Mathematics for Computer Science

revised June 17, 2010, 1329 minutes

Eric Lehman

Google Inc.

F Tom Leighton

Massachusets Institute of Technology

Albert R Meyer

Massachusets Institute of Technology

Contents

Part I

Proofs

Mathematical Proofs

This text is all about methods for constructing and understanding proofs. In fact, we could have titled the book *Proofs, Proofs, and More Proofs*. We will begin in Part I with a description of basic proof techniques. We then apply these techniques in chapter 4 to establish some very important facts about numbers, facts that form the underpinning of the world's most widely used cryptosystem.

Simply put, a proof is a method of establishing truth. Like beauty, "truth" sometimes depends on the eye of the beholder, however, and it should not be surprising that what constitutes a proof differs among fields. For example, in the judicial system, *legal* truth is decided by a jury based on the allowable evidence presented at trial. In the business world, *authoritative* truth is specified by a trusted person or organization, or maybe just your boss. In fields such as physics and biology, *scientific* truth[1] is confirmed by experiment. In statistics, *probable* truth is

[1]Actually, only scientific *falsehood* can really 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

established by statistical analysis of sample data.

Philosophical proof involves careful exposition and persuasion typically based on a series of small, plausible arguments. The best example begins with "Cogito ergo sum," a Latin sentence that translates as "I think, therefore I am." It comes from the beginning of a 17th century essay by the mathematician/philosopher, 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 idea. However, with just a few more lines of argument in this vein, Descartes goes on to conclude that there is an infinitely beneficent God. Whether or not you believe in a beneficent God, you'll probably agree that any very short proof of God's existence is bound to be far-fetched. So even in masterful hands, this approach is not reliable.

Mathematics has its own specific notion of "proof."

Definition. A *formal proof* of a *proposition* is a chain of *logical deductions* leading to

past, and anticipated future, experiments.

the proposition from a base set of *axioms*.

The three key ideas in this definition are highlighted: proposition, logical deduction, and axiom. These three ideas are explained in the following chapters, beginning with propositions in chapter 1. We will then provide *lots* of examples of proofs and even some examples of "false proofs" (*i.e.*, arguments that look like a proof but that contain mis-steps, or deductions that aren't so logical when examined closely).

Chapter 1

Propositions

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

For example, both of the following statements are propositions. The first is true and the second is false.

Proposition 1.0.1. *$2 + 3 = 5$.*

Proposition 1.0.2. *$1 + 1 = 3$.*

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

Unfortunately it is not always easy to decide if a proposition is true or false, or even what the proposition means. In part, this is because the English language is riddled with ambiguities. For example, here are some statements that illustrate the issue:

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

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

Americans have the same dream or might some of them have different dreams?

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

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

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

1.1 Compound 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.

1.1.1 NOT, AND, OR

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

P	NOT(P)
T	**F**
F	**T**

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

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

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

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

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

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

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

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

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

1.1.2 IMPLIES

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

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

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

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

The Riemann Hypothesis is a famous unresolved conjecture in mathematics (*i.e.*, no one knows if it is true or false). But that doesn't prevent you from answering the question! This proposition has the form $P \longrightarrow Q$ where the *hypothesis*, P, is "the Riemann Hypothesis is true" and the *conclusion*, Q, is "$x^2 \geq 0$ for every real number x". Since the conclusion is definitely true, we're on either line (tt) or line (ft) of the truth table. Either way, the proposition as a while is *true*!

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

"If pigs can 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 cannot 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!

1.1.3 IFF

Mathematicians commonly join propositions in one additional way that doesn't arise in ordinary speech. The proposition "P if and only if Q" asserts that P and Q

are logically equivalent; that is, either both are true or both are false.

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

For example, the following if-and-only-if statement is true for every real number x:

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

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

1.1.4 Notation

Mathematicians have devised symbols to represent words like "AND" and "NOT". The most commonly-used symbols are summarized in the table below.

