data and calculations, can you conclude that there is a high probability that the number of adult Americans who believe in evolution is 35 ± 3 percent?

Problem 19.10.
Suppose there are n students and d days in the year, and let D be the number of pairs of students with the same birthday. Let $B_1, B_2, \ldots, B_n$ be the birthdays of n independently chosen people, and let $E_{i,j}$ be the indicator variable for the event $[B_i = B_j]$.

(a) What are $\mathrm{E}[E_{i,j}]$ and $\mathrm{Var}[E_{i,j}]$?

(b) What is $\mathrm{E}[D]$?

(c) What is $\mathrm{Var}[D]$?

(d) In a 6.01 class of 500 students, the youngest student was born in 1995 and the oldest in 1975. Let S be the number of students in the class who were born on exactly the same day. What is the probability that $4 \leq S \leq 32$? (For simplicity, assume that the distribution of birthdays is uniform over the 7305 days in the two decade interval from 1975 to 1995.)

Problem 19.11.
A defendent in traffic court is trying to beat a speeding ticket on the grounds that —since virtually everybody speeds on the turnpike —the police have unconstitutional discretion in giving tickets to anyone they choose. (By the way, we don't recommend this defense :-) .)

To support his argument, the defendent arranged to get a random sample of trips by 3,125 cars on the turnpike and found that 94% of them broke the speed limit at some point during their trip. He says that as a consequence of sampling theory (in particular, the Pairwise Independent Sampling Theorem), the court can be 95% confident that the actual percentage of all cars that were speeding is $94 \pm 4\%$.

The judge observes that the actual number of car trips on the turnpike was never considered in making this estimate. He is skeptical that, whether there were a thousand, a million, or 100,000,000 car trips on the turnpike, sampling only 3,125 is sufficient to be so confident.

Suppose you were were the defendent. How would you explain to the judge why the number of randomly selected cars that have to be checked for speeding *does not depend on the number of recorded trips*? Remember that judges are not trained to understand formulas, so you have to provide an intuitive, nonquantitative explanation.

Problem 19.12.
The proof of the Pairwise Independent Sampling Theorem 19.5.1 was given for a sequence $R_1, R_2, \ldots$ of pairwise independent random variables with the same mean and variance.

The theorem generalizes straighforwardly to sequences of pairwise independent random variables, possibly with *different* distributions, as long as all their variances are bounded by some constant.

Theorem (Generalized Pairwise Independent Sampling)**.** *Let $X_1, X_2, \ldots$ be a sequence of pairwise independent random variables such that* $\text{Var}[X_i] \leq b$ *for some $b \geq 0$ and all $i \geq 1$. Let*

$$\begin{aligned} A_n &::= \frac{X_1 + X_2 + \cdots + X_n}{n}, \\ \mu_n &::= \text{E}[A_n]. \end{aligned}$$

Then for every $\epsilon > 0$,

$$\Pr\{|A_n - \mu_n| > \epsilon\} \leq \frac{b}{\epsilon^2} \cdot \frac{1}{n}. \tag{19.24}$$

(a) Prove the Generalized Pairwise Independent Sampling Theorem.

(b) Conclude that the following holds:
Corollary (Generalized Weak Law of Large Numbers)**.** *For every $\epsilon > 0$,*

$$\lim_{n \to \infty} \Pr\{|A_n - \mu_n| \leq \epsilon\} = 1.$$

Problem 19.13.
An *International Journal of Epidemiology* has a policy that they will only publish

the results of a drug trial when there were enough patients in the drug trial to be sure that the conclusions about the drug's effectiveness hold at the 95% confidence level. The editors of the Journal reason that under this policy, their readership can be confident that at most 5% of the published studies will be mistaken.

Later, the editors are astonished and embarrassed to learn that *every one* of the 20 drug trial results they published during the year was wrong. This happened even though the editors and reviewers had carefully checked the submitted data, and every one of the trials was *properly performed and reported* in the published paper.

The editors thought the probability of this was negligible (namely, $(1/20)^{20} < 10^{-25}$). Explain what's wrong with their reasoning and how it could be that all 20 published studies were wrong.

