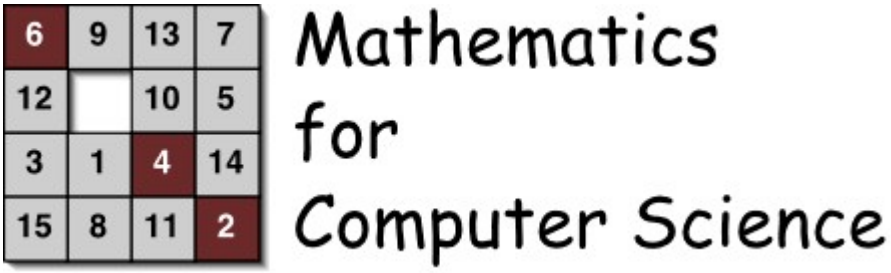
Massachusetts Institute of Technology
6.042J/18.062J, Spring '08: Mathematics for Computer Science
Prof. Albert R. Meyer
revised May 17, 2008, 783 minutes

6.042, Spring 2008



Lecture notes

Contents

Chapter 1

Proofs

1.1 What is a Proof?

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 based on consistency and plausibility. The best example is "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 proof in this vein, Descartes goes on to conclude that there is an infinitely beneficent God. This ain't Math.

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. In the next sections, we'll discuss these three ideas along with some basic ways of organizing proofs.

[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.

1.2 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 1.2.1. *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 1.2.2. *For every nonnegative integer, n, the value of $n^2 + n + 41$ is prime.*

Let's try some numerical experimentation to check this proposition. To begin, let $p(n) ::= n^2 + n + 41$. [2] Now $p(0) = 41$ which is prime. $p(1) = 43$ which is prime. $p(2) = 47$ which is prime. $p(3) = 53$ which is prime. $\ldots p(20) = 461$ 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 1.2.2 would be

$$\forall n \in \mathbb{N}.\ p(n) \text{ is prime.} \tag{1.1}$$

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

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

[2] 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.

In logical notation, Proposition 1.2.3 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 1.2.4. *$313(x^3 + y^3) = z^3$ has no solution when $x, y, z \in \mathbb{N}$.*

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

Proposition 1.2.5. *Every map can be colored with 4 colors so that adjacent[3] 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 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`). [4]

Proposition 1.2.6 (Goldbach)**.** *Every even integer greater than 2 is the sum of two primes.*

No one knows whether this proposition is true or false. This is the "Goldbach Conjecture," which dates back to 1742.

Problem 1.2.1. Show that no nonconstant polynomial can map all nonnegative integers into prime numbers.

1.3 The Axiomatic Method

The standard procedure for establishing truth in mathematics was invented by Euclid, a mathematician working in Alexadria, 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***.

[3] 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.

[4] The story of the four-color proof is told in a well-reviewed recent 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.

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, there are just a handful of axioms, called the axioms Zermel-Frankel with Choice (ZFC), which, together with a few logical deduction rules, appear to be sufficient to derive essentially all of mathematics.

1.3.1 Our Axioms

The ZFC axioms are important in studying and justifying the foundations of Mathematics. But for practical purposes, they are much too primitive— by one reckoning, proving 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 upfront about what you're assuming, and don't try to evade homework and exam problems by declaring everything an axiom!

1.3.2 Proofs in Practice

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.

The ZFC Axioms

We're *not* going to be working with the Axioms of Zermelo-Frankel Set Theory with Choice (ZFC) in this course, but we thought you might like to see them. Essentially all of mathematics can be derived from these axioms together with a few logical deduction rules.

Extensionality. Two sets are equal if they have the same members. In formal logical notation, this would be stated as:

$$(\forall z.\ (z \in x \longleftrightarrow z \in y)) \longrightarrow x = y.$$

Pairing. For any two sets x and y, there is a set, $\{x, y\}$, with x and y as its only elements.

Union. The union of a collection, z, of sets is also a set:

$$\exists u \forall x.\ (\exists y.\ x \in y \wedge y \in z) \longleftrightarrow x \in u.$$

(The symbol $\exists$ is read "there exists".)

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

Subset. Given any set, x, and any proposition $P(y)$, there is a set containing precisely those elements $y \in x$ for which $P(y)$ holds.

Power Set. All the subsets of a set form another set.

Replacement. The image of a set under a function is a set.

Foundation. For every non-empty set, x, there is a set $y \in x$ such that x and y have no elements in common. (This most technical of the axioms aims to capture the idea that sets are built up successively from simpler sets. In particular, this axiom prevents a set from being a member of itself.)

Choice. We can choose one element from each set in a collection of nonempty sets. More precisely, if x is a set, and every element of x is itself a set that is nonempty, then there is a "choice" function, g, such that $g(y) \in y$ for every $y \in x$.

1.4 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 standard methods for proving an implication.

1.4.1 Method #1

In order to prove that P implies Q:

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

Example

Theorem 1.4.1. *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.

1.4.2 Method #2 - Prove the Contrapositive

An implication ("P implies Q") is logically equivalent to its *contrapositive* "not Q 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 1.4.2. *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 exists 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. □

1.5 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 know for several thousand years:

> Two triangles have the same side lengths if and only if two side lengths and an angle are the same.

1.5.1 Method #1: Prove Each Statement Implies the Other

The statement "P if and only if Q" is equivalent to the two statements "P implies Q" and "Q implies P". So you can prove an "if and only if" 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 1.4.
3. Write, "Now, we show Q implies P." Again, do this by one of the methods in Section 1.4.

1.5.2 Method #2: Construct a Chain of Iffs

In order to prove that P is true if and only if 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 is generally more difficult 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:

$$\sqrt{\frac{(x_1-\mu)^2+(x_2-\mu)^2+\cdots+(x_n-\mu)^2}{n}} \tag{1.2}$$

where μ is the average of the values:

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

Theorem 1.5.1. *The standard deviation of a sequence of values $x_1, \ldots, x_n$ is zero if and only if 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 "if and only if" implications. The value of the standard deviation (1.2) is zero if and only if

$$(x_1 - \mu)^2 + (x_2 - \mu)^2 + \cdots + (x_n - \mu)^2 = 0,$$

since zero is the only number whose square root is zero. Every term in this equation is nonnegative, so this equation holds if and only if every term is actually 0. But this is true if and only if every value x_i is equal to the mean, μ. □

1.6 How to Write *Good* Proofs

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 6.042 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 are usually complementary; 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 a sequential 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 (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 which 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!

1.7 Propositional Formulas

It's really sort of amazing that people manage to communicate in the English language. Here are some typical sentences:

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 could sort of read this language, but you would miss all the subtleties and sometimes have trouble following along.

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.

1.7.1 Combining 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" 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 (F). 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**

This says that "P or Q" is true when P is true, Q is true, or *both* 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.

"Implies"

The least intuitive connecting word is "implies." Here is its truth table, with the lines labelled 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)

Let's experiment with this definition. For example, is the following proposition true or false?

"If Goldbach's Conjecture is true, then $x^2 \geq 0$ for every real number x."

Now, we told you before that no one knows whether Goldbach's Conjecture is true or false. But that doesn't prevent you from answering the question! This proposition has the form $P \longrightarrow Q$ where P is "Goldbach's Conjecture is true" and Q is "$x^2 \geq 0$ for every real number x". Since Q 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; 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 can understand 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*!

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!

"If and Only If"

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 if and only if 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:

"$x^2 - 4 \geq 0$ if and only if $|x| \geq 2$"

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.

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

1.7.2 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:

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

A truth table reveals that this complicated expression is logically equivalent to "A or B".