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

For example, using this notation, "If P AND NOT(Q), then R" would be written:

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

This symbolic language is helpful for writing complicated logical expressions compactly. But words such as "OR" and "IMPLIES" generally serve just as well as the cryptic symbols $\vee$ and $\longrightarrow$, and their meaning is easy to remember. We will use them interchangeably and you can feel free to use whichever convention is easiest for you.

1.1.5 Logically Equivalent Implications

Do these two sentences say the same thing?

If I am hungry, then I am grumpy.

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

We can settle the issue by recasting both sentences in terms of propositional logic.[1] 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

[1]This sounds scary, but don't worry, propositional logic is easy. **[Illegible]** compound propositions.

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

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

If I am grumpy, then I am hungry.

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

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

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

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

If I am grumpy, THEN I am hungry, AND

if I am hungry, THEN I am grumpy.

are equivalent to the single statement:

I am grumpy IFF I am hungry.

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

P	Q	$(P$ IMPLIES $Q)$	$(Q$ IMPLIES $P)$	$(P$ IMPLIES $Q)$ AND $(Q$ IMPLIES $P)$	Q IFF P
T	**T**	**T**	**T**	**T**	**T**
T	**F**	**F**	**T**	**F**	**F**
F	**T**	**T**	**F**	**F**	**F**
F	**F**	**T**	**T**	**T**	**T**

1.1.6 Problems

Class Problems

Homework Problems

1.2 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 "A OR (NOT(A) AND B)". A truth table reveals that this complicated expression is logically equivalent to "A OR B".

A	B	A OR (NOT(A) AND B)	A OR B
T	**T**	**T**	**T**
T	**F**	**T**	**T**
F	**T**	**T**	**T**
F	**F**	**F**	**F**

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

```
if ( x > 0 || y > 100 )
   ⋮
```

(further instructions)

Rewriting a logical expression involving many variables in the simplest form is both difficult and important. Simplifying expressions in software can increase the speed of your program. Chip designers face a similar challenge—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.

1.2.1 Problems

Class Problems

Homework Problems

1.2.2 Problems

Class Problems

1.3 Predicates and Quantifiers

1.3.1 Propositions with infinitely many cases

Most of the examples of propositions that we have considered thus far have been nice in the sense that it has been relatively easy to determine if they are true or false. At worse, there were only a few cases to check in a truth table. Unfortunately, not all propositions are so easy to check. That is because some propositions may involve a large or infinite number of possible cases. For example, consider the following proposition involving prime numbers. (A *prime* is an integer greater than 1 that is divisible only by itself and 1. For example, 2, 3, 5, 7, and 11 are primes, but 4, 6, and 9 are not. A number greater than 1 that is not prime is said to be *composite*.)

Proposition 1.3.1. *For every nonnegative integer, n, the value of $n^2 + n + 41$ is prime.*

It is not immediately clear whether this proposition is true or false. In such circumstances, it is tempting to try to determine its veracity by computing the value of[2]

$$p(n) ::= n^2 + n + 41. \tag{1.1}$$

for several values of$n$ and then checking to see if they are prime. If any of the computed values is not prime, then we will know that the proposition is false. If all the computed values are indeed prime, then we might be tempted to conclude that the proposition is true.

We begin with $p(0) = 41$ which is prime. $p(1) = 43$ is also prime. So is $p(2) = 47$, $p(3) = 53$,..., and $p(20) = 461$. Hmmm... It is starting to look like $p(n)$ is

[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.

a prime for every nonnegative integer n. In fact we can keep checking through $n = 39$ and confirm that $p(39) = 1601$ is prime. The proposition certainly does seem to be true.

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, and thus the proposition is false!

EDITING NOTE: In fact, it's not hard to show that *no* polynomial with integer coefficients can map all natural numbers into prime numbers, unless it's a constant.