Exam Problems

Problem 19.14.
Yesterday, the programmers at a local company wrote a large program. To estimate the fraction, b, of lines of code in this program that are buggy, the QA team will take a small sample of lines chosen randomly and independently (so it is possible, though unlikely, that the same line of code might be chosen more than once). For each line chosen, they can run tests that determine whether that line of code is buggy, after which they will use the fraction of buggy lines in their sample as their estimate of the fraction b.

The company statistician can use estimates of a binomial distribution to calculate a value, s, for a number of lines of code to sample which ensures that with 97% confidence, the fraction of buggy lines in the sample will be within 0.006 of the actual fraction, b, of buggy lines in the program.

Mathematically, the *program* is an actual outcome that already happened. The *sample* is a random variable defined by the process for randomly choosing s lines from the program. The justification for the statistician's confidence depends on some properties of the program and how the sample of s lines of code from the program are chosen. These properties are described in some of the statements below. Indicate which of these statements are true, and explain your answers.

1. The probability that the ninth line of code in the *program* is buggy is b.

2. The probability that the ninth line of code chosen for the *sample* is defective, is b.

3. All lines of code in the program are equally likely to be the third line chosen in the *sample*.

4. Given that the first line chosen for the *sample* is buggy, the probability that the second line chosen will also be buggy is greater than b.

5. Given that the last line in the *program* is buggy, the probability that the next-to-last line in the program will also be buggy is greater than b.

6. The expectation of the indicator variable for the last line in the *sample* being buggy is b.

7. Given that the first two lines of code selected in the *sample* are the same kind of statement —they might both be assignment statements, or both be conditional statements, or both loop statements,. . . —the probability that the first line is buggy may be greater than b.

8. There is zero probability that all the lines in the *sample* will be different.

19.7 The Chernoff Bound

Fussbook is a new social networking site oriented toward unpleasant people.

Like all major web services, Fussbook has a load balancing problem. Specifically, Fussbook receives 24,000 forum posts every 10 minutes. Each post is assigned to one of m computers for processing, and each computer works sequentially through its assigned tasks. Processing an average post takes a computer $1/4$ second. Some posts, such as pointless grammar critiques and snide witticisms, are easier. But the most protracted harangues require 1 full second.

Balancing the work load across the m computers is vital; if any computer is assigned more than 10 minutes of work in a 10-minute interval, then that computer is overloaded and system performance suffers. That would be bad, because Fussbook users are *not* a tolerant bunch.

An early idea was to assign each computer an alphabetic range of forum topics. ("That oughta work!", one programmer said.) But after the computer handling the "*pr*ivacy" and "*pr*eferred text editor" threads melted, the drawback of an ad hoc approach was clear: there are no guarantees.

If the length of every task were known in advance, then finding a balanced distribution would be a kind of "bin packing" problem. Such problems are hard to solve exactly, though approximation algorithms can come close. But in this case task lengths are not known in advance, which is typical for workload problems.

So the load balancing problem seems sort of hopeless, because there is no data available to guide decisions. Heck, we might as well assign tasks to computers at random!

As it turns out, random assignment not only balances load reasonably well, but also permits provable performance guarantees in place of "That oughta work!" assertions. In general, a randomized approach to a problem is worth considering when a deterministic solution is hard to compute or requires unavailable information.

Some arithmetic shows that Fussbook's traffic is sufficient to keep $m = 10$ computers running at 100% capacity with perfect load balancing. Surely, more than 10 servers are needed to cope with random fluctuations in task length and imperfect load balance. But how many is enough? 11? 15? 20? We'll answer that question with a new mathematical tool.

19.7.1 The Chernoff Bound

The Chernoff bound is a hammer that you can use to nail a great many problems. Roughly, the Chernoff bound says that certain random variables are very unlikely to significantly exceed their expectation. For example, if the expected load on a computer is just a bit below its capacity, then that computer is unlikely to be overloaded, provided the conditions of the Chernoff bound are satisfied.