A	B	A or $((\text{not } A)$ 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)
```

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.

1.7.3 A 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" or "if P then Q"	$P \longrightarrow Q$
"P if and only if Q"	$P \longleftrightarrow Q$

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

$$(P \wedge \neg Q) \longrightarrow R$$

This symbolic language is helpful for writing complicated logical expressions compactly. But words such as "or" and "implies," whose meaning is easy to remember, serve just as well as the symbols such as $\vee$ and $\longrightarrow$. So we'll use this symbolic language sparingly, and we advise you to do the same.

1.7.4 Logically Equivalent Implications

Are these two sentences saying 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	(not Q) implies (not 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 "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 if and only if 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 if and only if I am hungry.

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

P	Q	(P implies Q) and (Q implies P)	Q if and only if P
T	T	T	T
T	F	F	F
F	T	F	F
F	F	T	T

SAT

A proposition is **satisfiable** if some setting of the variables makes the proposition true. For example, $P \wedge \neg Q$ is satisfiable because the expression is true when P is true and Q is false. On the other hand, $P \wedge \neg P$ 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?

$$(P \vee Q \vee R) \wedge (\neg P \vee \neg Q) \wedge (\neg P \vee \neg R) \wedge (\neg R \vee \neg Q)$$

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. For a proposition with just 30 variables, that's already over a billion!

Is there an *efficient* solution to SAT? In other words, is there some, possibly very ingenious, procedure that *quickly* determines 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. 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.

At present, though, researchers are completely stuck. No one has a good idea how to either solve SAT more efficiently or to prove that no efficient solution exists. This is the outstanding unanswered question in theoretical Computer Science.

1.8 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 of $P \longrightarrow Q$ is a proof of Q.

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

Rule.

$$\frac{P, \quad P \longrightarrow Q}{Q}$$

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.

$$\frac{P \longrightarrow Q, \quad Q \longrightarrow R}{P \longrightarrow R}$$

Rule.

$$\frac{\neg P \longrightarrow Q, \quad \neg Q}{P}$$

Rule.

$$\frac{\neg P \longrightarrow \neg Q}{Q \longrightarrow P}$$

On the other hand,

Rule.

$$\frac{\neg P \longrightarrow \neg Q}{P \longrightarrow Q}$$

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

Problem 1.8.1. Prove that a propositional inference rule is sound iff 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.

Chapter 2

Predicates & Sets

2.1 More Proof Techniques

2.1.1 Proof by Cases

In Week1 Notes we verified by truth table that the two expressions $A \vee (\neg A \wedge B)$ and $A \vee B$ were equivalent. Another way to prove this would be to reason *by cases*:

A is **T**. Then $A \vee$ anything will have truth value **T**. Since both expressions are of this form, in this case, both have the same truth value, namely, **T**.

A is **F**. Then $A \vee P$ will have the same truth value as P for any proposition, P. So the second expression has the same truth value as B. Similarly, the first expression has the same truth value as $\neg\mathbf{F} \wedge B$ 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.

Here's a slightly more interesting 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 the remaining 5 people, at least 3 have met x.
2. Among the remaining 5 people, at least 3 have not met x.

We first argue that at least one of these two cases must hold.[2] We'll prove this itself by cases. Namely, let m be the number of the five remaining people that have met x and n the number that

[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 quite so simple.

have not met x. So $m + n = 5$. Now one case is that $m \geq 3$, in which case 1 holds. The other case is that $m < 3$, so $n = 5 - m > 5 - 3 = 2$, that is, $n \geq 3$ which means case 2 holds. So at least one of the cases 1 and 2 must hold.

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.

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. □

2.1.2 Proof by Contradiction

In a ***proof by contradiction*** or ***indirect proof***, you show that if a proposition were false, then some logical contradiction or absurdity would follow. Thus, the proposition must be true. So proof by contradiction would be described by the inference rule

Rule.

$$\frac{\neg P \longrightarrow \mathbf{F}}{P}$$

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.

2.1.3 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 2.1.1. *$\sqrt{2}$ 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 a/b in *lowest terms*.

Squaring both sides gives $2 = a^2/b^2$ and so $2b^2 = a^2$. This implies that a is even; that is, a is a multiple of 2. Therefore, a^2 must be a multiple of 4. Because of the equality $2b^2 = a^2$, we know $2b^2$ must also be a multiple of 4. This implies that b^2 is even and so b must be even. But since a and b are both even, the fraction a/b is not in lowest terms.

This is a contradiction. Therefore, $\sqrt{2}$ must be irrational. □

2.2 Predicates

A ***predicate*** is a proposition whose truth depends on the value of one or more variables. For example,

"n is a perfect square"

is a predicate whose truth depends on the value of n. The predicate is true for $n = 4$ since 4 is a perfect square, but false for $n = 5$ since 5 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 $n^2 + 1$, then $p(n)$ is a *numerical quantity*. Don't confuse these two!

2.2.1 Quantifying a Predicate

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

"$x^2 \geq 0$"

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

"$5x^2 - 7 = 0$"

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, $x^2 \geq 0$.
$P(n)$ is true for every n.	$x^2 \geq 0$ for every x.

Sometimes True

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

All these sentences quantify how often the predicate is true. Specifically, an assertion that a predicate is always true is called a ***universal*** quantification, and an assertion that a predicate is sometimes true is an ***existential*** quantification. Sometimes the English sentences are unclear with respect to 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 quantification:

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

or maybe

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

In any case, notice that this quantified phrase appears inside a larger if-then statement. This is quite normal; quantified statements are themselves propositions and can be combined with and, or, implies, etc., just like any other proposition.

2.2.2 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, is true for all values of x in some set, D, one writes:

$$\forall x \in D.\ P(x)$$

The symbol $\forall$ is read "for all", so this whole expression is read "for all x in D, $P(x)$ is true". To say that a predicate $P(x)$ is true for at least one value of x in D, one writes:

$$\exists x \in D.\ P(x)$$

The backward-E is read "there exists". So this expression would be read, "There exists an x in D such that $P(x)$ is true." The symbols $\forall$ and $\exists$ are always followed by a variable 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 A 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:

$$(\forall x \in \text{Probs. Solves}(x)) \longrightarrow A,$$

or maybe

$$(\exists x \in \text{Probs. Solves}(x)) \longrightarrow A.$$

2.2.3 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:

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

2.2.4 Order of Quantifiers

Swapping the order of different kinds of quantifiers (existential or universal) usually changes the meaning of a proposition. For another 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 their 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 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

$$\forall n.\ n \in \text{Evens} \longrightarrow (\exists p \exists q.\ p \in \text{Primes} \wedge q \in \text{Primes} \wedge n = p + q).$$

2.2.5 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:

$$\neg \forall x.\ P(x) \qquad \text{is equivalent to} \qquad \exists x.\ \neg P(x).$$

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:

$$(\neg \exists x.\ Q(x)) \longleftrightarrow \forall x.\ \neg Q(x). \tag{2.1}$$

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

2.2.6 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 $P \wedge (Q \vee R)$ is equivalent to $(P \wedge Q) \vee (P \wedge R)$. This is the same as saying that

$$[P \wedge (Q \vee R)] \longleftrightarrow [(P \wedge Q) \vee (P \wedge R)]$$

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 (2.1).

Another useful example of a valid assertion is

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

We could prove this as follows:

Proof. Let D be the domain for the variables and P_0 be some binary predicate[3] on D. We need to show that if $\exists x \in D\ \forall y \in D.\ P_0(x, y)$ holds under this interpretation, then so does $\forall y \in D\ \exists x \in D.\ P_0(x, y)$.

So suppose $\exists x \in D\ \forall y \in D.\ P_0(x, y)$. So some element $x_0 \in D$ has the property that $P_0(x_0, y)$ is true for all $y \in D$. So for every $y \in D$, there is some $x \in D$, namely x_0, such that $P_0(x, y)$ is true. That is, $\forall y \in D \exists x \in D.\ P_0(x, y)$ holds, that is, $\forall y\ \exists x.\ P(x, y)$ holds under this interpretation, as required. □

On the other hand,

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

is *not* valid. We can prove this simply 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 interpetation like this which falsifies an assertion is called a *counter model* to the assertion.

2.3 Mathematical Data Types

We've been assuming that the concepts of sets, sequences and functions are already familiar ones, and we've mentioned them repeatedly. Now we'll do a quick review of the definitions.

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

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:

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

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

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

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[4]. 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 at least one set mentioned on almost every page in these notes.

2.3.1 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.

2.3.2 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).

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

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, $X \cup Y = \{1, 2, 3, 4\}$.
- 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 **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 $Y - X = \{4\}$.

Complement of a Set

Sometimes we are focussed 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 aid 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, $A \subseteq \overline{B}$.

Power Set

The collection 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 $\mathcal{P}(A)$.

2.3.3 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.
- The empty set is usually denoted $\emptyset$. Texts differ on notation for the empty sequence; in 6.042, 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 natural number and whose second element is an a or a b:

$$\mathbb{N} \times \{a, b\} = \{(0, a), (0, b), (1, a), (1, b), (2, a), (2, b), \dots\}$$

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:

$$\{0, 1\}^3 = \{(0, 0, 0), (0, 0, 1), (0, 1, 0), (0, 1, 1), (1, 0, 0), (1, 0, 1), (1, 1, 0), (1, 1, 1)\}$$

2.3.4 Set Builder Notation

One specialized, but 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:

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

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

$$\text{“}n \text{ is a prime and } n = 4k + 1 \text{ for some integer } k\text{”}$$

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

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

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.

2.3.5 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

$$f_2(y, z) ::= y10yz$$

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, as in the function $f_4(P, Q)$ where P and Q are propositional variables:

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: $f_4(P, Q) = [P \longrightarrow Q]$.

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

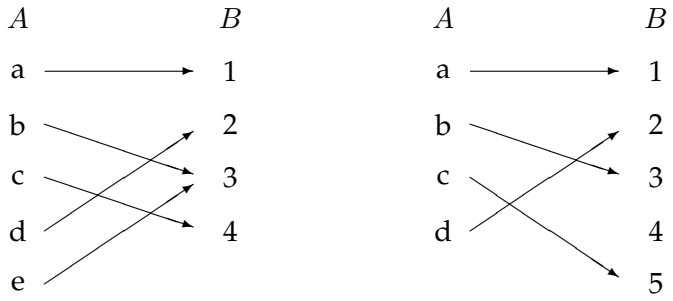
$$\begin{aligned} f_5(0010) &= 3, \\ f_5(100) &= 1, \\ f_5(0000) &\text{ is undefined.} \end{aligned}$$

There are a few properties of functions 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 function $f : A \to B$ is:

- ***total*** if every element of A is assigned to some element of B; otherwise, f is called a ***partial function***,

- ***surjective*** if every element of B is mapped to *at least once*[5],
- ***injective*** if every element of B is mapped to *at most once*, and
- ***bijective*** if f is total, surjective, and injective. In particular, each element of B is mapped to *exactly once*.

We can explain all these properties 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 $f(a) = b$. For example, here are diagrams for two functions:



Here is what the definitions say about such pictures:

- "f is a function" means that every point in the domain column, A, has at *most one arrow out of it*. (If more than one arrow came out of any point in the first column, then f would be a *relation*, but not a function. We'll take up the topic of relations in a couple of weeks.)
- "f is total" means that *every* point in the A column has *at least one arrow out of it*, which really means it has *exactly one arrow out of it* since f is a function.
- "f is surjective" means that *every* point in the codomain column, B, has *at least one arrow into it*.
- "f is injective" means that every point in the codomain column, B, has *at most one arrow into it*.
- "f 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 function on the left is total and surjective (every element in the A column has an 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 function on the right is total and injective (every element in the A column has an 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).

[5] The names "surjective" and "injective" are unmemorable and nondescriptive. Some authors use the term ***onto*** for surjective and *one-to-one* for injective, which are shorter but arguably no more memorable.

Everything about a function is captured by three sets: it domain, its codomain, and the set

$$\{(a, b) \mid f(a) = b\}$$

which is called the *graph* of f. Notice that the graph of f simply describes where the arrows go in a diagram for f.

The graph of f does not determine by itself whether f is total or surjective; we also need to know what the domain is to determine if f is total, and we need to know the codomain to tell if it's surjective. For example, a 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 $\mathbb{R}$.

Surjections and injections imply certain size relationships between domains and codomains. If A is a finite set, we let $|A|$ be its size, that is, the number of elements in A.

Lemma. *[Mapping Rule]*

- *If $f : A \to B$ is surjective, then $|A| \geq |B|$.*
- *If $f : A \to B$ is total and injective, then $|A| \leq |B|$.*
- *If $f : A \to B$ is bijective, then $|A| = |B|$.*

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 $A' \subseteq A$, we define $f(A')$ to be the set of all the values that f takes when it is applied to elements of A'. That is,

$$f(A') ::= \left\{b \in B \mid f(a') = b \text{ for some } a' \in A'\right\}.$$

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, A', of arguments is referred to as "applying f pointwise to A'." [6]

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 is important. The range and codomain of f are the same only when f is surjective.

[6] 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.

2.4 Does All This Really Work?

So this is where mainstream mathematics stands today: there is a handful of axioms from which 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.
- No one knows whether the ZFC axioms are logically consistent; there is some possibility that one person might prove a proposition P and another might prove the proposition $\neg P$. Then Math would be broken. This sounds like a crazy situation, but it has happened before. At the beginning of the 20th century, the logician Gotlob Frege made an initial attempt to axiomatize set theory using a few very plausible axioms. Several mathematicians— most famously Bertrand Russell[7]— discovered that Frege's axioms actually *were* self-contradictory!
- While the ZFC axioms largely generate the mathematics everyone wants— where $3 + 3 = 6$ and other basic facts are true— they also imply some disturbing conclusions. For example, the Banach-Tarski Theorem says that a solid ball can be divided into six pieces and then the pieces can be rearranged to give *two* solid balls, each the same size as the original!
- In the 1930's, Gödel proved that, assuming the ZFC axioms *are* consistent, then they are not *complete*: that is, there exist propositions that are true, but do not logically follow from the axioms. As a matter of fact, the proposition that ZFC is consistent (which is not too hard to express as a formula about sets) is an example of a true proposition that cannot be proved.

  There seems to be no way out of this disturbing situation; simply adding more axioms does not eliminate the problem.

These problems will not trouble us in 6.042, but they are interesting to think about!

[7] Bertrand Russell was a Mathematician/Logician at Oxford 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. He won two Nobel Prizes— one Literature Prize and one Peace Prize.

Russell's Paradox

Reasoning naively about sets quickly leads to the following contradiction— known as Russell's Paradox:

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

$$W \in W \text{ iff } W \notin W.$$

This paradox revealed a fatal flaw in Frege's initial effort to axiomatize set theory. This was an astonishing blow to efforts to provide an axiomatic foundation for Mathematics.

But a way out was clear at the time: *we cannot assume that W is a set*. So the step in the proof where we let S be W is invalid, because S ranges over sets, and W is not a set.

But denying that W is a set means we must reject the axiom that every mathematically well-defined collection of elements is actually a set.

The problem faced by Logicians was how to axiomatize rules determining which well-defined collections are sets. Russell and his colleague Whitehead immediately went to work on this problem and spent a dozen years developing a huge new axiom system in an even huger monograph called *Principia Mathematica*.

The modern ZFC axioms for set theory are much simpler than the Russell/Whitehead system and are close to Frege's original axioms. They specify that sets must be built up from "simpler" sets in certain standard ways. In particular, no set is ever a member of itself. So the modern resolution of Russell's paradox goes as follows: since $S \notin S$ for all sets S, it follows that W, defined above, contains every set. So W can't be a set or it would be a member of itself.

These issues rarely come up in mainstream Mathematics. And they don't come up at all in Computer Science, where the focus is generally on "countable," and often just finite, sets. In practice, only Logicians and Set Theorists have to worry about collections that are too big to be sets.

Chapter 3

Relations; Partial Orders

3.1 Binary Relations

Relations are another fundamental Mathematical data type. Equality and "less-than" are the most 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 and then focus on *partial orders*, which are a class of binary relations of particular importance in Computer Science, with direct applications that include task scheduling, database concurrency control, and proving that computations terminate.

3.1.1 Binary Relations and Functions

Binary relations are far more general than equality or less-than. Here's the official definition:

Definition 3.1.1. 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 3.1.1 is exactly the same as the definition 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 '08 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 ⟨instructor-name⟩ is in charge the subject with number ⟨subject-num⟩ in Spring '08. So the 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, Verghese,  6.011),
(G, Verghese,  6.UAT),
(G. Verghese,  6.881)
(G. Verghese,  6.882)
(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. Verghese 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.

3.1.2 Images and Inverse Images

The faculty in charge of 6.UAT in Spring '08 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 Verghese. Similarly to find out who is in-charge of introductory course 6 subjects this term, take all the pairs of the form $(\langle\text{instructor-name}\rangle, 6.0\ldots)$ and list the left hand sides of these pairs. This works because the set D, of introductory course 6 subject numbers are the ones that start with "6.0". For example, from the part of the graph of T shown above, we can see that Meyer, Leighton, Verghese, and Guttag are in-charge of introductory subjects this term.

These are all examples of taking an *inverse image* of a set under a relation. If R is a binary relation from A to B, and X is any set, define the *inverse image* of X under R, written simply as RX to be the set elements of A that are related to something in X.

For example, TD, the inverse image of the set D under the relation, T, is the set of all faculty members in-charge of introductory course 6 subjects in Spring '08. Notice that in inverse image

notation, D gets written to the right of T because, to find the faculty members in TD, we're looking at the right hand side of pairs in the graph of T for subject numbers in D.

Here's a nice, concise definition of the inverse image of a set X under a relation, R:

$$RX ::= \{a \in A \mid aRx \text{ for some } x \in X\}.$$

Similarly, the *image* of a set Y under R, written YR, is the set of elements of the codomain, B, that are related to some element in Y, namely,

$$YR ::= \{b \in B \mid yRb \text{ for some } y \in Y\}.]$$

So, $\{\text{A. Meyer}\}\,T$ gives the subject numbers that Meyer is in charge of in Spring '08. In fact, $\{\text{A. Meyer}\}\,T = \{6.042, 18.062, 6.844\}$. Since the domain, F, is the set of all in-charge faculty, FT is exactly the set of *all* Spring '08 subjects being taught. Similarly, TN is the set of people in-charge of a Spring '08 subject.

It gets interesting when we write composite expressions mixing images, inverse images and set operations. For example, $(TD)T$ is the set of Spring '08 subjects that have people in-charge who also are in-charge of introductory subjects. So $(TD)T - D$ are the advanced subjects with someone in-charge who is also in-charge of an introductory subject. Similarly, $TD \cap T(N - D)$ is the set of faculty teaching both an introductory *and* an advanced subject in Spring '08.

Warning: When R happens to be a function, the pointwise application, $R(Y)$, of R to a set Y described in Notes 2 is exactly the same as the image of Y under R. That means that when R is a function, $R(Y) = YR$ —*not* RY. Both notations are common in Math texts, so you'll have to live with the fact that they clash. Sorry about that.

3.1.3 Surjective and Total Relations

If R is a binary relation from A to B, we define AR to to be the *range* of R. This matches the definition of "range" when R happens to be a function.

A relation with the property that every codomain element is related to some domain element is called a *surjective* (or *onto*) relation —again, the same definition as for functions. More concisely, R is surjective iff $AR = B$. Likewise, a relation with the property that every domain element is related to some codomain element is called a *total* relation; more concisely, R is total iff $A = RB$.

3.1.4 The sizes of infinite sets

A finite set, A, may have no elements (the empty set), or one element, or two elements,... or any nonnegative integer number of elements. The *size*, $|A|$, of A is defined to be the number of elements in A. According to the Mapping Rule from Notes 2, the size of a finite set, A, is greater than or equal to the size of another finite set, B, iff there is a surjective function, $f : A \to B$. They have the same size iff there is a bijection from A to B. This idea generalizes in an interesting way to infinite sets:

Definition 3.1.2. Let A, B be sets. Then

1. A is *as big as* B iff there is a surjective function from A to B,

2. A is *the same size* as B iff there is a bijection from A to B.

3. A is *strictly* bigger than B iff A is as big as B, but B is not as big as A.

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 between sets.

But there's something else to watch out for, because we've taken an ordinary phrase and given it a purely technical meaning —something Mathematicians do all the time. There are a lot of properties you'd expect an "as big as" relation to have, but you can't assume *any* of them until they're proved. Of course most of the "as big as" and "same size" properties of finite sets actually do carry over to infinite sets, but some important ones don't, as we're about to show.

The first thing to confirm is that Definition 3.1.2 does match the usual one for finite sets. Since the Mapping Rule motivated Definition 3.1.2 in the first place, this should be no surprise. Namely,

Lemma 3.1.3. *For finite sets, A, B,*

$$A \text{ is as big as } B \quad \textit{iff} \quad |A| \geq |B|\,,$$

and

$$A \text{ is the same size as } B \quad \textit{iff} \quad |A| = |B|\,.$$

This Lemma is just a rephrasing of the surjection and bijection parts of the Mapping Rule, so it follows immediately.

Several further familiar properties of the "as big as" and "same size" relations on finite sets carry over exactly to infinite sets:

Lemma 3.1.4. *For any sets, A, B, C,*

1. *A is as big as B and B is as big as C, implies$A$ is as big as C.*

2. *A is the same size as B and B is the same size as C, implies A is the same size as C.*

3. *A is the same size as B implies B is the same size as A.*

Lemma 3.1.4.1 and 3.1.4.2 follow immediately from the fact that compositions of surjections are surjections, and likewise for bijections, and part 3. follows from the fact that the inverse of a bijection is a bijection. These facts make a good exercise:

Problem 3.1.1. 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.)

Another familiar property of finite sets carries over to infinite sets, but this time it's not so obvious:

Theorem 3.1.5 (Schröder-Bernstein)**.** *For any sets A, B, if A is as big as B, and B is as big as A, then A is the same size as B.*

The Schröder-Bernstein Theorem is actually pretty technical: it says that if there are surjections (which need not be bijections), $f : A \to B$ and $g : B \to A$, then there is a bijection $e : A \to B$. The idea is to construct e from parts of both f and g, but we won't go into the details here.

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 3.1.6. *Let A be a set and $b \notin A$. Then A is infinite iff A is the same size as $A \cup \{b\}$.*

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 $a_0, a_1, a_2, \ldots, a_n, \ldots$ of distinct elements of A. Now it's easy to define a bijection $e : A \cup \{b\} \to A$:

$$\begin{aligned} e(b) &::= a_0, \\ e(a_n) &::= a_{n+1} && \text{for } n \in \mathbb{N}, \\ e(a) &::= a && \text{for } a \in A - \{b, a_0, a_1, \ldots\}. \end{aligned}$$

□

Problem 3.1.2. Prove that every infinite set is as big as the set, $\mathbb{N}$, of nonnegative integers, *Hint:* The proof of Lemma 3.1.6.

Power sets are strictly bigger

All this finally brings us to our key concern about infinite sets: are they all the same size? It turns out that the ideas behind Russell's Paradox, which caused so much trouble for naive Set Theory, give the answer: no, they are not all the same size. In particular,

Theorem 3.1.7. *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 $A_g \in \mathcal{P}(A)$ that is not in the range of g. □

Infinities in Computer Science

We've run into a lot of Computer Science students who wonder why they need to learn all this abstract theory about infinite sets, and this is a good question.

Sometimes these students overstate their doubts by claiming that only finite sets come up in Computer Science, but it ain't so: the standard programming data types of integers, floating point numbers, strings, for example, each have an infinite number of data items of that type, although each individual datum of these types of course is finite. But higher order types, for example, the type of string-to-integer procedures, not only has an infinite number of data items, but each procedure datum generally behaves differently on different inputs, so that a single datum may embody an infinite number of behaviors.

On the other hand, if the romance of one infinity being bigger than another doesn't appeal to you, not knowing about it is not going to lower your professional abilities as a Computer Scientist. In fact, at the end of the 19th century, the general Mathematical community also doubted the relevance of what they called "Cantor's paradise" of unfamiliar sets of arbitrary infinite size.

But this theory, especially the proof in Theorem 3.1.7 that power sets are bigger, gives 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 (a variation of which is given in Pset 2, prob 5) 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.

Different infinities

So 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 3.1.7, 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. In this way you can keep going, building still bigger infinities.

Problem 3.1.3. Prove that the union of this sequence of sets is strictly bigger than each of the sets in the sequence.

So there is an endless variety of different size infinities. But things are not completely messy: sets can at least be "lined up" by size. Technically this means that any two sets A and B are *comparable* under the "as big as" relation: either A is as big as B, or B is as big as A, or both.

Comparability is another one of those familiar properties of finite sizes that you'd be tempted to assume in general, and it's true, but not obvious. In fact, proving comparability of arbitrary sets requires more than the "naive" Set Theory which is adequate for most purposes, and it goes beyond what belongs in a Mathematics for Computer Science course.

At this point in our overview of Set properties, it may come as a surprise to hear that the subject remains an active research area. For example, the nineteenth century Mathematician, Georg Cantor, who first developed the theory of infinite sizes (because he thought he needed it in his study of Fourier series) asked whether there is a set of size between $\mathbb{N}$ 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 $\mathbb{N}$.

The Continuum Hypothesis remains an open problem a century later. Its difficulty arises from one of the deepest results in 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 because 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 are 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.

3.2 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.

3.2.1 Axioms for Partial Orders

Definition 3.2.1. A binary relation, R, on a set A is:

- *transitive* iff

$$[a\,R\,b \text{ and } b\,R\,c] \quad \text{implies} \quad a\,R\,c,$$

for every $a, b, c \in A$,

[1] MIT's Committee on Curricula has the responsibility of watching out for such bugs that might creep into departmental requirements.

- *asymmetric* iff

$$a\,R\,b \quad \text{implies} \quad \neg(b\,R\,a)$$

for all $a, b \in A$,

- a *strict partial order* iff it is transitive and asymmetric.

So the prerequisite relation, $\rightarrow$, 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:

Definition 3.2.2. A binary relation, R, on a set A, is

- *reflexive* iff $a\,R\,a$ for all $a \in A$,
- *antisymmetric* if

$$a\,R\,b \quad \text{implies} \quad \neg(b\,R\,a)$$

for all $a \neq b \in A$,

- a *weak partial order* iff it is transitive, reflexive and antisymmetric.

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," presumably because of the confusion caused by using just his symbol.) Likewise, we generally use $\prec$ or $\sqsubset$ to indicate a strict partial order. We also write $b \succeq a$ to mean $a \preceq b$ and $b \succ a$ to mean $a \prec b$.

Two more examples of partial orders are worth mentioning:

Example 3.2.3. Let A be some family of sets and define aRb 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 3.2.4. The divides relation is a weak partial order on the nonnegative integers.

3.2.2 Representing Partial Orders by Set Containment

When a class of objects are defined by axioms, like the axioms for partial orders, it can help to have a way to "represent" them explicitly by known objects. Partial orders can be represented by the subset relation on a collection of sets. Namely, if R is a weak partial order on a set, A, we can let each element $a \in A$ correspond to the set $R\{a\}$. Since

$$a\,R\,b \quad \text{iff} \quad R\{a\} \subseteq R\{b\} \tag{3.1}$$

holds for all $a, b \in A$, we have completely captured the weak partial order R by the subset relation on the corresponding sets. A similar correspondence shows that strict partial orders can be represented by the proper subset relation, $\subset$.

Problem 3.2.1. Prove the iff assertion (3.1).

Problem 3.2.2. Verify that the relations in Examples 3.2.3 and 3.2.4 are partial orders.

Definition 3.2.5. A relation, R, on a set, A, is *irreflexive* iff for all $a \in A$, it is *not* true that $a \, R \, a$.

Problem 3.2.3. Prove that a binary relation is a strict partial order iff it is transitive and irreflexive.

3.2.3 Total Orders

The familiar order relations on numbers have an important additional property: given any two numbers, one will be bigger than the other. Partial orders with this property are said to be *total*[2] *orders*:

Definition 3.2.6. 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 ($a \, R \, b$ or $b \, R \, a$). A partial order under which every two distinct 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 distinct 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.

3.2.4 Products of Relations

Taking the product of two relations is a useful way to construct new relations from old ones.

Definition 3.2.7. 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 3.2.8. 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

$$(y_1, h_1)\, Y\, (y_2, h_2) \quad \text{iff} \quad y_1 \leq y_2 \wedge h_1 \leq h_2.$$

That is, Y is the product of the $\leq$-relation on ages and the $\leq$-relation on heights.

[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.

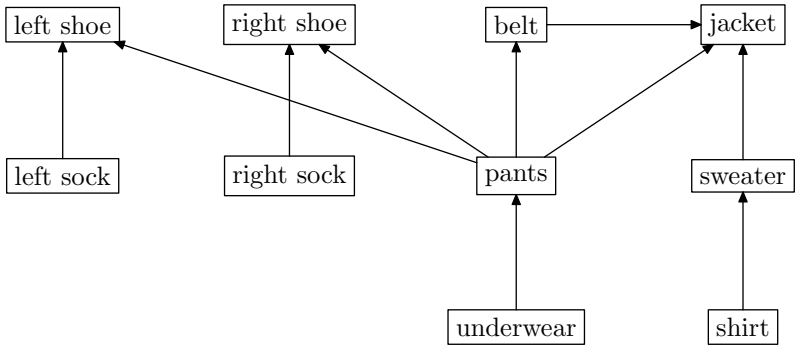
Products preserve several of the relational properties we have considered. Namely, it's not hard to verify that if R_1 and R_2 are both transitive, then so is $R_1 \times R_2$. The same holds for reflexivity, irreflexivity, and antisymmetry. This implies that if R_1 and R_2 are both partial orders, then 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.

3.2.5 Topological Sorting

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 3.2.9. 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 3.2.10. A *topological sort* of a partial order, $\prec$, on a set, A, is a total ordering, $\sqsubset$, on A such that

$$a \prec b \quad \text{implies} \quad a \sqsubset b.$$

For example,

shirt $\sqsubset$ sweater $\sqsubset$ underwear $\sqsubset$ leftsock $\sqsubset$ rightsock $\sqsubset$ pants $\sqsubset$ leftshoe $\sqsubset$ rightshoe $\sqsubset$ belt $\sqsubset$ jacket,

is one topological sort of the partial order of dressing tasks given by Example 3.2.9; 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 3.2.11. Let $\preceq$ be a partial order on a set, A. An element $a \in A$ is *minimum* iff it is $\preceq$ every other element of A. The element a is *minimal* iff no other element is $\preceq a$.

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.

For this method of topological sorting to work, we need to be sure there is always a minimal element. This is sort of obvious, but noting that an infinite partially ordered set might have no minimal element —consider $<$ on the $\mathbb{Z}$ —it would be good to prove that minimal elements exist.

Lemma 3.2.12. *Every partial order on a nonempty finite set has a minimal element.*

Proof. Let R be a strict partial order on a set, A. Define the *weight* of an element $a \in A$ to be $|R\{a\}|$ —the number of elements in the set $R\{a\}$. Since A is finite, the weights of all elements in A are nonnegative integers, so there must be an $a_0 \in A$ with the smallest weight.

Now suppose $|R\{a_0\}| \neq 0$. Then there is an element $a_1 \in R\{a_0\}$, which implies (by transitivity of R) that $R\{a_1\} \subseteq R\{a_0\}$, and hence $|R\{a_1\}| \leq |R\{a_0\}|$. But since R is strict, $a_1 \in R\{a_0\} - R\{a_1\}$, so in fact $|R\{a_1\}| < |R\{a_0\}|$, contradicting the fact the a_0 has the smallest weight.

This contradiction implies that $|R\{a_0\}| = 0$, which means that no element is related by R to a_0, that is, a_0 is minimal.

A similar argument works in the case that R is a weak partial order.

□

So our construction shows:

Theorem 3.2.13. *Every partial order on a finite set has a topological sort.*

In fact, the domain of the partial order need not be finite: we won't prove it, but *all* partial orders, even infinite ones, have topological sorts.

There are many other ways of constructing topological sorts. For example, instead of starting "from the bottom" with minimal elements, we could start "from the top" by picking *maximal* elements:

Definition 3.2.14. Let $\preceq$ be a partial order on a set, A. An element $a \in A$ is *maximum* iff it is $\succeq$ every other element of A. The element a is maxim*al* iff no other element is $\succeq a$.

Problem 3.2.4. **(a)** Prove that there is at most one minim*um* element in any partial order.

(b) Give an example of a partial order with exactly one minimal element, but no minimum element. *Hint:* It will have to be infinite.

3.2.6 Parallel Task Scheduling

For a partial order of task dependencies, topological sorting provides a way to execute tasks sequentially without violating 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 the same amount 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.

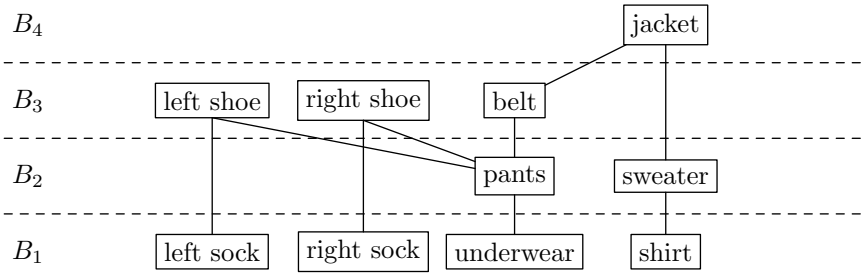
In the clothes example, we could do all the minimal elements first (leftsock, rightsock, underwear, shirt), remove them and repeat. We'd need lots of hands, or maybe dressing servants. We can do pants and sweater next, and then leftshoe, rightshoe, and belt, and finally jacket.

We can't do any better, because the sequence underwear, pants, belt, jacket must be done in that order. A set of tasks that must be done in sequence like this is called a *chain*.

Definition 3.2.15. A *chain* in a partial order is a set of elements such that any two elements in the set are comparable.

In other words, a chain is a totally ordered subset of the elements in a partial order. Clearly, the parallel time must be at least the size of any chain. For if we used less time, then two tasks in the chain would have to be done at the same time, violating the dependency constraints.

A largest chain is also known as a *critical path*. So we need at least t steps, where t is the size of a largest chain. Fortunately, it is always possible to use only t parallel steps. The idea is to let B_1 be all the minimal elements and schedule them first. Then remove all the elements in B_1, let B_2 be the elements that now become minimal, and schedule them next, and so on. For getting dressed, here is a picture of the schedule obtained in this way:



Theorem 3.2.16. *Let R be strict partial order on a set, A. If the longest chain in A is of size t, then there is a partition[3] of A into t blocks, $B_1, B_2, \ldots, B_t$, such that for each block, B_i, all tasks that have to precede*

[3]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 distinct sets, then $B \cap B' = \emptyset$, and
- $\bigcup_{B \in \mathcal{B}} B = A$.

tasks in B_i are in smaller-numbered groups. That is,

$$RB_1 = \emptyset, \text{ and} \tag{3.2}$$

$$RB_i \subseteq B_1 \cup B_2 \cup \cdots \cup B_{i-1}, \tag{3.3}$$

for $1 < i \leq t$.

Corollary 3.2.17. *For R and t as above, it is possible to schedule all tasks in t steps.*

Proof. Schedule all the elements of B_i at time i. This satisfies the dependency requirements, because all the tasks that any task depends on are scheduled at preceding times. □

Corollary 3.2.18. *Parallel time = Size of largest chain.*

So it remains to prove Theorem 3.2.16:

Proof. A chain is said to *begin* with its smallest element and *end* with its largest element, if any.

Construct the sets B_i as follows:

$$B_i ::= \{a \in A \mid \text{the largest chain ending in } a \text{ is of size } i\}.$$

This gives just t sets, because the largest chain is of size t. Also, each $a \in A$ belongs to exactly one B_i. To complete the proof, notice that if $a \in B_1$, then a must be minimal, and since R is strict we have $RB_1 = \emptyset$ proving (3.2).

Now suppose $1 < i \leq t$, and assume for the sake of contradiction that (3.3) does not hold. That is, there is an $a \in B_i$ and $b \in A$ such that $b\,R\,a$, and $b \notin B_1 \cup B_2 \cup \cdots \cup B_{i-1}$. Then by definition of the B_j's, there is a chain of size $> i-1$ ending at b. Also, since R is strict, a is not in the chain ending at b. So we can add a to the end of the chain to obtain a chain of size $> i$ ending in a, contradicting the fact that that $a \in B_i$. □

So with an unlimited number of processors, the time to complete all the tasks is the size of the largest chain. It turns out that this theorem is good for more than parallel scheduling. It is usually stated as follows.

Definition 3.2.19. An *antichain* in a partial order is a set of elements such that any two elements in the set are incomparable.

Corollary 3.2.20. *If the largest chain in a partial order is of size, t, then the domain can be partitioned into t antichains.*

Proof. Let the antichains be the sets B_i defined as in the proof of Theorem 3.2.16.