■

Although surprising, this example is not as contrived or rare as you might suspect. As we will soon see, there are many examples of propositions that seem to be true when you check a few cases, but which turn out to be false. The key to remember is that 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.

Propositions that involve *all* numbers are so common that there is a special notation for them. For example, Proposition 1.3.1 can also be written as

$$\forall n \in \mathbb{N}.\ p(n) \text{ is prime.} \tag{1.2}$$

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

Here is another example of a proposition that, at first, seems to be true but which turns out to be false.

Proposition 1.3.2. *$a^4 + b^4 + c^4 = d^4$ has no solution when a, b, c, d are positive integers.*

Euler (pronounced "oiler") conjectured proposition to be true this in 1769. Ultimately the proposition was proven false in 1987 by Noam Elkies. The solution he found was $a = 95800, b = 217519, c = 414560, d = 422481$. No wonder it took 218 years to show the proposition is false!

In logical notation, Proposition 1.3.2 could be written,

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

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

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

The following proposition is even nastier.

Proposition 1.3.3. $313(x^3 + y^3) = z^3$ *has no solution when* $x, y, z \in \mathbb{Z}^+$.

This proposition is also false, but the smallest counterexample values for x, y, and z have more than 1000 digits! Even the world's largest computers would not be able to get that far with brute force. Of course, you may be wondering why anyone would care whether or not there is a solution to $313(x^3 + y^3) = z^3$ where x, y, and z are positive integers. It turns out that finding solutions to such equations is important in the field of elliptic curves, which turns out to be important to the study of factoring large integers, which turns out (as we will see in Chapter 4) to be important in cracking commonly-used cryptosystems, which is why mathematicians went to the effort to find the solution with thousands of digits.

[Illegible] that have infinitely many cases to check turn out to be false. The following proposition (known as the "Four-Color Theorem") turns out to be true.

Proposition 1.3.4. *Every map can be colored with 4 colors so that adjacent*[3] *regions have different colors.*

The proof of this proposition is difficult and took over a century to perfect. Alon the way, many incorrect proofs were proposed, including one that stood for 10 years in the late 19th century before the mistake was found. An extremely laborious proof was finally found in 1976 by mathematicians Appel and Haken, who used a complex computer program to categorize the four-colorable maps; the program left a few thousand maps uncategorized, and these were checked by hand by Haken and his assistants—including his 15-year-old daughter. There was a lot of debate about whether this was a legitimate proof: the proof was too big to be checked without a computer, and no one could guarantee that the computer calculated correctly, nor did anyone have the energy to recheck the four-colorings of the thousands of maps that were done by hand. Within the past decade, a mostly

[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.

intelligible proof of the Four-Color Theorem was found, though a computer is still needed to check the colorability of several hundred special maps.[4]

In some cases, we do not know whether or not a proposition is true. For example, the following simple proposition (known as Goldbach's Conjecture) has been heavily studied since 1742 but we still do not know if it is true. Of course, it has been checked by computer for many values of n, but as we have seen, that is not sufficient to conclude that it is true.

Proposition 1.3.5 (Goldbach)**.** *Every even integer greater than 2 is the sum of two primes.*

While the preceding propositions are important in mathematics, computer scientists are often interested in propositions concerning the "correctness" of programs and systems, to determine whether a program or system does what it's

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

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

supposed to. Programs are notoriously buggy, and there's a growing community of researchers and practitioners trying to find ways to prove program correctness. These efforts have been successful enough in the case of CPU chips that they are now routinely used by leading chip manufacturers to prove chip correctness and avoid mistakes like the notorious Intel division bug in the 1990's.

EDITING NOTE: ref needed ■

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

1.3.2 Predicates

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

"n is a perfect square"

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

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

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

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

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

1.3.3 Quantifiers

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

$$\text{“}x^2 \geq 0\text{”}$$

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

$$\text{“}5x^2 - 7 = 0\text{”}$$

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

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

Always True

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

Sometimes True

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

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

Sometimes English sentences are unclear about quantification:

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

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

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

or maybe

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

In the preceding example, the 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.