More precisely, the Chernoff Bound says that *the sum of lots of little, independent random variables is unlikely to significantly exceed the mean*. The Markov and Chebychev bounds lead to the same kind of conclusion but typically provide much weaker conclusions.

Here is the theorem. The proof is at the end of the chapter.

Theorem 19.7.1 (Chernoff Bound)**.** *Let $T_1, \dots T_n$ be mutually independent random variables such that $0 \leq T_i \leq 1$ for all i. Let $T = T_1 + \cdots + T_n$. Then for all $c \geq 1$,*

$$\Pr\{T \geq c \operatorname{E}[T]\} \leq e^{-k \operatorname{E}[T]} \tag{19.25}$$

where $k = c \ln c - c + 1$.

The Chernoff bound applies only to distributions of sums of independent random variables that take on values in the interval $[0, 1]$. The binomial distribution is of course such a distribution, but are lots of other distributions because the Chernoff bound allows the variables in the sum to have differing, arbitrary, and even unknown distributions over the range $[0, 1]$. Furthermore, there is no direct dependence on the number of random variables in the sum or their expectations. In short, the Chernoff bound gives strong results for lots of problems based on little information —no wonder it is widely used!

A Simple Example

The Chernoff bound is pretty easy to apply, though the details can be daunting at first. Let's walk through a simple example to get the hang of it.

What are the odds that the number of heads that come up in 1000 independent tosses of a fair coin exceeds the expectation by 20% or more? Let T_i be an indicator variable for the event that the i-th coin is heads. Then the total number of heads is $T = T_1 + \cdots + T_{1000}$. The Chernoff bound requires that the random variables T_i be mututally independent and take on values in the range $[0, 1]$. Both conditions hold here. In fact, this example is similar to many applications of the Chernoff bound in that every T_i is *either* 0 or 1, since they're indicators.

The goal is to bound the probability that the number of heads exceeds its expectation by 20% or more; that is, to bound $\Pr\{T \geq c \operatorname{E}[T]\}$ where c = 1.2. To that end, we compute k as defined in the theorem:

$$k = c \ln c - c + 1 = 0.0187 \dots$$

Plugging this value into the Chernoff bound gives:

$$\begin{aligned}\Pr\{T \geq 1.2\,\mathrm{E}[T]\} &\leq e^{-k\,\mathrm{E}[T]} \\ &= e^{-(0.0187\ldots)\cdot 500} \\ &< 0.0000834\end{aligned}$$

So the probability of getting 20% or more extra heads on 1000 coins is less than 1 in 10,000.

The bound becomes much stronger as the number of coins increases, because the expected number of heads appears in the exponent of the upper bound. For example, the probability of getting at least 20% extra heads on a million coins is at most

$$e^{-(0.0187\ldots)\cdot 500000} < e^{-9392}$$

which is pretty darn small.

Alternatively, the bound also becomes stronger for larger deviations. For example, suppose we're interested in the odds of getting 30% or more extra heads in 1000 tosses, rather than 20%. In that case, $c = 1.3$ instead of 1.2. Consequently, the parameter k rises from 0.0187 to about 0.0410, which may seem insignificant. But because k appears in the exponent of the upper bound, the final probability decreases from around 1 in 10,000 to about 1 in a billion!

Pick-4

Pick-4 is a lottery game where you pick a 4-digit number between 0000 and 9999. If your number comes up in a random drawing, then you win. Your chance of winning is 1 in 10,000. And if 10 million people play, then the expected number of winners is 1000. The lottery operator's nightmare is that the number of winners is much greater; say, 2000 or more. What are the odds of that?

Let T_i be an indicator for the event that the i-th player wins. Then $T = T_1 + \cdots + T_n$ is the total number of winners. If we assume that the players' picks and the winning number are independent and uniform, then the indicators T_i are independent, as required by the Chernoff bound.