We should verify that each B_i is an antichain, namely, if a, b are distinct elements of B_i, then they are incomparable. But suppose to the contrary that there exist two elements $a, b \in B_i$ such that a and b are comparable, say $a\,R\,b$. Then, as in the proof of Theorem 3.2.16, by adding b at the end of the chain of size i ending at a, we obtain a chain of size $i+1$ ending at b, contradicting the assumption that $b \in B_i$. □

3.2.7 Dilworth's Lemma

We can use the Corollary 3.2.20 to prove a famous result[4] about partially ordered sets:

[4] Lemma 3.2.21 also follows from a more general result known as Dilworth's Theorem which we will not discuss.

Lemma 3.2.21 (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. Assume there is no chain of size greater than t, that is, the largest chain is of size $\leq t$. Then by Corollary 3.2.20, 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 3.2.22. *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 3.2.21. □

Example 3.2.23. 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 $4 \geq 10/4$.

Example 3.2.24. Suppose we have a class of 101 students. Then using the product partial order, Y, from Example 3.2.8, 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.

Quick Exercise: 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?

3.3 The Well Ordering Principle

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

Do you believe this statement? 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.

While the Well Ordering Principle may seem obvious, and 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 Lemma 3.2.12, that every finite partial order has a minimal element. We even implicitly relied on the Well Ordering Principle in the proof in Week 2 Notes 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

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

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! 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

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

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

Let's use this this template to prove

Theorem.

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

for all nonnegative integers, n.

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 (3.4) is false for $n = c$ but true for all nonnegative integers $n < c$. But (3.4) is true for $n = 0$, so $c > 0$. This means $c - 1$ is a nonnegative integer, and since it is less than c, equation (3.4) 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 (3.4) does hold for c, after all! This is a contradiction, and we are done. □

3.4 Well-Founded Partial Orderings

Definition 3.4.1. A partial order is *well-founded* iff every nonempty subset of its domain has a minim*al* element.

So the Well Ordering Principle is equivalent to the assertion that the nonnegative integers are well-founded under $\leq$. But well-foundedness makes sense much more generally than for just nonnegative integers.

For example, by Lemma 3.2.12, every partially ordered finite set has a minimal element, so all finite partial orders are automatically well founded. Note that in this case we can't expect to find a minim*um* element, since even in a finite partial order, there often isn't any minimum.

Quick Exercise: Give an example of a partial order with a *unique* minim*al* element but no minim*um* element.

Hint: It can't be a finite partial order.

There is another helpful way to characterize well-founded partial orders:

Lemma 3.4.2. *A partial order is well-founded iff it has no infinite decreasing chain.*

Saying that the partial order $\preceq$ has no infinite decreasing chain means there is no infinite sequence $p_1, p_2, \ldots, p_n \ldots$ of elements in P such that

$$p_1 \succ p_2 \succ \cdots \succ p_n \ldots.$$

Here we're using the notation "$p \succ q$" to mean [$q \preceq p$ and $q \neq p$]. That's so we can read the decreasing chain left to right, as usual in English.

Proof. (**left to right**) (By contradiction) If there was such an infinite decreasing sequence, then the set of elements in the sequence itself would be a nonempty subset without a minimal element.

(**right to left**) (By contradiction) Suppose $\preceq$ was not well-founded. So there is some subset $S \subseteq$ domain $(\preceq)$ P, such that S has at least one element, s_1, but S has no minimal element. In particular, since s_1 is not minimal, there must be *another* element $s_2 \in S$ such that $s_1 \succ s_2$. Similarly, since s_2 is not minimal, there must be still another element $s_3 \in S$ such that $s_2 \succ s_3$. Continuing in this way, we can construct an infinite decreasing chain $s_1 \succ s_2 \succ s_3 \cdots$ in S. □

It now follows immediately from Lemma 3.4.2 that every finite partial order is well-founded — without needing the counting argument used to prove this in Lemma 3.2.12.

Problem 3.4.1. Let D be the usual *dictionary order* on finite sequences of letters of the alphabet. Show that neither D nor D^{-1} is well-founded.

3.4.1 Product Partial Orders

An easy way to construct well-founded partial orders is by taking products of well-founded partial orders. For example, the nonnegative integers are well-founded under $\leq$, so the product partial order $(\leq \times \leq)$ on pairs of nonnegative integers is going to be well-founded.

To prove this, we first generalize partial orders on pairs of integers to partial orders pairs of elements from *any* partial orders.

Definition 3.4.3. Let $\preceq_1$ and $\preceq_2$ be partial orders with domains A_1 and A_2.

The *coordinatewise partial order*, $\preceq_c$, is defined to be the product relation, $(\preceq_1 \times \preceq_2)$, which we observed in Section 3.2.4 really is a partial order when $\preceq_1$ and $\preceq_2$ are.

The *lexicographic partial order*, $\preceq_{\text{lex}}$, for $\preceq_1$ and $\preceq_2$ is defined by the conditions:

$$\begin{array}{rcl} \text{domain}\,(\preceq_{\text{lex}}) & ::= & A_1 \times A_2 \\ (a_1, a_2) \preceq_{\text{lex}} (b_1, b_2) & \text{iff} & a_1 \prec_1 b_1 \text{ or } (a_1 = b_1 \text{ and } a_2 \preceq_2 b_2). \end{array}$$

It's easy to check that $\preceq_{\text{lex}}$ also a partial order, so we're justified in calling it that. But not only are coordinatewise and lexicographic partial orders really partial orders, but they will also be well-founded providing $\preceq_1$ and $\preceq_2$ are well-founded. Namely,

Theorem 3.4.4. *For well-founded partial orders, $\preceq_1$ and $\preceq_2$, let $\preceq_c$ be their coordinatewise partial order and $\preceq_{lex}$ be their lexicographic partial order. Then*

1. *$\preceq_c$ is well-founded,*
2. *$\preceq_{lex}$ is well-founded,*
3. *if $\preceq_1$ and $\preceq_2$ are both total orders, then so is $\preceq_{lex}$.*

The proof of part 1 is easy: a coordinatewise minimal element of any nonempty subset, S, of $A_1 \times A_2$ is simply a pair (m_1, m_2) consisting of a minimal element, $m_1 \in A_1$, among the first coordinates of pairs in S, and a minimal element, $m_2 \in A_2$, among the second coordinates of pairs in S.

A refinement of this idea works for lexicographic order. To find a minimal element of S under $\preceq_{\text{lex}}$, let $m_1 \in A_1$, be minimal among the first coordinates of pairs in S, as before. But now find a minimal element $n_2 \in A_2$ among the second coordinates pairs in S *whose first coordinate is* m_1. It's not hard to see that the element $(m_1, n_2) \in S$ must be a lexicographically minimal element of S.

Watch out! When we, or any other author, says "It's not hard to see," a savvy reader should quickly check if it really is easy, and if it doesn't seem to be, they should suspect that the author is being lazy or possibly wrong.

In this case, we're not being lazy —we're just omitting an almost automatic (and uninformative) series of simple proof steps, which it would be more educational for you work out yourself than to see written out by us. For the same reason, we'll let you verify the last Part 3. that lexicographic order determined by two total orders is also a total order.

This completes the proof of Theorem 3.4.4.

3.4.2 Well founded recursive definitions

Ackermann's function is an extremely fast-growing function of two nonegative arguments, which means its inverse is extremely slow-growing. For example, its inverse grows slower than $\log n$, $\log\log n$, $\log\log\log n$, ..., but it does grow unboundly. It turns out that this extremely slow growing function actually comes up in the study of algorithms. It is the coefficient of n in a formula for the average number of steps used by a *Union-Find algorithm* to process n queries and declarations of equivalence between pairs of elements of a set. Since the coefficient of n grows unboundly, the Union-Find algorithm theoretically uses a nonlinear number of operations. But it may as well be linear, since the coefficient is less than 5 for any n that could conceivably be achieved.

You will learn about Union-Find if you take the Algorithms course, 6.046. We're mentioning this story to motivate an examination of the somewhat unusual recursive definition of Ackermann's function, $A(m, n)$.

Ackermann's function can be defined recursively as the function, f, given by the following rules:

$$f(m,n) = 2n, \qquad \text{if } m = 0 \text{ or } n \leq 1, \tag{3.5}$$

$$f(m,n) = f(m-1, f(m, n-1)), \qquad \text{otherwise.} \tag{3.6}$$

Now these rules are unusual because the definition of $f(m, n)$ involves an evaluation of f at arguments that may be a lot bigger than m and n. Definitions of values at small arguments in terms of larger arguments can easily lead to divergent (non terminating) evaluations, so how can we be sure this one is ok? The simple answer is that the definition of $f(m, n)$ actually doesn't involve evaluation at bigger arguments than (m, n) if we think of the right way to order the arguments. In this case, lexicographic order does the job.

Being an ok definition means that equations (3.5) and (3.6) define a *unique*, *total* function, f. We'll focus on proving uniqueness. Namely, if g satisfies the same equations, that is,

$$g(m,n) = 2n, \qquad \text{if } m = 0 \text{ or } n \leq 1, \tag{3.7}$$

$$g(m,n) = g(m-1, g(m, n-1)), \qquad \text{otherwise,} \tag{3.8}$$

then $f = g$.

We'll give a simple proof of this using the well foundedness of lexicographic order, $\preceq_{\text{lex}}$, on $\mathbb{N}^2$. Namely, assume for the sake of contradiction that $f \neq g$. Then the set, S, of pairs (m, n) such that $f(m, n) \neq g(m, n)$ is not empty, so by well-foundedness of lexicographic order, there is a $\prec_{\text{lex}}$-*least* element $(m_0, n_0) \in S$. In other words,

$$f(m_0, n_0) \neq g(m_0, n_0), \tag{3.9}$$

but

$$f(m, n) = g(m, n) \text{ for } (m, n) \prec_{\text{lex}} (m_0, n_0). \tag{3.10}$$

Now if $m_0 = 0$ or $n_0 \leq 1$, then $f(m_0, n_0)$ and $g(m_0, n_0)$ both equal $2n_0$ by (3.5) and (3.7), which contradicts (3.9). So it must be that $m_0 > 0$ and $n_0 > 1$. This implies

$$f(m_0, n_0) = f(m_0 - 1, f(m_0, n_0 - 1)), \tag{3.11}$$

and

$$g(m_0, n_0) = g(m_0 - 1, g(m_0, n_0 - 1)), \tag{3.12}$$

by (3.6) and (3.8).

Next, by definition of lexicographic order, $(m_0, n_0 - 1) \prec_{\text{lex}} (m_0, n_0)$, so (3.10) implies that

$$f(m_0, n_0 - 1)) \text{ and } g(m_0, n_0 - 1)) \text{ have the same value, } v.$$

Similarly, $(m_0 - 1, v) \prec_{\text{lex}} (m_0, n_0)$ by definition of $\prec_{\text{lex}}$, so (3.10) also implies that

$$f(m_0 - 1, v) = g(m_0 - 1, v) \tag{3.13}$$

which by (3.11) and (3.12), implies that $f(m_0, n_0) = g(m_0, n_0)$. But this contradicts (3.9), which completes the proof.

Chapter 4

Induction

4.1 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 induction works, suppose there is a professor who brings to class a bottomless bag of assorted miniature candy bars. She offers to share in accordance with two rules. First, she numbers the students 0, 1, 2, 3, and so forth for convenient reference. Now here are the two rules:

1. Student 0 gets candy.
2. If student n gets candy, then student $n + 1$ also gets candy, for every $n \in \mathbb{N}$.

You can think of the second rule as a compact way of writing a whole sequence of statements, one for each possible value of n:

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

$$\vdots$$

Now suppose you are student 17. By these rules, are you entitled to a miniature candy bar? Well, student 0 gets candy by the first rule. Therefore, by the second rule, student 1 also gets candy, which means student 2 gets candy as well, which means student 3 get candy, and so on. So the professor's two rules actually guarantee candy for *every* student, no matter how large the class. You win!

This kind of reasoning is an instance of

The Principle of Induction. Let $P(n)$ be a predicate. If

- $P(0)$ is true, and
- for all $n \in \mathbb{N}$, $P(n)$ implies $P(n+1)$,

then $P(n)$ is true for all $n \in \mathbb{N}$.

Here's the correspondence between the induction principle and sharing candy bars. Suppose that $P(n)$ is the predicate, "student n gets candy". Then the professor's first rule asserts that $P(0)$ is true, and her second rule is that for all $n \in \mathbb{N}$, $P(n)$ implies $P(n+1)$. Given these facts, the induction principle says that $P(n)$ is true for all $n \in \mathbb{N}$. In other words, everyone gets candy.

The intuitive justification for the general induction principle is the same as for everyone getting a candy bar under the professor's two rules. So the Principle of Induction is universally accepted as an obvious, sound proof method. What's not so obvious is how much mileage we get by using it.

4.2 Using Induction

Induction often works directly in proving that some statement about nonnegative integers holds for all of them. For example, here is a classic formula:

Theorem 4.2.1. *For all $n \in \mathbb{N}$,*

$$1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2} \tag{4.1}$$

The left side of equation (4.1) represents the sum of all the numbers from 1 to n. Here the dots ($\cdots$) indicate a pattern you're supposed to be able to guess so you can mentally fill in the remaining terms.

The meaning of this sum is not so obvious in a couple of special cases:

- If $n = 1$, then there is only one term in the summation, and so $1 + 2 + 3 + \cdots + n = 1$. 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.)

We could eliminate the need for guessing by rewriting the left side of (4.1) 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 on by the expression to the right of the sigma as the variable, i, ranges from 1 to n. Both these summation expressions make it clear what (4.1) 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).

Now let's use the induction principle to prove Theorem 4.2.1. Suppose that we define predicate $P(n)$ to be "$1+2+3+\cdots+n = n(n+1)/2$". 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:

- $P(0)$ 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 $0(0+1)/2 = 0$.

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 $P(n)$:

$$1+2+3+\cdots+n = \frac{n(n+1)}{2} \tag{4.2}$$

in order to prove $P(n+1)$:

$$1+2+3+\cdots+n+(n+1) = \frac{(n+1)(n+2)}{2} \tag{4.3}$$

These two equations are quite similar; in fact, adding $(n+1)$ to both sides of equation (4.2) and simplifying the right side gives the equation (4.3):

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

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. In effect, we've just proved that $P(0)$ implies $P(1)$, $P(1)$ implies $P(2)$, $P(2)$ implies $P(3)$, etc., all in one fell swoop.

With these two facts in hand, the induction principle says that the predicate $P(n)$ is true for all nonnegative n, so the theorem is proved.

4.2.1 A Template for Induction Proofs

The proof of Theorem 4.2.1 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". (Sometimes the base case will be $n = 1$ or even some larger number, in which case the starting value of n also should be stated.)

4. **Prove that $P(n)$ implies $P(n+1)$ for every nonnegative integer n.** This is called the "inductive step" or "induction 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 $P(0) \to P(1)$, $P(1) \to P(2)$, $P(2) \to P(3)$, 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 many writers leave this step implicit.

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

4.2.2 A Clean Writeup

The proof of Theorem 4.2.1 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 (4.1).

Base case: $P(0)$ is true, because both sides of equation (4.1) 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. We'll show you some tricks for finding such formulas in a few weeks.

4.2.3 Powers of Odd Numbers

A proof in class that $\sqrt[n]{2}$ is irrational used the "obvious":

Fact. The nth power of an odd number is odd, for all nonnegative integers, n.

Instead of taking this fact for granted, we can prove it by induction. The proof will require a simple Lemma.

Lemma. *The product of two odd numbers is odd.*

To prove the Lemma, note that the odd numbers are, by definition, the numbers of the form $2k+1$ where k is an integer. But

$$(2k+1)(2k'+1) = 2(2kk' + k + k') + 1,$$

so the product of two odd numbers also has the form of an odd number, which proves the Lemma.

Now we will prove the Fact using the induction hypothesis

$$P(n) ::= \text{if } a \text{ is an odd integer, then so is } a^n.$$

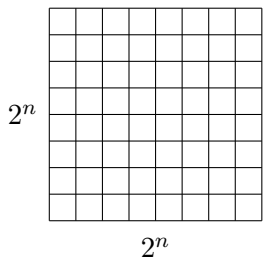
The base case $P(0)$ holds because $a^0 = 1$, and 1 is odd.

For the inductive step, suppose $n \geq 0$, a is an odd number and $P(n)$ holds. So a^n is an odd number. Therefore, $a^{n+1} = a^n a$ is a product of odd numbers, and by the Lemma a^{n+1} is also odd. This proves $P(n+1)$, and we conclude by induction that $P(n)$ holds for nonnegative integers n.

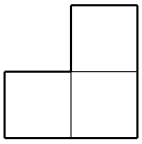
4.2.4 Courtyard Tiling

Induction served purely as a proof technique in the preceding examples. But induction sometimes can serve as a more general reasoning tool.

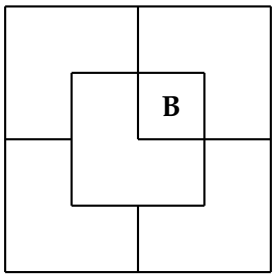
MIT recently constructed the Stata Center which houses the Computer Science and AI Laboratory. During development, the project went further and further over budget, and there were some radical fundraising ideas. One rumored plan was to install a big courtyard with dimensions $2^n \times 2^n$:



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, supposedly insisted 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 4.2.2. *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 4.2.2 by induction, we're going to need some *other* induction hypothesis than simply the statement about n that we're trying to prove.

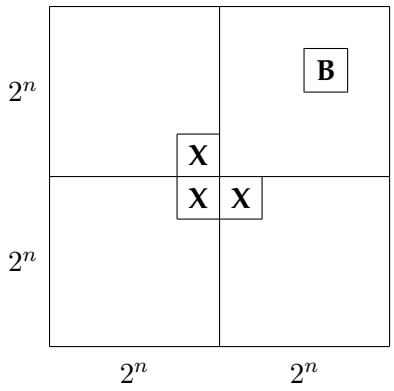
When this happens, your first fallback should be to look for a *stronger* induction hypothesis; that is, one which implies your previous hypothesis. For example, we could make $P(n)$ the proposition that for *every* location of Bill in a $2^n \times 2^n$ courtyard, there exists a tiling of the remainder.

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 $P(n) \longrightarrow P(n+1)$, 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"[1]. 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!

4.2.5 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 4.2.3. *In every set of $n \geq 1$ horses, all are the same color.*

[1] 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 two weeks.

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

$$\text{In every set of } n \text{ horses, all are the same color.} \tag{4.4}$$

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:

$$h_1,\ \underbrace{h_2,\ \ldots,\ h_n,\ h_{n+1}}_{\text{same color}}$$

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 "$\ldots$" 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 6.042 exam.

4.3 Strong Induction

4.3.1 The Strong Induction Principle

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 $n \in \mathbb{N}$.

Principle of Strong Induction. Let $P(n)$ be a predicate. If

- $P(0)$ is true, and
- for all $n \in \mathbb{N}$, $P(0)$, $P(1)$, ..., $P(n)$ *together* imply $P(n+1)$,

then $P(n)$ is true for all $n \in \mathbb{N}$.

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 $P(0)$, $P(1)$, ..., and $P(n)$ are *all* true when you go to prove $P(n+1)$. These extra assumptions can only make your job easier.

4.3.2 Products of Primes

As a first example, we'll use strong induction to prove one of those familiar facts that is almost, but maybe not entirely, obvious:

Lemma 4.3.1. *Every integer greater than 1 is a product of primes.*

Note that, by convention, any number is considered to be a product consisting of one term, namely itself. In particular, every prime is considered to be a product whose terms are all primes.

Proof. We will prove Lemma 4.3.1 by strong induction, letting the induction hypothesis, $P(n)$, be

$$n+2 \text{ is a product of primes.}$$

So Lemma 4.3.1 will follow if we prove that $P(n)$ holds for all $n \geq 0$.

Base Case: $P(0)$ is true because $0+2$ is prime, and so is a product of primes by convention.

Inductive step: Suppose that $n \geq 0$ and that $i+2$ is a product of primes for every nonnegative integer $i < n+1$. We must show that $P(n+1)$ holds, namely, that $n+3$ is also a product of primes. We argue by cases:

If $n+3$ is itself prime, then it is a product of primes by convention, so $P(n+1)$ holds in this case.

Otherwise, $n+3$ is not prime, which by definition means $n+3 = km$ for some integers k, m such that $2 \leq k, m < n+3$. Now $0 \leq k-2 < n+1$, so by strong induction hypothesis, we know that

$(k-2)+2=k$ is a product of primes. Likewise, m is a product of primes. it follows immediately that $km = n+3$ 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.

□

Despite the name, strong induction is actually no more powerful than ordinary induction: any theorem that can be proved with strong induction could also be proved with ordinary induction (using a slightly more complicated induction hypothesis). But strong induction can make some proofs a bit easier. On the other hand, if $P(n)$ is easily sufficient to prove $P(n+1)$, then it's better to use ordinary induction for simplicity.

4.3.3 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 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:

> If $n \geq 8$, then there is a collection of coins whose value is n Strongs.

Notice that $P(n)$ is an implication. When the hypothesis of an implication is false, we know the whole implication is true. In this situation, the implication is said to be *vacuously* true. So $P(n)$ will be vacuously true whenever $n < 8$.[2]

We now proceed with the induction proof:

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.

Case ($n+1$ = 8): $P(8)$ holds because the Inductians can use one 3Sg coin and one fiveSg coins.

Case ($n+1$ = 9): Use a three 3Sg coins.

[2] Another approach that avoids these vacuous cases is to define

$$P'(n) ::= \text{there is a collection of coins whose value is } n+8 \text{ Strongs}$$

and prove that $P'(n)$ holds for all $n \geq 0$.

Case ($n + 1 = 10$): Use two 5Sg coins.

Case ($n + 1 \geq 11$): Then $n \geq (n+1) - 3 \geq 8$, so by the strong induction hypothesis, the Inductians can make change for $(n + 1) - 3$ 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 $n \geq 8$, the Inductians can make change for n Strong.

□

4.3.4 Unstacking

Here is another exciting 6.042 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 4.3.2. *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. □

Problem 4.3.1. Define the ***potential***, $p(S)$, of a stack, S, of blocks to be $k(k+1)/2$ where k is the number of blocks in S. Define the potential, $p(A)$, of a set, A, of stacks to be the sum of the potentials of the stacks in A.

Generalize Theorem 4.3.2 to show that for any set, A, of stacks, if a sequence of moves starting with A leads to another set, B, of stacks, then the score for this sequence of moves is $p(A) - p(B)$.

4.4 Induction versus Well Ordering

The induction axiom looks nothing like the Well Ordering Principle, but these two proof methods are closely related. In fact, we can take any proof by Strong Induction and reformat it into a Well Ordering proof.

Here's how: suppose that we have a proof by Strong Induction with induction hypothesis $P(n)$. Then we start a Well Ordering proof by defining the set of counterexamples to P, and then assuming there is a smallest counterexample, s. This means that $P(s)$ is false, but also $P(0), P(1), \ldots, P(s-1)$ are all true. At this point we reuse the proof of the inductive step in the Strong Induction proof, which shows that since $P(0), P(1), \ldots, P(s-1)$ are all true, then $P(s)$ is also true. This contradicts the assumption that $P(s)$ is false, so we have the contradiction needed to complete the Well Ordering Proof that $P(n)$ holds for all $n \in \mathbb{N}$.

Problem 4.4.1. Use strong induction to prove the Well Ordering Principle. *Hint:* Prove that if a set of nonnegative integers contains an integer, n, then it has a smallest element.

Mathematicians commonly use the Well Ordering Principle because it can lead to shorter proofs than induction. On the other hand, well ordering proofs typically involve proof by contradiction, so using it is not always the best approach. The choice of method is really a matter of style—but style does matter.

4.5 Recursive Data Types

Recursive data types play a central role in modern programming languages. 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.

Example.

Definition 4.5.1. Define a set, E, recursively as follows:

- **Base case:** $0 \in E$,
- **Constructor cases:** if $n \in E$, then
  1. $n + 2 \in E$, when $n \geq 0$;
  2. $-n \in E$, when $n > 0$.

Using this definition, we can see that $0 \in E$ by the Base case, so $0 + 2 = 2 \in E$ by Constructor case 1., and so $2 + 2 = 4 \in E$, $4 + 2 = 6 \in E$, ..., and in fact any nonnegative even number is in E by successive application of case 1. Also, by case 2., $-2, -4, -6, \cdots \in E$. So clearly all the even integers are in E.

Is anything else in E? The definition doesn't say so explicitly, but an implicit condition on a recursive definition is that the only way things get into E is as a consequence of the Base and Constructor cases. In other words, E will be exactly the set of even integers.

Another example is the set, M, of strings of *m*atched right and left parentheses. These are the strings that would be obtained if we took a sequence of fully parenthesized arithmetic (or Scheme) expressions and erased all the characters except the parentheses. Here's a recursive definition:

Definition 4.5.2. Define the set, M, of strings of matched right and left parentheses recursively as follows:

- **Base case:** $\lambda \in M$, where λ is the *empty* string,
- **Constructor case:** if $s, t \in M$, then $(s)\, t \in M$.

Here we're writing $(s)\,t$ to indicate the string that starts with a left parenthesis, followed by the sequence of parentheses (if any) in the string s, followed by a right parenthesis, and ending with the sequence of parentheses in the string t.

Using this definition, we can see that $\lambda \in S$ by the Base case, so

$$(\lambda)\,\lambda = () \in M$$

by the Constructor case, and so

$$\begin{aligned}(\lambda)\,() &= ()\,(),\\ (()\,)\,\lambda &= (()\,),\\ (()\,)\,() &\end{aligned}$$

are further strings in M by repeated applications of the Constructor case.

4.6 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 set E given by Definition 4.5.1 is exactly the set of even numbers. We already explained why all the even numbers are in E. So what's left is to show that:

Lemma. *Every number in the set E in Definition 4.5.1 is even.*

Proof. The proof is by structural induction on $n \in E$. The induction hypothesis is

$$Q(n) ::= n \text{ is even.}$$

Base case: $Q(0)$ holds since 0 is even.

Constructor cases: assuming $n \in E$ and $Q(n)$ holds, prove that

- $Q(n+2)$ holds. This is immediate, since adding 2 to an even number gives an even number.
- $Q(-n)$ holds. This is also immediate, since n is even iff $-n$ is even.

This completes the proof of the Constructor cases, and we conclude by structural induction at $Q(n)$ holds for all $n \in E$. □

Another example of structural induction comes in proving that strings of matched parentheses always have an equal number of left and right parentheses. To do this, define a predicate on strings

$$P(s) ::= \quad s \text{ has an equal number of left and right parentheses.}$$

Proof. We'll prove that $P(s)$ holds for all $s \in M$ by structural induction on the definition of $s \in M$, using $P(s)$ as the induction hypothesis.

Base case: $P(\lambda)$ holds because the empty string has zero left and zero right parentheses.

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 parentheses in s and t. So the number of left parentheses in r is $1 + n_s + n_t$.

Now from the respective hypotheses $P(s)$ and $P(t)$, we conclude that the number of right parentheses in s and t, respectively, is also n_s and n_t. So the number of right parentheses in r is $1+n_s+n_t$, which is the same as the number of left parentheses. This proves $P(r)$. We conclude by structural induction that $P(s)$ holds for all $s \in M$. □

In fact, ordinary induction can be understood as an instance of structural induction if we regard the nonnegative integers as a recursive data type:

Definition 4.6.1. 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 $\mathbb{N}$.

4.6.1 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, M, of strings of matched parentheses, we define:

Definition 4.6.2. The *depth*, $d(s)$, of a string, $s \in M$, 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 *un*ambiguous definitions of the sets M and E to ensure that functions defined recursively on the definitions of these data types would always be well-defined.

As an example of the trouble ambiguous definitions can cause, let consider defining the set, E, of even numbers as in Definitions 4.5.1, but without the conditions 1. and 2. that restrict application of the rules. Namely,

Definition 4.6.3. Define a set, E', recursively as follows:

- **Base case:** $0 \in E'$,
- **Constructor cases:** if $n \in E'$, then
  1. $n + 2 \in E'$,
  2. $-n \in E'$.

Now Definition 4.6.3 is perfectly legitimate, and we could us it to prove by structural induction that E' also is the set of even integers, that is, $E' = E$. But Definition 4.6.3 is ambiguous. For example, $0 \in E'$ by the base case, but also $0 = -0 \in E'$ by applying constructor case 2 to the base case. This begins to matter when we try to define a function, s, from E' to nonnegative integers based on Definition 4.6.3:

$$\begin{aligned} s(0) &::= 1, \\ s(n+2) &::= 1 + s(n), \\ s(-n) &::= 1 + s(n). \end{aligned}$$

So $s(0) ::= 1$ by the base case of this definition, and also $s(0) = s(-0) ::= 1 + s(0) = 1 + 1 = 2$ by the second constructor case, which shows that these rules are inconsistent.

On the other hand, using the unambiguous Definition 4.5.1 of E, essentially the same definition of S works just fine. Namely, define

$$\begin{aligned} s(0) &::= 1, && \\ s(n+2) &::= 1 + s(n), && \text{for } n \geq 0 \\ s(-n) &::= 1 + s(n) && \text{for } n > 0. \end{aligned}$$

Now $s(n)$ is unambiguously defined, and in fact is precisely the (unique) number of steps required to construct $n \in E$ according to the unambiguous Definition 4.5.1 of E.

4.6.2 Recursive Functions on Nonnegative Integers

Definition 4.6.1 of the nonnegative integers as a recursive data type 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:

$$\begin{aligned} \text{fib}(0) &::= 0, \\ \text{fib}(1) &::= 1, \\ \text{fib}(n) &::= \text{fib}(n-1) + \text{fib}(n-2) \qquad \text{for } n \geq 2. \end{aligned}$$

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.

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.

$$f_1(n) ::= 2 + f_1(n-1). \tag{4.5}$$

This "definition" has no base case. If some function, f_1, satisfied (4.5), so would a function obtained by adding a constant to the value of f_1. So equation (4.5) does not uniquely define an f_1.

$$f_2(n) ::= \begin{cases} 0, & \text{if } n \text{ is divisible by 2,} \\ 1, & \text{if } n \text{ is divisible by 3,} \\ 2, & \text{otherwise.} \end{cases} \tag{4.6}$$

This "definition" is inconsistent: it requires $f_2(6) = 0$ and $f_2(6) = 1$, so (4.6) doesn't define anything.

$$f_3(n) ::= \begin{cases} 0, & \text{if } n = 0, \\ f_3(n+1) & \text{otherwise.} \end{cases} \tag{4.7}$$

Any function that is 0 at 0 and constant everywhere else satisfies (4.7), so it also does not uniquely define anything.

A Mysterious Function

Mathematicians have been wondering about this function specification for a while:

$$f_4(n) ::= \begin{cases} 1, & \text{if } n \leq 1, \\ f_4(n/2) & \text{if } n > 1 \text{ is even}, \\ f_4(3n+1) & \text{if } n > 1 \text{ is odd}. \end{cases} \tag{4.8}$$

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 (4.8), 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 (4.8) equals 1 for all n up to over a billion.

Quick exercise: Why does the constant function 1 satisfy (4.8)?

4.7 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.[3]

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.

4.7.1 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 4.1.

[3]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.

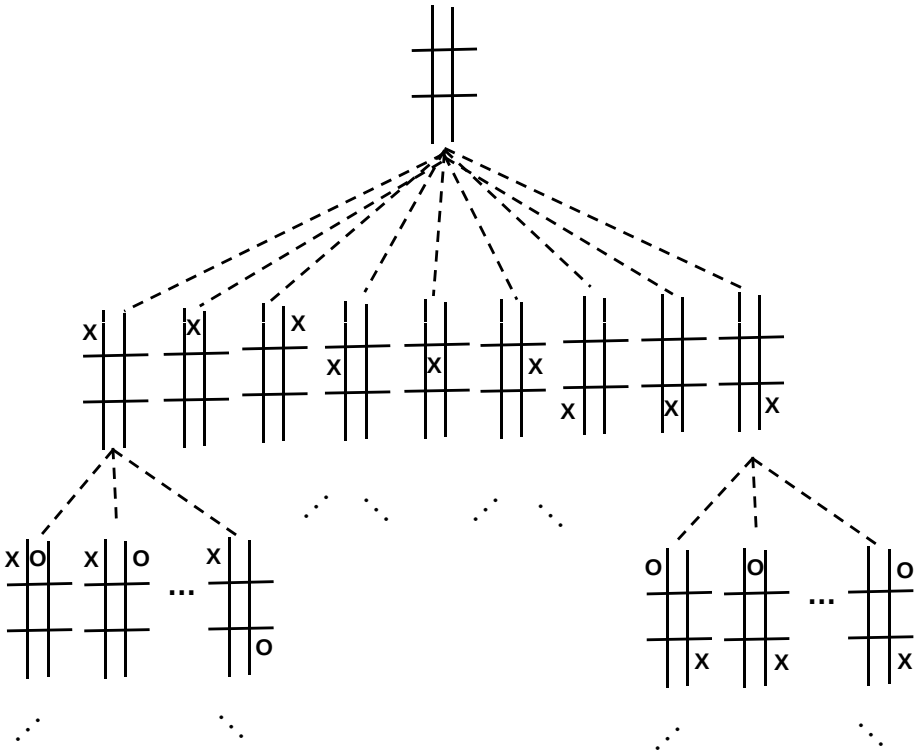


Figure 4.1: The Top of the Game Tree for Tic-Tac-Toe.

Definition 4.7.1. 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.
- $$\langle P, \langle \texttt{lose} \rangle \rangle,$$
  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 games trees starting at possible next patterns after P.

For example, if

$$P_0 = \begin{array}{|c|c|c|}\hline O & X & O \\ \hline X & O & X \\ \hline X & & \\ \hline \end{array}$$

$$Q_1 = \begin{array}{|c|c|c|}\hline O & X & O \\ \hline X & O & X \\ \hline X & & O \\ \hline \end{array}$$

$$Q_2 = \begin{array}{|c|c|c|}\hline O & X & O \\ \hline X & O & X \\ \hline X & O & \\ \hline \end{array}$$

$$R = \begin{array}{|c|c|c|}\hline O & X & O \\ \hline X & O & X \\ \hline X & O & X \\ \hline \end{array}$$

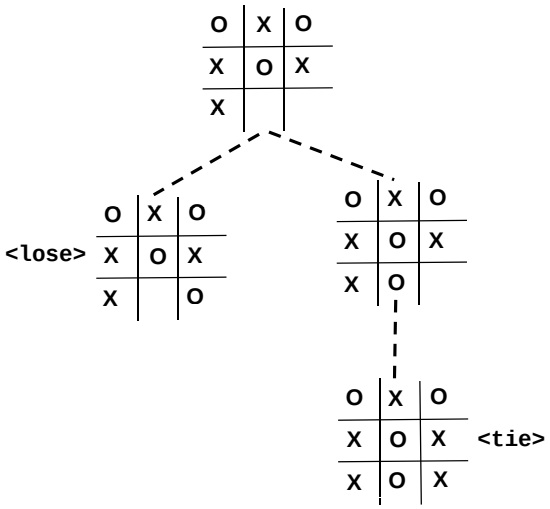


Figure 4.2: Game Tree for the Tic-Tac-Toe game starting at P_0.

the game tree starting at P_0 is pictured in Figure 4.2.

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.")

4.7.2 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 suppose we consider an *n-game Tic-Tac-Toe tournament* where the tournament winner is the one who wins more of $n > 0$ individual Tic-Tac-Toe games. (If they each win an equal number of individual games, then the tournament is a tie.)

Now we can consolidate all these tournaments into a single game we can call *Tournament-Tic-Tac-Toe*: the first player in Tournament-Tic-Tac-Toe chooses any integer $n > 0$, and then the players play an n-game tournament. Now there are infinitely many possible first moves: the first player can choose $n = 1$, or $n = 2$, or $n = 3$, or .... But still, it's obvious that every possible play of Tournament-Tic-Tac-Toe is finite, 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 Tournament-Tic-Tac-Toe: even though every play of Tournament-Tic-Tac-Toe must come to an end, there is no longer any 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.

With Tournament-Tic-Tac-Toe recognized as a 2PTG, we can go on to Tournament^2-Tic-Tac-Toe where the first player chooses a number, $m > 0$, of Tournament-Tic-Tac-Toe games to play, and the second player acts as the first player in each of the m Tournament-Tic-Tac-Toe games to be played. Then, of course, there's Tournament^3-Tic-Tac-Toe....

4.7.3 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.

Like 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. 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. This leads to the following very simple —perhaps deceptively simple —general definition.

Definition 4.7.2. The 2PTG, *game trees for two-person terminating games of perfect information* are defined recursively as follows:

- **Base cases:**
$$\begin{array}{ll} \langle \texttt{leaf}, \texttt{win} \rangle & \in \text{2PTG, and} \\ \langle \texttt{leaf}, \texttt{lose} \rangle & \in \text{2PTG.} \end{array}$$

- **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 games 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 4.7.3. 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 4.7.4. *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. □

4.7.4 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 4.7.5. ***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 4.7.5 guarantees a winning strategy, its proof gives no clue which player has it. For most familiar 2PTG's like Checkers, Chess, Go, ..., no one knows which player has a winning strategy.

4.7.5 Structural Induction versus Ordinary Induction

In Computer Science, structural induction is the natural, preferred approach to proving properties of recursive data types. So you really should learn it.

Students will sometimes try to avoid structural induction in favor of ordinary induction by assigning nonnegative integer sizes to data items and then using ordinary induction on size. This works pretty generally, since each recursive datum can be assigned a size equal to the *smallest number of constructor steps* needed to build it. In this way, ordinary induction remains a viable, though more cumbersome, alternative proof method for all recursive data types that come up in practice. But this doesn't work for infinite games, because not only are such games infinite, but the number of constructor steps to build them is infinite, so there's no apparent measure of game size that allows structural induction to be replaced by ordinary induction. In fact, it can't be done: it's a *meta*mathematical fact (that is, a mathematical fact *about* properties of Mathematics) that structural induction is more powerful than ordinary induction when it comes to reasoning about such infinite data items.[4]

We admit that infinite games are a far fetched concern for Computer Science. Truth be told, we introduced infinite games just to help teach about structural induction by making it impossible to avoid. But we do think it's cool that infinite games are no harder to handle than finite ones, because the proofs by structural induction work the same way for both.

[4]There is a generalization of induction on nonnegative integers to induction on things called *ordinals*. Induction on ordinals is as powerful as structural induction, but the theory of ordinals is not a topic that belongs in an introduction to discrete Math.

Chapter 5

State Machines; Stable Marriage

5.1 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.

5.1.1 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" and a pair (p, q) in the graph of the relation is called a "transition." The transition from state p to state q will be written $p \longrightarrow q$. 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]

Example 5.1.1. A bounded counter, which counts from 0 to 99 and overflows at 100. The transitions are pictured in Figure 5.1, with start state zero.

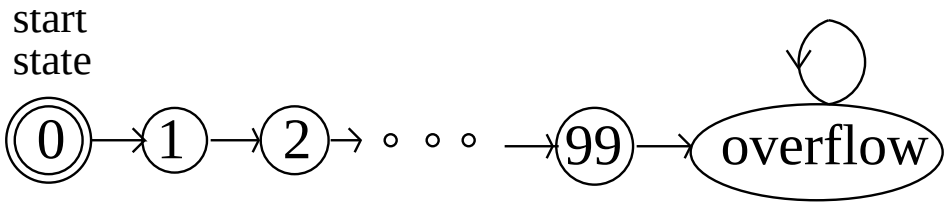


Figure 5.1: *State transitions for the 99-bounded counter.*

This machine isn't much use once it overflows, since it has no way to get out of its overflow state.

[1] We do name states, as in Figure 5.1, so we can talk about them, but the names aren't part of the state machine.

Example 5.1.2. An unbounded counter is similar, but has an infinite state set. This is harder to draw :-)

Example 5.1.3. **I**n 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.

Quickie exercise: Which states of the Die Hard 3 machine have direct transitions to exactly two states?

5.1.2 Reachability and 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) sequence of transitions through successive states beginning at the start state corresponds to a possible system behavior; such a sequence 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 *invariant* properties of states.

Definition 5.1.4. An *invariant* for 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.

Now we can reformulate Induction in a convenient form for state machines:

The Invariant Principle
If a predicate is an invariant of a state machine, and the predicate holds for the start state, then it holds for all reachable states.

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 invariant predicate, $P(b, l)$, to be that b and l are nonnegative integer mulitples of 3. So P obviously holds for the state state $(0, 0)$.

To prove that P is an 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 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!

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 prove 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 invariant predicate, 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 at an invariant is to let $P(q)$ be the predicate 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 an 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 an invariant, then we have proven that the robot never reaches $(1, 0)$ because the sum $1 + 0$ of its coordinates is not an even number, but the sum $0 + 0$ of the coordinates of the start state is an even number.

Theorem 5.1.5. *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 an 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 5.1.6. *The robot cannot reach* $(1, 0)$.

Problem 5.1.1. 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).

Robert W. Floyd

The Invariant Principle was formulated by Robert Floyd at Carnegie Tech in 1967[a]. 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. A eulogy describing Floyd's life and work by his closest colleague, Don Knuth, can be found at http://www.acm.org/pubs/membernet/stories/floyd.pdf.

[a] The following year, Carnegie Tech was renamed Carnegie-Mellon Univ.

5.1.3 Sequential algorithm examples

Proving Correctness

Robert Floyd, who pioneered modern approaches 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.

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 invariance of P follows immediately from

Lemma 5.1.7. *For all $m, n \in \mathbb{N}$ such that $n \neq 0$,*

$$\gcd(m, n) = \gcd(n, \text{remainder}(m, n)). \tag{5.1}$$

Lemma 5.1.7 is easy to prove, and we'll leave it to the reader (a proof will appear in later Notes on elementary Number Theory). So 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 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.

The Extended Euclidean Algorithm

An important fact about the $\gcd(a, b)$ is that it equals an integer linear combination of a and b, that is,

$$\gcd(a, b) = sa + tb \tag{5.2}$$

for some $s, t \in \mathbb{Z}$. We'll see some nice proofs of (5.2) in later Notes, but there is also an extension of the Euclidean Algorithm that efficiently, if obscurely, produces the desired s and t. In particular, given nonnegative integers x,y, with $y > 0$, we claim the following procedure[2] halts with integers s, t in registers `S` and `T` satisfying (5.2).

Inputs: $a, b \in \mathbb{N}, b > 0$.

Registers: `X,Y,S,T,U,V,Q`.

Extended Euclidean Algorithm:

```
X := a; Y := b; S := 0; T := 1; U := 1; V := 0;
loop:
if Y divides X, then halt
else
  Q := quotient(X,Y);
         ;;the following assignments in braces are SIMULTANEOUS
 {X := Y,
  Y := remainder(X,Y);
  U := S,
  V := T,
  S := U - Q * S,
  T := V - Q * T};
goto loop;
```

Note that `X,Y` behave exactly as in the Euclidean GCD algorithm in Section 5.1.3, except that this extended procedure stops one step sooner, ensuring that $\gcd(x, y)$ is in `Y` at the end. So for all inputs x, y, this procedure terminates for the same reason as the Euclidean algorithm: the contents,

[2] This procedure is adapted from Aho, Hopcroft, and Ullman's text on algorithms.

y, of register `Y` is a nonnegative integer-valued variable that strictly decreases each time around the loop.

We claim that invariant properties that can be used to prove partial correctness are:

$$\gcd(X, Y) = \gcd(a, b), \tag{5.3}$$
$$Sa + Tb = Y, \text{ and} \tag{5.4}$$
$$Ua + Vb = X. \tag{5.5}$$

To verify these invariants, note that invariant (5.3) is the same one we observed for the Euclidean algorithm. To check the other two invariants, let x, y, s, t, u, v be the contents of registers `X,Y,S,T,U,V` at the start of the loop and assume that all the invariants hold for these values. We must prove that (5.4) and (5.5) hold (we already know (5.3) does) for the new contents x', y', s', t', u', v' of these registers at the next time the loop is started.

Now according to the procedure, $u' = s, v' = t, x' = y$, so invariant (5.5) holds for u', v', x' because of invariant (5.4) for s, t, y. Also, $s' = u - qs, t' = v - qt, y' = x - qy$ where $q = \text{quotient}(x, y)$, so

$$s'a + t'b = (u - qs)a + (v - qt)b = ua + vb - q(sa + tb) = x - qy = y',$$

and therefore invariant (5.4) holds for s', t', y'.

Also, it's easy to check that all three invariants are true just before the first time around the loop. Namely, at the start $X = a, Y = b, S = 0, T = 1$ so $Sa + Tb = 0a + 1b = b = Y$ so (b) holds; also $U = 1, V = 0$ and $Ua + Vb = 1a + 0b = a = X$ so (5.5) holds. So by the Invariant Principle, they are true at termination. But at termination, the contents, Y, of register `Y` divides the contents, X, of register `X`, so invariants (5.3) and (5.4) imply

$$\gcd(a, b) = \gcd(X, Y) = Y = Sa + Tb.$$

So we have the gcd in register `Y` and the desired coefficients in `S`, `T`.

Now we don't claim that this verification offers much insight. In fact, if you're not wondering how somebody came up with this concise program and invariant, you:

- are blessed with an inspired intellect allowing you to see how this program and its invariant were devised,
- have lost interest in the topic, or
- haven't read this far.

If none of the above apply to you, we can offer some reassurance: you're not expected to understand this program. It was presented here simply as another example of application of the invariant method (plus, we'll need a procedure like this when we take up number theory based cryptography in a couple of weeks).

An invariant is really just an induction hypothesis, and as with induction, finding the right hypothesis is usually the hard part. Once the right hypothesis or invariant is found, checking that it works is usually routine, as this program illustrates:

> **Given the right invariant, it can be easy to verify a program even if you don't understand it.**

5.1.4 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 : 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 5.1.8. 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 $\mathtt{Y}$, respectively, that were nonnegative integer-valued and strictly decreasing. We can summarize this approach to proving termination as follows:

Theorem 5.1.9. *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 5.1.9 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.

There are cases where it's easier to prove termination based on more general partial orders than "less-than" on $\mathbb{N}$. Termination is guaranteed whever there is a derived variable that strictly decreases with respect to any well-founded partial order.

Definition 5.1.10. Let $\prec$ be a strict partial order on a set, A. A derived variable $f : Q \to A$ is *strictly decreasing* with respect to $\prec$ iff

$$q \longrightarrow q' \text{ implies } f(q') \prec f(q).$$

Also, f is *weakly decreasing* iff

$$q \longrightarrow q' \text{ implies } f(q') \preceq f(q).$$

where $\preceq$ is the weak partial order corresponding to $\prec$, namely,

$$[a_1 \preceq a_2] ::= [(a_1 \prec a_2) \text{ or } (a_1 = a_2)].$$

Strictly increasing and *weakly increasing* derived variables are defined similarly.[3]

[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.

Theorem 5.1.11. *If there exists a derived variable for a state machine that is strictly decreasing with respect to some well-founded partial order, then every execution terminates.*

Theorem 5.1.11 follows immediately from the observation in Notes 3 that a well-founded partial order has no infinite decreasing sequences.

Note that the existence of a nonnegative integer-valued *weakly* decreasing derived variable does not guarantee that every execution terminates. That's because an infinite execution could proceed through states in which a weakly decreasing variable remained constant.

A Southeast Jumping Robot

Here's a contrived but simple example of proving termination based on a variable that is strictly decreasing over a well-founded order. Let's think about a robot positioned at an integer lattice-point in the Northeast quadrant of the plane, that is, at $(x, y) \in \mathbb{N}^2$.

At every second when it is away from the origin, $(0, 0)$, the robot must make a move, which may be

- a unit distance West when it is not at the boundary of the Northeast quadrant (that is, $(x, y) \longrightarrow (x - 1, y)$ for $x > 0$), or
- a unit distance South combined with an arbitrary jump East (that is, $(x, y) \longrightarrow (z, y - 1)$ for $z \geq x$).

Claim 5.1.12. *The robot will always get stuck at the origin.*

If we think of the robot as a nondeterministic state machine, then Claim 5.1.12 is a termination assertion. The Claim may seem obvious, but it really has a different character than termination based on nonnegative integer-valued variables. That's because, even knowing that the robot is at position $(0, 1)$, for example, there is no way to bound the time it takes for the robot to get stuck. It can delay getting stuck for as many seconds as it wants by making its next move to a distant point in the Far East. This rules out proving termination using Theorem 5.1.9.

So does Claim 5.1.12 still seem obvious?

Well it is if you see the trick: if we reverse the coordinates, then every robot move goes to a position that is smaller under lexicographic order. More precisely, let $f : \mathbb{N}^2 \to \mathbb{N}^2$ be the derived variable mapping a robot state —its position (x, y) —to $(y, x) \in \mathbb{N}^2$. Now $(x, y) \longrightarrow (x', y')$ is a legitimate robot move iff $f((x', y')) \prec_{\text{lex}} f((x, y))$. In particular, f is a strictly $\prec_{\text{lex}}$-decreasing derived variable, so Theorem 5.1.11 proves that the robot always get stuck as claimed.

5.2 The Stable Marriage Problem

Okay, frequent public reference to derived variables may not help your mating prospects. But they can help with the analysis!

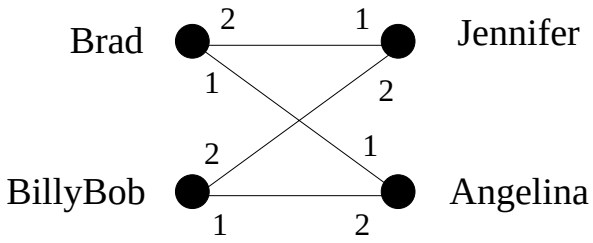
5.2.1 The Problem

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 her rank 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 doing 6.042 homework 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 5.2.1. A matching is *stable* iff it has 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 5.2 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:

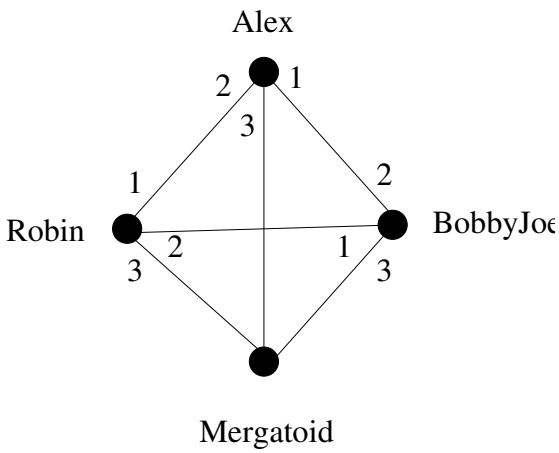


Figure 5.2: Some preferences with no stable buddy matching.

Lemma. *There is no stable buddy matching among the four people in Figure 5.2.*

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.

5.2.2 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 suitor, "We might get engaged. Come back tomorrow." To the others, 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.

5.2.3 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 5.2.2. 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 $\to$ 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 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.

5.2.4 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.

5.2.5 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 5.2.3. *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 5.2.4. *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. □

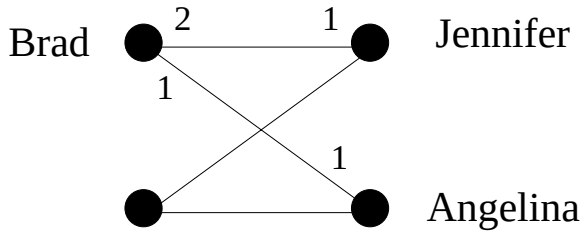
5.2.6 ...Especially the Boys

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 5.2.5. 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 5.2.6. *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 5.2.7. *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 5.2.6, 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. □

5.2.7 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 unbeknownst to 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.) 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.[4]

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.

[4]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.

Chapter 6

Simple Graphs

6.1 Simple Graphs

6.1.1 Introduction

Graphs are an incredibly useful structure in Computer Science! They arise 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 first we are going to talk about something else. Namely, sex. The question that we'll address is, on average, who has more opposite-gender partners, men or women? This has 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, which fits right in with stereotype of male promiscuity versus female sexual selectivity.

Other studies have found that the disparity is even larger. In particular, ABC News claims 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, claimed to be one of the most scientific ever done with only a 2.5% margin of error. It was called "American Sex Survey: A peak between the sheets"—- hmmmmmm, doesn't sound so scientific. The promotion for the study is even better:

> A ground breaking ABC News "Primetime Live" survey finds a range of eye-popping sexual activities, fantasies and attitudes in this country, confirming some conventional wisdom, exploding some myths – and venturing where few scientific surveys have gone before.

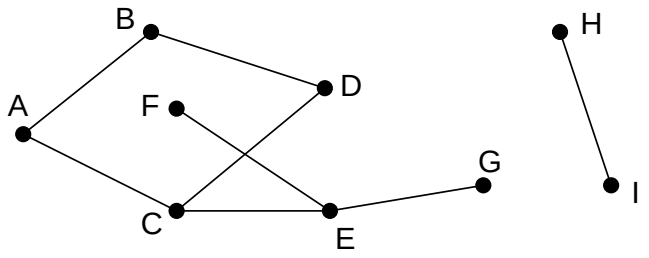
Probably that last part about going where few scientific surveys have gone before is pretty accurate!

And 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.

6.1.2 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 6.1.1. 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 in dots-and-lines diagram above is a captured by the following simple graph:

$$\begin{aligned} V &= \{A, B, C, D, E, F, G, H, I\} \\ E &= \{\{A, B\}, \{A, C\}, \{B, D\}, \{C, D\}, \{C, E\}, \{E, F\}, \{E, G\}, \{H, I\}\}. \end{aligned}$$

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.

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*, and adjacent vertices are sometimes called *neighbors*. 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 6.1.1 are worth noting right from the start:

1. Simple graphs do not have edges going from a vertex back around to itself (called a *self-loop*).
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 these Notes we'll only be considering simple graphs, so we'll just call them "graphs" from now on.

6.1.3 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 men, and W, which contains all the women.[1] We'll put an edge between a man and a woman iff they have been sexual partners. This graph is pictured in Figure 6.1 with men on the left and women on the right.

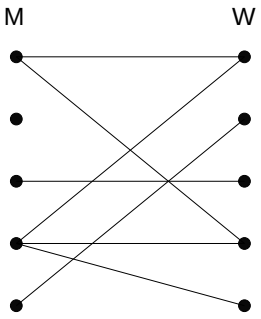


Figure 6.1: The sex partners graph

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 $|W| \approx 152.4M$. 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

[1] For simplicity, we'll ignore the possibility of someone being both, or neither, a man and a woman.

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 men, so they have a higher ratio. So 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 men exaggerate their number of partners —or maybe women downplay theirs —but these explanations are speculative. Interestingly, the principal author of the government study reported that she knew the results had to be wrong, but felt she had no choice but to publish the data collected in the survey.

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. Well, that just follows from the definition of "minority"!

6.1.4 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:

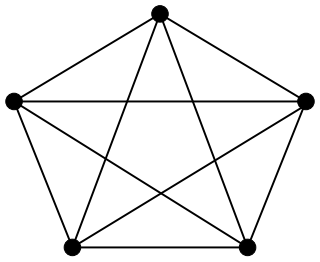
Theorem 6.1.2. *The sum of the degrees of the vertices in a graph equals twice the number of edges.*

Proof. Every edge contributes two to the sum of the degrees, one for each of its endpoints. □

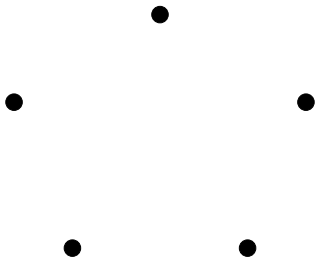
Theorem 6.1.2 is sometimes called the *Handshake Theorem*: if we total up the number of people each person at a party shakes hands with, the total will be twice the number of handshakes that occurred.

6.1.5 Some Common Graphs

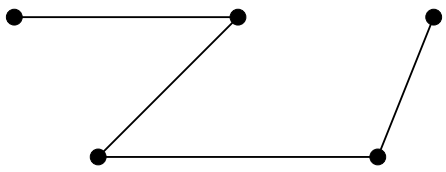
Some graphs come up so frequently that they have names. The *complete graph* on n vertices, also called K_n, has an edge between every two vertices. Here is K_5:



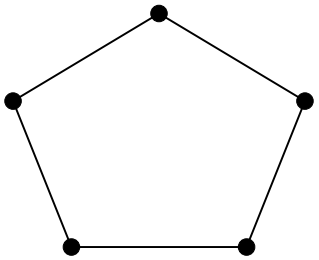
The *empty graph* has no edges at all. Here is the empty graph on 5 vertices:



Another 5 vertex graph is L_4, the *line graph* of length four:



And here is C_5, a *simple cycle* with 5 vertices:



6.1.6 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." Graphs G_1 and G_2 are *isomorphic* if there exists a bijection between the vertices in G_1 and the vertices in G_2 such that there is an edge between two vertices in G_1 if and only if there is an edge between the two corresponding vertices in G_2. For example, take the following bijection between vertices in the two graphs above:

A corresponds to 1	B corresponds to 2
D corresponds to 4	C corresponds to 3.

Now 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. Therefore, the two graphs are isomorphic. The bijection itself is called an *isomorphism*.

Definition 6.1.3. 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.

Two isomorphic graphs may be drawn very differently. For example, here are two different ways of drawing C_5:

Isomorphism captures all 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. So a property like "having three vertices of degree 4" is preserved under isomorphism, while "having a vertex that is an integer" is not preserved. In particular, if one graph has three vertices of degree 4 and another does not, they can't be isomorphic. Similarly, if one graph has an edge that is incident to a degree 8 vertex and a degree 3 vertex, then any isomorphic graph must also have such an edge.

Looking for properties like these 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 which 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.

6.2 Connectedness

6.2.1 Paths and Simple Cycles

A *path* in a graph describes how to get from one vertex to another following edges of the graph. Formally,

Definition 6.2.1. 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 ***length*** of the path is defined to be k. An edge, u—w, is ***traversed*** n ***times*** by the path if there are n different values of i such that v_i—$v_{i+1} = v$—w.

The path is ***simple***[2] iff all the v_i's are different, that is, $v_i = v_j$ only if $i = j$.

For example, the graph in Figure 6.2 has a length 6 simple path A,B,C,D,E,F,G. This is the longest simple path in the graph.

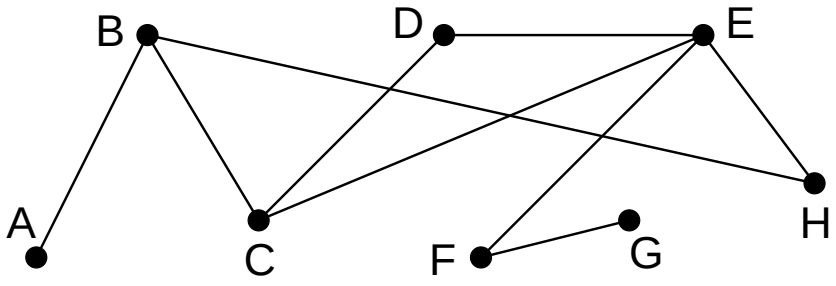


Figure 6.2: *A graph with 3 simple cycles.*

Notice that 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. The length 6 path A,B,C,D,E,F,G is actually a sequence of seven vertices.

A *cycle* can 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 6.2. This path suggests that the cycle begins and ends at vertex B, but a cycle isn't intended to have a 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 6.2 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. [3]

[2]Heads up if you read another graph theory text: what amounts to paths are commonly referred to as "walks," and simple paths are referred to as a just "paths". Likewise, what we will call *cycles* and *simple cycles* are commonly called "closed walks" and just "cycles".

[3]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.

6.2.2 Connected Components

Definition 6.2.2. Two vertices in a graph are said to be ***connected*** if 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 6.3 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.

Figure 6.3: *One graph with 3 connected components.*

Definition 6.2.3. A graph is said to be ***connected*** if 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.

6.2.3 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 remains connected as long as fewer than k "direct connections between components fail," that is, it stays connected even if as many as $k-1$ edges are deleted. More precisely:

Definition 6.2.4. 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[4] 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.

[4]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."

For example, in the graph in Figure 6.2, 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 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 takes some ingenuity to prove this even for the case $k = 2$.

6.2.4 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 6.2.5. *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 6.2.6. *For any path of length k in a graph, there is a simple path of length* at most *k with the same endpoints.*

6.2.5 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 6.2.7. *Every graph with v vertices and e edges has at least $v - e$ connected components.*

Of course for Theorem 6.2.7 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 Induction step.

The theorem now follows by induction. □

Corollary 6.2.8. *Every connected graph with e vertices has at least $e - 1$ edges.*

A couple of points about the proof of Theorem 6.2.7 are worth noting. First, notice that 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.

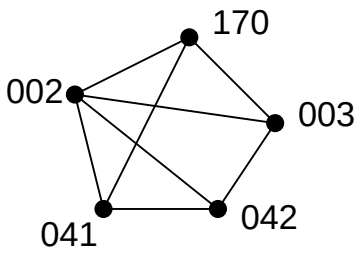
The second point is more subtle. Notice that in the inductive step, we took an arbitrary $(n + 1)$-edge graph, threw out an edge so that we could apply the induction assumption, and then put the edge back. You'll see this shrink-down, grow-back process very often in the inductive steps of proofs related to graphs. This might seem like needless effort; why not start with an e-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. You'll see an example in class. Always use shrink-down, grow-back arguments, and you'll never fall into this trap.

6.3 Coloring Graphs

In section 6.1.3, 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. For example, 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

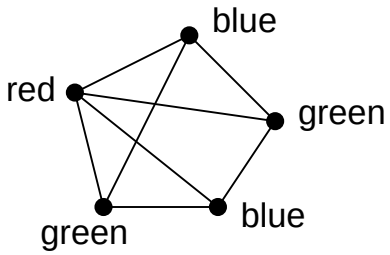
slots for the hundreds of courses, and exam period would run all year! So, the Schedules Office would also like to keep exam period short.

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. The minimum value of k for which a coloring exists is called the ***chromatic number***, $\chi(G)$, of 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).

6.3.1 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 6.3.1. *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.

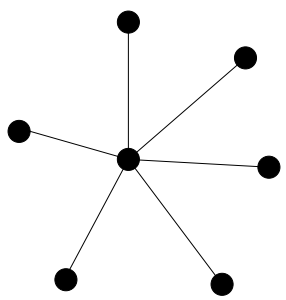
Proof. We use induction on the number of vertices in the graph, which we denote by n. Let $P(n)$ be the proposition that an n-vertex graph with maximum degree at most k is $(k+1)$-colorable.

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 6.3.1 gives the best possible bound. This means that Theorem 6.3.1 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!

6.3.2 Why coloring?

Coloring problems come up in all sorts of applications. 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 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 mentioned in Week 1 Notes. 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 6.5. Then in Section 6.7, we'll develop enough planar graph theory to present an easy proof that planar graphs are 5-colorable.

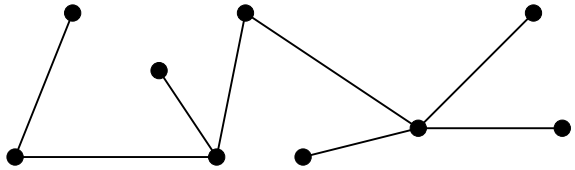
6.4 Trees

Trees are a fundamental data structure in Computer Science, and there are many kinds, for example rooted, ordered, or binary trees. In this section we focus on the purest kind of tree. Namely, we use the ***tree*** to mean a connected graph without simple cycles.

A graph with no simple cycles is called ***acyclic***; so trees are acyclic connected graphs.

6.4.1 Tree Properties

Here is an example of a tree:



A vertex of degree at most one is called a ***leaf***. Note that the only case where a tree can have a vertex of degree zero is a graph with a single vertex. In this example, there are 5 leaves.

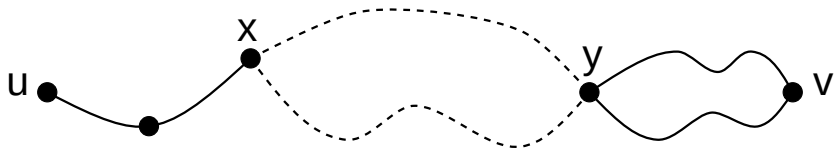
The graph shown above would no longer be a tree if any edge were removed, because it would no longer be connected. The graph would also not remain a tree if any edge were added between two of its vertices, because then it would contain a simple cycle. Furthermore, note that there is a unique path between every pair of vertices. These features of the example tree are actually common to all trees.

Theorem 6.4.1. *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 a 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 $0 + 1 = 1$.

   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.

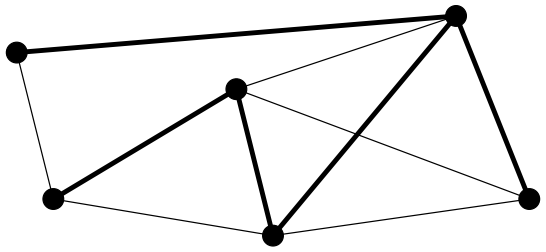
   We will let the reader verify that deleting a vertex of degree 1 (and its incident edge) from any connected graph leaves a connected subgraph. So by (1), deleting v and its incident edge gives a smaller tree, and this smaller tree has one more vertex than edge by induction. If we reattach the vertex, 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.

6.4.2 Spanning Trees

Trees are everywhere. In fact, every connected graph contains a 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 6.4.2. *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:

$$v_0\text{—}v_1, v_1\text{—}v_2, \ldots, v_n\text{—}v_0$$

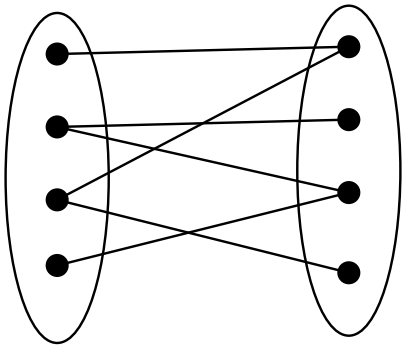
Suppose that we remove the last edge, v_n—v_0. If a pair of vertices x and y was joined by a path not containing v_n—v_0, then they remain joined by that path. On the other hand, if x and y were joined by a path containing v_n—v_0, then they remain joined by a path containing the remainder of the cycle. 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. □

6.5 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 6.5.1. 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.

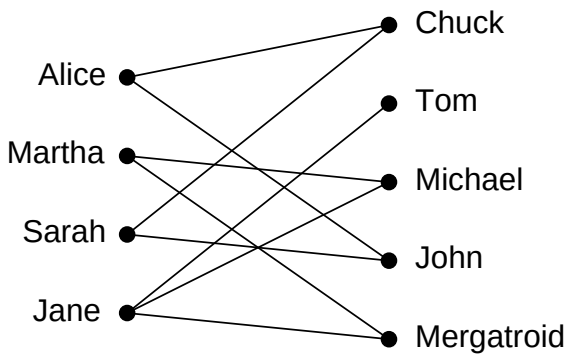
Theorem 6.5.2. *A graph is bipartite iff it has no odd-length cycle.*

The proof of Theorem 6.5.2 will be on a problem set.

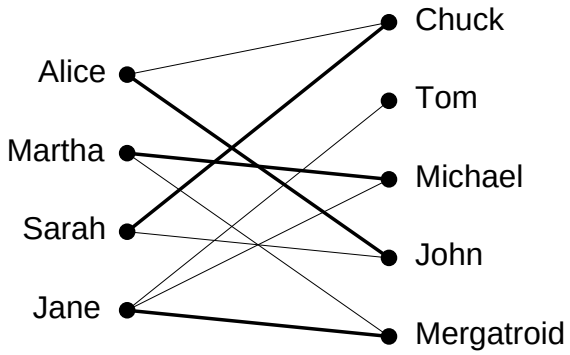
6.6 Bipartite Matchings

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.

6.6.1 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 6.6.1. *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. If $|G| = 1$, then the matching condition implies that the lone girl likes at least one boy, and so a matching exists. Now suppose that $|G| \geq 2$. There are two possibilities:

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.

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 will show that the matching condition holds for the remaining boys and girls, and so we can match the rest of the girls by induction as well.

   To that end, 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:

   $$\left|X \cup X'\right| \leq \left|Y \cup Y'\right|$$

   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.

In both cases, there is a matching for the girls. 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.

6.6.2 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*[5] 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 6.6.2 (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.*

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 6.6.3. 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 6.6.4. *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 degree-constrained graphs come up in several natural examples given in class and the problem set.

[5] 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.

6.7 Planar Graphs

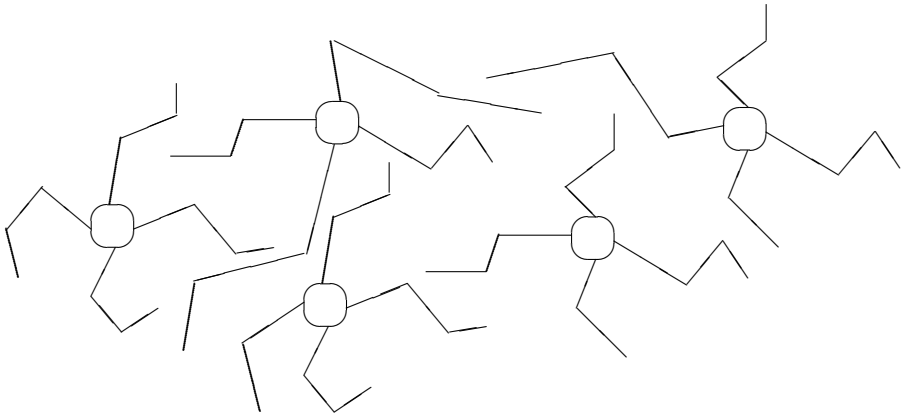
6.7.1 Drawing Graphs in the Plane

Here are three dogs and three houses.

Dog Dog Dog

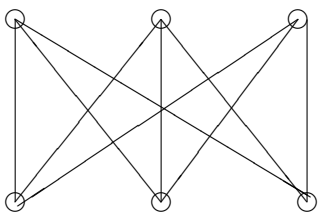
Can you find a path from each dog to each house such that no two paths intersect?

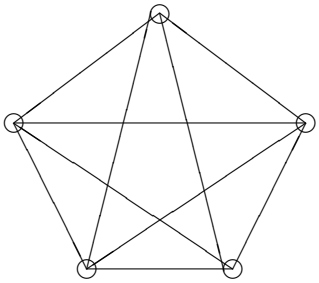
A *quadapus* is a little-known animal similar to an octopus, but with four arms. Here are five quadapi resting on the seafloor:



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.

6.7.2 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 6.4 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 6.4 form a simple cycle. For example, labeling the vertices as in Figure 6.5, the simple cycles for the face boundaries are

`abca` `abda` `bcdb` `acda`.

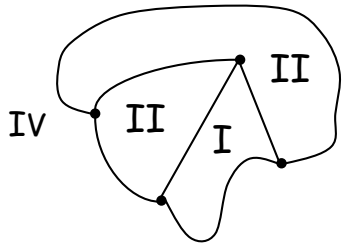


Figure 6.4: A Planar Drawing with Four Faces.

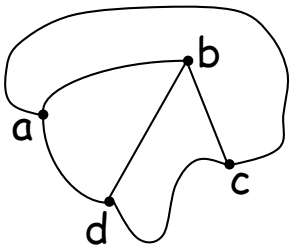


Figure 6.5: The Drawing with Labelled Vertices.

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 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 6.6. Now the cycle around the outer face is

`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 6.7. Now the cycle around the inner face is

`rstvxyxvwvtur`,

because it has to traverse *every* edge of the dongle twice —once "coming" and once "going."

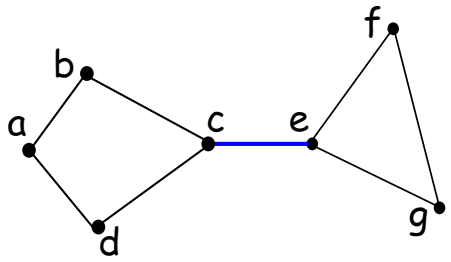


Figure 6.6: A Planar Drawing with a *Bridge*.

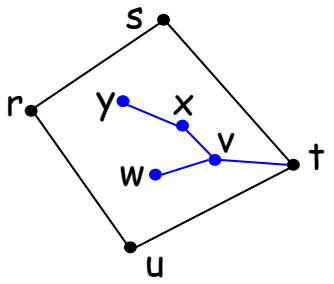


Figure 6.7: A Planar Drawing with a *Dongle*.

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.

6.7.3 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 6.7.1. 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[6]

$$a \ldots ba \quad \text{and} \quad ab \cdots a,$$

as illustrated in Figure 6.8.

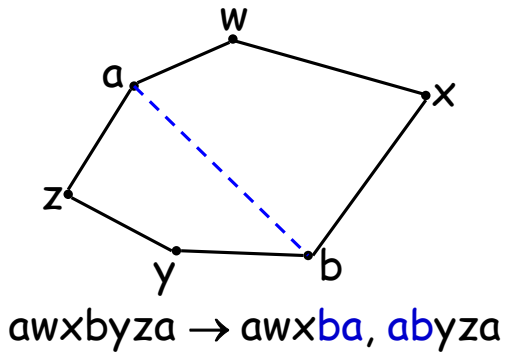


Figure 6.8: The Split a Face Case.

- **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

  $$a \ldots a.$$

  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 6.9, where the faces of G and H are:

  $$G : \{\texttt{axyza}, \texttt{axya}, \texttt{ayza}\} \qquad H : \{\texttt{btuvwb}, \texttt{btvwb}, \texttt{tuvt}\},$$

[6] 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.

and after adding the bridge a—b, there is a single connected graph with faces

$$\{\texttt{axyzabtuvwba}, \texttt{axya}, \texttt{ayza}, \texttt{btvwb}, \texttt{tuvt}\}.$$

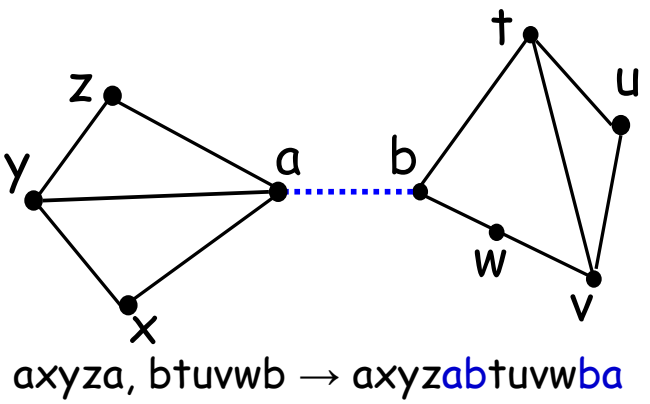


Figure 6.9: The Add Bridge Case.

An arbitrary graph is *planar* iff each of its connected components has a planar embedding.

6.7.4 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 an 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.

6.7.5 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 6.7.2 (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 6.4, $|V| = 4$, $|E| = 6$, 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 cross 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. □

6.7.6 Number of Edges versus Vertices

Like Euler's formula, the following lemmas follow by structural induction directly from the definition of planar embedding.

Lemma 6.7.3. *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 6.7.4. *In a planar embedding of a connected graph with at least three vertices, each face is of length at least three.*

Corollary 6.7.5. *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 6.7.3, 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 6.7.4, when $v \geq 3$, each face boundary is of length at least three, so this sum is at least $3f$. This implies that

$$3f \leq 2e. \tag{6.1}$$

But $f = e - v + 2$ by Euler's formula, and substituting into (6.1) gives

$$\begin{aligned} 3(e - v + 2) &\leq 2e \\ e - 3v + 6 &\leq 0 \\ e &\leq 3v - 6 \end{aligned}$$

□

Corollary 6.7.5 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 6.7.5 required for K_5 to be planar, which proves

Lemma 6.7.6. *K_5 is not planar.*

Another consequence is

Lemma 6.7.7. *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 6.7.5. □

Planar 5-colorability

There are two more familiar properties of planar graphs that we need to know.

Lemma 6.7.8. *Deleting a vertex from a planar graph, along with all its incident edges of course, leaves another planar graph.*

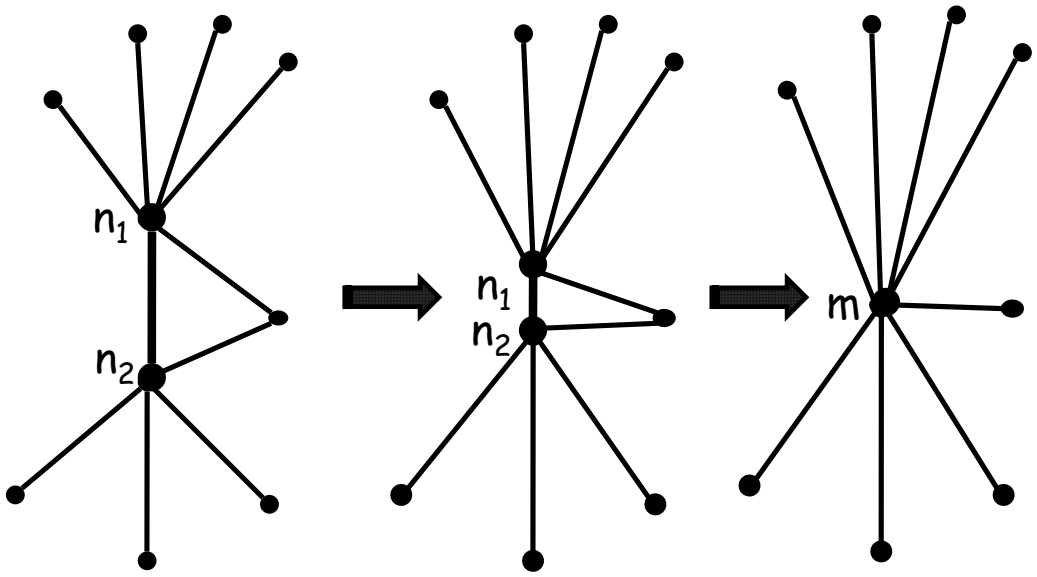


Figure 6.10: Merging adjacent vertices n_1 and n_2 into new vertex, m.

Lemma 6.7.9. *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 6.10.

Of course Lemmas 6.7.6 and 6.7.7 can be proved by structural induction, but the proofs are kind of boring, and we hope you'll be relieved that we're going to omit them. (If you insist, we can add proving one of them as a problem on the next problem set.)

Now we've got all the simple facts we need to prove

Theorem 6.7.10. *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 6.7.7 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 6.7.8, 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 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 subgraph isomorphic to K_5. But then by deleting all but these five vertices from G, we could conclude from Lemma 6.7.8 that K_5 was planar, contradicting Lemma 6.7.6. 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 6.10. In this new graph, n_2 is adjacent to m, and the graph is planar by Lemma 6.7.9. So we can then merge m and n_2 into a another new vertex, m', resulting in a new graph, G', which by Lemma 6.7.9 is also planar. Now G' has $v - 1$ vertices and so is five-colorable by the induction hypothesis.

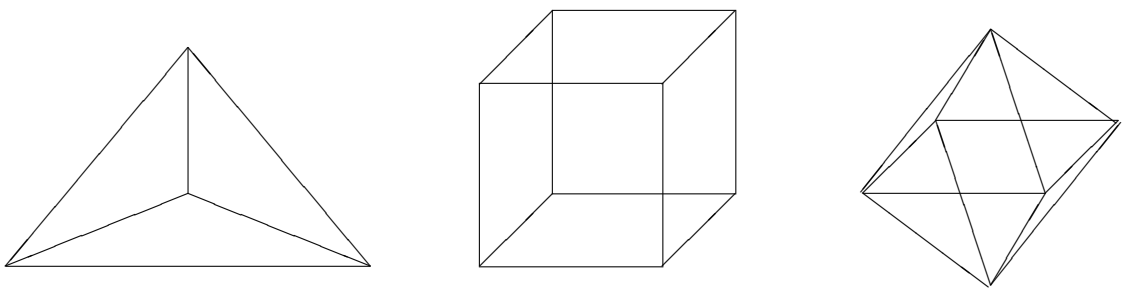
Now define a five coloring of G as follows: use the five-coloring of G' for all the vertices besides g, n_1 and n_2. Next assign the color of m' in G' to be the color of the neighbors n_1 and n_2. Since n_1 and n_2 are not adjacent in G, this defines a proper five-coloring of G except for vertex g. But 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.

□

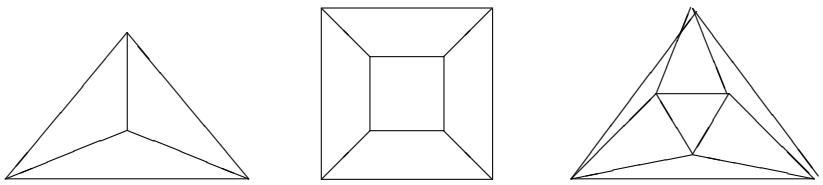
6.7.7 Classifying Polyhedra

The Pythagoreans had two great mathematical secrets, the irrationality of 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.



How many more polyhedra are there? Imagine putting your eye very close to one face of a translucent polyhedron. The edges of that face would ring the periphery of your vision and all other edges would be visible within. For example, the three polyhedra above would look something like this:



Thus, we can regard the corners and edges of these polyhedra as the vertices and edges of a connected planar graph. (This is another logical leap based on geometric intuition.) This means Euler's formula for planar graphs can help guide our search for regular polyhedra.

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:

$$\frac{2e}{m} - e + \frac{2e}{n} = 2$$
$$\frac{1}{m} + \frac{1}{n} = \frac{1}{e} + \frac{1}{2}$$

The last equation 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!

Chapter 7

Digraphs

7.1 Digraphs

A *directed graph* (*digraph* for short) is formally the same as a binary relation, R, on a set, A, but we picture the digraph geometrically by representing elements of A as points on the plane, with an arrow from the point for a to the point for b exactly when $(a, b) \in \text{graph}(R)$. The elements of A are referred to as the *vertices* of the digraph, and the pairs $(a, b) \in \text{graph}(R)$ are called its *directed edges*. We use the notation $a \to b$ as an alternative notation for the pair (a, b).

For example, the divisibility relation on $\{1, 2, \ldots, 12\}$ is represented by the digraph:

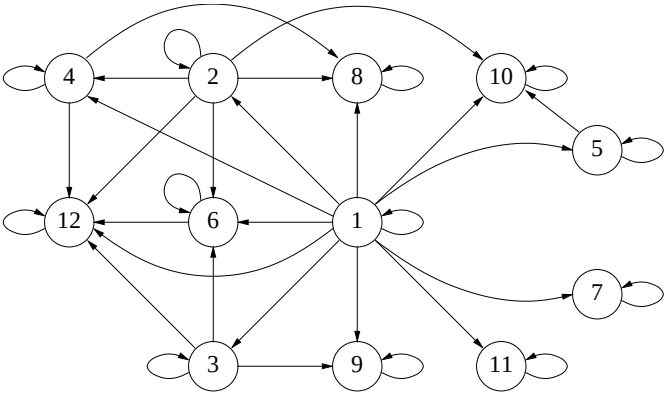


Figure 7.1: The Digraph for Divisibility on $\{1, 2, \ldots, 12\}$.

7.1.1 Paths in Digraphs

Pictured with points and arrows, a length k path in a digraph looks like a line that starts at a point, a_0, and traverses k arrows between successive points, $a_1, a_2, \ldots$ to end at a point, a_k. Note that k may be 0 —a single vertex counts as length zero path to itself, just as for simple graphs. The precise definitions are very similar to those for simple graphs:

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 for every $i \geq 0$ such that $i < k$. 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, $a_i = a_j$ only if $i = j$.

Many of the relational properties have geometric descriptions in terms of digraphs. 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.

Asymmetry: No self-loops and at most one (directed) edge between any two vertices.

Symmetry: A binary relation R is ***symmetric*** iff aRb implies bRa for all a, b in the domain of R. So if there is an edge from a to b, there is also one in the reverse direction. So edges may as well be represented without arrows, indicating that they can be followed in either direction.

Transitivity: Short-circuits—for any path through the graph, there is an arrow from the first vertex to the last vertex on the path.

We can define some new relations based on paths. Let R be the edge relation of a digraph. Define relations R^* and R^+ on the vertices by the conditions that for all vertices a, b:

$$\begin{aligned} a\, R^*\, b &::= \text{there is a path in } R \text{ from } a \text{ to } b, \\ a\, R^+\, b &::= \text{there is a positive length path in } R \text{ from } a \text{ to } b. \end{aligned}$$

R^* is called the *path relation*[1] of R. It follows from the definition of path that R^* is transitive. It is also reflexive (because of the length-zero paths) and it contains the graph of R (because of the length-one paths). R^+ is called the *positive-length path relation*; it also contains graph (R) and is transitive.

7.1.2 Directed Acyclic Graphs

Definition 7.1.2. A *cycle* in a digraph is a path that begins and ends at the same vertex. Note that by convention, a single vertex is considered to be a cycle of length 0 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 are an economical way to represent partial orders. For example, the direct prerequisite relation between MIT subjects described in Week 3 Notes 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 7.1.3. *If D is a DAG, then D^+ is a strict partial order.*

[1] In many texts, R^* is called the *transitive closure* of R.

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 Week 3, Wednesday, Class Problem 2). □

It's easy to check that conversely, the graph of any strict partial order is a DAG.

Problem 7.1.1. Verify that 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.

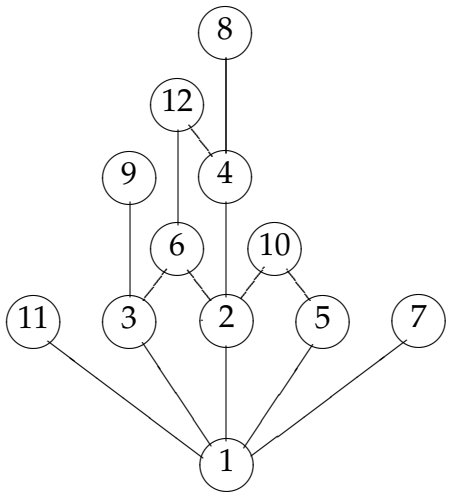


Figure 7.2: A DAG whose Path Relation is Divisibility on $\{1, 2, \ldots, 12\}$.

The minimum edge DAG whose path relation is a given finite partial order is unique, and is easy to find. This is explained in the following problem.

Problem 7.1.2. 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 there is no other path from a to b. If a covers b, the edge from a to b is called a covering edge.

(a) Show that if two DAG's have the same positive path relation, then they have the same set of covering edges.

(b) For any DAG, D, let $\widehat{D}$ be the subgraph of D consisting of only the covering edges. Show that if D is finite, then D and $\widehat{D}$ have the same positive path relation, that is $D^+ = \widehat{D}^+$.

(c) Conclude that if D is a finite DAG, then $\widehat{D}$ is the unique DAG with the smallest number of edges among all digraphs with the same positive path relation.

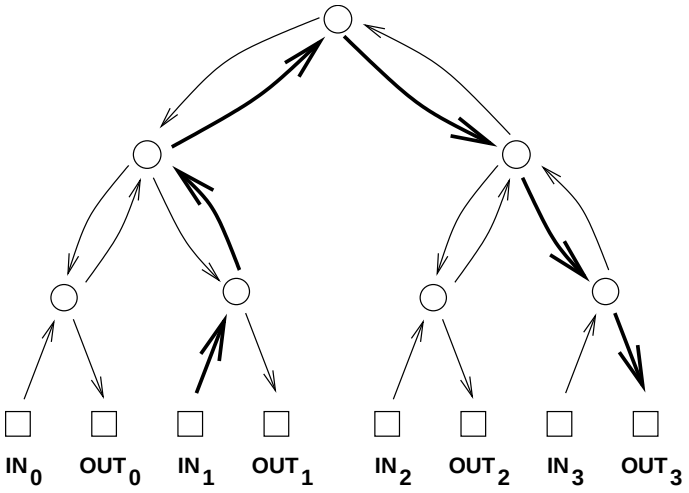
(d) Show that the previous result is not true for the infinite DAG corresponding to the total order on the rational numbers.

7.2 Communication Networks

Modeling communication networks is an important application of graphs in Computer Science. Here, 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. However, there do exist more organized networks, such as certain telephone switching networks and the communication networks inside parallel computers. For these, the corresponding graphs are highly structured. In this section, we'll look at some of the nicest and most commonly used communication networks.

7.2.1 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.

7.2.2 Latency and Diameter

Latency is a critical issue in communication networks. This is the largest delay between the time a packet is sent from an input until it arrives at its designated output. Assuming it takes one time

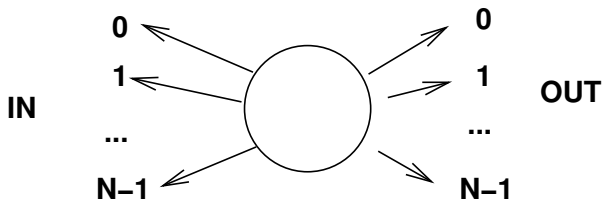
unit to travel across a wire with no delays at switches, the delay of a packet is the number of wires it crosses going from input to output, that is, the packet delay is the *length* of the path the packet follows.

The latency of a network will depend on how packets are routed, but 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[2] 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.

We're going to consider several different communication networks. For a fair comparison, let's assume that each network has N inputs and N outputs, where N is a power of two. For example, 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 still have a latency of only $2 \log(2^{10}) + 2 = 22$.