1.3.4 Notation

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

$$\forall x \in D.\ P(x) \tag{1.3}$$

The *universal quantifier* symbol $\forall$ is read "for all," so this whole expression (1.3) is read "For all x in D, $P(x)$ is true." Remember that upside-down "A" stands for "**A**ll."

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

$$\exists x \in D.\ P(x) \tag{1.4}$$

The *existential quantifier* symbol $\exists$, is read "there exists." So expression (1.4) is read, "There exists an x in D such that $P(x)$ is true." Remember that backward "E" stands for "**E**xists."

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

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

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

can be written as follows:

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

or maybe

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

1.3.5 Mixing Quantifiers

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

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

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

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

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

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

The proposition can also be written more simply as

$$\forall n \in \text{Evens}\; \exists p, q \in \text{Primes}.\; p + q = n.$$

1.3.6 Order of Quantifiers

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

"Every American has a dream."

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

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

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

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

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

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

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

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

1.3.7 Variables Over One Domain

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

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

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

1.3.8 Negating Quantifiers

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

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

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

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

Similarly, these sentences mean the same thing:

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

We can express the equivalence in logic notation this way:

$$\text{NOT}\,(\exists x.\, P(x))\ \text{ IFF }\ \forall x.\ \text{NOT}(P(x)). \tag{1.5}$$

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

1.4 Validity

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

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

is valid.

The same idea extends to predicate formulas, but to be valid, a formula now must evaluate to true no matter what values its variables may take over any unspecified domain, and no matter what interpretation a predicate variable may be given. For example, we already observed that the rule for negating a quantifier is captured by the valid assertion (1.5).

Another useful example of a valid assertion is

$$\exists x \forall y.\ P(x, y) \text{ IMPLIES } \forall y \exists x.\ P(x, y). \tag{1.6}$$

Here's an explanation why this is valid:

Let D be the domain for the variables and P_0 be some binary predicate[5] on D. We need to show that if

$$\exists x \in D\ \forall y \in D.\ P_0(x, y) \tag{1.7}$$

holds under this interpretation, then so does

$$\forall y \in D\ \exists x \in D.\ P_0(x, y). \tag{1.8}$$

So suppose (1.7) is true. Then by definition of $\exists$, this means that some element $d_0 \in D$ has the property that

$$\forall y \in D. P_0(d_0, y).$$

By definition of $\forall$, this means that

$$P_0(d_0, d)$$

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

is true for all $d \in D$. So given any $d \in D$, there is an element in D, namely, d_0, such that $P_0(d_0, d)$ is true. But that's exactly what (1.8) means, so we've proved that (1.8) holds under this interpretation, as required.

We hope this is helpful as an explanation, although purists would not really want to call it a "proof." The problem is that with something as basic as (1.6), it's hard to see what more elementary axioms are ok to use in proving it. What the explanation above did was translate the logical formula (1.6) into English and then appeal to the meaning, in English, of "for all" and "there exists" as justification.

In contrast to (1.6), the formula

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

is *not* valid. We can prove this 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, for example, choose the value of x to be $n+1$. But under this interpretation the conclusion asserts that there is an integer that is bigger than all integers, which is certainly false. An interpretation like this which falsifies an assertion is called a *counter model* to the assertion.

1.5 Satisfiability

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

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

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

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

Recently there has been exciting progress on *sat-solvers* for practical applications like digital circuit verification. These programs find satisfying assignments with amazing efficiency even for formulas with millions of variables. Unfortunately, it's hard to predict which kind of formulas are amenable to sat-solver methods, and for formulas that are NOT satisfiable, sat-solvers generally take exponential time to verify that.

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

1.6 Problems

1.6.1 Problems

Class Problems

Homework Problems

Chapter 2

Patterns of Proof

2.1 The Axiomatic Method

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

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

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

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

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

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

Our Axioms

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

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

Logical Deductions

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

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

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

Rule.