Now, 2000 winners would be twice the expected number. So we choose $c = 2$, compute $k = c\ln c - c + 1 = 0.386\ldots$, and plug these values into the Chernoff bound:

$$\begin{aligned}\Pr\{T \geq 2000\} &= \Pr\{T \geq 2\,\mathrm{E}[T]\} \\ &\leq e^{-k\,\mathrm{E}[T]} \\ &= e^{-(0.386\ldots)\cdot 1000} \\ &< e^{-386}\end{aligned}$$

So there is almost no chance that the lottery operator pays out double. In fact, the number of winners won't even be 10% higher than expected very often. To prove that, let $c = 1.1$, compute $k = c \ln c - c + 1 = 0.00484\ldots$, and plug in again:

$$\begin{aligned}\Pr\{T \geq 1.1\,\mathrm{E}[T]\} &\leq e^{-k\,\mathrm{E}[T]} \\ &= e^{-}0.00484\cdots * 1000 \\ &< 0.01\end{aligned}$$

So the Pick-4 lottery may be exciting for the players, but the lottery operator has little doubt about the outcome!

19.7.2 Randomized Load Balancing

Now let's return to Fussbook and its load balancing problem. Specifically, we need to determine how many machines suffice to ensure that no server is overloaded; that is, assigned to do more than 10 minutes of work in a 10-minute interval.

To begin, let's find the probability that the first server is overloaded. Let T_i be the number of seconds that the first server spends on the i-th task. So T_i is zero if the task is assigned to another machine, and otherwise T_i is the length of the task. Then $T = \sum T_i$ is the total length of tasks assigned to the server. We need to upper bound $\Pr\{T \geq 600\}$; that is, the probability that the first server is assigned more than 600 seconds (or, equivalently, 10 minutes) of work.

The Chernoff bound is applicable only if the T_i are mutually independent and take on values in the range $[0, 1]$. The first condition is satisfied if we assume that tasks lengths and assignments are independent. And the second condition is satisfied because processing even the most interminable harangue takes at most 1 second.

In all, there are 24,000 tasks each with an expected length of 1/4 second. Since tasks are assigned to computers at random, the expected load on the first server is:

$$\begin{aligned}\mathrm{E}[T] &= \frac{24,000 \text{ tasks} \cdot 1/4 \text{ second per task}}{m \text{ machines}} \\ &= 6000/m \text{ seconds}\end{aligned}$$

For example, if there are $m = 10$ machines, then the expected load on the first server is 600 seconds, which is 100% of its capacity.

Now we can use the Chernoff bound to upper bound the probability that the first server is overloaded:

$$\begin{aligned}\Pr\{T \geq 600\} &= \Pr\{T \geq c\,\mathrm{E}[T]\} \\ &\leq e^{-(c \ln c - c + 1)\cdot 6000/m}\end{aligned}$$

Equality holds on the first line when $c = m/10$, since $c\,\mathrm{E}[T] = (m/10)\cdot(6000/m) = 600$. The probability that *some* server is overloaded is at most m times the probability that the first server is overloaded:

$$\Pr\{\text{some server is overloaded}\} \leq m e^{-(c \ln c - c + 1)\cdot 6000/m}$$

Some values of this upper bound are tabulated below:

$$\begin{aligned}
m &= 11: \quad 0.784\ldots \\
m &= 12: \quad 0.000999\ldots \\
m &= 13: \quad 0.0000000760\ldots
\end{aligned}$$

These values suggest that a system with $m = 11$ machines might suffer immediate overload, $m = 12$ machines could fail in a few days, but $m = 13$ should be fine for a century or two!

19.7.3 Proof of the Chernoff Bound

The proof of the Chernoff bound is somewhat involved. Heck, even *Chernoff* didn't come up with it! His friend, Herman Rubin, showed him the argument. Thinking the bound not very significant, Chernoff did not credit Rubin in print. He felt pretty bad when it became famous!

For clarity, we'll go through the proof "top down"; that is, we'll use facts that are proved immediately afterward.