7.2.3 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 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 elementary devices, like 3×3 switches. Following this approach, we can build arbitrarily large networks just by adding in more building blocks.

7.2.4 Switch Count

Another goal in designing a communication network is to use as few switches as possible since routing hardware has a cost. The number of switches in a complete binary tree is $1 + 2 + 4 + 8 +$

[2]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.

$\cdots + N$, since there is 1 switch at the top (the "root switch"), 2 below it, 4 below those, and so forth. By the formula for the sum of a geometric series (see Slides 4W, Feb. 27, 2008, slide #6) the total number of switches is $2N - 1$, which is nearly the best possible with 3×3 switches.

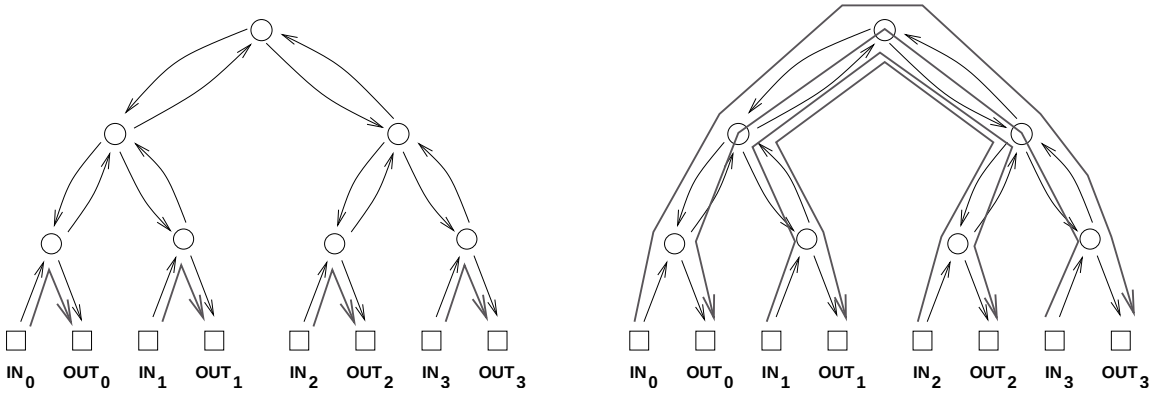
7.2.5 Congestion

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. The *max congestion* of a network is a measure of its bottlenecks; defining max congestion requires some preliminary definitions.

A ***permutation*** is a function π that maps each number in the set $\{0, 1, \ldots, N-1\}$ to another number in the set such that no two numbers are mapped to the same value. In other words, π is a bijection from $\{0, 1, \ldots, N-1\}$ to itself.

For each permutation π, there is a corresponding ***permutation routing problem***: we think of a packet starting out at each input. In particular, the packet starting at input i is called packet i. The problem is to direct each packet i through the network from input i to output $\pi(i)$.

A *solution*, P, to a permutation 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 $\pi(i)$. For example, if $\pi_1(i) = i$, then there is an easy solution, P, for π_1: let P_i be the path from input i up through one switch and back down to output i. On the other hand, if $\pi_2(i) = (N-1)-i$, then in *any* solution, Q, for π_2, 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 set of paths, P, is equal to the largest number of paths that pass through a single switch. For example, the congestion of the set of paths in the diagram at left is 1, since at most 1 path passes through each switch. However, the congestion of the paths 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, we can also distinguish between "good" and "bad" networks with respect to bottleneck problems. The ***max congestion*** of a network is the *maximum* over all problems, π, of the *minimum* over all problem π solutions, P, of the congestion of P.

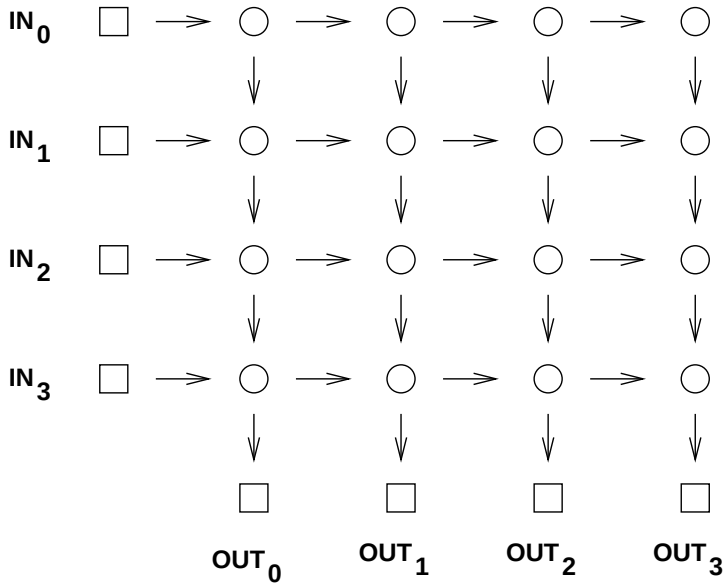
So for the complete binary tree, the worst permutation would be $\pi_2(i) = (N-1) - i$. Then in every possible solution for π_2, *every* packet, would have to follow a path passing through the root switch. Thus, the max congestion of the complete binary tree is N —which is horrible!

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

7.2.6 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 7.2.1. *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 permutation routing problem, π, where $\pi(0) = 0$ and $\pi(N-1) = N-1$, two packets must pass through the lower left switch. □

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.

7.2.7 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 7.3.

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 7.4. That is, the ith and $2^n + i$th new input switches are connected to 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 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.

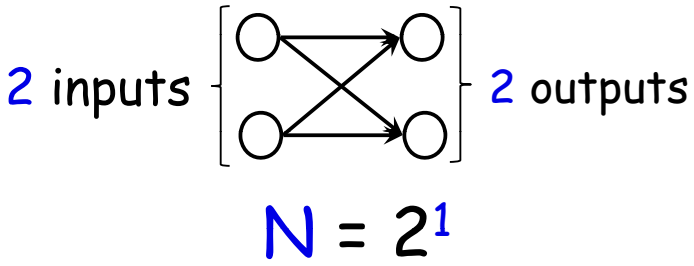


Figure 7.3: F_1, the Butterfly Net switches with $N = 2^1$.

There is a simple argument that shows that 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.

Problem 7.2.1. Show that the congestion of F_n is exactly $\sqrt{N}$ when n is even.

Hints:

- 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 matching of inputs to outputs.
- If v is a vertex at level 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 level of the butterfly network must the congestion be worst? What is the congestion at the node whose binary representation is all 0s at that level of the network?

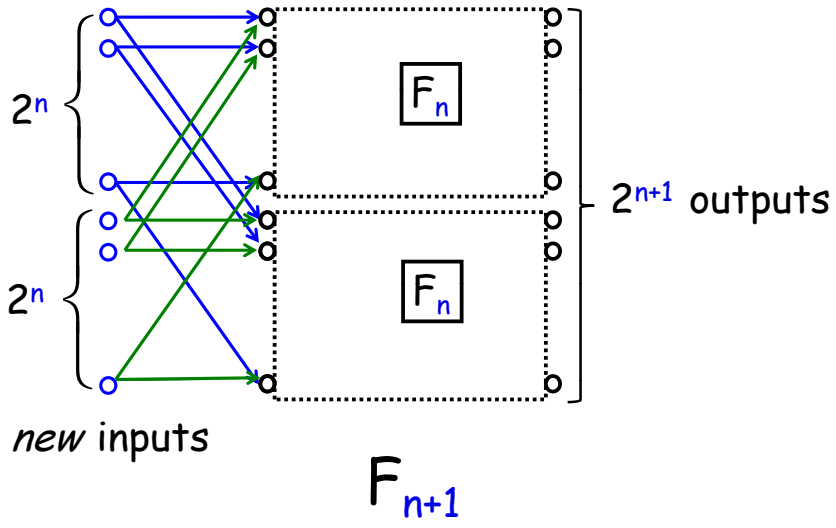


Figure 7.4: F_{n+1}, the Butterfly Net switches with 2^{n+1} inputs and outputs.

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.

7.2.8 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 7.3.

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 7.5.

Namely, the ith and $2^n + i$th new input switches are connected to 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 two switches, namely, to the ith output switches of each of two B_n components.

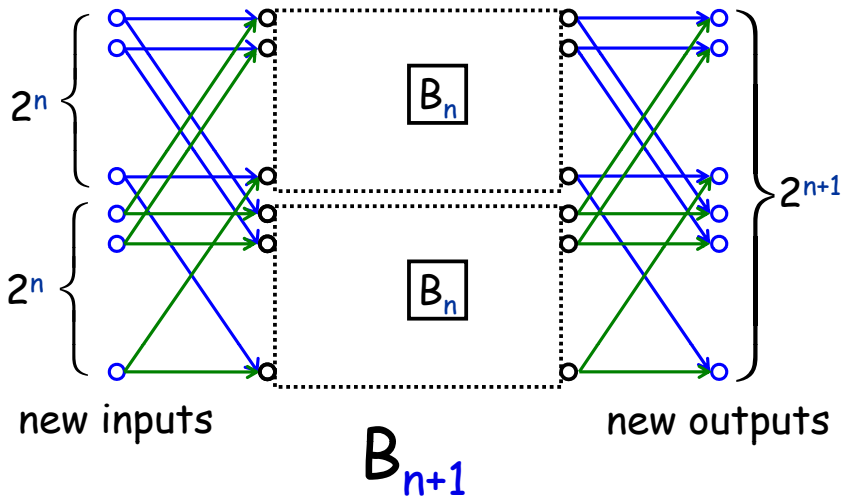


Figure 7.5: B_{n+1}, the Beneš Net switches with 2^{n+1} inputs and outputs.

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 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 7.2.2. *The congestion of the N-input Beneš network is 1.*

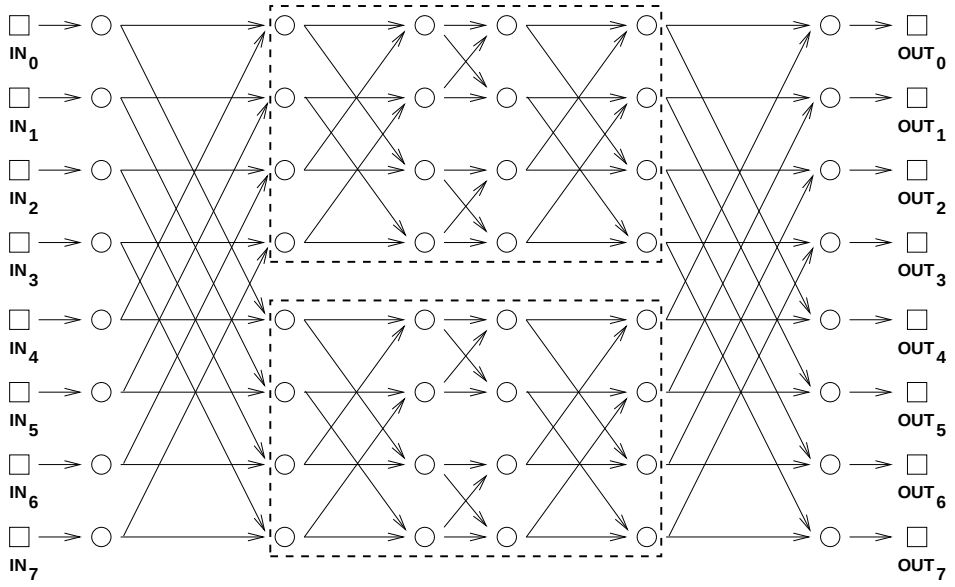
Proof. By induction on n where $N = 2^n$. So the induction hypothesis is

$$P(n) ::= \text{the congestion of } B_n \text{ is } 1.$$

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.

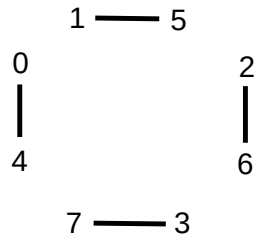


By the inductive assumption, the subnetworks can each route an arbitrary permutation with congestion 1. So if we can guide packets safely through just the first and last levels, then we can rely on induction for the rest! Let's see how this works in an example. Consider the following

permutation routing problem:

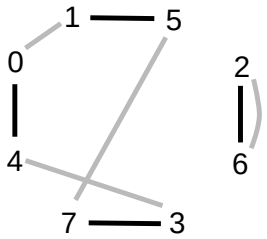
$$\begin{aligned} \pi(0) &= 1 & \pi(4) &= 3 \\ \pi(1) &= 5 & \pi(5) &= 6 \\ \pi(2) &= 4 & \pi(6) &= 0 \\ \pi(3) &= 7 & \pi(7) &= 2 \end{aligned}$$

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:



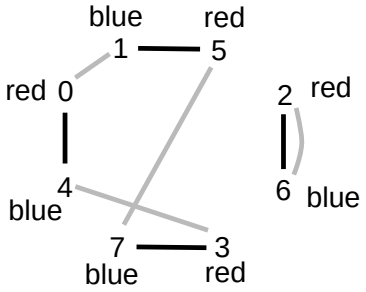
Notice that at most one new edge is incident to each vertex. The two lines drawn between vertices 2 and 6 reflect the two different reasons why these packets must be routed through different networks. However, we intend this to be a simple graph; the two lines still signify a single edge.

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:

Problem 7.2.2. Prove that if a graph, G_1, with vertices, V, and edges, E_1, and a graph, G_2, with the same set, V, of vertices and edges, E_2, both have maximum degree 1, then the graph, G, also with vertices, V, but with edges $E_1 \cup E_2$, is 2-colorable.

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 $\{0, 1, \ldots, N\}$. Let G_1 be a graph where the vertices are packets $0, 1, \ldots, N$ and edges E_1 with an edge u—$v \in E_1$ iff $|u - v| = N/2$. Let G_2 be a graph with the same vertices and edges E_2 with u—$v \in E_2$ iff $|\pi(u) - \pi(v)| = N/2$. Now according to Problem 7.2.2, the graph, G, on these vertices with edges $E_1 \cup E_2$ is 2-colorable, so color the vertices red and blue. Now route red packets through the upper subnetwork and blue packets 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 $P(n)$. □

In class, you will work through an example in which you route packets using this recursive idea!

Chapter 8

Introduction to 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, along with their negatives. Which one don't you understand? After all, the mathematician G. H. Hardy 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.

What most concerned Hardy was that number theory not be used in warfare; he was a pacifist. Good for him, but if number theory is remote from *all* human activity, then why study it? We'll come back to that question later on, but ironically, we'll see that poor Hardy must be turning in his grave.

8.1 Divisibility

We'll be examining integer properties in these notes, so we'll adopt the convention that variables range over integers.

The nature of number theory emerges as soon as we consider the ***divides*** relation

$$a \textbf{ ***divides*** } b \quad \text{iff} \quad ak = b \text{ for some } k.$$

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. As we've seen, 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, computer scientists have found ways to turn these difficulties to their advantage. 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.

DON'T PANIC— we're going to stick to some relatively benign parts of number theory. We won't put any of these super-hard unsolved problems on exams!

8.1.1 Facts About Divisibility

The lemma below states some basic facts about divisibility that are *not* difficult to prove:

Lemma 8.1.1. *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 $ak_1k_2 = c$, 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. The number 1 is considered neither prime nor composite. This is just a matter of definition, but reflects the fact that 1 does not behave like a prime in many contexts, such as the Fundamental Theorem of Arithmetic, which we'll come to shortly.

8.1.2 When Divisibility Goes Bad

As you learned in elementary school, if one number does *not* evenly divide another, then there is a "remainder" left over. More precisely, if you divide n by d, then you get a quotient q and a remainder r:

Theorem 8.1.2 (Division Theorem)**.** *Let n and d be integers such that $d > 0$. Then there exists a unique pair of integers q and r such that $n = qd + r$ and $0 \leq r < d$.*[1]

[1] This theorem is often called the "Division Algorithm," even though it is not an algorithm in the modern sense.

Famous Problems in Number Theory

Fermat's Last Theorem Do there exist 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 Is every even integer greater than or equal to 4 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 Are there 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 Is there an efficient way to determine whether n is prime? A naive search for factors of n takes a number of steps exponential in $\log n$, which is the size of n in bits. 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 $n = pq$, is there an 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.

As an example, suppose that $n = 2716$ and $d = 10$. Then the quotient is $q = 271$ and the remainder is $r = 6$, since $2716 = 271 \cdot 10 + 6$.

The remainder r in the Division Theorem is denoted $\text{rem}(n, d)$. In other words, $\text{rem}(n, d)$ is the remainder when n is divided by d. For example, $\text{rem}(32, 5)$ is the remainder when 32 is divided by 5, which is 2. Similarly, $\text{rem}(-11, 7) = 3$, 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.

8.2 Die Hard

We've previously looked at the Die Hard water jug problem with jugs of sizes 3 and 5, and 3 and 9. It would be nice if we could solve all these silly water jug questions at once. In particular, how can one form g gallons using jugs with capacities a and b? Here's where number theory comes in handy.

8.2.1 Finding an Invariant Property

Suppose that we have water jugs with capacities a and b. 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:

$$\begin{aligned}
(0,0) &\to (a,0) && \text{fill first jug}\\
&\to (0,a) && \text{pour first into second}\\
&\to (a,a) && \text{fill first jug}\\
&\to (2a-b,b) && \text{pour first into second (assuming } 2a \geq b)\\
&\to (2a-b,0) && \text{empty second jug}\\
&\to (0,2a-b) && \text{pour first into second}\\
&\to (a,2a-b) && \text{fill first}\\
&\to (3a-2b,b) && \text{pour first into second (assuming } 3a \geq 2b)
\end{aligned}$$

What leaps out is that at every step, the amount of water in each jug is of the form

$$s \cdot a + t \cdot b \tag{8.1}$$

for some integers s and t. An expression of the form (8.1) is called an ***integer linear combination*** of a and b, but in these notes we'll just call it a ***linear combination***, since we're only talking integers. So we're suggesting:

Lemma 8.2.1. *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 8.2.1 is easy to prove by induction on the number of pourings.

But Lemma 8.2.1 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 and that will help us solve the water jug problem.

8.3 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.

8.3.1 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 8.3.1. *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

$$\gcd(a, b) \mid sa + tb$$

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 and rearranging terms gives:

$$a = q \cdot (sa + tb) + r$$
$$r = (1 - qs)a + (-qt)b$$

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 $m \mid a$. □

The proof notes that 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. This establishes a corollary:

Corollary 8.3.2. *Every linear combination of a and b is a multiple of $\gcd(a, b)$ and vice versa.*

Now we can restate the water jugs lemma in terms of the greatest common divisor:

Corollary 8.3.3. *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 $\gcd(1247, 899) = 29$.

8.3.2 Properties of the Greatest Common Divisor

We'll often make use of some basic $\gcd$ facts:

Lemma 8.3.4. *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. $\gcd(a, b) = \gcd(b, \text{rem}(a, b))$.

Here's the trick to proving these statements: translate the $\gcd$ world to the linear combination world using Theorem 8.3.1, argue about linear combinations, and then translate back using Theorem 8.3.1 again.

Proof. We prove only parts 3. and 4.

Proof of 3.: The assumptions together with Theorem 8.3.1 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 8.3.1.

Proof of 4.: Theorem 8.3.1 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 8.3.4, part 5. is the fact we assumed when we proved 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.

8.3.3 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 8.3.2 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

$$3 = s' \cdot 21 + t' \cdot 26$$

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 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, there must be exactly 3 gallons in the 26-gallon jug! Here's why: we've taken $s' \cdot 21$ gallons of water from the fountain, we've poured out some multiple of 26 gallons, and in the end the 26-gallon jug holds somewhere between 0 and 26 gallons. Furthermore, we know:

$$s' \cdot 21 + t' \cdot 26 = 3$$

Thus, we must have emptied the 26-gallon jug exactly $-t'$ times; if we had emptied it fewer times, then there would be more than 26 gallons left. And we did not withdraw enough water from the fountain to empty the 26-gallon jug more than $-t'$ times. Thus, by the equation above, there must be 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}{llllllll}
(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}} \quad (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}} \quad (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}} \quad (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}} \quad (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}} \quad (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}} \quad (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}} \quad (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}} \quad (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}} \quad (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}} \quad (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}} \quad (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}} \quad (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!

8.3.4 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 the Notes on State Machines we defined and verified the "Extended Euclidean GCD algorithm" which is a much more efficient way to find these coefficients. In this section we give a more straightforward description of this procedure for finding s and t 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:

$$\begin{array}{rcll} \gcd(259, 70) & = & \gcd(70, 49) & \text{since } \mathrm{rem}(259, 70) = 49 \\ & = & \gcd(49, 21) & \text{since } \mathrm{rem}(70, 49) = 21 \\ & = & \gcd(21, 7) & \text{since } \mathrm{rem}(49, 21) = 7 \\ & = & \gcd(7, 0) & \text{since } \mathrm{rem}(21, 7) = 0 \\ & = & 7. & \end{array}$$

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	$(\mathrm{rem}(x, y))$	$=$	$x - q \cdot y$
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 $\mathrm{rem}(x, y)$, 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.

8.4 The Fundamental Theorem of Arithmetic

We now have almost enough tools to prove something that you probably already know.

Theorem (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 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:

$$2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, \ldots$$

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 8.4.1. *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 part (4) of Lemma 8.3.4. □

A routine induction argument extends this statement to:

Lemma 8.4.2. *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.

Theorem 8.4.3 (Fundamental Theorem of Arithmetic)**.** *Every positive integer n can be written in a unique way as a product of primes:*

$$n = p_1 \cdot p_2 \cdots p_j \qquad (p_1 \leq p_2 \leq \cdots \leq p_j)$$

Proof. We proved earlier 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 the well-ordering principle 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 8.4.2 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. □

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" mathematicans 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:

$$\begin{aligned} e = & 2.71828182845904523536028747135266249775724709369995957496 \\ & 96762772407663035354759457138217852516642\mathbf{7427466391}9320030 \\ & 59921817413596629043572900334295260595630738132328627943 4\ldots \end{aligned}$$

8.5 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 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.

8.6 Turing's Code

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 digital cash systems. 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. 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.

8.6.1 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:

	"v	i	c	t	o	r	y"
$\rightarrow$	22	09	03	20	15	18	25

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 betting 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.

8.6.2 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.

8.7 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 denoted $a \equiv b \pmod{n}$. For example:

$$29 \equiv 15 \pmod{7} \quad \text{because } 7 \mid (29-15).$$

There is a close connection between congruences and remainders:

Lemma 8.7.1 (Congruences and Remainders)**.**

$$a \equiv b \pmod{n} \quad \textit{iff} \quad \text{rem}(a,n) = \text{rem}(b,n).$$

Proof. By the Division Theorem, there exist unique pairs of integers q_1, r_1 and q_2, r_2 such that:

$$\begin{aligned} a &= q_1 n + r_1 && \text{where } 0 \le r_1 < n, \\ b &= q_2 n + r_2 && \text{where } 0 \le r_2 < n. \end{aligned}$$

In these terms, $\text{rem}(a,n) = r_1$ and $\text{rem}(b,n) = r_2$. 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 $r_1 = r_2$, which is equivalent to $\text{rem}(a,n) = \text{rem}(b,n)$. □

So we can also see that

$$29 \equiv 15 \pmod{7} \quad \text{because } \text{rem}(29,7) = 1 = \text{rem}(15,7).$$

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 is in no sense more strongly associated with the 15 than 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 8.7.1:

Corollary 8.7.2.

$$a \equiv \text{rem}(a,n) \pmod{n}$$

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:

$$\begin{array}{llllllll} \{ & \ldots, & -6, & -3, & 0, & 3, & 6, & 9, & \ldots & \} \\ \{ & \ldots, & -5, & -2, & 1, & 4, & 7, & 10, & \ldots & \} \\ \{ & \ldots, & -4, & -1, & 2, & 5, & 8, & 11, & \ldots & \} \end{array}$$

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 8.7.3 (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 8.7.1. 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

$$a \equiv b \pmod{n} \tag{8.2}$$

and

$$c \equiv d \pmod{n}. \tag{8.3}$$

Then

$$\begin{aligned}
a + c &\equiv b + c \pmod{n} && \text{(by part 4. and (8.2)),}\\
c + b &\equiv d + b \pmod{n} && \text{(by part 4. and (8.3)), so}\\
b + c &\equiv b + d \pmod{n} && \text{and therefore}\\
a + c &\equiv b + d \pmod{n} && \text{(by part 3.)}
\end{aligned}$$

Part 7. has a similar proof. □

8.8 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: the British had broken Germany's naval code, Enigma. Through much of the war, the Allies were able to route convoys around German submarines by listening into German communications. The British government didn't explain *how* Enigma was broken until 1996. When the analysis was finally released (by

the US), the author was none other than Alan Turing. In 1939 he had joined the secret British codebreaking effort at Bletchley Park. There, he played a central role in cracking the German's Enigma code and thus in preventing Britain from falling into Hitler's hands.

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:

$$m^* = \text{rem}(mk, p) \tag{8.4}$$

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.

8.8.1 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 8.8.1. *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:

$$\begin{aligned} m^* \cdot k^{-1} &= \operatorname{rem}(mk, p) \cdot k^{-1} && \text{(def. (8.4) of } m^*) \\ &\equiv (mk)k^{-1} \pmod{p} && \text{(by Cor. 8.7.2)} \\ &\equiv m \pmod{p}. \end{aligned}$$

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:

$$m = \operatorname{rem}(m^*k^{-1}, p)$$

So now we can decrypt!

8.8.2 Cancellation

Another sense in which real numbers are nice is that one can cancel multiplicative terms. In other words, if we know that $m_1k = m_2k$, 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, this congruence is correct:

$$2 \cdot 3 \equiv 4 \cdot 3 \pmod{6}$$

But if we cancel the 3's, we reach a false conclusion:

$$2 \equiv 4 \pmod{6}$$

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 8.8.2. *Suppose p is a prime and k is not a multiple of p. Then*

$$ak \equiv bk \pmod{p} \qquad \textit{implies} \qquad a \equiv b \pmod{p}$$

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 8.8.3. *Suppose p is a prime and k is not a multiple of p. Then the sequence:*

$$\text{rem}((0 \cdot k), p), \quad \text{rem}((1 \cdot k), p), \quad \text{rem}((2 \cdot k), p), \quad \ldots, \quad \text{rem}(((p-1) \cdot k), p)$$

is a permutation[2] *of the sequence:*

$$0, \quad 1, \quad 2, \quad \ldots, \quad (p-1)$$

This remains true if the first term is deleted from each sequence.

Proof. The first sequence contains p numbers, which are all in the range 0 to $p-1$ by the definition of remainder. Furthermore, the numbers in the first sequence are all different; by Lemma 8.8.2, $ik \equiv jk \pmod p$ if and only if $i \equiv j \pmod p$, and no two numbers in the range 0, 1, ..., p - 1 are congruent modulo p. Thus, the first sequence must contain *all* of the numbers from 0 to $p-1$ in some order. The claim remains true if the first terms are deleted, because both sequences begin with 0. □

For example, suppose $p = 5$ and $k = 3$. Then the sequence:

$$\underbrace{\text{rem}((0 \cdot 3), 5)}_{=0}, \quad \underbrace{\text{rem}((1 \cdot 3), 5)}_{=3}, \quad \underbrace{\text{rem}((2 \cdot 3), 5)}_{=1}, \quad \underbrace{\text{rem}((3 \cdot 3), 5)}_{=4}, \quad \underbrace{\text{rem}((4 \cdot 3), 5)}_{=2}$$

is a permutation of 0, 1, 2, 3, 4 and the last four terms are 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.

8.8.3 Fermat's 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 8.8.1: $k^{-1} = \text{rem}(t, p)$ 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 Theorem, which is much easier than his famous Last Theorem —and more useful.

Theorem 8.8.4 (Fermat's Theorem)**.** *Suppose p is a prime and k is not a multiple of p. Then:*

$$k^{p-1} \equiv 1 \pmod p$$

[2] 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:

$$\begin{aligned} 1 \cdot 2 \cdots (p-1) &= \operatorname{rem}(k,p) \cdot \operatorname{rem}(2k,p) \cdots \operatorname{rem}((p-1)k,p) && \text{(by Cor 8.8.3)} \\ &\equiv k \cdot 2k \cdots (p-1)k \pmod{p} && \text{(by Cor 8.7.2)} \\ &\equiv (p-1)! \cdot k^{p-1} \pmod{p} && \text{(rearranging terms)} \end{aligned}$$

Now $(p-1)!$ can not be a multiple of p, because the prime factorizations of $1, 2, \ldots, (p-1)$ contain only primes smaller than p. Therefore, we can cancel $(p-1)!$ from the first expression and the last by Lemma 8.8.2, 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 $\operatorname{rem}(6^{15}, 17)$, 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, $\operatorname{rem}(6^{15}, 17) = 3$. 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.

8.8.4 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 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:

$$\begin{aligned} m^{p-2} \cdot m^* &= m^{p-2} \cdot \text{rem}(mk, p) && \text{(def. (8.4) of } m^*) \\ &\equiv m^{p-2} \cdot mk \pmod{p} && \text{(by Cor 8.7.2)} \\ &\equiv m^{p-1} \cdot k \pmod{p} \\ &\equiv k \pmod{p} && \text{(Fermat's Theorem)} \end{aligned}$$

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 cracking of Enigma surely saved thousands of lives, if not the whole of Britain.

8.9 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, at that time, homosexuality was a crime in Britain, punishable by up to two years in prison. Turing was sentenced to a humiliating 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.

8.10 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. To send her a message, one encrypts using her widely-distributed public

The Riemann Hypothesis

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 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:

$$1 + \frac{1}{2^s} + \frac{1}{2^{2s}} + \frac{1}{2^{3s}} + \cdots = \frac{1}{1-1/2^s}$$
$$1 + \frac{1}{3^s} + \frac{1}{3^{2s}} + \frac{1}{3^{3s}} + \cdots = \frac{1}{1-1/3^s}$$
$$1 + \frac{1}{5^s} + \frac{1}{5^{2s}} + \frac{1}{5^{3s}} + \cdots = \frac{1}{1-1/5^s}$$
$$\text{etc.}$$

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:

> *The Riemann Hypothesis*: Every nontrivial zero of the zeta function $\zeta(s)$ lies on the line $s = 1/2 + ci$ in the complex plane.

Researchers continue to work intensely to settle this conjecture, as they have for over a century. A proof would immediately imply, among other things, a strong form of the Prime Number Theorem— and earn the prover a $1 million prize! (We're not sure what the cash would be for a counter-example, but the discoverer would be wildly applauded by mathematicians everywhere.)

key. Then she decrypts the 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, in the same sense that a doctor says "This is only going to hurt a little" before he jams a big needle in your arm.

8.10.1 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 every integer is relatively prime to a genuine prime number p, except for multiples of 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.

Theorem 8.10.1. *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 $k \geq 1$.

A proof of Theorem 8.10.1 will appear in a Problem Set, and we'll give another proof in a few weeks after we've developed a few principles for counting things. In the meanwhile, here's an example of using Theorem 8.10.1 to compute $\phi(300)$:

$$\begin{aligned}
\phi(300) &= \phi(2^2 \cdot 3 \cdot 5^2) \\
&= \phi(2^2) \cdot \phi(3) \cdot \phi(5^2) && \text{(by Theorem 8.10.1.(a))} \\
&= (2^2 - 2^1)(3^1 - 3^0)(5^2 - 5^1) && \text{(by Theorem 8.10.1.(b))} \\
&= 80.
\end{aligned}$$

8.10.2 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 8.8.1 of an inverse for k modulo p extends to an inverse for k relatively prime to n:

Lemma 8.10.2. *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 8.10.3. *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.

8.10.3 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 8.10.4. *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 $0 \leq k_i < n$. 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 numbers in the first sequence are all distinct and all appear in the second sequence. Since the two sequences have the same length, the first must be a permutation of the second.

First, we show that the numbers in the first sequence are all distinct. Suppose that $\text{rem}(k_i k, n) = \text{rem}(k_j k, n)$. This is equivalent to $k_i k \equiv k_j k \pmod{n}$, which implies $k_i \equiv k_j \pmod{n}$ by Corollary 8.10.3. This, in turn, means that $k_i = k_j$ since both are between 1 and $n-1$. Thus, a term in the first sequence is not equal to any other term.

Next, we show that each number in the first sequence appears in the second. By assumption, $\gcd(k_i, n) = 1$ and $\gcd(k, n) = 1$, which means that

$$\begin{aligned} \gcd(n, \text{rem}(k_i k, n)) &= \gcd(k_i k, n) && \text{(by Lemma 8.3.4.5)} \\ &= 1 && \text{(by Lemma 8.3.4.3).} \end{aligned}$$

So $\text{rem}(k_i k, n)$ is relatively prime to n and is in the range from 0 to $n-1$ by the definition of remainder. The second sequence is defined to consist of all such integers. □

We can now prove Euler's Theorem:

Theorem 8.10.5 (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:

$$\begin{aligned}
&k_1 \cdot k_2 \cdots k_r \\
&\quad = \operatorname{rem}(k_1 \cdot k, n) \cdot \operatorname{rem}(k_2 \cdot k, n) \cdots \operatorname{rem}(k_r \cdot k, n) && \text{(by Lemma 8.10.4)} \\
&\quad \equiv (k_1 \cdot k) \cdot (k_2 \cdot k) \cdot \cdots \cdot (k_r \cdot k) \pmod{n} && \text{(by Cor 8.7.2)} \\
&\quad \equiv (k_1 \cdot k_2 \cdots k_r) \cdot k^r \pmod{n} && \text{(rearranging terms)}
\end{aligned}$$

Lemma 8.3.4.3. implies that $k_1 \cdot k_2 \cdots k_r$ is prime relative to n. Therefore, we can cancel this product from the first expression and the last by Corollary 8.10.3. 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 $\phi(n)$. Our best method for doing so requires factoring n, which can be quite difficult in general. Fortunately, when we know how to factor n, we can use Theorem 8.10.1 to compute $\phi(n)$ efficiently!

8.10.4 RSA

Finally, we are ready to see how the RSA public key encryption scheme works:

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:

$$m' = \operatorname{rem}(m^e, n).$$

Decoding The receiver decrypts message m' back to message m using the secret key:

$$m = \operatorname{rem}((m')^d, n).$$

We'll explain in class why this way of Decoding works!

Chapter 9

Sums & Asymptotics; Introduction to Counting

9.1 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 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.

[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 twenty 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 to U.S. economists why the Japanese populace keeps any money in their banks.

9.1.1 The Future Value of Money

Our goal is to determine the value of 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:

$$V = \sum_{i=1}^{n} \frac{m}{(1+p)^{i-1}}.$$

To compute the real value of the annuity, we need to evaluate this sum. One way is to plug in m, n, and p, compute each term explicitly, and then add them up. However, this sum has a special closed form that makes the job easier. (The phrase "closed form" refers to a mathematical expression without any summation or product notation.) First, lets make the summation prettier with some substitutions.

$$\begin{aligned} V &= \sum_{i=1}^{n} \frac{m}{(1+p)^{i-1}} \\ &= \sum_{j=0}^{n-1} \frac{m}{(1+p)^{j}} \quad \text{(substitute } j = i-1\text{)} \\ &= m\sum_{j=0}^{n-1} x^{j} \quad \text{(substitute } x = \frac{1}{1+p}\text{)}. \end{aligned}$$

The goal of these substitutions is to put the summation into a special form so that we can bash it with a theorem given in the next section.

9.1.2 Geometric Sums

Theorem 9.1.1. *For all $n \geq 1$ and all $x \neq 1$,*[2]

$$\sum_{i=0}^{n-1} x^{i} = \frac{1-x^{n}}{1-x}.$$

The summation in this theorem is a *geometric sum*. The distinguishing feature of a geometric sum is that each of the terms

$$1,\ x,\ x^2,\ x^3,\ \ldots,\ x^{n-1}$$

in the sum is a constant times the one before; in this case, the constant is x. The theorem gives a closed form for a geometric sum that starts with 1.

We already saw one proof of this theorem in the lecture on induction. As is often the case, the proof by induction gave no hint about how the formula was found in the first place. Here is a more insightful derivation. The trick is to let S be the value of the sum and then observe what $-xS$ is:

$$\begin{array}{rcccccccc} S &=& 1 & +x & +x^2 & +x^3 & + & \cdots & +x^{n-1} \\ -xS &=& & -x & -x^2 & -x^3 & - & \cdots & -x^{n-1} - x^{n}. \end{array}$$

[2] To make this equality hold for $x = 0$, we adopt the convention that $0^0 ::= 1$.

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*.

9.1.3 Return of the Annuity Problem

Now we can solve the annuity pricing problem. The value of an annuity that pays m dollars at the start of each year for n years is computed as follows:

$$\begin{aligned} V &= m\sum_{j=0}^{n-1} x^j \\ &= m\frac{1 - x^n}{1 - x} \\ &= m\frac{1 - (\frac{1}{1+p})^n}{1 - \frac{1}{1+p}} \\ &= m\frac{1 + p - (\frac{1}{1+p})^{n-1}}{p}. \end{aligned}$$

The first line is a restatement of the summation we obtained earlier for the value of an annuity. The second line uses the closed form formula for a geometric sum. In the third line, we undo the earlier substitution $x = 1/(1 + p)$. In the final step, both the numerator and denominator are multiplied by $1 + p$ to simplify the expression.

The resulting formula 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 $V \approx \$530,180$. 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!

9.1.4 Infinite Geometric Series

The question at the beginning of this section 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!

We can compute the value of an annuity with an infinite number of payments by taking the limit of our geometric sum in Theorem 9.1.1 as n tends to infinity. This one is worth remembering!

Theorem 9.1.2. *If $|x| < 1$, then*

$$\sum_{i=0}^{\infty} x^i = \frac{1}{1-x}.$$

Proof.

$$\begin{aligned}\sum_{i=0}^{\infty} x^i &= \lim_{n\to\infty} \sum_{i=0}^{n-1} x^i \\ &= \lim_{n\to\infty} \frac{1-x^n}{1-x} \\ &= \frac{1}{1-x}.\end{aligned}$$

The first equality follows from the definition of an infinite summation. In the second line, we apply the formula for the sum of an n-term geometric sum given in Theorem 9.1.1. The final line follows by evaluating the limit; the x^n term vanishes since we assumed that $|x| < 1$. □

In our annuity problem, $x = 1/(1+p) < 1$, so the theorem applies. Substituting for x, we get an annuity value of

$$\begin{aligned}V &= m \cdot \frac{1}{1-x} \\ &= m \cdot \frac{1}{1-1/(1+p)} \\ &= m \cdot \frac{1+p}{(1+p)-1} \\ &= m \cdot \frac{1+p}{p}.\end{aligned}$$

Plugging in $m = \$50,000$ and $p = 0.08$ gives 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 the answer makes some sense.

9.1.5 Examples

We now have closed form formulas for geometric sums and series. Some examples are given below. In each case, the solution follows immediately from either Theorem 9.1.1 (for finite sums)

or Theorem 9.1.2 (for infinite series).

$$1 + 1/2 + 1/4 + 1/8 + \cdots = \sum_{i=0}^{\infty} (1/2)^i = \frac{1}{1-(1/2)} = 2 \tag{9.1}$$

$$0.999999999\ldots = 0.9 \sum_{i=0}^{\infty} (1/10)^i = 0.9 \frac{1}{1-1/10} = 0.9 \frac{10}{9} = 1 \tag{9.2}$$

$$1 - 1/2 + 1/4 - 1/8 + \cdots = \sum_{i=0}^{\infty} (-1/2)^i = \frac{1}{1-(-1/2)} = 2/3 \tag{9.3}$$

$$1 + 2 + 4 + 8 + \cdots + 2^{n-1} = \sum_{i=0}^{n-1} 2^i = \frac{1-2^n}{1-2} = 2^n - 1 \tag{9.4}$$

$$1 + 3 + 9 + 27 + \cdots + 3^{n-1} = \sum_{i=0}^{n-1} 3^i = \frac{1-3^n}{1-3} = \frac{3^n - 1}{2} \tag{9.5}$$

If the terms in a geometric sum or series grow smaller, as in equation (9.1), then the sum is said to be *geometrically decreasing*. If the terms in a geometric sum grow progressively larger, as in (9.4) and (9.5), then the sum is said to be *geometrically increasing*.

Here is a good rule of thumb: *a geometric sum or series is approximately equal to the term with greatest absolute value*. In equations (9.1) and (9.3), the largest term is equal to 1 and the sums are 2 and 2/3, both relatively close to 1. In equation (9.4), the sum is about twice the largest term. In the final equation (9.5), the largest term is 3^{n-1} and the sum is $(3^n - 1)/2$, which is only about a factor of 1.5 greater.

9.2 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!

9.2.1 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 9.1.

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

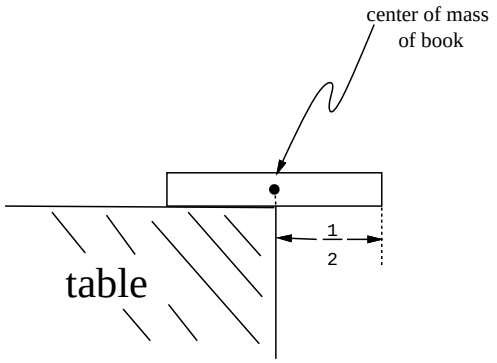


Figure 9.1: One book can overhang half a book length.

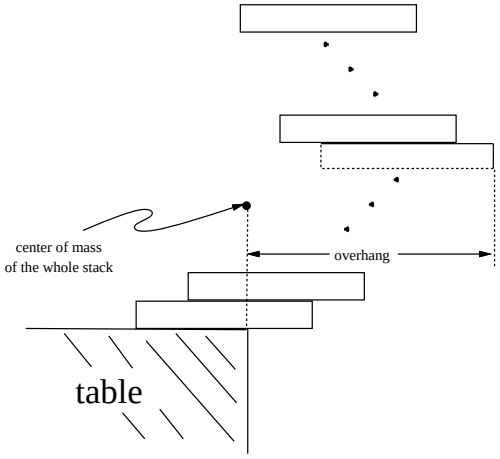


Figure 9.2: Overhanging the edge of the table.

of mass of the stable stack at the edge of the table as in Figure 9.2, that's how far we can get a book in the stack to stick out past the edge.

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 9.3.

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,

$$B_{n+1} = B_n + \frac{1}{2(n+1)}, \tag{9.6}$$

as shown in Figure 9.3.

Expanding equation (9.6), we have

$$\begin{aligned} B_{n+1} &= B_{n-1} + \frac{1}{2n} + \frac{1}{2(n+1)} \\ &= B_1 + \frac{1}{2 \cdot 2} + \cdots + \frac{1}{2n} + \frac{1}{2(n+1)} \\ &= \frac{1}{2} \sum_{i=1}^{n+1} \frac{1}{i}. \end{aligned}$$

Define

$$H_n ::= \sum_{i=1}^{n} \frac{1}{i}.$$

H_n is called the nth *Harmonic number*, and we have just shown that

$$B_n = \frac{H_n}{2}.$$

The first few Harmonic numbers are easy to compute. For example, $H_1 = 1$, $H_2 = 1 + \frac{1}{2} = \frac{3}{2}$, $H_3 = 1 + \frac{1}{2} + \frac{1}{3} = \frac{11}{6}$, $H_4 = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} = \frac{25}{12}$. The fact that H_4 is greater than 2 has special

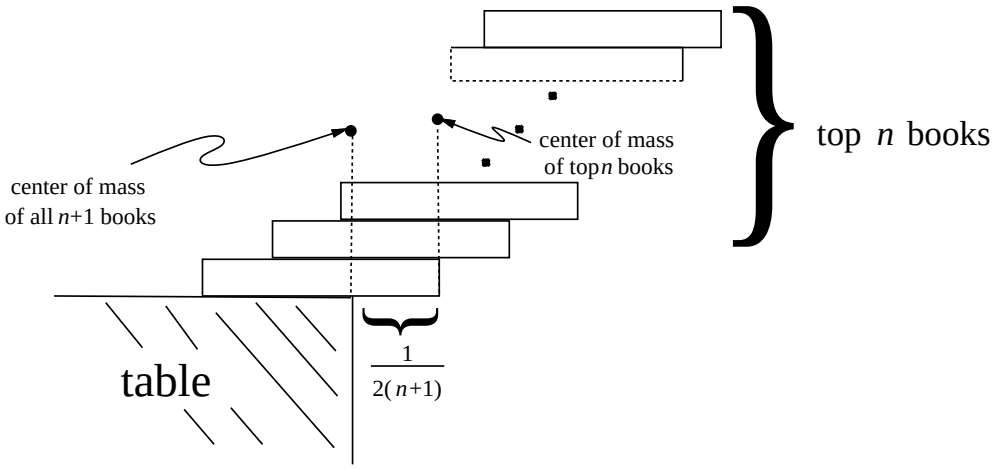


Figure 9.3: Additional overhang with $n + 1$ books.

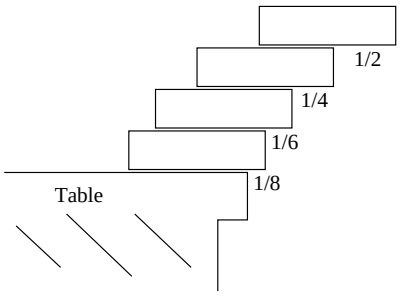