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

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

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

You can see why modus ponens is a sound inference rule by checking the truth table of P IMPLIES Q. There is only one case where P and P IMPLIES Q are both true, and in that case Q is also true.

P	Q	$P \longrightarrow Q$
F	**F**	**T**
F	**T**	**T**
T	**F**	**F**
T	**T**	**T**

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

Rule.

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

EDITING NOTE:

Rule.

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

■

Rule.

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

On the other hand,

Rule.

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

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

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

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

2.2 Proof Templates

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

Here's the good news: many proofs follow one of a handful of standard templates. Each proof has it own details, of course, but these templates at least provide you with an outline to fill in. In the remainder of this chapter, we'll 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 in Chapter 3.

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

2.2.1 Proof by Cases

Breaking a complicated proof into cases and proving each case separately is a useful and common proof strategy. In fact, we have already implicitly used this strategy when we used truth tables to show that certain propositions were true or valid. For example, in section 1.1.5, we showed that an implication $P \longrightarrow Q$ is equivalent to its contrapositive $\neg Q \longrightarrow P$ by considering all 4 possible assignments of **T** or **F** to P and Q. In each of the four cases, we showed that $P \longrightarrow Q$ was true if and only if $\neg Q \longrightarrow P$ was true. (For example, if $P =$ **T** and $Q =$ **F**, then both $P \longrightarrow Q$ and $\neg Q \longrightarrow P$ are false, thereby establishing that $(P \longrightarrow Q) \longleftrightarrow (\neg Q \longrightarrow P)$ is true in for this case.) Hence we could conclude that $P \longrightarrow Q$ was true if and only if $\neg Q \longrightarrow P$ are equivalent.

Proof by cases works in much more general environments than propositions involving Boolean variables. In what follows, we will use this approach to prove a simple fact about acquaintances. As background, we will assume that for any pair of 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 other 5 people besides x, at least 3 have met x.

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

Now we have to be sure that at least one of these two cases must hold,[2] but that's easy: we've split the 5 people into two groups, those who have shaken hands

[1]Describing your approach at the outset helps orient the reader. Try to remember to always do this.

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

with x and those who have not, so one of the groups must have at least half the people.

Case 1: Suppose that at least 3 people have met x.

This case splits into two subcases:

Case 1.1: Among the people who have met x, none have met each other. Then the people who have met x are a group of at least 3 strangers. So the Theorem holds in this subcase.

Case 1.2: Among the people who have met x, some pair 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 have not met x.

This case also splits into two subcases:

Case 2.1: Among the people who have not met x, every pair has met each other. Then the people who have not met x are a club of at least 3 people. So the Theorem holds in this subcase.

Case 2.2: Among the people who have not met x, some pair 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.

■

2.2.2 Proving an Implication

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

Here are some examples of implications:

- (Quadratic Formula) If $ax^2 + bx + c = 0$ and $a \neq 0$, then

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

- (Goldbach's Conjecture) If n is an even integer greater than 2, then n is a sum of two primes.

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

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

Method #1: Assume P is true

This method is really an example of proof by cases in disguise. In particular, when proving P IMPLIES Q, there are two cases to consider: P is true and P is false. The case when P is false is easy since, by definition, **T** IMPLIES Q is true no matter what Q is. This case is so easy that we usually just forget about it and start right off by assuming that P is true when proving an implication, since this is the only case that is interesting. Hence, in order to prove that P IMPLIES Q:

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

For example, we will use this method to prove

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

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

The inequality certainly holds for $x = 0$; then the left side is equal to 1 and $1 > 0$. As x grows, the $4x$ term (which is positive) initially seems to have greater magnitude than $-x^3$ (which is negative). For example, when $x = 1$, we have $4x = 4$, but $-x^3 = -1$. In fact, it looks like $-x^3$ doesn't begin to dominate $4x$ 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 ■ or "q.e.d". The only purpose for these conventions is to clarify where proofs begin and end.