Proof. The key step is to exponentiate both sides of the inequality $T > c\,\mathrm{E}[T]$ and then apply the Markov bound.

$$\begin{aligned}
\Pr\{T \geq c\,\mathrm{E}[T]\} &= \Pr\left\{c^T \geq c^{c\,\mathrm{E}[T]}\right\} \\
&\leq \frac{\mathrm{E}\left[c^T\right]}{c^{c\,\mathrm{E}[T]}} && \text{(by Markov)} \\
&\leq \frac{e^{(c-1)\,\mathrm{E}[T]}}{c^{c\,\mathrm{E}[T]}} \\
&= e^{-(c\ln c - c + 1)\,\mathrm{E}[T]}
\end{aligned}$$

In the third step, the numerator is rewritten using the inequality

$$\mathrm{E}\left[c^T\right] \leq e^{(c-1)\,\mathrm{E}[T]}$$

which is proved below in Lemma 19.7.2. The final step is simplification. (Recall that c^c is equal to $e^{c\ln c}$.) ■

Algebra aside, there is a brilliant idea in this proof: in this context, exponentiating somehow supercharges the Markov bound. This is not true in general! One unfortunate side-effect is that we have to bound some nasty expectations involving exponentials in order to complete the proof. This is done in the two lemmas below, where variables take on values as in Theorem 19.7.1.

Lemma 19.7.2.

$$\mathrm{E}\left[c^T\right] \leq e^{(c-1)\,\mathrm{E}[T]}$$

Proof.

$$\begin{aligned}
\mathrm{E}\left[c^T\right] &= \mathrm{E}\left[c^{T_1+\cdots+T_n}\right] \\
&= \mathrm{E}\left[c^{T_1}\cdots c^{T_n}\right] \\
&= \mathrm{E}\left[c^{T_1}\right]\cdots \mathrm{E}\left[c^{T_n}\right] \\
&\leq e^{(c-1)\,\mathrm{E}[T_1]}\cdots e^{(c-1)\,\mathrm{E}[T_n]} \\
&= e^{(c-1)(\mathrm{E}[T_1]+\cdots+\mathrm{E}[T_n])} \\
&= e^{(c-1)\,\mathrm{E}[T_1+\cdots+T_n]} \\
&= e^{(c-1)\,\mathrm{E}[T]}
\end{aligned}$$

The first step uses the definition of T, and the second is just algegra. The third step uses the fact that the expectation of a product of independent random variables is the product of the expectations. (This is where the requirement that the T_i be independent is used.) Then we bound each term using the inquality

$$\mathrm{E}\left[c^{T_i}\right] \leq e^{(c-1)\,\mathrm{E}[T_i]}$$

which is proved in Lemma 19.7.3. The last steps are simplifications using algebra and linearity of expectation. ■

Lemma 19.7.3.

$$\mathrm{E}\left[c^{T_i}\right] \leq e^{(c-1)\,\mathrm{E}[T_i]}$$

Proof. All summations below range over values v taken on by the random variable T_i, which are all required to be in the interval $[0, 1]$.

$$\begin{aligned}
\mathrm{E}\left[c^{T_i}\right] &= \sum c^v \Pr\{T_i = v\} \\
&\leq \sum (1 + (c-1)v) \Pr\{T_i = v\} \\
&= \sum \Pr\{T_i = v\} + (c-1)v \Pr\{T_i = v\} \\
&= \sum \Pr\{T_i = v\} + \sum (c-1)v \Pr\{T_i = v\} \\
&= 1 + (c-1) \sum v \Pr\{T_i = v\} \\
&= 1 + (c-1)\,\mathrm{E}[T_i] \\
&\leq e^{(c-1)\,\mathrm{E}[T_i]}
\end{aligned}$$

The first step uses the definition of expectation. The second step relies on the inequality $c^v \leq 1 + (c-1)v$, which holds for all v in $[0, 1]$ and $c \geq 1$. This follows from the general principle that a convex function, namely c^v, is less than the linear function, $1 + (c-1)v$, between their points of intersection, namely $v = 0$ and 1. This inequality is why the variables T_i are restricted to the interval $[0, 1]$. We then multiply out inside the summation and split into two sums. The first sum adds the probabilities of all possible outcomes, so it is equal to 1. After pulling the constant $c - 1$ out of the second sum, we're left with the definition of $\mathrm{E}[T_i]$. The final step uses the standard inequality $1 + z \leq e^z$, which holds for all real z. ■

Index