Figure 9.4: Stack of four books with maximum overhang.

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 9.4.

In the next section we will prove that H_n grows slowly, but *unboundedly* with n. That means we can get books to overhang *any distance* past the edge of the table by piling them high enough!

9.2.2 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 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 9.5. The integrals of these functions then bound the value of the sum above and below.

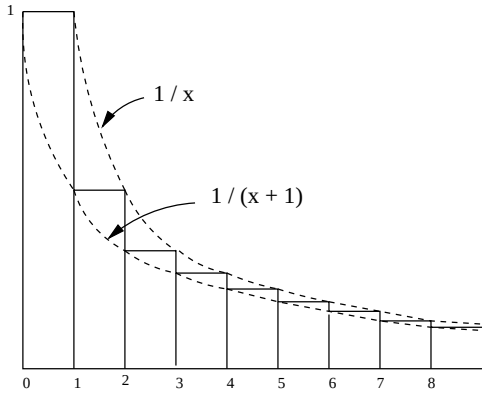


Figure 9.5: *This figure illustrates the Integral Method for bounding a sum. The area under the "stairstep" curve over the interval* $[0, n]$ *is equal to* $H_n = \sum_{i=1}^{n} 1/i$. *The function* $1/x$ *is everywhere greater than or equal to the stairstep and so the integral of* $1/x$ *over this interval 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:

$$\begin{aligned} H_n &\leq 1 + \int_1^n \frac{1}{x}\, dx = 1 + \ln n \\ H_n &\geq \int_0^n \frac{1}{x+1}\, dx = \int_1^{n+1} \frac{1}{x}\, dx = \ln(n+1). \end{aligned} \tag{9.7}$$

These bounds imply that the harmonic number H_n is around $\ln n$. Since $\ln n$ grows without bound, albeit slowly, we can make a stack of books that extends arbitrarily far.

For example, to build a stack extending three book lengths beyond the table, we need a number of books n so that $H_n \geq 6$. Exponentiating the above inequalities gives

$$e^{H_n - 1} \leq n \leq e^{H_n} - 1.$$

This implies that we will need somewhere between 149 and 402 books. Actual calculation of H_n shows that 227 books will be the minimum number to overhang three book lengths.

9.2.3 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 9.2.1. 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

$$\lim_{x\to\infty} f(x)/g(x) = 1.$$

We also might write $H_n \sim \ln n + \gamma$ to indicate two leading terms. While this notation is widely used, it is not really right. Referring to the definition of $\sim$, we see that while $H_n \sim \ln n + \gamma$ is a true statement, so is $H_n \sim \ln n + c$ 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:

$$|H_n - \ln n - \gamma| \leq \left|\frac{1}{200} - \frac{1}{120000} + \frac{1}{120 \cdot 100^4}\right| < \frac{1}{200}.$$

9.3 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 6.042.

9.3.1 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 9.6. This gives bounds on $\ln(n!)$ as follows:

$$\begin{aligned}\int_1^n \ln x \, dx \leq &\quad \textstyle\sum_{i=1}^n \ln i &&\leq \int_0^n \ln(x+1)\, dx \\ n\ln(\frac{n}{e}) + 1 \leq &\quad \textstyle\sum_{i=1}^n \ln i &&\leq (n+1)\ln\left(\frac{n+1}{e}\right) + 1 \\ \left(\frac{n}{e}\right)^n e \leq &\quad n! &&\leq \left(\frac{n+1}{e}\right)^{n+1} e.\end{aligned}$$

The second line follows from the first by completing the integrations. The third line is obtained by exponentiating.

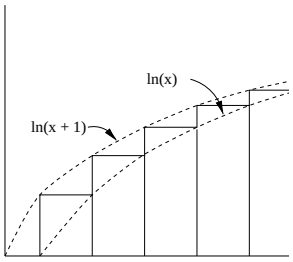


Figure 9.6: *This figure illustrates the Integral Method for bounding the sum* $\sum_{i=1}^{n} ln\ i$.

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)**.**

$$n! \sim \left(\frac{n}{e}\right)^n \sqrt{2\pi n},$$

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)**.**

$$\sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{1/(12n+1)} \leq n! \leq \sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{1/12n}.$$

The Approximation implies the asymptotic Formula, 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:

$$\begin{aligned} 100! &\geq \sqrt{200\pi}\left(\frac{100}{e}\right)^{100} e^{1/1201} \\ 100! &\leq \sqrt{200\pi}\left(\frac{100}{e}\right)^{100} e^{1/1200} \end{aligned}$$

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.

9.4 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.

9.4.1 Little Oh

The asymptotic notation $\sim$ of Definition 9.2.1 is a binary relation indicating that two functions grow at the *same* rate. There is a related strict partial order on functions indicating that one function grows at a significantly *slower* rate. Namely,

Definition 9.4.1. For functions $f, g : \mathbb{R} \to \mathbb{R}$, 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 9.4.2. *$x^a = o(x^b)$ for all nonnegative constants $a < b$.*

Using the familiar fact that $\log x < x$ for all $x > 1$, we can prove

Lemma 9.4.3. $\log x = o(x^\epsilon)$ *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

$$\log z < z^\delta/\delta = o(z^\epsilon) \qquad \text{by Lemma 9.4.2.} \tag{9.8}$$

□

Corollary 9.4.4. $x^b = o(a^x)$ *for any* $a, b \in \mathbb{R}$ *with* $a > 1$.

Proof. From (9.8),

$$\log z < z^\delta/\delta$$

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 9.4.3 and Corollary 9.4.4 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.

Problem 9.4.1. Prove the initial claim that $\log x < x$ for all $x > 1$ (requires elementary calculus).

Problem 9.4.2. Prove that the relation, R, on functions such that $f R g$ iff $f = o(g)$ is a strict partial order, namely, R is transitive and *asymmetric*: if $f R g$ then $\neg g R f$.

Problem 9.4.3. Prove that $f \sim g$ iff $f = g + h$ for some function $h = o(g)$.

9.4.2 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 9.4.5. Given functions $f, g : \mathbb{R} \to \mathbb{R}$, with g nonnegative, we say that

$$f = O(g)$$

iff

$$\limsup_{x \to \infty} |f(x)| / g(x) < \infty.$$

This definition[3] makes it clear that

Lemma 9.4.6. *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 9.4.6 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 9.4.7. 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 9.4.8. *Assume that g is nonnegative. If $f = o(g)$, then it is* not *true that $g = O(f)$.*

Proof.

$$\lim_{x\to\infty} \frac{g(x)}{f(x)} = \frac{1}{\lim_{x\to\infty} \frac{f(x)}{g(x)}} = \frac{1}{0} = \infty,$$

so $g \neq O(f)$.

□

Proposition 9.4.9. $100x^2 = O(x^2)$.

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 9.4.10. $x^2 + 100x + 10 = O(x^2)$.

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 9.4.10 generalizes to an arbitrary polynomial:

---

[3] $$\limsup_{x\to\infty} h(x) ::= \lim_{x\to\infty} \text{lub}_{y\geq x} h(y).$$

We need the $\limsup$ 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. However, in this case we would have $\limsup_{x\to\infty} f(x)/g(x) = 5$.

Proposition 9.4.11. *For $a_k \neq 0$, $a_k x^k + a_{k-1}x^{k-1} + \cdots + a_1 x + a_0 = O(x^k)$.*

The routine proof is left to the reader.

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. It is even conceivable that there is an $O(n^2)$ matrix multiplication procedure, but none is known.

9.4.3 Theta

Definition 9.4.12.

$$f = \Theta(g) \quad \text{iff} \quad f = O(g) \text{ and } g = O(f).$$

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.

9.4.4 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.

[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 $T(n) \sim n^3$.

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 9.4.13. $4^x \neq O(2^x)$

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 9.4.14.

$$\sum_{i=1}^{n} i = O(n)$$

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.

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 "$O(f) = g$."

9.5 Rules for Counting

20480135385502964448038	31710048321735013941130 17	57632573310834796474093 98	82473310000429953116460 21
48944599186691567624099 2	32082344215973686470192 65	58009491235489891226286 63	84962439971234759227663 10
10826620324303796513709 81	34372546563551578648691 13	60429008011992802180260 01	85183991406760026607474 77
11784808947697061789949 93	35748833903586539237113 65	61161717891377378967014 05	85436912834701914523337 63
12531273516832396938513 27	36449099460404801899691 49	61448689730015823697235 12	86753092583741370924613 52
13015051292340778110690 11	37900441327370840944172 46	62473145938511692347461 52	86943211123639968672966 65
13115671111438664338821 94	38703321274379713553228 15	68144289442668749634882 74	87723212036084772458511 54
14700294527212035876862 14	40805058045778014513631 00	68708529455438868491478 81	87914221617225825463410 91
15782710472862574994338 86	41672834610257023481249 20	69149555081209500937323 97	90626280245921262839732 85
16382439218521762431923 54	42359968311237777882112 49	69496324513659871524235 41	91378455669255263498977 94
17635802191319859631023 65	46709394457494390421112 20	71282111436136198284156 50	91537629668031892919344 19
18262277956018422310296 94	48153793518653842796134 27	71739200836518623079253 94	92708801940776364069842 49
18439718626751020372014 20	48370529482129226044421 90	72156548742117556762205 87	93243014807221034903792 04
23969511937221345261772 37	51063894238550185506715 30	72569328471643910402330 50	94360908321466951471405 81
27813945682685998010963 54	51423681920047692180699 10	73328226570752354316203 17	94753081597345382490132 38
27966051967136104054080 19	51812340961301440840418 56	74264418295415734449641 39	94923766239174869749232 02
29310163947619752631903 47	51982673981256179943913 48	76321981265318093271863 21	95119725587798802882529 79
29334580582944051551972 96	53175929403162312197583 72	77121544322119128823105 11	96024134246191871125522 64
30755144104909759203153 48	53843581267717941283569 47	78589186642402623566100 10	96312171149061292194611 11
31114749852527934528600 17	54392117122489019954234 41	78981567867632129631786 79	99081898531027533359813 19
31456215879361201184387 01	56103798260928381927604 58	81475910170375733378486 16	99132374763417642998139 87
31489012556288811031985 49	56323175554652286776760 44	81494367168713711619320 35	
31576931053251112843219 93	56921683746370196174237 12	81760638316825365713067 91	

Are there two different subsets of the ninety 25-digit numbers shown above 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? 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, *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, Eric Lehman, a 6.042 instructor who wrote a good part of the class notes, offered a $100 prize for being the first 6.042 student to actually find two different subsets of the above ninety 25-digit numbers that have the same sum. Eric didn't expect to have to pay off this bet, but he underestimated the ingenuity and initiative of 6.042 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 explicit 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 in turn is perhaps the most important topic this term.
- 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 counting as a practical skill, like integration.

9.5.1 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 S by finding the size of a related set T.* We've already seen a general statement of this idea in the Mapping Rule from Notes 2. Assuming for simplicity that all functions mentioned in the rest of these notes are total, we can restate these rules as:

Rule 1 (Mapping Rule)**.**

1. *If $f : X \to Y$ is surjective, then $|X| \geq |Y|$.*
2. *If $f : X \to Y$ is injective, then $|X| \leq |Y|$.*
3. *If $f : X \to Y$ is bijective, then $|X| = |Y|$.*

Now let's start to put these rules to work.

9.5.2 The Bijection Rule

We've already implicitly used the Bijection Rule a lot: when stable marriage and bipartite matching were considered in earlier notes, we used 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 function, f, defines how they are 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:

$$\underbrace{0\,0}_{\text{chocolate}} \;1\; \underbrace{}_{\text{lemon-filled}} \;1\; \underbrace{0\,0\,0\,0\,0\,0}_{\text{sugar}} \;1\; \underbrace{0\,0}_{\text{glazed}} \;1\; \underbrace{0\,0}_{\text{plain}}$$

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:

$$c \text{ chocolate, } l \text{ lemon-filled, } s \text{ sugar, } g \text{ glazed, and } p \text{ plain}$$

to the sequence:

$$\underbrace{0\ldots0}_{c}\;1\;\underbrace{0\ldots0}_{l}\;1\;\underbrace{0\ldots0}_{s}\;1\;\underbrace{0\ldots0}_{g}\;1\;\underbrace{0\ldots0}_{p}$$

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 boringly routine.

9.5.3 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!

9.5.4 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 2 (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.

9.5.5 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 3 (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:

$$B = \{\text{pancakes}, \text{bacon and eggs}, \text{bagel}, \text{Doritos}\}$$
$$L = \{\text{burger and fries}, \text{garden salad}, \text{Doritos}\}$$
$$D = \{\text{macaroni}, \text{pizza}, \text{frozen burrito}, \text{pasta}, \text{Doritos}\}$$

Then $B \times L \times D$ is the set of all possible daily diets. Here are some sample elements:

$$(\text{pancakes}, \text{burger and fries}, \text{pizza})$$
$$(\text{bacon and eggs}, \text{garden salad}, \text{pasta})$$
$$(\text{Doritos}, \text{Doritos}, \text{frozen burrito})$$

The Product Rule tells us how many different daily diets are possible:

$$\begin{aligned} |B \times L \times D| &= |B| \cdot |L| \cdot |D| \\ &= 4 \cdot 3 \cdot 5 \\ &= 60 \end{aligned}$$

9.5.6 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.

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.

$$F = \{a, b, \ldots, z, A, B, \ldots, Z\}$$
$$S = \{a, b, \ldots, z, A, B, \ldots, Z, 0, 1, \ldots, 9\}$$

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:

$$\begin{array}{cccc} \{\} & \{x_1\} & \{x_2\} & \{x_1, x_2\} \\ \{x_3\} & \{x_1, x_3\} & \{x_2, x_3\} & \{x_1, x_2, x_3\} \end{array}$$

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:

$$\begin{array}{rccccccccccc} \text{subset:} & \{ & & x_2, & x_3, & & x_5, & & x_7, & & & x_{10} & \} \\ \text{sequence:} & (& 0, & 1, & 1, & 0, & 1, & 0, & 1, & 0, & 0, & 1 &) \end{array}$$

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.

9.5.7 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?

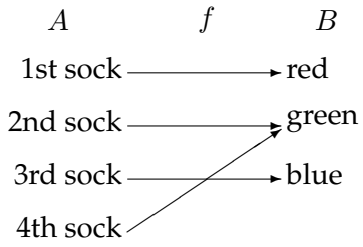
For example, picking out three socks is not enough; you might end up with one red, one green, and one blue. The solution relies on the Pigeonhole Principle, which is a friendly name for the contrapositive of part (2) of the Mapping Rule. Let's write it down:

> If $|X| > |Y|$, then no function $f : X \to Y$ is injective.

And now rewrite it again to eliminate the word "injective."

Rule 4 (Pigeonhole Principle)**.** *If $|X| > |Y|$, then for every function $f : X \to Y$, there exist two different elements of X that are mapped to the same element of Y.*

Perhaps the relevance of this abstract mathematical statement to selecting footwear under poor lighting conditions is 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.



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).

Hairs on Heads

There are a number of generalizations of the pigeonhole principle. For example:

Rule 5 (Generalized Pigeonhole Principle)**.** *If $|X| > k \cdot |Y|$, then every 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.

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 $B = \{1, \ldots, 200,000\}$, 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!

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 $\left\{0, 1, \ldots, 90 \cdot 10^{25}\right\}$, 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.

Chapter 10

More Counting Rules

10.1 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 in 6.042. 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 6 (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.

10.1.1 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

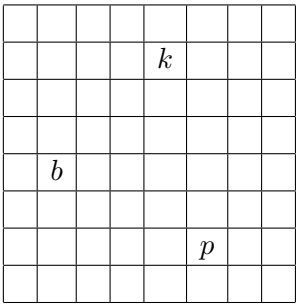
$$\begin{aligned} 10 \cdot 9 \cdot 8 \cdot 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 &= \frac{10!}{2} \\ &= 1,814,400 \end{aligned}$$

serial numbers with all digits different. Plugging these results into the equation above, we find:

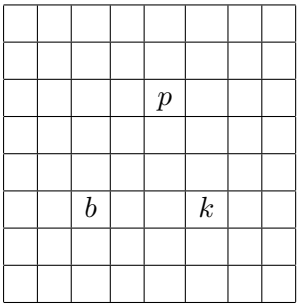
$$\begin{aligned} \text{fraction dollars that are nondefective} &= \frac{1,814,400}{100,000,000} \\ &= 1.8144\% \end{aligned}$$

10.1.2 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 $(8 \cdot 7 \cdot 6)^2$.

10.1.3 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 $\{a, b, c\}$:

$$\begin{array}{lll} (a,b,c) & (a,c,b) & (b,a,c) \\ (b,c,a) & (c,a,b) & (c,b,a) \end{array}$$

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.

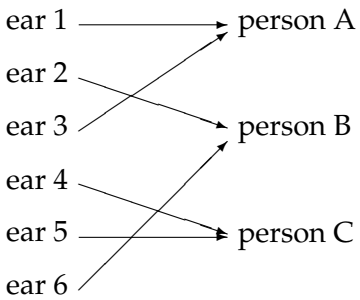
Permutations will come up again in this course approximately 1.6 bazillion times. In fact, permutations are the reason why factorial comes up so often and why we taught you Stirling's approximation:

$$n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

10.2 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:

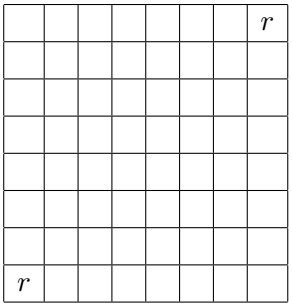
Rule 7 (Division Rule)**.** *If $f : A \to B$ is k-to-1, then $|A| = k \cdot |B|$.*

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.

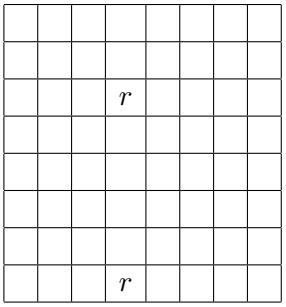
10.2.1 Another Chess Problem

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:

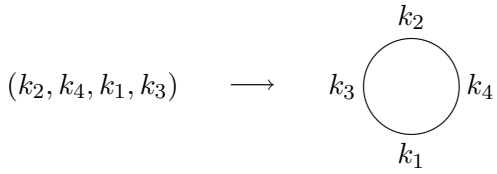
$$\begin{aligned} |B| &= \frac{|A|}{2} \\ &= \frac{(8 \cdot 7)^2}{2} \end{aligned}$$

On the second line, we've computed the size of A using the General Product Rule just as in the earlier chess problem.

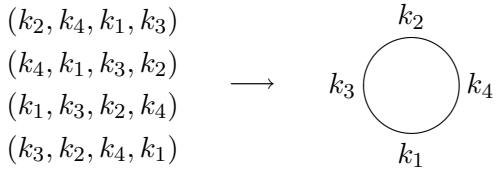
10.2.2 Knights of the Round Table

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:



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:



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.

10.3 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.

10.3.1 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 8 (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.)

10.3.2 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 $\binom{n}{k}$.

10.4 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.[1]

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).

[1] There are 52 cards in a standard deck. Each card has a *suit* and a *rank*. There are four suits:

♠(spades) ♡(hearts) ♣(clubs) ♢(diamonds)

And there are 13 ranks, listed here from lowest to highest:

$$\overset{\text{Ace}}{A},\ 2\ ,\ 3\ ,\ 4\ ,\ 5\ ,\ 6\ ,\ 7\ ,\ 8\ ,\ 9\ ,\ \overset{\text{Jack}}{J}\ ,\ \overset{\text{Queen}}{Q}\ ,\ \overset{\text{King}}{K}$$

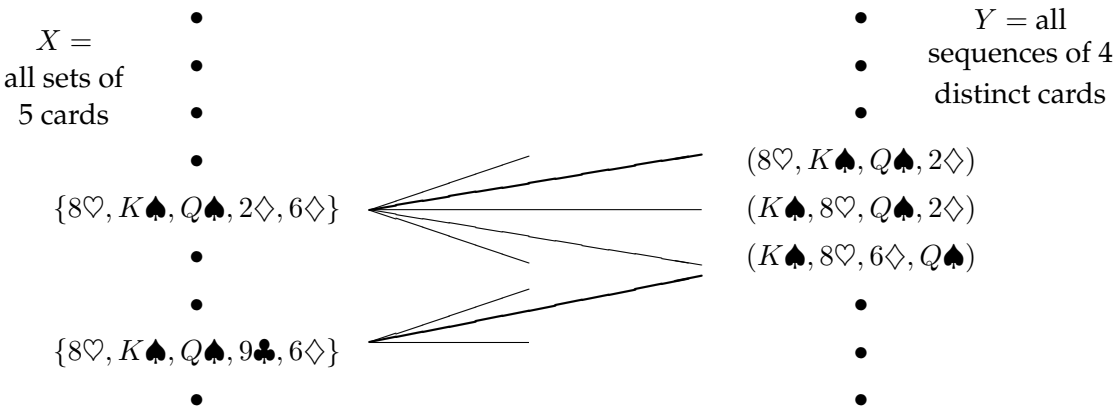
Thus, for example, $8\heartsuit$ is the 8 of hearts and $A\spadesuit$ is the ace of spades.

10.4.1 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.



For example, $\{8\heartsuit, K\spadesuit, Q\spadesuit, 2\diamondsuit, 6\diamondsuit\}$ 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 $\{8\heartsuit, K\spadesuit, Q\spadesuit, 9\clubsuit, 6\diamondsuit\}$, then the Assistant reveals the corresponding sequence $(K\spadesuit, 8\heartsuit, 6\diamondsuit, Q\spadesuit)$. Using the matching, the Magician sees that $(K\spadesuit, 8\heartsuit, 6\diamondsuit, Q\spadesuit)$ is indeed matched to $\{8\heartsuit, K\spadesuit, Q\spadesuit, 9\clubsuit, 6\diamondsuit\}$, 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, the Assistant could have revealed the same sequence, $(K\spadesuit, 8\heartsuit, 6\diamondsuit, Q\spadesuit)$, if the audience picked a different set $\{8\heartsuit, K\spadesuit, Q\spadesuit, 2\diamondsuit, 6\diamondsuit\}$, but if that happened, then the Magician would have no way to tell if remaining card was the $9\clubsuit$ or $2\diamondsuit$.

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* (see Notes 7), 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!

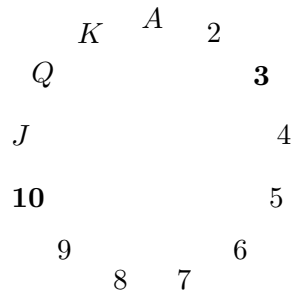
10.4.2 The Real Secret

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:

$$\begin{array}{lllll} (\ \text{small}, & \text{medium}, & \text{large} &) & = 1 \\ (\ \text{small}, & \text{large}, & \text{medium} &) & = 2 \\ (\ \text{medium}, & \text{small}, & \text{large} &) & = 3 \\ (\ \text{medium}, & \text{large}, & \text{small} &) & = 4 \\ (\ \text{large}, & \text{small}, & \text{medium} &) & = 5 \\ (\ \text{large}, & \text{medium}, & \text{small} &) & = 6 \end{array}$$

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).

10.4.3 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!

10.5 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.

10.5.1 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:

$$\begin{array}{llllll} \{ & 8\spadesuit, & 8\diamondsuit, & Q\heartsuit, & 8\heartsuit, & 8\clubsuit \ \} \\ \{ & A\clubsuit, & 2\clubsuit, & 2\heartsuit, & 2\diamondsuit, & 2\spadesuit \ \} \end{array}$$

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!

10.5.2 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:

$$\begin{array}{llllll} \{ & 2\spadesuit, & 2\clubsuit, & 2\diamondsuit, & J\clubsuit, & J\diamondsuit \ \} \\ \{ & 5\diamondsuit, & 5\clubsuit, & 5\heartsuit, & 7\heartsuit, & 7\clubsuit \ \} \end{array}$$

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:

$$\begin{array}{lcl} (2, \{\spadesuit, \clubsuit, \diamondsuit\}, J, \{\clubsuit, \diamondsuit\}) & \leftrightarrow & \{\ 2\spadesuit,\ \ 2\clubsuit,\ \ 2\diamondsuit,\ \ J\clubsuit,\ \ J\diamondsuit\ \} \\ (5, \{\diamondsuit, \clubsuit, \heartsuit\}, 7, \{\heartsuit, \clubsuit\}) & \leftrightarrow & \{\ 5\diamondsuit,\ \ 5\clubsuit,\ \ 5\heartsuit,\ \ 7\heartsuit,\ \ 7\clubsuit\ \} \end{array}$$

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.

10.5.3 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:

$$\begin{array}{l} \{\ 3\diamondsuit,\ \ 3\spadesuit,\ \ Q\diamondsuit,\ \ Q\heartsuit,\ \ A\clubsuit\ \} \\ \{\ 9\heartsuit,\ \ 9\diamondsuit,\ \ 5\heartsuit,\ \ 5\clubsuit,\ \ K\spadesuit\ \} \end{array}$$

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:

$$\begin{array}{ll} (3,\{\diamondsuit,\spadesuit\},Q,\{\diamondsuit,\heartsuit\},A,\clubsuit) & \searrow \\ & \{\ 3\diamondsuit,\ \ 3\spadesuit,\ \ Q\diamondsuit,\ \ Q\heartsuit,\ \ A\clubsuit\ \ \} \\ (Q,\{\diamondsuit,\heartsuit\},3,\{\diamondsuit,\spadesuit\},A,\clubsuit) & \nearrow \end{array}$$

$$\begin{array}{ll} (9,\{\heartsuit,\diamondsuit\},5,\{\heartsuit,\clubsuit\},K,\spadesuit) & \searrow \\ & \{\ 9\heartsuit,\ \ 9\diamondsuit,\ \ 5\heartsuit,\ \ 5\clubsuit,\ \ K\spadesuit\ \ \} \\ (5,\{\heartsuit,\clubsuit\},9,\{\heartsuit,\diamondsuit\},K,\spadesuit) & \nearrow \end{array}$$

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:

$$\begin{array}{rcl} (\{3,Q\},\{\diamondsuit,\spadesuit\},\{\diamondsuit,\heartsuit\},A,\clubsuit) & \leftrightarrow & \{\ 3\diamondsuit,\ \ 3\spadesuit,\ \ Q\diamondsuit,\ \ Q\heartsuit,\ \ A\clubsuit\ \} \\ (\{9,5\},\{\heartsuit,\clubsuit\},\{\heartsuit,\diamondsuit\},K,\spadesuit) & \leftrightarrow & \{\ 9\heartsuit,\ \ 9\diamondsuit,\ \ 5\heartsuit,\ \ 5\clubsuit,\ \ K\spadesuit\ \} \end{array}$$

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.

10.5.4 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:

$$(7,K,A,2,\diamondsuit,3)\ \ \leftrightarrow\ \ \{\ \ 7\diamondsuit,\ \ K\clubsuit,\ \ A\heartsuit,\ \ 2\spadesuit,\ \ 3\diamondsuit\ \ \}$$

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:

$$\begin{array}{ll} (7,K,A,2,\diamondsuit,3) & \searrow \\ & \{\ \ 7\diamondsuit,\ \ K\clubsuit,\ \ A\heartsuit,\ \ 2\spadesuit,\ \ 3\diamondsuit\ \ \} \\ (3,K,A,2,\diamondsuit,7) & \nearrow \end{array}$$

Therefore, the number of hands with every suit is:

$$\frac{13^4\cdot 4\cdot 12}{2}$$

10.6 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 sides of this equation, once as an element of M and once as an element of P. Worse, there might be a triple-major[2] counting *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.

10.6.1 Union of Two Sets

For two sets, S_1 and S_2, the Inclusion-Exclusion rule is that the size of their union is:

$$|S_1 \cup S_2| = |S_1| + |S_2| - |S_1 \cap S_2| \tag{10.1}$$

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:

$$S_1 \cup S_2 = (S_1 - S_2) \cup (S_2 - S_1) \cup (S_1 \cap S_2). \tag{10.2}$$

Similarly, we can decompose each of S_1 and S_2 into the elements exclusively in each set and the elements in both:

$$S_1 = (S_1 - S_2) \cup (S_1 \cap S_2), \tag{10.3}$$
$$S_2 = (S_2 - S_1) \cup (S_1 \cap S_2). \tag{10.4}$$

Now we have from (10.3) and (10.4)

$$\begin{aligned} |S_1| + |S_2| &= (|S_1 - S_2| + |S_1 \cap S_2|) + (|S_2 - S_1| + |S_1 \cap S_2|) \\ &= |S_1 - S_2| + |S_2 - S_1| + 2\,|S_1 \cap S_2|\,, \end{aligned} \tag{10.5}$$

which shows the double-counting of $S_1 \cap S_2$ in the sum. On the other hand, we have from (10.2)

$$|S_1 \cup S_2| = |S_1 - S_2| + |S_2 - S_1| + |S_1 \cap S_2|\,. \tag{10.6}$$

Subtracting (10.6) from (10.5), we get

$$(|S_1| + |S_2|) - |S_1 \cup S_2| = |S_1 \cap S_2|$$

which proves (10.1).

[2] . . . though not at MIT anymore.

10.6.2 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:

$$|P_{42} \cup P_{04} \cup P_{60}| = 9! + 9! + 9! - 8! - 8! - 8! + 7!$$

10.6.3 Union of *n* Sets

The size of a union of n sets is given by the following rule.

Rule 9 (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.*

There are various ways to write the Inclusion-Exclusion formula in mathematical symbols, but none are particularly clear, so we've just used words. The formulas for unions of two and three sets are special cases of this general rule.

10.6.4 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

$$S = \bigcup_{i=1}^{m} C_i.$$

Next, observe that if r is a positive divisor of n, then exactly n/r nonnegative integers less than n are divisible by r, namely, $0, r, 2r, \ldots, ((n/r) - 1)r$.

Now by inclusion-exclusion,

$$\begin{aligned}
|S| &= \left|\bigcup_{i=1}^{m} C_i\right| \\
&= \sum_{i=1}^{m} |C_i| - \sum_{1\le i<j\le m} |C_i \cap C_j| + \sum_{1\le i<j<k\le m} |C_i \cap C_j \cap C_k| - \cdots \pm \left|\bigcap_{i=1}^{m} C_i\right| \\
&= \sum_{i=1}^{m} \frac{n}{p_i} - \sum_{1\le i<j\le m} \frac{n}{p_i p_j} + \sum_{1\le i<j<k\le m} \frac{n}{p_i p_j p_k} - \cdots \pm \frac{n}{\prod_{i=1}^{m} p_i} \\
&= n\left(1 - \prod_{i=1}^{m}\left(1 - \frac{1}{p_i}\right)\right)
\end{aligned}$$

But $\phi(n) = n - |S|$ by definition, so

$$\phi(n) = n - n\left(1 - \prod_{i=1}^{m}\left(1 - \frac{1}{p_i}\right)\right) = n\prod_{i=1}^{m}\left(1 - \frac{1}{p_i}\right). \tag{10.7}$$

Notice that in case $n = p^k$ for some prime, p, then (10.7) simplifies to

$$\phi(p^k) = p^k\left(1 - \frac{1}{p}\right) = p^k - p^{k-1}$$

as claimed in the Notes on Number Theory.

Problem 10.6.1. Use equation (10.7) to prove that, as claimed in the Notes on Number Theory,

$$\phi(ab) = \phi(a)\phi(b)$$

for relatively prime integers $a, b > 1$.

10.7 The Bookkeeper Rule

10.7.1 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 subsets of A and $|A_i| = k_i$ for $i = 1, \ldots, m$.

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 10 (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.

10.7.2 Sequences over an alphabet

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 11 (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:

$$\frac{20!}{5!^4}$$

10.7.3 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'll skip it.

Chapter 11

Combinatorial Identities; Generating Functions

11.1 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

$$\begin{array}{rcccccccc} (a+b)^4 & = & aaaa & + & aaab & + & aaba & + & aabb \\ & + & abaa & + & abab & + & abba & + & abbb \\ & + & baaa & + & baab & + & baba & + & babb \\ & + & bbaa & + & bbab & + & bbba & + & bbbb \end{array}$$

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:

$$(a+b)^4 = \binom{4}{0} \cdot a^4b^0 + \binom{4}{1} \cdot a^3b^1 + \binom{4}{2} \cdot a^2b^2 + \binom{4}{3} \cdot a^1b^3 + \binom{4}{4} \cdot a^0b^4$$

In general, this reasoning gives the Binomial Theorem:

Theorem 11.1.1 (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:

Theorem 11.1.2 (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.

11.2 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.

11.2.1 Boxing

Jay, famed 6.042 TA, 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 6.042 TA, 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.

11.2.2 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 11.2.1.

$$\sum_{r=0}^{n} \binom{n}{r}\binom{2n}{n-r} = \binom{3n}{n}$$

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 11.2.1 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 11.2.1 is $\binom{3n}{n}$, which suggests choosing S to be all n-element subsets of some $3n$-element set.

11.3 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 these notes, 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 $\langle g_0, g_1, g_2, g_3 \dots \rangle$ 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 these notes, we'll indicate the correspondence between a sequence and its generating function with a double-sided arrow as follows:

$$\langle g_0, g_1, g_2, g_3, \dots \rangle \longleftrightarrow g_0 + g_1 x + g_2 x^2 + g_3 x^3 + \cdots$$

For example, here are some sequences and their generating functions:

$$\begin{aligned}
\langle 0, 0, 0, 0, \dots \rangle &\longleftrightarrow 0 + 0x + 0x^2 + 0x^3 + \cdots = 0 \\
\langle 1, 0, 0, 0, \dots \rangle &\longleftrightarrow 1 + 0x + 0x^2 + 0x^3 + \cdots = 1 \\
\langle 3, 2, 1, 0, \dots \rangle &\longleftrightarrow 3 + 2x + 1x^2 + 0x^3 + \cdots = 3 + 2x + x^2
\end{aligned}$$

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:

$$\begin{aligned}
\langle 1,1,1,1,\ldots\rangle &\longleftrightarrow 1+x+x^2+x^3+\cdots &&= \frac{1}{1-x}\\
\langle 1,-1,1,-1,\ldots\rangle &\longleftrightarrow 1-x+x^2-x^3+x^4-\cdots &&= \frac{1}{1+x}\\
\langle 1,a,a^2,a^3,\ldots\rangle &\longleftrightarrow 1+ax+a^2x^2+a^3x^3+\cdots &&= \frac{1}{1-ax}\\
\langle 1,0,1,0,1,0,\ldots\rangle &\longleftrightarrow 1+x^2+x^4+x^6+\cdots &&= \frac{1}{1-x^2}
\end{aligned}$$

11.4 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.

11.4.1 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 12 (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:

$$\begin{aligned}
\langle cf_0,cf_1,cf_2,\ldots\rangle &\longleftrightarrow cf_0+cf_1x+cf_2x^2+\cdots\\
&= c\cdot(f_0+f_1x+f_2x^2+\cdots)\\
&= cF(x)
\end{aligned}$$

11.4.2 Addition

Adding generating functions corresponds to adding the two sequences term by term. For example, adding two of our earlier examples gives:

$$\begin{array}{rccccccccc} & \langle & 1, & 1, & 1, & 1, & 1, & 1, & \ldots & \rangle \quad \longleftrightarrow \quad \dfrac{1}{1-x} \\ + & \langle & 1, & -1, & 1, & -1, & 1, & -1, & \ldots & \rangle \quad \longleftrightarrow \quad \dfrac{1}{1+x} \\ \hline & \langle & 2, & 0, & 2, & 0, & 2, & 0, & \ldots & \rangle \quad \longleftrightarrow \quad \dfrac{1}{1-x}+\dfrac{1}{1+x} \end{array}$$

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 13 (Addition Rule)**.** *If*

$$\begin{array}{ll} \langle f_0, f_1, f_2, \ldots \rangle \longleftrightarrow F(x), & \textit{and} \\ \langle g_0, g_1, g_2, \ldots \rangle \longleftrightarrow G(x), & \end{array}$$

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:

$$\begin{aligned} \langle f_0 + g_0,\ f_1 + g_1,\ f_2 + g_2,\ \ldots \rangle \quad &\longleftrightarrow \quad \sum_{n=0}^{\infty} (f_n + g_n) x^n \\ &= \quad \left(\sum_{n=0}^{\infty} f_n x^n \right) + \left(\sum_{n=0}^{\infty} g_n x^n \right) \\ &= \quad F(x) + G(x) \end{aligned}$$

11.4.3 Right Shifting

Let's start over again with a simple sequence and its generating function:

$$\langle 1, 1, 1, 1, \ldots \rangle \longleftrightarrow \frac{1}{1-x}$$

Now let's *right-shift* the sequence by adding k leading zeros:

$$\begin{aligned} \langle \underbrace{0, 0, \ldots, 0}_{k \text{ zeroes}}, 1, 1, 1, \ldots \rangle \quad &\longleftrightarrow \quad x^k + x^{k+1} + x^{k+2} + x^{k+3} + \cdots \\ &= \quad x^k \cdot (1 + x + x^2 + x^3 + \cdots) \\ &= \quad \frac{x^k}{1-x} \end{aligned}$$

Evidently, adding k leading zeros to the sequence corresponds to multiplying the generating function by x^k. This holds true in general.

Rule 14 (Right-Shift Rule)**.** *If* $\langle f_0, f_1, f_2, \ldots \rangle \longleftrightarrow F(x)$*, then:*

$$\langle \underbrace{0, 0, \ldots, 0}_{k \text{ zeroes}}, f_0, f_1, f_2, \ldots \rangle \longleftrightarrow x^k \cdot F(x)$$

The idea behind this rule is that:

$$\begin{aligned} \langle \overbrace{0, 0, \ldots, 0}^{k \text{ zeroes}}, f_0, f_1, f_2, \ldots \rangle &\longleftrightarrow f_0 x^k + f_1 x^{k+1} + f_2 x^{k+2} + \cdots \\ &= x^k \cdot (f_0 + f_1 x + f_2 x^2 + f_3 x^3 + \cdots) \\ &= x^k \cdot F(x) \end{aligned}$$

11.4.4 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.

$$\begin{aligned} \frac{d}{dx}\left(1 + x + x^2 + x^3 + x^4 + \cdots\right) &= \frac{d}{dx}\left(\frac{1}{1-x}\right) \\ 1 + 2x + 3x^2 + 4x^3 + \cdots &= \frac{1}{(1-x)^2} \\ \langle 1, 2, 3, 4, \ldots \rangle &\longleftrightarrow \frac{1}{(1-x)^2} \end{aligned}$$

We found a generating function for the sequence $\langle 1, 2, 3, 4, \ldots \rangle$ 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 15 (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:

$$\begin{aligned} \langle f_1, 2f_2, 3f_3, \ldots \rangle &\longleftrightarrow f_1 + 2f_2 x + 3f_3 x^2 + \cdots \\ &= \frac{d}{dx}\left(f_0 + f_1 x + f_2 x^2 + f_3 x^3 + \cdots\right) \\ &= \frac{d}{dx} F(x) \end{aligned}$$

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 14 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:

$$\frac{x(1+x)}{(1-x)^3}$$

11.4.5 Products

Rule 16 (Product Rule). *If*

$$\begin{aligned}
\langle a_0, a_1, a_2, \ldots \rangle &\longleftrightarrow A(x), \qquad \textit{and} \\
\langle b_0, b_1, b_2, \ldots \rangle &\longleftrightarrow B(x),
\end{aligned}$$

then

$$\langle c_0,\ c_1,\ c_2,\ \ldots \rangle \longleftrightarrow A(x) \cdot B(x),$$

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	$\ldots$
a_0x^0	$a_0b_0x^0$	$a_0b_1x^1$	$a_0b_2x^2$	$a_0b_3x^3$	$\ldots$
a_1x^1	$a_1b_0x^1$	$a_1b_1x^2$	$a_1b_2x^3$	$\ldots$	
a_2x^2	$a_2b_0x^2$	$a_2b_1x^3$	$\ldots$		
a_3x^3	$a_3b_0x^3$	$\ldots$			
$\vdots$	$\ldots$				

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,

$$a_0b_n + a_1b_{n-1} + a_2b_{n-2} + \cdots + a_nb_0. \tag{11.1}$$

This expression (11.1) may be familiar from a signal processing course; the sequence $\langle c_0, c_1, c_2, \ldots \rangle$ is called the ***convolution*** of sequences $\langle a_0, a_1, a_2, \ldots \rangle$ and $\langle b_0, b_1, b_2, \ldots \rangle$.

11.5 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:

$$\langle 0, 1, 1, 2, 3, 5, 8, 13, 21, \ldots \rangle \longleftrightarrow \frac{x}{1 - x - x^2}$$

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.

11.5.1 Finding a Generating Function

Let's begin by recalling the definition of the Fibonacci numbers:

$$\begin{aligned} f_0 &= 0 \\ f_1 &= 1 \\ f_n &= f_{n-1} + f_{n-2} \qquad (\text{for } n \geq 2) \end{aligned}$$

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:

$$\begin{array}{rccccccccc} & \langle & 0, & 1, & 0, & 0, & 0, & \ldots & \rangle & \longleftrightarrow & x \\ & \langle & 0, & f_0, & f_1, & f_2, & f_3, & \ldots & \rangle & \longleftrightarrow & xF(x) \\ + & \langle & 0, & 0, & f_0, & f_1, & f_2, & \ldots & \rangle & \longleftrightarrow & x^2F(x) \\ \hline & \langle & 0, & 1+f_0, & f_1+f_0, & f_2+f_1, & f_3+f_2, & \ldots & \rangle & \longleftrightarrow & x + xF(x) + x^2F(x) \end{array}$$

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:

$$\begin{array}{rcccccccccc} F(x) & = & f_0 + & f_1 & x + & f_2 & x^2 + & f_3 & x^3 + & f_4 & x^4 + \ldots \\ \| & & \| & \| & & \| & & \| & & \| & \\ x + xF(x) + x^2F(x) & = & 0 + & (1+f_0) & x + & (f_1+f_0) & x^2 + & (f_2+f_1) & x^3 + & (f_3+f_2) & x^4 + \cdots \end{array}$$

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.

11.5.2 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:

$$\frac{1}{1 - \alpha_1 x} = 1 + \alpha_1 x + \alpha_1^2 x^2 + \cdots$$
$$\frac{1}{1 - \alpha_2 x} = 1 + \alpha_2 x + \alpha_2^2 x^2 + \cdots$$

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.

11.6 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.

11.6.1 Choosing Distinct Items from a Set

The generating function for binomial coefficients follows directly from the Binomial Theorem:

$$\begin{aligned} \left\langle \binom{k}{0}, \binom{k}{1}, \binom{k}{2}, \ldots, \binom{k}{k}, 0, 0, 0, \ldots \right\rangle &\longleftrightarrow \binom{k}{0} + \binom{k}{1}x + \binom{k}{2}x^2 + \cdots + \binom{k}{k}x^k \\ &= (1+x)^k \end{aligned}$$

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.)

11.6.2 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 choose n elements from this set is simply $1+x$: we have 1 way to choose zero elements, 1 way to choose one element, and 0 ways to choose more than one element. Similarly, the number of ways to choose 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 choose n elements from the $\{a_1, a_2\}$ is:

$$\underbrace{(1+x)}_{\substack{\text{gen func for}\\ \text{choosing from}\\ \{a_1\}}} \cdot \underbrace{(1+x)}_{\substack{\text{gen func for}\\ \text{choosing from}\\ \{a_2\}}} = \underbrace{(1+x)^2}_{\substack{\text{gen func for}\\ \text{choosing from}\\ \{a_1, a_2\}}} = 1 + 2x + x^2$$

Sure enough, for the set $\{a_1, a_2\}$, we have 1 way to choose zero elements, 2 ways to choose one element, 1 way to choose two elements, and 0 ways to choose more than two elements.

Repeated application of this rule gives the generating function for choosing n items from a k-element set $\{a_1, a_2, \ldots, a_k\}$:

$$\underbrace{(1+x)}_{\substack{\text{gen func for}\\ \text{choosing from}\\ \{a_1\}}} \cdot \underbrace{(1+x)}_{\substack{\text{gen func for}\\ \text{choosing from}\\ \{a_2\}}} \cdots \underbrace{(1+x)}_{\substack{\text{gen func for}\\ \text{choosing from}\\ \{a_k\}}} = \underbrace{(1+x)^k}_{\substack{\text{gen func for}\\ \text{choosing from}\\ \{a_1, a_2, \ldots, a_k\}}}$$

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 17 (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 $A(x)B(x)$.

11.6.3 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:

$$\begin{aligned} \langle 1,1,1,1,\ldots\rangle &\longleftrightarrow 1 + x + x^2 + x^3 + \cdots \\ &= \frac{1}{1-x} \end{aligned}$$