Pitfall

For the purpose of proving an implication P IMPLIES Q, it's OK, and typical, to begin by assuming P. But when the proof is over, it's no longer OK to assume that P holds! For example, Theorem 2.2.1 has the form "if P, then Q" with P being "$0 \leq x \leq 2$" and Q being "$-x^3 + 4x + 1 > 0$," and its proof began by assuming that $0 \leq x \leq 2$. But of course this assumption does not always hold. Indeed, if you were going to prove another result using the variable x, it could be disastrous to have a step where you assume that $0 \leq x \leq 2$ just because you assumed it as part of the proof of Theorem 2.2.1.

Method #2: Prove the Contrapositive

We have already seen that an implication "P IMPLIES Q" is logically equivalent to its *contrapositive*

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

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

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

2. Proceed as in Method #1.

For example, we can use this approach to prove

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

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

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

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

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

Squaring both sides gives:

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

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

2.2.3 Proving an "If and Only If"

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

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

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

Method #1: Prove Each Statement Implies the Other

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

1. Write, "We prove P implies Q and vice-versa."
2. Write, "First, we show P implies Q." Do this by one of the methods in Section 2.2.2.
3. Write, "Now, we show Q implies P." Again, do this by one of the methods in Section 2.2.2.

Method #2: Construct a Chain of IFFs

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

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

This method sometimes requires more ingenuity than the first, but the result can be a short, elegant proof, as we see in the following example.

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

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

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

where μ is the *mean* of the values:

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

As an example, Theorem 2.2.3 says that the standard deviation of test scores is zero if and only if everyone scored exactly the class average. (We will talk a lot more about means and standard deviations in Part IV of the book.)

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

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

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

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

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

Every term on the left hand side of (2.3) is zero. (2.4)

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

Every x_i equals the mean.

■

2.2.4 Proof by Contradiction

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

EDITING NOTE:

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.

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.”

As an example, we will use proof by contradiction to prove that $\sqrt{2}$ is irrational. Recall 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.

Theorem 2.2.4. *$\sqrt{2}$ is irrational.*

Proof. We use proof by contradiction. Suppose the claim is false; that is, $\sqrt{2}$ is rational. Then we can write $\sqrt{2}$ as a fraction n/d where n and d are positive integers. Furthermore, let's take n and d so that n/d is in *lowest terms*, namely, there is no number greater than 1 that divides both n and d).

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

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

EDITING NOTE:

Potential Pitfall

Often students use an indirect proof when a direct proof would be simpler. Such proofs aren't wrong; they just aren't excellent. Let's look at an example. A function f is *strictly increasing* if $f(x) > f(y)$ for all real x and y such that $x > y$.

Theorem 2.2.5. *If f and g are strictly increasing functions, then $f + g$ is a strictly increasing function.*

Let's first look at a simple, direct proof.

Proof. Let x and y be arbitrary real numbers such that $x > y$. Then:

$$f(x) > f(y) \qquad \text{(since } f \text{ is strictly increasing)}$$

$$g(x) > g(y) \qquad \text{(since } g \text{ is strictly increasing)}$$

Adding these inequalities gives:

$$f(x) + g(x) > f(y) + g(y)$$

Thus, $f + g$ is strictly increasing as well. ■

Now we *could* prove the same theorem by contradiction, but this makes the argument needlessly convoluted.

Proof. We use proof by contradiction. Suppose that $f + g$ is not strictly increasing. Then there must exist real numbers x and y such that $x > y$, but

$$f(x) + g(x) \leq f(y) + g(y)$$

This inequality can only hold if either $f(x) \leq f(y)$ or $g(x) \leq g(y)$. Either way, we have a contradiction because both f and g were defined to be strictly increasing. Therefore, $f + g$ must actually be strictly increasing. ■

■

A proof of a proposition P by contradiction is really the same as proving the implication **T** IMPLIES P by contrapositive. Indeed, the contrapositive of T IMPLIES P is NOT(P) IMPLIES **F**. As we saw in Section 2.2.2(**???**), such a proof would be begin by assuming NOT(P) in an effort to derive a falsehood, just as you do in a proof by contradiction.