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:

$$\underbrace{\frac{1}{1-x}}_{\substack{\text{gen func for} \\ \text{repeated choice from} \\ \{a_1\}}} \cdot \underbrace{\frac{1}{1-x}}_{\substack{\text{gen func for} \\ \text{repeated choice from} \\ \{a_2\}}} \cdots \underbrace{\frac{1}{1-x}}_{\substack{\text{gen func for} \\ \text{repeated choice from} \\ \{a_k\}}} = \underbrace{\frac{1}{(1-x)^k}}_{\substack{\text{gen func for} \\ \text{repeated choice from} \\ \{a_1, a_2, \ldots, a_k\}}}$$

Therefore, the generating function for selecting 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 select 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 11.6.1 (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. Let

$$G(x) ::= \frac{1}{(1-x)^k} = (1-x)^{-k}.$$

Then we have:

$$\begin{aligned}
G'(x) &= k(1-x)^{-(k+1)} \\
G''(x) &= k(k+1)(1-x)^{-(k+2)} \\
G'''(x) &= k(k+1)(k+2)(1-x)^{-(k+3)} \\
G^{(n)}(x) &= k(k+1)\cdots(k+n-1)(1-x)^{-(k+n)}
\end{aligned}$$

Thus, the coefficient of x^n in the generating function is:

$$\begin{aligned}
G^{(n)}(0)/n! &= \frac{k(k+1)\cdots(k+n-1)}{n!} \\
&= \frac{(k+n-1)!}{(k-1)!\,n!} \\
&= \binom{n+k-1}{n}.
\end{aligned}$$

So if we didn't already know the Bookkeeper Rule, we could have proved it using generating functions.

11.7 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 selecting apples. We can select 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 selecting bananas is:

$$B(x) = 1 + x^5 + x^{10} + x^{15} + \cdots = \frac{1}{1-x^5}$$

Now, we can select a set of 0 oranges in one way, a set of 1 orange in one way, and so on. However, we can not select 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 select only zero or one pear, so we have:

$$P(x) = 1 + x$$

The Convolution Rule says that the generating function for selecting 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!*

Chapter 12

Introduction to Probability

Probability is the last topic in this course and perhaps the most important. Many algorithms rely on randomization. Investigating their correctness and performance requires probability theory. Moreover, many aspects of computer systems, such as memory management, branch prediction, packet routing, and load balancing are designed around probabilistic assumptions and analyses. Probability also comes up in information theory, cryptography, artificial intelligence, and game theory. Beyond these engineering applications, an understanding of probability gives insight into many everyday issues, such as polling, DNA testing, risk assessment, investing, and gambling.

So probability is good stuff.

12.1 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 roughly describes a situation 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. 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.

Yet this is is an elementary problem with an elementary solution. Why was there so much dispute? Apparently, most people *believe* they have an intuitive grasp of probability. (This is in stark contrast to other branches of mathematics; few people believe they have an intuitive ability to compute integrals or factor large integers!) Unfortunately, approximately 100% of those people

are *wrong*. In fact, everyone who has studied probability at length can name a half-dozen problems in which their intuition led them astray— often embarrassingly so.

The way to avoid errors is to distrust informal arguments and rely instead on a rigorous, systematic approach. If you insist on relying on intuition, then there are lots of compelling financial deals we'd love to offer you! In short: intuition *bad*, rigor *good* —at least until you've gained some solid experience with probabilities.

12.1.1 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 reduces many famously-confusing problems to near triviality. 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!

12.1.2 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?"

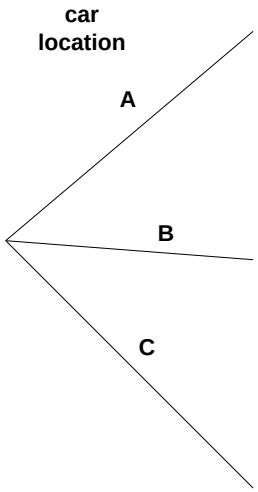
12.1.3 Step 1: Find the Sample Space

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.

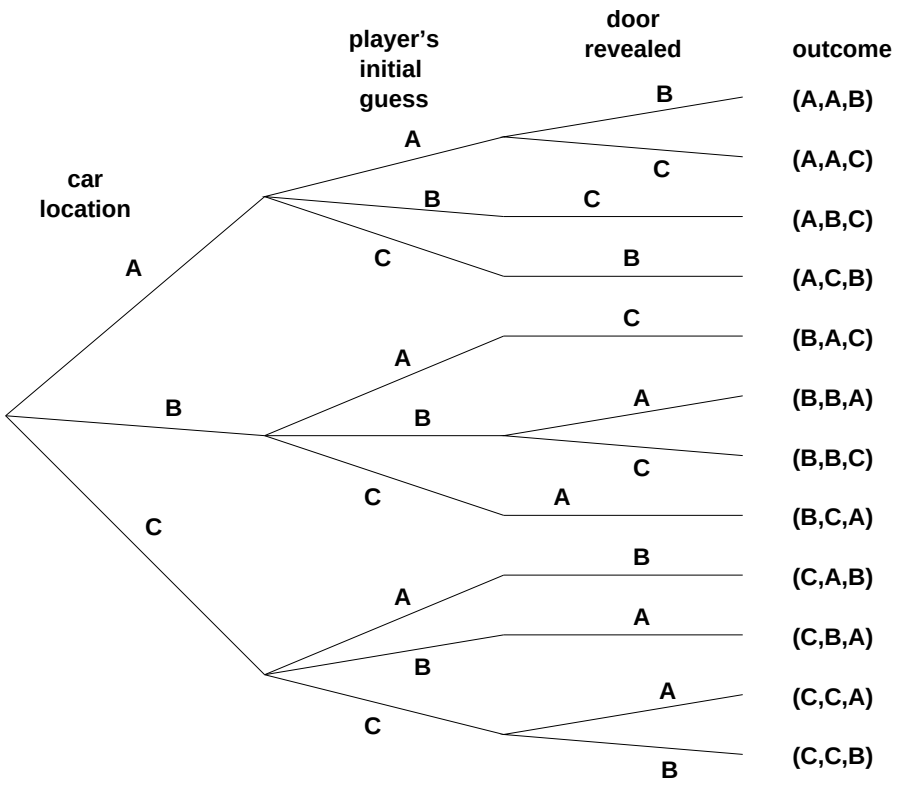
Every possible combination of these randomly-determined quantities is called an ***outcome***. The set of all possible outcomes is called the ***sample space*** for the experiment.

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:

$$\mathcal{S} = \left\{ \begin{array}{llllll} (A,A,B), & (A,A,C), & (A,B,C), & (A,C,B), & (B,A,C), & (B,B,A), \\ (B,B,C), & (B,C,A), & (C,A,B), & (C,B,A), & (C,C,A), & (C,C,B) \end{array} \right\}$$

The tree diagram has a broader interpretation as well: we can regard the whole experiment as "walk" from the root down to a leaf, where the branch taken at each stage is randomly determined. Keep this interpretation in mind; we'll use it again later.

12.1.4 Step 2: Define Events of Interest

Our objective is to answer questions of the form "What is the probability that —— ?", where the horizontal line stands for some phrase such as "the player wins by switching", "the player initially

picked the door concealing the prize", or "the prize is behind door C". Almost any such phrase can be modeled mathematically as an ***event***, which is defined to be a subset of the sample space.

For example, the event that the prize is behind door C is the set of outcomes:

$$\{(C,A,B),(C,B,A),(C,C,A),(C,C,B)\}$$

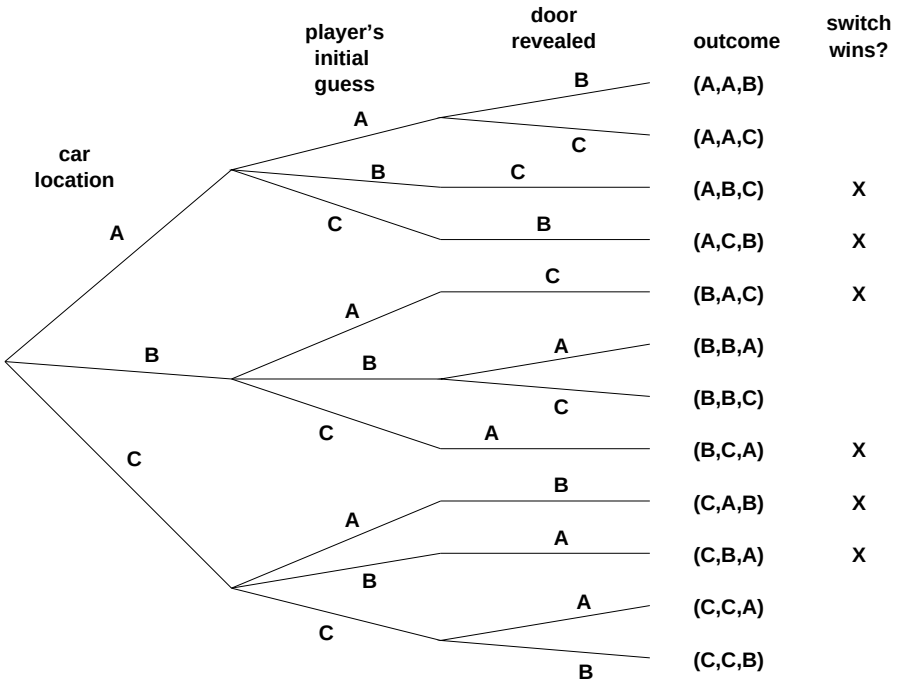
The event that the player initially picked the door concealing the prize is the set of outcomes:

$$\{(A,A,B),(A,A,C),(B,B,A),(B,B,C),(C,C,A),(C,C,B)\}$$

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.

12.1.5 Step 3: Determine Outcome Probabilities

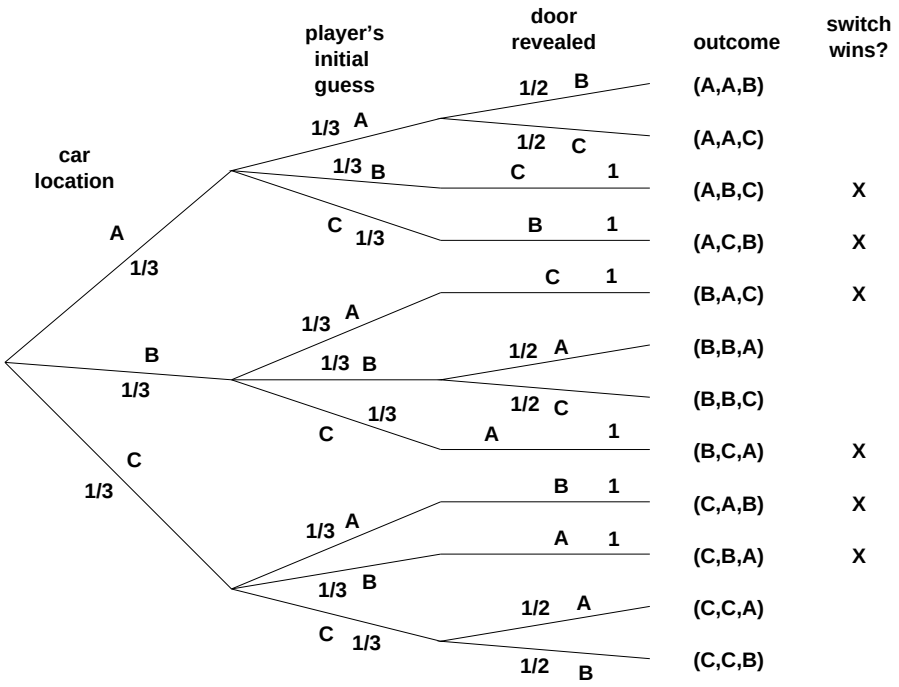
So far we've enumerated all the possible outcomes of the experiment. Now we must start assessing the likelihood of those outcomes. In particular, the goal of this step is to assign each outcome

a probability, which is a real number between 0 and 1. The sum of all outcome probabilities must be 1, reflecting the fact that exactly one outcome must occur.

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

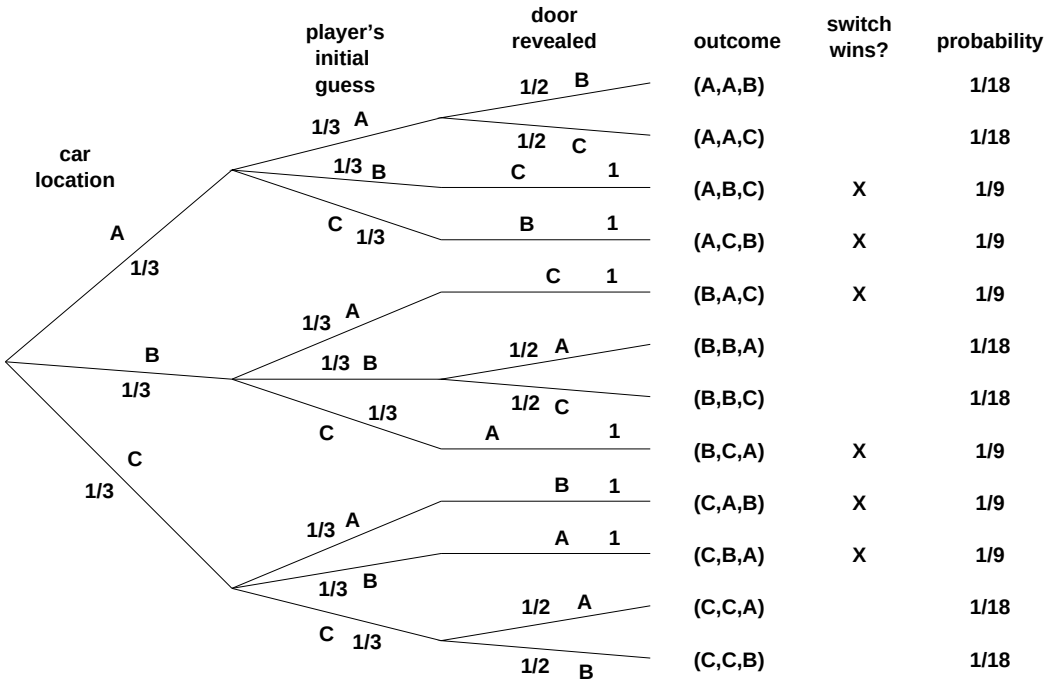
Our next job is to convert edge probabilities into outcome probabilities. This is a purely mechanical process: *the probability of an outcome is equal to the product of the edge-probabilities on the path from the root to that outcome*. For example, the probability of the topmost outcome, (A, A, B) is

$$\frac{1}{3} \cdot \frac{1}{3} \cdot \frac{1}{2} = \frac{1}{18}.$$

We'll justify this process formally later on. In the meantime, here is a nice informal justification to tide you over. Remember that the whole experiment can be regarded as a walk from the root of the tree diagram down to a leaf, where the branch taken at each step is randomly determined. In particular, the probabilities on the edges indicate how likely the walk is to proceed along each path. For example, a walk 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:

$$\Pr\{(A, A, B)\} = \frac{1}{18}$$
$$\Pr\{(A, A, C)\} = \frac{1}{18}$$
$$\Pr\{(A, B, C)\} = \frac{1}{9}$$
$$\text{etc.}$$

Earlier, we noted that the sum of all outcome probabilties must be 1 since exactly one outcome must occur. We can now express this symbolically:

$$\sum_{x \in \mathcal{S}} \Pr\{x\} = 1$$

In this equation, $\mathcal{S}$ denotes the sample space.

Though $\Pr$ is an ordinary function, just like your old friends f and g from calculus, we will subject it to all sorts of horrible notational abuses that f and g were mercifully spared. Just for starters, all of the following are common notations for the probability of an outcome x:

$$\Pr\{x\} \qquad \Pr(x) \qquad \Pr[x] \qquad \Pr x \qquad p(x)$$

A sample space $\mathcal{S}$ and a probability function $\Pr : \mathcal{S} \to [0,1]$ together form a ***probability space***. Thus, a probability space describes all possible outcomes of an experiment *and* the probability of each outcome. A probability space is a complete mathematical model of an experiment.

12.1.6 Step 4: Compute Event Probabilities

We now have a probability for each *outcome*, but we want to determine the probability of an *event*. We can bridge this gap with a definition:

> The *probability of an event* is the sum of the probabilities of the outcomes it contains.

As a notational matter, the probability of an event $E \subseteq \mathcal{S}$ is written $\Pr\{E\}$. Thus, our definition of the probability of an event can be written:

$$\Pr\{E\} ::= \sum_{x \in E} \Pr\{x\}.$$

For example, the probability of the event that the player wins by switching is:

$$\begin{aligned}\Pr\{\text{switching wins}\} &= \Pr\{A,B,C\} + \Pr\{A,C,B\} + \Pr\{B,A,C\} + \\ &\quad \Pr\{B,C,A\} + \Pr\{C,A,B\} + \Pr\{C,B,A\} \\ &= \frac{1}{9} + \frac{1}{9} + \frac{1}{9} + \frac{1}{9} + \frac{1}{9} + \frac{1}{9} \\ &= \frac{2}{3}\end{aligned}$$

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.

12.1.7 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!

12.1.8 Probability Identities

The definitions we've introduced lead to some useful identities involving probabilities. Many probability problems can be solved quickly with such identities, once you're used to them. If E is an event, then the ***complement of*** E consists of all outcomes not in E and is denoted $\overline{E}$. The probabilities of complementary events sum to 1:

$$\Pr\{E\} + \Pr\left\{\overline{E}\right\} = 1.$$

About half of the time, the easiest way to compute the probability of an event is to compute the probability of its complement and then apply this formula.

Suppose that events $E_1, \ldots, E_n$ are disjoint; that is, every outcome is in at most one event E_i. The ***sum rule*** says that the probability of the union of these events is equal to the sum of their probabilities:

$$\Pr\{E_1 \cup \cdots \cup E_n\} = \Pr\{E_1\} + \cdots + \Pr\{E_n\}.$$

The probability of the union of events that are not necessarily disjoint is given by an ***inclusion-exclusion formula*** analogous to the one for set sizes:

$$\Pr\{E_1 \cup \cdots \cup E_n\} = \sum_i \Pr\{E_i\} - \sum_{i,j} \Pr\{E_i \cap E_j\} + \sum_{i,j,k} \Pr\{E_i \cap E_j \cap E_k\} - \cdots.$$

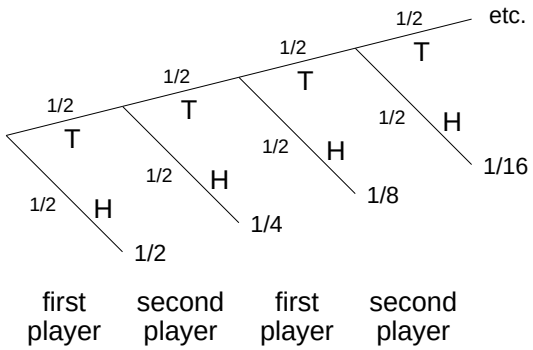
The following inequality, called the ***Union Bound***, also holds even if events $E_1, \ldots, E_n$ are not disjoint:

$$\Pr\{E_1 \cup \cdots \cup E_n\} \leq \Pr\{E_1\} + \cdots + \Pr\{E_n\}.$$

The Union Bound is simple and "good enough" for many probability 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 gives an upper bound on this vital probability and does not require engineers to estimate all the terms in the gigantic inclusion-exclusion formula.

12.2 Infinite Sample Spaces

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:

$$\begin{aligned}\Pr\{\text{first player wins}\} &= \frac{1}{2} + \frac{1}{8} + \frac{1}{32} + \frac{1}{128} + \cdots \\ &= \frac{2}{3}.\end{aligned}$$

The second step uses the formula for the sum of a geometric series. Similarly, we can compute the probability that the second player wins:

$$\begin{aligned}\Pr\{\text{second player wins}\} &= \frac{1}{4} + \frac{1}{16} + \frac{1}{64} + \frac{1}{256} + \cdots \\ &= \frac{1}{3}.\end{aligned}$$

In principle, the game could go on forever if both players kept flipping tails. In our tree diagram, this situation does not correspond to any leaf —rather, it corresponds to the infinite path. So, this is not an outcome. If we wanted to consider it as such, we could add an extra leaf as a child of the root. The probability on the edge to this leaf should then be the probability of the infinite path

$$\frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} \cdots = \lim_{n\to\infty} \left(\frac{1}{2}\right)^n = 0.$$

Since this probability is 0, there would be no impact on our calculations.

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 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

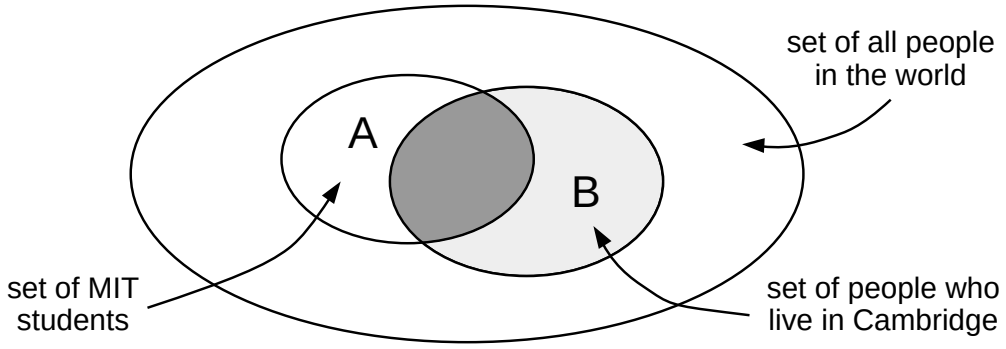
$$\frac{b_1}{2^1} + \frac{b_2}{2^2} + \frac{b_3}{2^3} + \cdots$$

is a rational number? We won't take up such problems in this course.

12.3 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 12.3.1.

$$\Pr\{A \mid B\} ::= \frac{\Pr\{A \cap B\}}{\Pr\{B\}}$$

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.

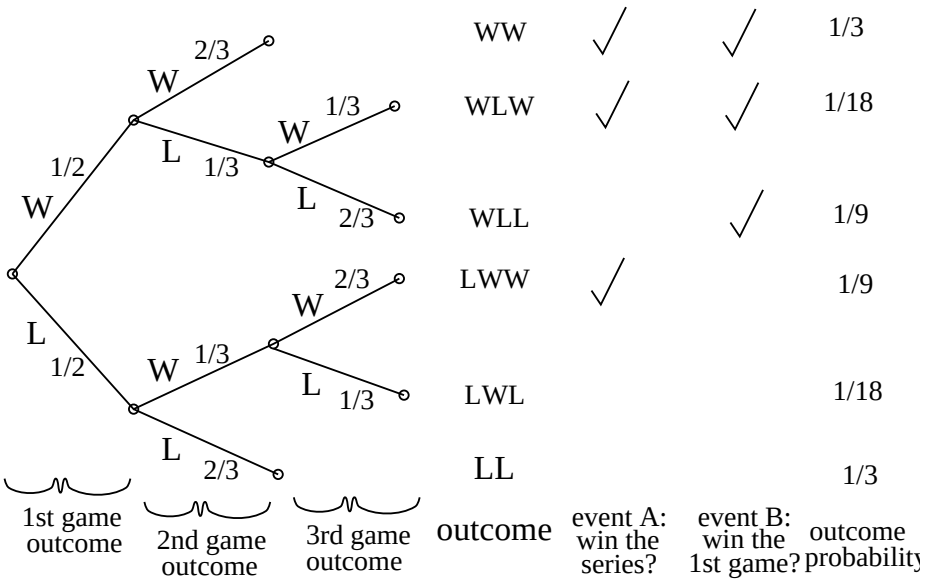
12.3.1 The Halting Problem

The *Halting Problem* is the canonical undecidable problem in computation theory that was first introduced by Alan Turing in his seminal 1936 paper. The problem is to determine whether a Turing machine halts on a given . . . blah, blah, blah. But what's *much more important*, it is 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 $7/9$.

12.3.2 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 the definition 12.3.1 of conditional probability which could be written in a form called the ***Product Rule*** for probabilities:

Lemma (Product Rule for 2 Events)**.** *If* $\Pr\{A_1\} \neq 0$*, then:*

$$\Pr\{A_1 \cap A_2\} = \Pr\{A_1\} \cdot \Pr\{A_2 \mid A_1\}$$

Multiplying edge probabilities in a tree diagram amounts to evaluating the right side of this equation. For example:

$$\begin{aligned}&\Pr\{\text{win first game} \cap \text{win second game}\}\\ &\quad= \Pr\{\text{win first game}\} \cdot \Pr\{\text{win second game} \mid \text{win first game}\}\\ &\quad= \frac{1}{2}\cdot\frac{2}{3}\end{aligned}$$

So the Product Rule is the formal justification for multiplying edge probabilities to get outcome probabilities!

To justify multiplying edge probabilities along longer paths, we need a more general form of the Product Rule:

Definition 12.3.2 (Product Rule for n Events)**.** If $\Pr\{A_1 \cap \ldots \cap A_{n-1}\} \neq 0$, then:

$$\Pr\{A_1 \cap \ldots \cap A_n\} = \Pr\{A_1\} \cdot \Pr\{A_2 \mid A_1\} \cdot \Pr\{A_3 \mid A_1 \cap A_2\} \cdots \Pr\{A_n \mid A_1 \cap \ldots \cap A_{n-1}\}$$

Let's interpret this big formula in terms of tree diagrams. Suppose we want to compute the probability that an experiment traverses a particular root-to-leaf path of length n. Let A_i be the event that the experiment traverses the i-th edge of the path. Then $A_1 \cap \ldots \cap A_n$ is the event that the experiment traverses the whole path. The Product Rule says that the probability of this is the probability that the experiment takes the first edge times the probability that it takes the second, *given* it takes the first edge, times the probability it takes the third, *given* it takes the first two edges, and so forth. In other words, the probability of an outcome is the product of the edge probabilities along the corresponding root-to-leaf path.

12.3.3 The Law of Total Probability

The following identity

$$\Pr\{A\} = \Pr\{A \mid E\} \cdot \Pr\{E\} + \Pr\left\{A \mid \overline{E}\right\} \cdot \Pr\left\{\overline{E}\right\}.$$

is called the Law of Total Probability and lets you compute the probability of an event A using case analysis based on whether or not event E occurs. 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, $\Pr\{E\} = \Pr\left\{\overline{E}\right\} = 1/2$. 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 $\Pr\left\{A \mid \overline{E}\right\} = 1/36$. Therefore, the probability that the whole process yields a 2 is

$$\Pr\{A\} = \frac{1}{2} \cdot \frac{1}{6} + \frac{1}{2} \cdot \frac{1}{36} = \frac{7}{72}.$$

More generally, if $E_1, \ldots, E_n$ are disjoint events whose union is the whole sample space, then:

$$\Pr\{A\} = \sum_{i=1}^{n} \Pr\{A \mid E_i\} \cdot \Pr\{E_i\}.$$

12.3.4 *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 $1/2$.

12.3.5 Medical Testing

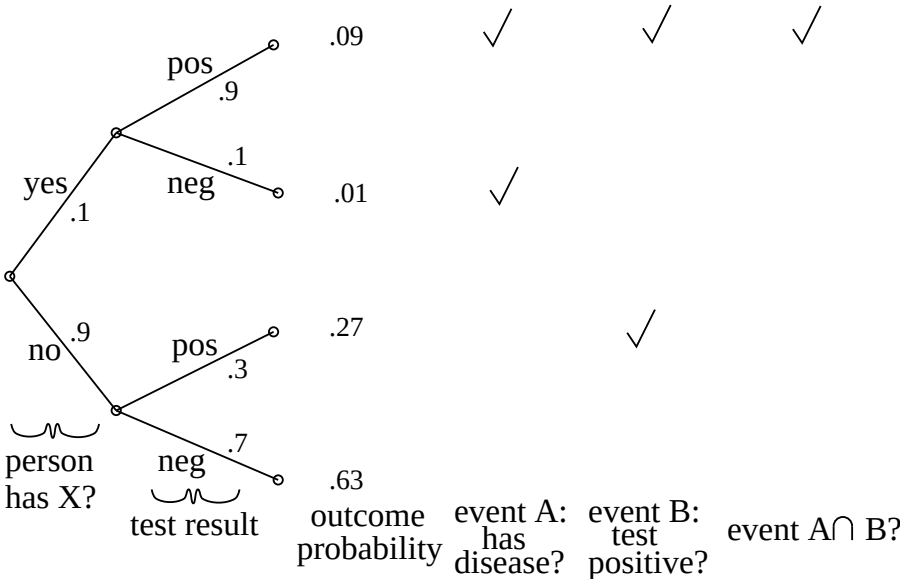
There is a deadly disease called X that has infected 10% of the population. There are no symptoms; victims just drop dead one day. Fortunately, there is a test for the disease. The test is not perfect, however:

- If you have the disease, there is a 10% chance that the test will say you do not. (These are called "false negatives".)
- If you do not have the disease, there is a 30% chance that the test will say you do. (These are "false positives".)

A random person is tested for the disease. If the test is positive, then what is the probability that the person has the disease?

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 the disease. 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 disease X, 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

$$\begin{aligned}\Pr\{A \mid B\} &= \frac{\Pr\{A \cap B\}}{\Pr\{B\}} \\ &= \frac{0.09}{0.09 + 0.27} \\ &= \frac{1}{4}\end{aligned}$$

If you test positive, then there is only a 25% chance that you have the disease!

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 disease. 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!

12.3.6 Other Identities

There is a close relationship between computing the size of a set and computing the probability of an event. The inclusion-exclusion formula is one such example; the probability of a union of events and the cardinality of a union of sets are computed using similar formulas.

In fact, all of the methods we developed for computing sizes of sets carry over to computing probabilities. This is because a probability space is just a weighted set; the sample space is the set and the probability function assigns a weight to each element. Earlier, we were counting the number of items in a set. Now, when we compute the probability of an event, we are just summing the weights of items. We'll see many examples of the close relationship between probability and counting over the next few weeks.

Many general probability identities still hold when all probabilities are conditioned on the same event. For example, the following identity is analogous to the Inclusion-Exclusion formula for two sets, except that all probabilities are conditioned on an event C.

$$\Pr\{A \cup B \mid C\} = \Pr\{A \mid C\} + \Pr\{B \mid C\} - \Pr\{A \cap B \mid C\}.$$

As a special case we have

$$\Pr\{A \cup B \mid C\} = \Pr\{A \mid C\} + \Pr\{B \mid C\} \qquad \text{when } A \cap B = \emptyset.$$

Be careful not to mix up events before and after the conditioning bar! For example, the following is *not* a valid identity:

False Claim.

$$\Pr\{A \mid B \cup C\} = \Pr\{A \mid B\} + \Pr\{A \mid C\} \qquad \textit{when } B \cap C = \emptyset. \tag{12.1}$$

12.4 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.

12.4.1 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 $1/10$.

12.4.2 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:

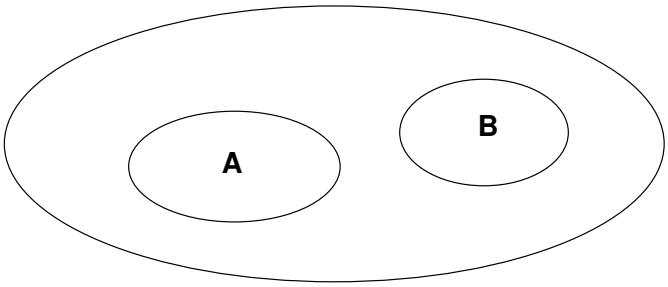
Proposition. *If* $\Pr\{B\} \neq 0$*, then events* A *and* B *are independent if and only if*

$$\Pr\{A \mid B\} = \Pr\{A\}. \tag{12.2}$$

Equation (12.2) 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.

12.4.3 Some Intuition

Suppose that A and B are disjoint events, as shown in the figure below.



Are these events independent? Let's check. On one hand, we know

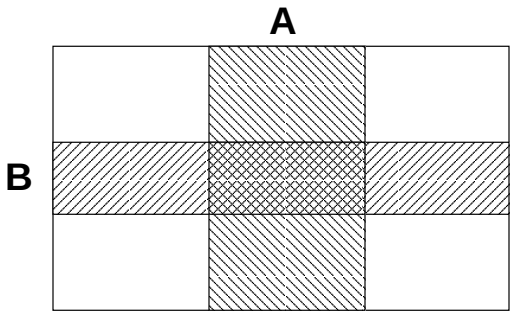
$$\Pr\{A \cap B\} = 0$$

because $A \cap B$ contains no outcomes. On the other hand, we have

$$\Pr\{A\} \cdot \Pr\{B\} > 0$$

except in degenerate cases where A or B has zero probability. Thus, *disjointness and independence are very different ideas*.

Here's a better mental picture of what independent events look like.



The sample space is the whole rectangle. Event A is a vertical stripe, and event B is a horizontal stripe. Assume that the probability of each event is proportional to its area in the diagram. Now if A covers an α-fraction of the sample space, and B covers a β-fraction, then the area of the intersection region is $\alpha \cdot \beta$. In terms of probability:

$$\Pr\{A \cap B\} = \Pr\{A\} \cdot \Pr\{B\}$$

12.5 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.

12.5.1 DNA Testing

This is testimony from the O. J. Simpson murder trial on May 15, 1995:

MR. CLARKE: When you make these estimations of frequency— and I believe you touched a little bit on a concept called independence?

DR. COTTON: Yes, I did.

MR. CLARKE: And what is that again?

DR. COTTON: It means whether or not you inherit one allele that you have is not— does not affect the second allele that you might get. That is, if you inherit a band at 5,000 base pairs, that doesn't mean you'll automatically or with some probability inherit one at 6,000. What you inherit from one parent is what you inherit from the other. *(Got that? – EAL)*

MR. CLARKE: Why is that important?

DR. COTTON: Mathematically that's important because if that were not the case, it would be improper to multiply the frequencies between the different genetic locations.

MR. CLARKE: How do you— well, first of all, are these markers independent that you've described in your testing in this case?

The jury was told that genetic markers in blood found at the crime scene matched Simpson's. Furthermore, the probability that the markers would be found in a randomly-selected person was at most 1 in 170 million. This astronomical figure was derived from statistics such as:

- 1 person in 100 has marker A.
- 1 person in 50 marker B.
- 1 person in 40 has marker C.
- 1 person in 5 has marker D.
- 1 person in 170 has marker E.

Then these numbers were multiplied to give the probability that a randomly-selected person would have all five markers:

$$\begin{aligned}\Pr\{A \cap B \cap C \cap D \cap E\} &= \Pr\{A\} \cdot \Pr\{B\} \cdot \Pr\{C\} \cdot \Pr\{D\} \cdot \Pr\{E\} \\ &= \frac{1}{100} \cdot \frac{1}{50} \cdot \frac{1}{40} \cdot \frac{1}{5} \cdot \frac{1}{170} \\ &= \frac{1}{170,000,000}\end{aligned}$$

The defense pointed out that this assumes that the markers appear mutually independently. Furthermore, all the statistics were based on just a few hundred blood samples. The jury was widely mocked for failing to "understand" the DNA evidence. If you were a juror, would *you* accept the 1 in 170 million calculation?

12.5.2 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 all *pairs* of events are independent!

A set of events is ***pairwise independent*** if every pair is independent. Pairwise independence is a much weaker property than mutual independence. For example, suppose that the prosecutors in the O. J. Simpson trial were wrong and markers A, B, C, D, and E appear only *pairwise* independently. Then the probability that a randomly-selected person has all five markers is no more than:

$$\begin{aligned}\Pr\{A \cap B \cap C \cap D \cap E\} &\leq \Pr\{A \cap E\} \\ &= \Pr\{A\} \cdot \Pr\{E\} \\ &= \frac{1}{100} \cdot \frac{1}{170} \\ &= \frac{1}{17,000}\end{aligned}$$

The first line uses the fact that $A \cap B \cap C \cap D \cap E$ is a subset of $A \cap E$. (We picked out the A and E markers because they're the rarest.) We use pairwise independence on the second line. Now the probability of a random match is 1 in 17,000— a far cry from 1 in 170 million! And this is the strongest conclusion we can reach assuming only pairwise independence.

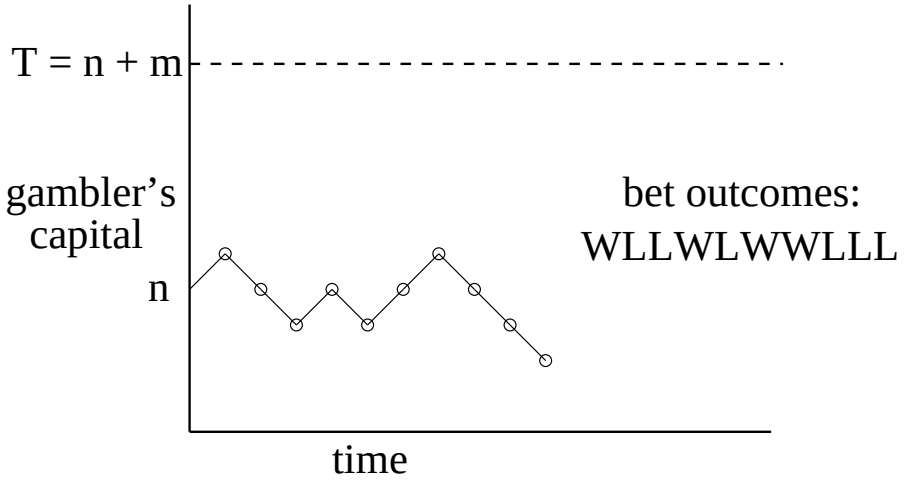


Figure 12.1: *This is a graph of the gambler's capital versus time for one possible sequence of bet outcomes. At each time step, the graph goes up with probability p and down with probability $1 - p$. The gambler continues betting until the graph reaches either 0 or $T = n + m$.*

12.6 Gamblers' Ruin

Random Walks are used to model situations in which an object moves in a sequence of steps in randomly chosen directions. Next week we'll explain how the Google search engine uses random walks through the graph of world-wide web links to determine the relative importance of web-sites. In this section we'll look at a simple 1-dimensional random walk —a walk along a straight line.

It's helpful and fun to describe these walks as a gambling experience. In particular, we 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. So the position on the line at any time corresponds to the gambler's cash-on-hand or *capital*. Now 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* 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 positive probability, p, less than 1 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 12.1. 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 $p < 1/2$; the corresponding random walks are called *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.

12.6.1 The Probability of Winning

The Unbiased Game

Let's begin by considering the case of a fair coin, that is, $p = 1/2$. For example, suppose the gambler starts with 100 dollars and 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 $1/2$.

OK, suppose he want to win the same \$100, but instead starts out with \$500. It turns out that now his probability of making the 100 dollars is $5/6$. So his chances of winning are really very good, namely, 5 chances out of 6. In general, 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.

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. For example, if the gambler started with one million dollars but still aimed to win 100 dollars, the probability he wins increases to $1M/(1M + 100) > .9999$. 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.

The Biased Game

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 in this game, 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 gamblers 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.

12.6.2 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

$$w_0 = 0, \tag{12.3}$$
$$w_T = 1. \tag{12.4}$$

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}. Overall, he is a winner with probability $w_n = pw_{n+1} + qw_{n-1}$. Solving for w_{n+1} we have

$$w_{n+1} = \frac{w_n}{p} - w_{n-1}\frac{q}{p}. \tag{12.5}$$

This recurrence holds only for $n+1 \leq T$, but there's no harm in using (12.5) 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 have from (12.5) and (12.3) that

$$\frac{xW(x)}{p} - \frac{q}{p}x^2 W(x) = W(x) - w_1 x,$$

so

$$W(x) = \frac{w_1 x}{(q/p)x^2 - (x/p) + 1} = \frac{w_1 x}{(1-x)(1-(q/p)x)}. \tag{12.6}$$

Now if $p \neq q$, then W can be expressed in partial fractions as

$$W(x) = \frac{A}{1-x} + \frac{B}{1-(q/p)x}. \tag{12.7}$$

From (12.6) and (12.7), we have

$$w_1 x = A(1-(q/p)x) + B(1-x).$$

Letting $x = 1$, we get $A = w_1/(1-(q/p))$, and letting $x = p/q$, we get $B = -w_1/(1-(q/p))$, so

$$W(x) = \frac{w_1}{1-(q/p)}\left(\frac{1}{1-x} - \frac{1}{1-(q/p)x}\right),$$

which implies

$$w_n = \frac{w_1}{(q/p)-1}\left(\left(\frac{q}{p}\right)^n - 1\right). \tag{12.8}$$

We can use (12.8) to solve for w_1 by letting $n = T$:

$$1 = w_T = \frac{w_1}{(q/p)-1}\left(\left(\frac{q}{p}\right)^T - 1\right)$$

so

$$w_1 = \frac{(q/p)-1}{(q/p)^T - 1},$$

and plugging this value of w_1 into (12.8), we finally arrive at the solution;

$$w_n = \frac{(q/p)^n - 1}{(q/p)^T - 1}. \tag{12.9}$$

The expression (12.9) 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.

Suppose that $p < 1/2$; that is, the game is biased *against* the gambler. Then both the numerator and denominator in the quotient in (12.9) are positive, and the quotient is less than one. This implies that

$$w_n < \frac{(q/p)^n}{(q/p)^T} = \left(\frac{p}{q}\right)^{T-n},$$

which proves:

Corollary 12.6.1. *In the Gambler's Ruin game biased against the Gambler, that is, with probability $p < 1/2$ of winning each individual bet, with initial capital, n, and target, T,*

$$\Pr\{\textit{the gambler is a winner}\} < \left(\frac{p}{q}\right)^{T-n}. \tag{12.10}$$

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 amazing consequences. For example, suppose that the gambler starts with \$500 aiming to profit \$100, this time by making \$1 bets on red in roulette. By (12.10), the probability, w_n, that he is a winner is less than

$$\left(\frac{18/38}{20/38}\right)^{100} = \left(\frac{9}{10}\right)^{100} < \frac{1}{37,648},$$

which is a dramatic contrast to the unbiased game, where he won with probability $5/6$. In fact his chance of going up a mere \$100 is less than 1 in 37,648 *no matter how much money he starts with*!

The bound (12.10) 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 odds of winning a little money are not so bad. Applying the exact formula (12.9), we find that the probability of winning \$10 before losing \$10 is

$$\frac{\left(\frac{20/38}{18/38}\right)^{10} - 1}{\left(\frac{20/38}{18/38}\right)^{20} - 1} = 0.2585\ldots.$$

This is somewhat worse than the 1 in 2 chance in the fair game, but not dramatically so.

The solution (12.9) only applies to biased walks, since we require $p \neq q$ so the denominator is not zero. But we can solve the fair case by taking the limit as q/p approaches 1 of (12.9), which by L'Hopital's Rule is n/T.

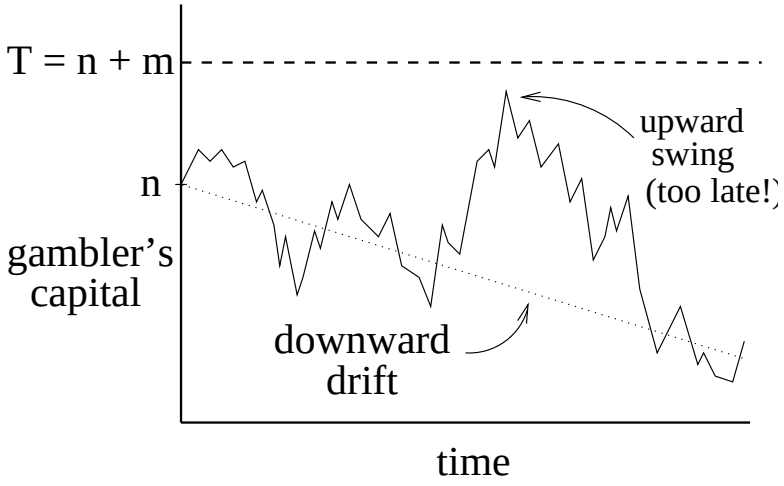


Figure 12.2: *In an unfair game, the gambler's capital swings randomly up and down, but steadily drifts downward. If the gambler does not have a winning swing early on, then his capital drifts downward, and later upward swings are insufficient to make him a winner.*

12.6.3 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 12.2.

For example, in roulette the gambler wins a dollar with probability $9/19$ and loses a dollar with probability $10/19$. Therefore, his average return on each bet is $9/10 - 10/19 = -1/19 \approx -0.053$ dollars. That is, on each bet his capital is can be expected to drift downward by a little over 5 cents.

Our intuition is that if the gambler starts with a trillion dollars, then he will 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.

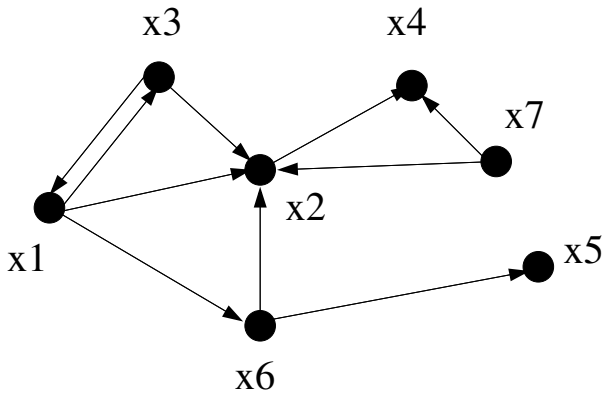
Chapter 13

Random Variables & Sampling

13.1 Random Walks on Graphs

Random walks on graphs arise in all sorts of applications. One interesting example is Google and page rank, which we'll explore in this section.

The hyperlink structure of the World Wide Web can be described as a digraph. The nodes are the web pages or web-accessible files. There is a directed edge from node x to node y if the page associated with x has a link to the page associated with 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 nodes. At first glance, this graph wouldn't seem to be very interesting. But in 1995, two students at Stanford, Larry Page and 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 straighforward 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, doing a search on Google for "math for computer science notes" gives $378,000$ 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.

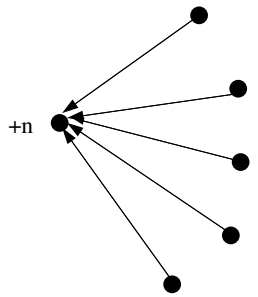
One way to get placed high on the list is to pay Google an advertising fees —and Google gets an enormous revenue stream from these fees. Of course an early listing is worth a fee only if an advertiser's target audience is attracted to the listing. But an audience does get attracted to Google listings because its ranking method is really good at determining the most important relevant web pages. For example, Google demonstrated its accuracy in our case by giving first rank to the Fall 2002 open courseware page for 6.042 :-) . So how did Google know to pick 6.042 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.

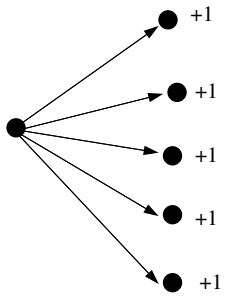
13.1.1 A First Crack at Page Rank

Looking at the web graph, any idea which node/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.

13.1.2 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 node, 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

$$\begin{aligned}\Pr\{\text{go to } y\} &= \sum_{\text{edges } x\to y} \Pr\{\text{follow link } x \to y \mid \text{ at page } x\} \cdot \Pr\{\text{at page } x\} \\ &= \sum_{\text{edges } x\to y} \frac{\Pr\{\text{at } x\}}{\text{outdegree}(x)} \end{aligned} \tag{13.1}$$

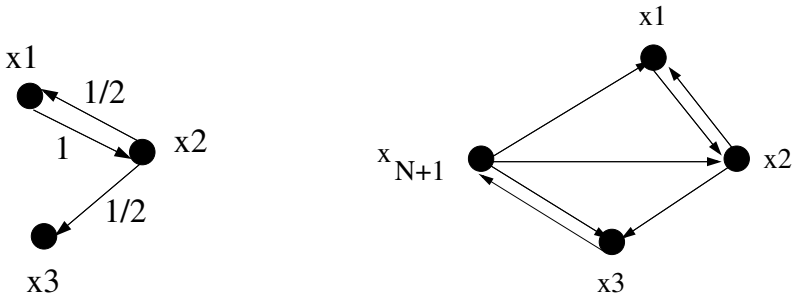
For example, in our web graph, we have

$$\Pr\{\text{go to } x_4\} = \frac{\Pr\{\text{at } x_7\}}{2} + \frac{\Pr\{\text{at } x_2\}}{1}.$$

One can think of this equation as x_7 sending $\frac{1}{2}$ of 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 supernode to the web graph and had every page with no hyperlinks point to it. Moreover, the supernode points to every other node 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 supernode x_{N+1}.



The addition of the supernode also removes the possibility that the value $1/\text{outdegree}(x)$ might involve a division by zero.

13.1.3 Stationary Distribution & Page Rank

Page rank is basically just a stationary distribution over the web graph (there are some more details, but this is the main idea), so let's define a stationary distribution.

Suppose each node is assigned a probability that corresponds, intuitively, to the likelihood that a random walker is at that node at a randomly chosen time. We assume that the walk never leaves the nodes in the graph, so we require that

$$\sum_{\text{nodes } x} \Pr\{\text{at } x\} = 1. \tag{13.2}$$

Definition 13.1.1. An assignment of probabililties to nodes in a digraph is a ***stationary distribution*** if for all nodes 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 node, x, such that

$$\text{PR}(x) = \sum_{\text{edges } y \to x} \frac{\text{PR}(y)}{\text{outdegree}(y)}, \tag{13.3}$$

corresponding to the intuitive equations given in (13.1). These numbers must also satisfy the additional constraint corresponding to (13.2):

$$\sum_{\text{nodes } x} \text{PR}(x) = 1. \tag{13.4}$$

So if there are n nodes, then equations (13.3) and (13.4) provide a system of $n+1$ linear equations in the n variables, $\text{PR}(x)$. Note that constraint (13.4) is needed because the remaining constraints (13.3) 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 supernode ensures there is always a *unique* stationary distribution. Moreover, starting from *any* node 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 supernodes 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.

[Optional] Here's a note on solving the system of linear constraints, for the interested reader.

Let W be the $n \times n$ with the entry w_{ij} (in row i and column j) having the value $w_{ij} = 1/\text{outdegree}(x_i)$ if edge $x_i \to x_j$ exists, and $w_{ij} = 0$ otherwise. For example, in our last example with the 4-node graph (including the supernode), we have W given by:

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 1 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 \end{pmatrix}$$

The system of linear equations can now be described by a single matrix vector product equation $W^T \vec{P} = \vec{P}$, where W^T denotes the transpose of W, and $\vec{P}$ is the column vector of page probabilities (ranks):

$$\vec{P} ::= \begin{pmatrix} \text{PR}(x_1) \\ \text{PR}(x_2) \\ \vdots \\ \text{PR}(x_n) \end{pmatrix}$$

So the jth entry of the solultion vector, $\vec{P}$, is

$$\sum_{1 \le i \le n} w_{ij} \cdot \text{PR}(x_i) = \sum_{i | x_i \to x_j} \frac{\text{PR}(x_i)}{\text{outdegree}(x_i)},$$

which is exactly the constraint corresponding to node x_j in (13.3).

If you have taken a linear algebra or numerical analysis course, you realize that the vector of page ranks is just the *principle eigenvector* of the matrix, W, of the web graph! Once you've had such a course, these values are easy to compute. Of course, when you are dealing with matrices of this size, the problem gets a little more interesting.

Now just keeping track of the digraph whose nodes 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 6.042 ranked first out of $378,000$ matches. Lots of other universities use our notes and probably have links to the 6.042 open courseware site and they are themselves legitimate, which ultimately leads 6.042 to get a high page rank in the web graph.

13.2 Random Variables

Last week we focused on probabilities of ***events***: what is the probability of the event that you win the Monty Hall game? ...that you have a rare disease, given that you tested positive. ...that the Red Sox won the pennant, given that they lost the second game?

This week we focus on quantitative questions: *How many* contestants must play the Monty Hall game until one of them finally wins? . . . *How long* will this illness last? *How much* will I lose playing 6.042 games all day? A *random variable* is the mathematical tool for addressing such questions.

Definition 13.2.1. A random variable, R, is a total function whose domain is $\mathcal{S}$, the sample space of outcomes.

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!

13.2.1 Examples

Consider the experiment of tossing 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:

$$\begin{array}{rclrcl} C(HHH) & = & 3 & C(THH) & = & 2 \\ C(HHT) & = & 2 & C(THT) & = & 1 \\ C(HTH) & = & 2 & C(TTH) & = & 1 \\ C(HTT) & = & 1 & C(TTT) & = & 0. \end{array}$$

Similarly, M is a function mapping each outcome another way:

$$\begin{array}{rclrcl} M(HHH) & = & 1 & M(THH) & = & 0 \\ M(HHT) & = & 0 & M(THT) & = & 0 \\ M(HTH) & = & 0 & M(TTH) & = & 0 \\ M(HTT) & = & 0 & M(TTT) & = & 1. \end{array}$$

So C and M are ***random variables***.

13.2.2 Indicator Random Variables

An ***indicator random variable*** (or simply an ***indicator***, or a ***Bernoulli random variable***) is a random variable that maps every outcome to either 0 or 1. 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, for the event, 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.

13.2.3 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 equations involving random variables. For example:

$$\begin{aligned} \Pr\{C = 2\} &= \Pr\{THH\} + \Pr\{HTH\} + \Pr\{HHT\} \\ &= \frac{1}{8} + \frac{1}{8} + \frac{1}{8} = \frac{3}{8}. \end{aligned}$$

As another example:

$$\begin{aligned} \Pr\{M = 1\} &= \Pr\{TTT\} + \Pr\{HHH\} \\ &= \frac{1}{8} + \frac{1}{8} = \frac{1}{4}. \end{aligned}$$

13.2.4 Conditional Probability

Mixing conditional probabilities and events involving random variables creates no new difficulties. For example, $\Pr\{C \geq 2 \mid M = 0\}$ is the probability that at least two coins are heads ($C \geq 2$), given that not all three coins are the same ($M = 0$). We can compute this probability using the definition of conditional probability:

$$\begin{aligned} \Pr\{C \geq 2 \mid M = 0\} &= \frac{\Pr\{[C \geq 2] \cap [M = 0]\}}{\Pr\{M = 0\}} \\ &= \frac{\Pr\{\{THH, HTH, HHT\}\}}{\Pr\{\{THH, HTH, HHT, HTT, THT, TTH\}\}} \\ &= \frac{3/8}{6/8} = \frac{1}{2}. \end{aligned}$$

The expression $[C \geq 2] \cap [M = 0]$ on the first line may look odd; what is the set operation $\cap$ doing between an inequality and an equality? But recall that, in this context, $[C \geq 2]$ and $[M = 0]$ are *events*, namely, *sets* of outcomes.

13.2.5 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:

$$\Pr\{[R_1 = x_1] \cap [R_2 = x_2]\} = \Pr\{R_1 = x_1\} \cdot \Pr\{R_2 = x_2\}.$$

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:

$$\Pr\{[C = x_1] \cap [M = x_2]\} \quad \neq \quad \Pr\{C = x_1\} \cdot \Pr\{M = x_2\}.$$

One appropriate choice of values is $x_1 = 2$ and $x_2 = 1$. In this case, we have:

$$\Pr\{[C = 2] \cap [M = 1]\} = 0 \neq \frac{1}{4} \cdot \frac{3}{8} = \Pr\{M = 1\} \cdot \Pr\{C = 2\}.$$

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

$$\begin{aligned}\Pr\{M = 1\} = 1/4 = \Pr\{M = 1 \mid H_1 = 1\} = \Pr\{M = 1 \mid H_1 = 0\}\\ \Pr\{M = 0\} = 3/4 = \Pr\{M = 0 \mid H_1 = 1\} = \Pr\{M = 0 \mid H_1 = 0\}\end{aligned}$$

In fact, this example is an instance of the important observation that 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 13.2.2. Random variables $R_1, R_2, \ldots, R_n$ are ***mutually independent*** iff

$$\begin{aligned}&\Pr\{[R_1 = x_1] \cap [R_2 = x_2] \cap \cdots \cap [R_n = x_n]\}\\ &= \Pr\{R_1 = x_1\} \cdot \Pr\{R_2 = x_2\} \cdots \Pr\{R_n = x_n\}.\end{aligned}$$

for all $x_1, x_2, \ldots, x_n$.

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 probabilites that the individual variables take their values. Thus, for example, if $R_1, R_2, \ldots, R_{100}$ are mutually independent random variables, then it follows that:

$$\begin{aligned}&\Pr\{[R_1 = 7] \cap [R_7 = 9.1] \cap [R_{23} = \pi] \cap [R_{87} = -1]\}\\ &= \Pr\{R_1 = 7\} \cdot \Pr\{R_7 = 9.1\} \cdot \Pr\{R_{23} = \pi\} \cdot \Pr\{R_{87} = -1\}.\end{aligned}$$

13.3 Probability Distributions

A random variable is defined to be a function whose domain is the sample space of an experiment. Often, however, random variables with essentially the same properties show up in completely different experiments. For example, some random variables that come up in polling, in primality testing, and in coin flipping all share some common properties. If we could study such random variables in the abstract, divorced from the details of any particular experiment, then our conclusions would apply to *all* the experiments where that sort of random variable turned up. Such general conclusions could be very useful. There are a couple tools that capture the essential properties of a random variable, but leave other details of the associated experiment behind.

The ***probability density function (pdf)*** for a random variable R with codomain V is a function $\text{PDF}_R : V \to [0,1]$ defined by:

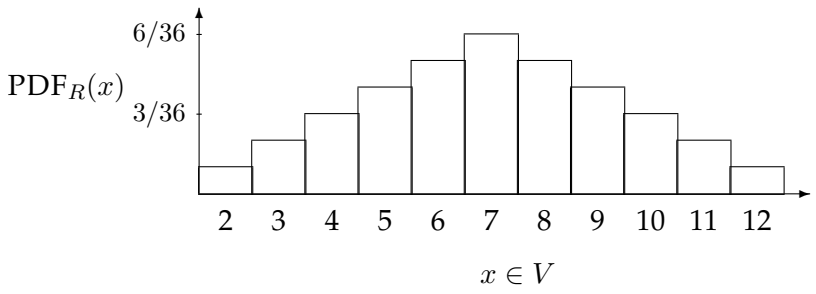
$$\text{PDF}_R(x) = \Pr\{R = x\}$$

A consequence of this definition is that

$$\sum_{x \in V} \text{PDF}_R(x) = 1$$

since the random variable always takes on exactly one value in the set V.

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 $V = \{2, 3, \ldots, 12\}$. A plot of the probability density function is shown below:

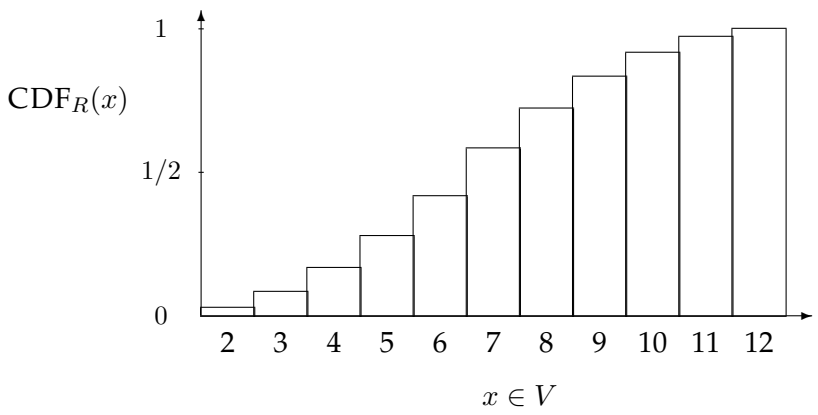


The lump in the middle indicates that sums close to 7 are the most likely. The total area of all the rectangles is 1 since the dice must take on exactly one of the sums in $V = \{2, 3, \ldots, 12\}$.

A closely-related idea is the ***cumulative distribution function (cdf)*** for a random variable R whose range 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. Thus, through these functions, we can study random variables without reference to a particular experiment.

We'll now look at three important distributions and some applications.

13.3.1 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

$$\begin{aligned}\text{PDF}_B(0) &= p \\ \text{PDF}_B(1) &= 1 - p\end{aligned}$$

where $0 \leq p \leq 1$. The corresponding cumulative distribution function is:

$$\begin{aligned}\text{CDF}_B(0) &= p \\ \text{CDF}_B(1) &= 1\end{aligned}$$

13.3.2 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:

$$\text{PDF}_U(k) = \frac{1}{N}$$

And the cumulative distribution function is:

$$\text{CDF}_U(k) = \frac{k}{N}$$

Uniform distributions come up all the time. For example, the number rolled on a fair die is uniform on the set $\{1, 2, \ldots, 6\}$.

13.3.3 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 evelope 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:

$$\left\{\frac{1}{2},\ 1\frac{1}{2},\ 2\frac{1}{2},\ \ldots,\ n-\frac{1}{2}\right\}$$

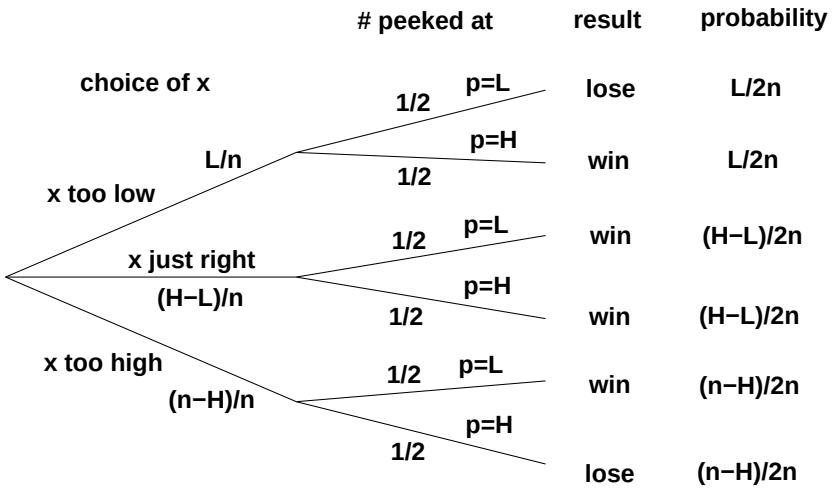
But what probability distribution should you use?

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!

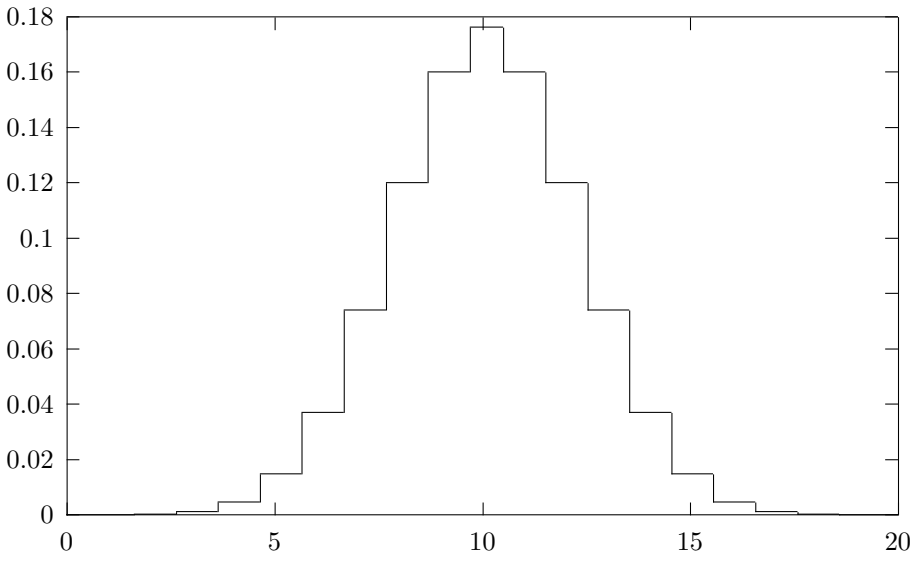
13.3.4 Binomial Distribution

Of the more complex distributions, the ***binomial distribution*** is surely the most important in Computer Science. 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*:

$$\text{PDF}_{H_n}(k) = \binom{n}{k} 2^{-n}.$$

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 $\text{PDF}_{H_n}(k)$ 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***.



An enormous number of analyses in Computer Science come down to proving that the tails of the binomial and similar distributions are very small. 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.

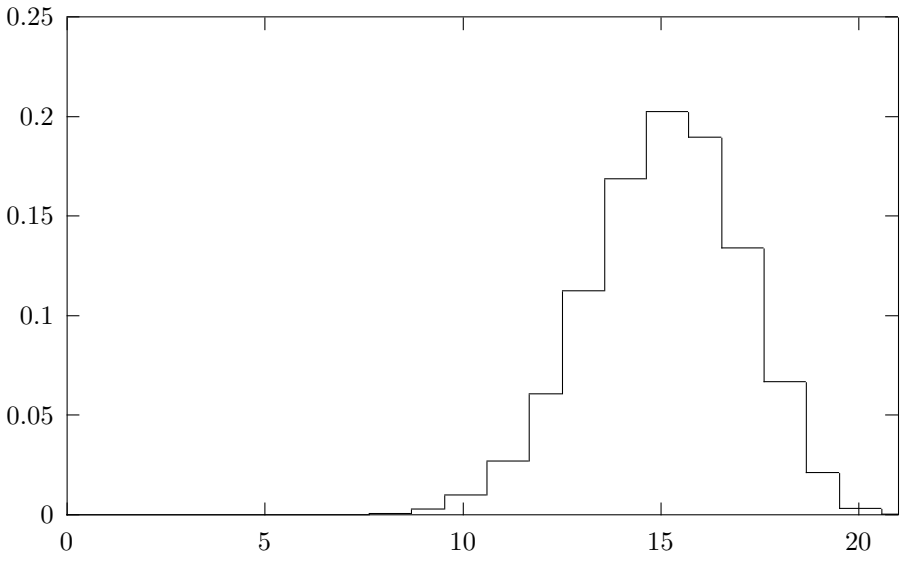
The General Binomial Distribution

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*:

$$\text{PDF}_J(k) = \binom{n}{k} p^k (1-p)^{n-k}.$$

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 $\text{PDF}_J(k)$ 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.



Approximating the Binomial Density Function

Computing the general binomial density function is daunting if not impossible when n is large –say $n \geq 2000$. Fortunately, there is an approximate closed-form formula for this function, which, though a bit unwieldy, is easy to calculate. First, we need an approximation for the binomial coefficient in the exact formula. For convenience, let's replace k by αn where α is a number between 0 and 1. Then, from Stirling's formula, we find that:

$$\binom{n}{\alpha n} \leq \frac{2^{nH(\alpha)}}{\sqrt{2\pi\alpha(1-\alpha)n}}$$

where $H(\alpha)$ is the famous ***entropy function***:

$$H(\alpha) ::= \alpha \log_2 \frac{1}{\alpha} + (1-\alpha)\log_2 \frac{1}{1-\alpha}$$

Its graph is shown in Figure 13.1.

This upper bound on $\binom{n}{\alpha n}$ is very tight and serves as an excellent approximation.

Now let's plug this formula into the general binomial density function. The probability of flipping αn heads in n tosses of a coin that comes up heads with probability p is:

$$\text{PDF}_J(\alpha n) \leq \frac{2^{nH(\alpha)}}{\sqrt{2\pi\alpha(1-\alpha)n}} \cdot p^{\alpha n}(1-p)^{(1-\alpha)n} \tag{13.5}$$

This formula is ugly as a bowling shoe, but quite useful. For example, suppose we flip a fair coin n times. What is the probability of getting *exactly* $\frac{1}{2}n$ heads? Plugging $\alpha = 1/2$ and $p = 1/2$ into this formula gives:

$$\begin{aligned}\text{PDF}_J(\alpha n) &\leq \frac{2^{nH(1/2)}}{\sqrt{2\pi(1/2)(1-(1/2))n}} \cdot 2^{-n} \\ &= \sqrt{\frac{2}{\pi n}}\end{aligned}$$

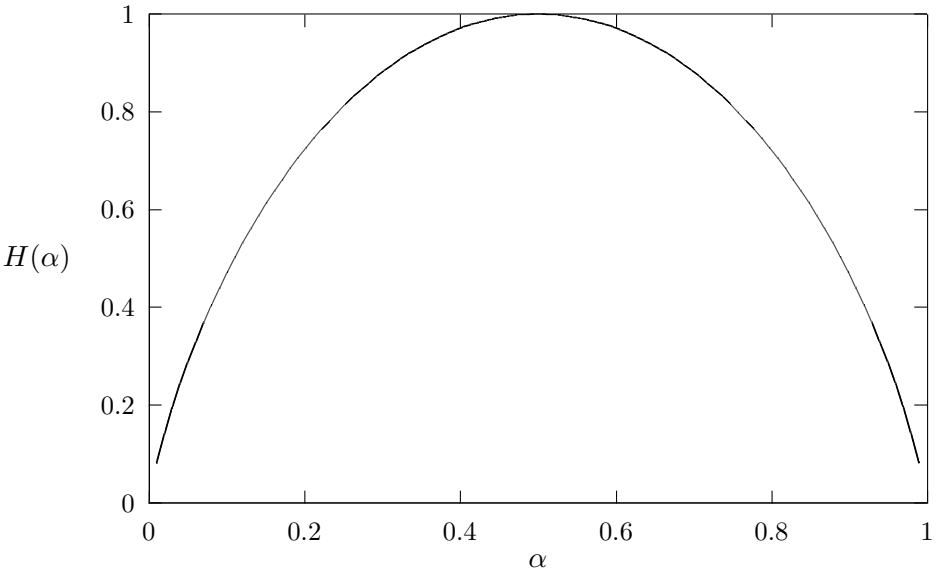


Figure 13.1: The Entropy Function

Thus, for example, if we flip a fair coin 100 times, the probability of getting exactly 50 heads is about $1/\sqrt{50\pi} \approx 0.079$ or around 8%.

13.3.5 Approximating the Cumulative Binomial Distribution Function

Suppose a coin comes up heads with probability p. As before, let the random variable J be the number of heads that come up on n independent flips. Then the probability of getting *at most* k heads is given by the cumulative binomial distribution function:

$$\begin{aligned} \text{CDF}_J(k) &= \Pr\{J \leq k\} \\ &= \sum_{i=0}^{k} \text{PDF}_J(i) \\ &= \sum_{i=0}^{k} \binom{n}{i} p^i (1-p)^{n-i} \end{aligned} \tag{13.6}$$

Evaluating this expression directly would be a lot of work for large k and n, so now an approximation would be really helpful. Once again, we can let $k = \alpha n$; that is, instead of thinking of the absolute number of heads (k), we consider the fraction of flips that are heads (α). The following approximation holds provided $\alpha < p$:

$$\begin{aligned} \text{CDF}_J(\alpha n) &\leq \frac{1-\alpha}{1-\alpha/p} \cdot \text{PDF}_J(\alpha n) \\ &\leq \frac{1-\alpha}{1-\alpha/p} \cdot \frac{2^{nH(\alpha)}}{\sqrt{2\pi\alpha(1-\alpha)n}} \cdot p^{\alpha n}(1-p)^{(1-\alpha)n} \qquad \text{(by (13.5))} \end{aligned} \tag{13.7}$$

The first inequality above can be derived by bounding the summation (13.6) with a geometric sum and applying the formula for the sum of a geometric series. (The details are dull and omitted.)

It would be awkward to evaluate (13.7) with a calculator, but it's easy to write a program to do it. So don't look gift blessings in the mouth before they hatch. Or something.

As an example, the probability of flipping at most 25 heads in 100 tosses of a fair coin is obtained by setting $\alpha = 1/4$, $p = 1/2$ and $n = 100$:

$$\text{CDF}_J\left(\frac{n}{4}\right) \leq \frac{1-(1/4)}{1-(1/4)/(1/2)} \cdot \text{PDF}_J\left(\frac{n}{4}\right) \leq \frac{3}{2} \cdot 1.913 \cdot 10^{-7}.$$

This says that flipping 25 or fewer heads is extremely unlikely, which is consistent with our earlier claim that the tails of the binomial distribution are very small. In fact, notice that the probability of flipping *25 or fewer* heads is only 50% more than the probability of flipping *exactly 25* heads. Thus, flipping exactly 25 heads is twice as likely as flipping any number between 0 and 24!

Caveat: The upper bound on $\text{CDF}_J(\alpha n)$ holds only if $\alpha < p$. If this is not the case in your problem, then try thinking in complementary terms; that is, look at the number of tails flipped instead of the number of heads. In our example, the probability of flipping 75 or more heads is the same as the probability of flipping 25 or fewer tails. By the above analysis, this is also extremely small.

13.4 Polling

Suppose we want to estimate the fraction of the U.S. voting population who would favor Hillary Clinton over Rudy Giuliani in the year 2008 presidential election.

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 Clinton, 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 Clinton. That is, we define variables $K_1, K_2, \ldots$, where K_i is interpreted to be the indicator variable for the event that the ith chosen voter prefers Clinton. 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 $K_1, K_2, \ldots$, each with the same probability, p, of being equal to 1. Now let S_n be their sum, that is,

$$S_n ::= \sum_{i=1}^{n} K_i. \tag{13.8}$$

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 *in our sample* who favor Clinton. We would expect that S_n/n should be something like p. We will use the sample value, S_n/n, as our *statistical estimate* of p.

[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.

13.4.1 Sampling

Suppose we want our estimate of p to be within 0.04 of p at least 95% of the time. Namely, we want

$$\Pr\left\{\left|\frac{S_n}{n} - p\right| \leq 0.04\right\} \geq 0.95\,.$$

We let ϵ be the margin of error we can tolerate, and let δ be the probability that our result lies outside this margin, so in this case we'd have $\epsilon = 0.04$ and $\delta \leq 0.05$.

We want to determine the number, n, of times we must poll voters so that the value, S_n/n, of our estimate will, with probability at least $1-\delta$, be within ϵ of the actual fraction in the nation favoring Clinton.

We can define δ, the probability that our poll is off by more than the margin of error ϵ, as follows:

$$\begin{aligned}\delta &= \underbrace{\Pr\left\{\frac{S_n}{n} \leq p-\epsilon\right\}}_{\substack{\text{too many in sample}\\ \text{prefer ``Giuliani''}}} + \underbrace{\Pr\left\{\frac{S_n}{n} \geq p+\epsilon\right\}}_{\substack{\text{too many in sample}\\ \text{prefer ``Clinton''}}}\\ &= \Pr\{S_n \leq (p-\epsilon)n\} + \Pr\{S_n \geq (p+\epsilon)n\}\,.\end{aligned}$$

Now

$$\mathrm{CDF}_{S_n}((p-\epsilon)n) ::= \Pr\{S_n \leq (p-\epsilon)n\}$$

Also,

$$\Pr\{S_n \geq (p+\epsilon)n\} = \Pr\{n - S_n \leq ((1-p)-\epsilon)n\}\,.$$

But $T_n ::= n - S_n$ is simply the number of voters in the sample who prefer Giuliani, which is a sum of Bernoulli random variables with parameter $1-p$, and therefore

$$\Pr\{T_n \leq ((1-p)-\epsilon)n\} = \mathrm{CDF}_{T_n}(((1-p)-\epsilon)n).$$

Hence

$$\delta = \mathrm{CDF}_{S_n}((p-\epsilon)n) + \mathrm{CDF}_{T_n}(((1-p)-\epsilon)n). \tag{13.9}$$

So we have reduced getting a good estimate of the required sample size to finding good bounds on two cumulative binomial distributions with parameters p and $1-p$ respectively.

Using the bound on the cumulative binomial distribution function allows us to calculate an expression bounding (13.9) in terms of n, ϵ and p. The problem is that this bound would contain the fraction, p, of Americans that prefer Clinton which is the unknown number we are trying to determine by polling in the first place! Fortunately, there is a simple way out of this circularity. Since (13.9) is symmetric in p, it has an inflection point when $p = 1/2$, and this inflection point is, in fact, its maximum:

Fact. For all ϵ, n, the maximum value of δ in equation (13.9) occurs when $p = 1/2$.

In other words, the binomial tails fall off most slowly when $p = 1/2$. Using this fact, and plugging into the inequality (13.7) bounding $\mathrm{CDF}_{S_n}((p-\epsilon)n)$ and $\mathrm{CDF}_{T_n}(((1-p)-\epsilon)n)$, we get the following theorem:

Theorem 13.4.1. *[Binomial Sampling] Let $K_1, K_2, \ldots,$ be a sequence of mutually independent 0-1-valued random variables with the same probability, p, that $K_i = 1$, and let*

$$S_n ::= \sum_{i=1}^{n} K_i.$$

Then, for $1/2 > \epsilon > 0$,

$$\Pr\left\{\left|\frac{S_n}{n} - p\right| \geq \epsilon\right\} \leq \frac{1+2\epsilon}{2\epsilon} \cdot \frac{2^{-n(1-H((1/2)-\epsilon))}}{\sqrt{2\pi(1/4-\epsilon^2)n}}. \tag{13.10}$$

We want $\epsilon = 0.04$, so plugging into (13.10) gives

$$\delta \leq 13.5 \cdot \frac{2^{-n(0.00462)}}{1.2492\sqrt{n}} \tag{13.11}$$

where δ is the probability that our estimate is not within ϵ of p. We want to poll enough people so that $\delta \leq 0.05$. The easiest way to find the necessary sample size n is to plug in values for n to find the smallest one where the righthand side of (13.11) is ≤ 0.05:

n = people polled	upper bound on probability poll is wrong	
500	9.7%	
600	6.4%	
623	5.9%	
650	5.3%	
664	5.0%	← our poll size
700	4.3%	

So 95% of the time, polling 664[2] people will yield a fraction that is within 0.04 of the actual fraction of voters preferring Clinton. This method of estimation by sampling a quantity —voting preference in this example— is a technique that can obviously be used to estimate many other unknown quantities.

We just showed that sampling merely 664 voters will yield a fraction that, 95% of the time, is within 0.04 of the actual fraction of the voting population who prefer Clinton. Notice that the actual size of the voting population was never considered because *it did not matter*. Polling only a few hundred is always sufficient, whether there are a thousand, a million, or a billion voters in the country —which people often find remarkable.

13.4.2 Confidence Levels

Suppose a pollster uses a sample of 664 random voters to estimate the fraction of voters who prefer Clinton, and the pollster finds that 364 of them prefer Clinton. It's tempting, **but sloppy**, to say that this means:

False Claim. *With probability 0.95, the fraction, p, of voters who prefer Clinton is $364/664 \pm 0.04$. Since $364/664 - 0.04 > 0.50$, there is a 95% chance that more than half the voters prefer Clinton.*

[2] An exact calculation of the binomial CDF shows that a somewhat smaller poll size of 589 would be sufficient.

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 Clinton is more than 0.50. 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 0.49; then it's nonsense to ask about the probability that it is within 0.04 of 364/664 —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 Clinton is $364/664 \pm 0.04$.

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.

Chapter 14

The Expected Value of a Random Variable

14.1 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 knowb about a random variable is its expected value.

Definition 14.1.1.

$$\begin{aligned}\mathrm{E}[R] ::= & \sum_{x \in \text{range}(R)} x \cdot \Pr\{R = x\} \\ = & \sum_{x \in \text{range}(R)} x \cdot \mathrm{PDF}_R(x).\end{aligned} \tag{14.1}$$

Let's work through an example. Let R be the number that comes up on a fair, six-sided die. Then by (14.1), 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 can't roll a $3\frac{1}{2}$ on an ordinary die!

There is an even simpler formula for expectation:

Theorem 14.1.2. *If R is a random variable defined on a sample space, $\mathcal{S}$, then*

$$\mathrm{E}[R] = \sum_{\omega \in \mathcal{S}} R(\omega) \Pr\{\omega\} \tag{14.2}$$

The proof of Theorem 14.1.2, like many of the elementary proofs about expectation in these notes, follows by judicious regrouping of terms in the defining sum (14.1):

Proof.

$$\begin{aligned}
\mathrm{E}[R] ::= & \sum_{x \in \text{range}(R)} x \cdot \Pr\{R = x\} && \text{(Def 14.1.1 of expectation)} \\
= & \sum_{x \in \text{range}(R)} x \left(\sum_{\omega \in [R=x]} \Pr\{\omega\} \right) && \text{(def of } \Pr\{R = x\}\text{)} \\
= & \sum_{x \in \text{range}(R)} \sum_{\omega \in [R=x]} x \Pr\{\omega\} && \text{(distributing } x \text{ over the inner sum)} \\
= & \sum_{x \in \text{range}(R)} \sum_{\omega \in [R=x]} R(\omega) \Pr\{\omega\} && \text{(def of the event } [R = x]\text{)} \\
= & \sum_{\omega \in \mathcal{S}} R(\omega) \Pr\{\omega\}
\end{aligned}$$

The last equality follows because the events $[R = x]$ for $x \in \text{range}(R)$ partition the sample space, $\mathcal{S}$, so summing over the outcomes in $[R = x]$ for $x \in \text{range}(R)$ is the same as summing over $\mathcal{S}$. □

In general, the defining sum (14.1) 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 (14.2)is sometimes easier to use in proofs about expectation.

14.2 Expected Value of an Indicator Variable

The expected value of an indicator random variable for an event is just the probability of that event. (Remember that a random variable I_A is the indicator random variable for event A, if $I_A = 1$ when A occurs and $I_A = 0$ otherwise.)

Lemma 14.2.1. *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, $\mathrm{E}[I_A] = \Pr\{I_A = 1\} = p$.

14.3 Mean Time to Failure

A computer program crashes at the end of each hour of use with probability p, if it has not crashed already. What is the expected time until the program crashes?

If we let C be the number of hours until the crash, then the answer to our problem is $\mathrm{E}[C]$. Now the probability that, for $i > 0$, the first crash occurs in the ith hour is the probability that it does not crash in each of the first $i-1$ hours and it does crash in the ith hour, which is $(1-p)^{i-1}p$. So from formula (14.1) for expectation, we have

$$\begin{aligned}
\mathrm{E}[C] &= \sum_{i\in\mathbb{N}} i \cdot \Pr\{R = i\} \\
&= \sum_{i\in\mathbb{N}^+} i(1-p)^{i-1}p \\
&= p\sum_{i\in\mathbb{N}^+} i(1-p)^{i-1} \\
&= p\frac{1}{(1-(1-p))^2} && \text{(by (14.3))} \\
&= \frac{1}{p}
\end{aligned}$$

As an alternative to applying the formula

$$\sum_{i\in\mathbb{N}^+} ix^{i-1} = \frac{1}{(1-x)^2} \tag{14.3}$$

from Notes 11 (which you remembered, right?), there is a useful trick for calculating expectations of nonegative integer valued variables:

Lemma 14.3.1. *If R is a nonegative integer-valued random variable, then:*

$$\mathrm{E}[R] = \sum_{i\in\mathbb{N}} \Pr\{R > i\} \tag{14.4}$$

Proof. Consider the sum:

$$\begin{array}{ccccccc}
\Pr\{R=1\} & + & \Pr\{R=2\} & + & \Pr\{R=3\} & + & \cdots \\
 & + & \Pr\{R=2\} & + & \Pr\{R=3\} & + & \cdots \\
 & & & + & \Pr\{R=3\} & + & \cdots \\
 & & & & & + & \cdots
\end{array}$$

The successive columns sum to $1\cdot\Pr\{R=1\}$, $2\cdot\Pr\{R=2\}$, $3\cdot\Pr\{R=3\}$, ... Thus, the whole sum is equal to:

$$\sum_{i\in\mathbb{N}} i\cdot\Pr\{R=i\}$$

which equals $\mathrm{E}[R]$ by (14.1). On the other hand, the successive rows sum to $\Pr\{R>0\}$, $\Pr\{R>1\}$, $\Pr\{R>2\}$, ... Thus, the whole sum is also equal to:

$$\sum_{i\in\mathbb{N}} \Pr\{R>i\},$$

which therefore must equal $\mathrm{E}[R]$ as well. □

Now $\Pr\{C > i\}$ is easy to evaluate: a crash happens later than the ith hour iff the system did not crash during the first i hours, which happens with probability $(1-p)^i$. Plugging this into (14.4) gives:

$$\begin{aligned}
\mathrm{E}[C] &= \sum_{i\in\mathbb{N}} (1-p)^i \\
&= \frac{1}{1-(1-p)} && \text{(sum of geometric series)} \\
&= \frac{1}{p}
\end{aligned}$$

So, for example, if there is a 1% chance that the program crashes at the end of each hour, then the expected time until the program crashes is $1/0.01 = 100$ hours. The general principle here is well-worth remembering: if a system fails at each time step with probability p, then the expected number of steps up to the first failure is $1/p$.

As a further example, suppose a couple really wants to have a baby girl. For simplicity assume there is a 50% chance that each child they have is a girl, and the genders of their children are mutually independent. If the couple insists on having children until they get a girl, then how many baby boys should they expect first?

This is really a variant of the previous problem. The question, "How many hours until the program crashes?" is mathematically the same as the question, "How many children must the couple have until they get a girl?" In this case, a crash corresponds to having a girl, so we should set $p = 1/2$. By the preceding analysis, the couple should expect a baby girl after having $1/p = 2$ children. Since the last of these will be the girl, they should expect just one boy.

Something to think about: If every couple follows the strategy of having children until they get a girl, what will eventually happen to the fraction of girls born in this world?

14.4 Linearity of Expectation

Expected values obey a simple, very helpful rule called *Linearity of Expectation*. Its simplest form says that the expected value of a sum of random variables is the sum of the expected values of the variables.

Theorem 14.4.1. *For any random variables R_1 and R_2,*

$$\mathrm{E}[R_1 + R_2] = \mathrm{E}[R_1] + \mathrm{E}[R_2].$$

Proof. Let $T ::= R_1 + R_2$. The proof follows straightforwardly by rearranging terms in the sum (14.2)

$$\begin{aligned}
\mathrm{E}[T] &= \sum_{\omega\in\mathcal{S}} T(\omega)\cdot \Pr\{\omega\} && \text{(Theorem 14.1.2)} \\
&= \sum_{\omega\in\mathcal{S}} (R_1(\omega) + R_2(\omega))\cdot \Pr\{\omega\} && \text{(def of } T\text{)} \\
&= \sum_{\omega\in\mathcal{S}} R_1(\omega)\Pr\{\omega\} + \sum_{\omega\in\mathcal{S}} R_2(\omega)\Pr\{\omega\} && \text{(rearranging terms)} \\
&= \mathrm{E}[R_1] + \mathrm{E}[R_2]. && \text{(Theorem 14.1.2)}
\end{aligned}$$

□

A small extension of this proof, which we leave to the reader, implies

Theorem 14.4.2 (Linearity of Expectation)**.** *For random variables R_1, R_2 and constants $a_1, a_2 \in \mathbb{R}$,*

$$\mathrm{E}\left[a_1 R_1 + a_2 R_2\right] = a_1 \mathrm{E}\left[R_1\right] + a_2 \mathrm{E}\left[R_2\right].$$

In other words, expectation is a linear function. A routine induction extends the result to more than two variables:

Corollary 14.4.3. *For any random variables $R_1, \ldots, R_k$ and constants $a_1, \ldots, a_k \in \mathbb{R}$,*

$$\mathrm{E}\left[\sum_{i=1}^{k} a_i R_i\right] = \sum_{i=1}^{k} a_i \mathrm{E}\left[R_i\right].$$

The great thing about linearity of expectation is that *no independence is required*. This is really useful, because dealing with independence is a pain, and we often need to work with random variables that are not independent.

14.4.1 Expected Value of Two Dice

What is the expected value of the sum of two fair dice?

Let the random variable R_1 be the number on the first die, and let R_2 be the number on the second die. We observed earlier that the expected value of one die is 3.5. We can find the expected value of the sum using linearity of expectation:

$$\mathrm{E}\left[R_1 + R_2\right] = \mathrm{E}\left[R_1\right] + \mathrm{E}\left[R_2\right] = 3.5 + 3.5 = 7.$$

Notice that we did *not* have to assume that the two dice were independent. The expected sum of two dice is 7, even if they are glued together (provided the dice remain fair after the gluing). Proving that this expected sum is 7 with a tree diagram would be a bother: there are 36 cases. And if we did not assume that the dice were independent, the job would be really tough!

14.4.2 The Hat-Check Problem

There is a dinner party where n men check their hats. The hats are mixed up during dinner, so that afterward each man receives a random hat. In particular, each man gets his own hat with probability $1/n$. What is the expected number of men who get their own hat?

Letting G be the number of men that get their own hat, we want to find the expectation of G. But all we know about G is that the probability that a man gets his own hat back is $1/n$. There are many different probability distributions of hat permutations with this property, so we don't know enough about the distribution of G to calculate its expectation directly. But linearity of expectation makes the problem really easy.

The trick is to express G as a sum of indicator variables. In particular, let G_i be an indicator for the event that the ith man gets his own hat. That is, $G_i = 1$ if he gets his own hat, and $G_i = 0$ otherwise. The number of men that get their own hat is the sum of these indicators:

$$G = G_1 + G_2 + \cdots + G_n. \tag{14.5}$$

These indicator variables are *not* mutually independent. For example, if $n - 1$ men all get their own hats, then the last man is certain to receive his own hat. But, since we plan to use linearity of expectation, we don't have worry about independence!

Now since G_i is an indicator, we know $1/n = \Pr\{G_i = 1\} = \mathrm{E}[G_i]$ by Lemma 14.2.1. Now we can take the expected value of both sides of equation (14.5) and apply linearity of expectation:

$$\begin{aligned}\mathrm{E}[G] &= \mathrm{E}[G_1 + G_2 + \cdots + G_n] \\ &= \mathrm{E}[G_1] + \mathrm{E}[G_2] + \cdots + \mathrm{E}[G_n] \\ &= \frac{1}{n} + \frac{1}{n} + \cdots + \frac{1}{n} = n\left(\frac{1}{n}\right) = 1.\end{aligned}$$

So even though we don't know much about how hats are scrambled, we've figured out that on average, just one man gets his own hat back!

14.4.3 Expectation of a Binomial Distribution

Suppose that we independently flip n biased coins, each with probability p of coming up heads. What is the expected number that come up heads?

Let J be the number of heads after the flips, so J has the (n, p)-binomial distribution. Now let I_k be the indicator for the kth coin coming up heads. By Lemma 14.2.1, we have

$$\mathrm{E}[I_k] = p.$$

But

$$J = \sum_{k=1}^{n} I_k,$$

so by linearity

$$\mathrm{E}[J] = \mathrm{E}\left[\sum_{k=1}^{n} I_k\right] = \sum_{k=1}^{n} \mathrm{E}[I_k] = \sum_{k=1}^{n} p = pn.$$

In short, the expectation of an (n, p)-binomially distributed variable is pn.

14.4.4 The Coupon Collector Problem

Every time I purchase a kid's meal at Taco Bell, I am graciously presented with a miniature "Racin' Rocket" car together with a launching device which enables me to project my new vehicle across any tabletop or smooth floor at high velocity. Truly, my delight knows no bounds.

There are n different types of Racin' Rocket car (blue, green, red, gray, etc.). The type of car awarded to me each day by the kind woman at the Taco Bell register appears to be selected uniformly and independently at random. What is the expected number of kid's meals that I must purchase in order to acquire at least one of each type of Racin' Rocket car?

The same mathematical question shows up in many guises: for example, what is the expected number of people you must poll in order to find at least one person with each possible birthday? Here, instead of collecting Racin' Rocket cars, you're collecting birthdays. The general question is commonly called the ***coupon collector problem*** after yet another interpretation.

A clever application of linearity of expectation leads to a simple solution to the coupon collector problem. Suppose there are five different types of Racin' Rocket, and I receive this sequence:

$$\text{blue} \quad \text{green} \quad \text{green} \quad \text{red} \quad \text{blue} \quad \text{orange} \quad \text{blue} \quad \text{orange} \quad \text{gray}$$

Let's partition the sequence into 5 segments:

$$\underbrace{\text{blue}}_{X_0} \quad \underbrace{\text{green}}_{X_1} \quad \underbrace{\text{green} \quad \text{red}}_{X_2} \quad \underbrace{\text{blue} \quad \text{orange}}_{X_3} \quad \underbrace{\text{blue} \quad \text{orange} \quad \text{gray}}_{X_4}$$

The rule is that a segment ends whenever I get a new kind of car. For example, the middle segment ends when I get a red car for the first time. In this way, we can break the problem of collecting every type of car into stages. Then we can analyze each stage individually and assemble the results using linearity of expectation.

Let's return to the general case where I'm collecting n Racin' Rockets. Let X_k be the length of the kth segment. The total number of kid's meals I must purchase to get all n Racin' Rockets is the sum of the lengths of all these segments:

$$T = X_0 + X_1 + \cdots + X_{n-1}$$

Now let's focus our attention on X_k, the length of the kth segment. At the beginning of segment k, I have k different types of car, and the segment ends when I acquire a new type. When I own k types, each kid's meal contains a type that I already have with probability k/n. Therefore, each meal contains a new type of car with probability $1 - k/n = (n-k)/n$. Thus, the expected number of meals until I get a new kind of car is $n/(n-k)$ by the "mean time to failure" formula. So we have:

$$\mathrm{E}[X_k] = \frac{n}{n-k}$$

Linearity of expectation, together with this observation, solves the coupon collector problem:

$$\begin{aligned}
\mathrm{E}[T] &= \mathrm{E}[X_0 + X_1 + \cdots + X_{n-1}] \\
&= \mathrm{E}[X_0] + \mathrm{E}[X_1] + \cdots + \mathrm{E}[X_{n-1}] \\
&= \frac{n}{n-0} + \frac{n}{n-1} + \cdots + \frac{n}{3} + \frac{n}{2} + \frac{n}{1} \\
&= n\left(\frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{3} + \frac{1}{2} + \frac{1}{1}\right) \\
&\quad n\left(\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n-1} + \frac{1}{n}\right) \\
&= nH_n \sim n \ln n.
\end{aligned}$$

Let's use this general solution to answer some concrete questions. For example, the expected number of die rolls required to see every number from 1 to 6 is:

$$6H_6 = 14.7\ldots$$

And the expected number of people you must poll to find at least one person with each possible birthday is:

$$365H_{365} = 2364.6\ldots$$

14.5 The Expected Value of a Product

While the expectation of a sum is the sum of the expectations, the same is usually not true for products. But it is true in an important special case, namely, when the random variables are *independent*.

For example, suppose we throw two *independent*, fair dice and multiply the numbers that come up. What is the expected value of this product?

Let random variables R_1 and R_2 be the numbers shown on the two dice. We can compute the expected value of the product as follows:

$$\mathrm{E}\left[R_1 \cdot R_2\right] = \mathrm{E}\left[R_1\right] \cdot \mathrm{E}\left[R_2\right] = 3.5 \cdot 3.5 = 12.25. \tag{14.6}$$

Here the first equality holds because the dice are independent.

At the other extreme, suppose the second die is always the same as the first. Now $R_1 = R_2$, and we can compute the expectation, $\mathrm{E}\left[R_1^2\right]$, of the product of the dice explicitly, confirming that it is not equal to the product of the expectations.

$$\begin{aligned}
\mathrm{E}\left[R_1 \cdot R_2\right] &= \mathrm{E}\left[R_1^2\right] \\
&= \sum_{i=1}^{6} i^2 \cdot \Pr\left\{R_1^2 = i^2\right\} \\
&= \sum_{i=1}^{6} i^2 \cdot \Pr\left\{R_1 = i\right\} \\
&= \frac{1^2}{6} + \frac{2^2}{6} + \frac{3^2}{6} + \frac{4^2}{6} + \frac{5^2}{6} + \frac{6^2}{6} \\
&= 15\ 1/6 \\
&\neq 12\ 1/4 \\
&= \mathrm{E}\left[R_1\right] \cdot \mathrm{E}\left[R_2\right].
\end{aligned}$$

Theorem 14.5.1. *For any two* independent *random variables R_1, R_2,*

$$\mathrm{E}\left[R_1 \cdot R_2\right] = \mathrm{E}\left[R_1\right] \cdot \mathrm{E}\left[R_2\right].$$

Proof. The event $[R_1 \cdot R_2 = r]$ can be split up into events of the form $[R_1 = r_1 \text{ and } R_2 = r_2]$ where

$r_1 \cdot r_2 = r$. So

$$\begin{aligned}
&\mathrm{E}[R_1 \cdot R_2] \\
&::= \sum_{r \in \text{range}(R_1 \cdot R_2)} r \cdot \Pr\{R_1 \cdot R_2 = r\} \\
&= \sum_{r_i \in \text{range}(R_i)} r_1 r_2 \cdot \Pr\{R_1 = r_1 \text{ and } R_2 = r_2\} \\
&= \sum_{r_1 \in \text{range}(R_1)} \sum_{r_2 \in \text{range}(R_2)} r_1 r_2 \cdot \Pr\{R_1 = r_1 \text{ and } R_2 = r_2\} && \text{(ordering terms in the sum)} \\
&= \sum_{r_1 \in \text{range}(R_1)} \sum_{r_2 \in \text{range}(R_2)} r_1 r_2 \cdot \Pr\{R_1 = r_1\} \cdot \Pr\{R_2 = r_2\} && \text{(indep. of } R_1, R_2) \\
&= \sum_{r_1 \in \text{range}(R_1)} \left(r_1 \Pr\{R_1 = r_1\} \cdot \sum_{r_2 \in \text{range}(R_2)} r_2 \Pr\{R_2 = r_2\} \right) && \text{(factoring out } r_1 \Pr\{R_1 = r_1\}) \\
&= \sum_{r_1 \in \text{range}(R_1)} r_1 \Pr\{R_1 = r_1\} \cdot \mathrm{E}[R_2] && \text{(def of } \mathrm{E}[R_2]) \\
&= \mathrm{E}[R_2] \cdot \sum_{r_1 \in \text{range}(R_1)} r_1 \Pr\{R_1 = r_1\} && \text{(factoring out } \mathrm{E}[R_2]) \\
&= \mathrm{E}[R_2] \cdot \mathrm{E}[R_1]. && \text{(def of } \mathrm{E}[R_1])
\end{aligned}$$

□

Theorem 14.5.1 extends routinely to a collection of mutually independent variables.

Corollary 14.5.2. *If random variables $R_1, R_2, \ldots, R_k$ are mutually independent, then*

$$\mathrm{E}\left[\prod_{i=1}^{k} R_i\right] = \prod_{i=1}^{k} \mathrm{E}[R_i].$$

14.6 Conditional Expectation

Just like event probabilities, expectations can be conditioned on some event.

Definition 14.6.1. The *conditional expectation*, $\mathrm{E}[R \mid A]$, of a random variable, R, given event, A, is:

$$\mathrm{E}[R \mid A] ::= \sum_{r \in \text{range}(R)} r \cdot \Pr\{R = r \mid A\}. \tag{14.7}$$

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 (14.7),

$$\mathrm{E}[R \mid R \geq 4] = \sum_{i=1}^{6} i \cdot \Pr\{R = i \mid R \geq 4\} = 1 \cdot 0 + 2 \cdot 0 + 3 \cdot 0 + 4 \cdot \tfrac{1}{3} + 5 \cdot \tfrac{1}{3} + 6 \cdot \tfrac{1}{3} = 5.$$

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 14.6.2 (Law of Total Expectation)**.** *Let $A_1, A_2, \ldots$ be a partition of the sample space. Then*

$$\mathrm{E}[R] = \sum_i \mathrm{E}[R \mid A_i]\Pr\{A_i\}.$$

Proof.

$$\begin{aligned}\mathrm{E}[R] ::= &\sum_{r\in \text{range}(R)} r\cdot \Pr\{R = r\} && \text{(Def 14.1.1 of expectation)}\\ = &\sum_r r\cdot \sum_i \Pr\{R = r \mid A_i\}\Pr\{A_i\} && \text{(Law of Total Probability)}\\ = &\sum_r \sum_i r\cdot \Pr\{R = r \mid A_i\}\Pr\{A_i\} && \text{(distribute constant } r\text{)}\\ = &\sum_i \sum_r r\cdot \Pr\{R = r \mid A_i\}\Pr\{A_i\} && \text{(exchange order of summation)}\\ = &\sum_i \Pr\{A_i\}\sum_r r\cdot \Pr\{R = r \mid A_i\} && \text{(factor constant } \Pr\{A_i\}\text{)}\\ = &\sum_i \Pr\{A_i\}\,\mathrm{E}[R \mid A_i]. && \text{(Def 14.6.1 of cond. expectation)}\end{aligned}$$

□

14.6.1 Properties of Conditional Expectation

Many rules for conditional expectation correspond directly to rules for ordinary expectation.

For example, linearity of conditional expectation carries over with the same proof:

Theorem 14.6.3. *For any two random variables R_1, R_2, constants $a_1, a_2 \in \mathbb{R}$, and event A,*

$$\mathrm{E}[a_1R_1 + a_2R_2 \mid A] = a_1\,\mathrm{E}[R_1 \mid A] + a_2\,\mathrm{E}[R_2 \mid A].$$

Likewise,

Theorem 14.6.4. *For any two* independent *random variables R_1, R_2, and event, A,*

$$\mathrm{E}\left[R_1 \cdot R_2 \mid A\right] = \mathrm{E}\left[R_1 \mid A\right] \cdot \mathrm{E}\left[R_2 \mid A\right].$$

14.7 Why the Mean?

In these final Notes, we've taken it for granted that expectation is important, and we've developed a bunch of techniques for calculating expectations, but we haven't really explained why. One important reason is that the technique of estimation by sampling, which we illustrated just for binomial sampling, has far-reaching generalizations beyond the fraction-of-voters by polling we examined in previous Notes.

For example, suppose instead of the fraction of voters favoring one candidate, we wanted to estimate the average age, income, family size, or other measure of a population. Then the same idea works: suppose we determine a random process for selecting people. Then the selected person's age, income, and so on are random variables whose expected values are the actual average age or income of the population. Now if we actually select a random sample of people, we can use the average of people in the sample as an estimate for 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. We don't have time left to explore such results any further, but we hope you've gotten the basic idea from the results we have presented.