Pitfall

No matter how you think about it, it is important to remember that when you start by assuming NOT(P), you will derive conclusions along the way that are not necessarily true. (Indeed, the whole point of the method is to derive a falsehood.) This means that you cannot rely on such intermediate results after the proof is completed, for example that n is even in the proof of Theorem 2.2.4). There was not much risk of that happening in the proof of Theorem 2.2.4, but when you are doing more complicated proofs that build up from several lemmas, some of which utilize a proof by contradiction, it will be important to keep track of which follow from a (false) assumption in a proof by contradiction.

2.3 *Good* Proofs in Practice

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

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

In practice, the notion of proof is a moving target. Proofs in a professional research journal are generally unintelligible to all but a few experts who know all the terminology and prior results used in the proof. Conversely, proofs in the first weeks of an introductory course like *Mathematics for Computer Science* 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 than what we consider to be a good proof in the first couple of weeks of this course. But even so, we can offer some general tips on writing good proofs:

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

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

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

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

Revise and simplify. Your readers will be grateful.

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

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

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

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

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

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

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

2.3.1 Problems

Class Problems

Homework Problems

Chapter 3

Induction

3.1 The Well Ordering Principle

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

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

3.1.1 Well Ordering Proofs

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

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

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

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

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

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

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

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

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

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

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

3.1.2 Template for Well Ordering Proofs

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

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

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

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

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

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

3.1.3 Summing the Integers

Let’s use this this template to prove

Theorem.

$$1+2+3+\cdots+n = n(n+1)/2 \tag{3.1}$$

for all nonnegative integers, n.

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

- If $n = 1$, then there is only one term in the summation, and so $1+2+3+\cdots+n$ is just the term 1. Don't be misled by the appearance of 2 and 3 and the suggestion that 1 and n are distinct terms!
- If $n \leq 0$, then there are no terms at all in the summation. By convention, the sum in this case is 0.

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

We could have eliminated the need for guessing by rewriting the left side of (3.1) with *summation notation*:

$$\sum_{i=1}^{n} i \qquad \text{or} \qquad \sum_{1 \le i \le n} i.$$

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

OK, back to the proof:

Proof. By contradiction. Assume that the theorem is *false*. Then, some nonnegative integers serve as *counterexamples* to it. Let's collect them in a set:

$$C ::= \left\{ n \in \mathbb{N} \mid 1 + 2 + 3 + \cdots + n \neq \frac{n(n+1)}{2} \right\}.$$

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

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

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

But then, adding c to both sides we get

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

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

3.1.4 Factoring into Primes

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

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

gives an easy proof that every integer greater than one can be expressed as *some* product of primes.

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

Proof. The proof is by Well Ordering.

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

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

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

3.1.5 Problems

Practice Problems

Class Problems

Homework Problems

3.2 Induction

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

1. The student at the beginning of the line gets a candy bar.

2. If a student gets a candy bar, then the following student in line also gets a

candy bar.

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

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

$$\vdots$$

Of course this sequence has a more concise mathematical description:

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

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

3.2.1 Ordinary Induction

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

The Principle of Induction.

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

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

then

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

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

Rule. ***Induction Rule***

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

This general induction rule works for the same intuitive reason that all the students get candy bars, and we hope the explanation using candy bars makes it clear why the soundness of the ordinary induction can be taken for granted. In fact, the rule is so obvious that it's hard to see what more basic principle could be used to justify it.[2] What's not so obvious is how much mileage we get by using it.

Using Ordinary Induction

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

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

$$1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2} \tag{3.2}$$

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

Suppose that we define predicate $P(n)$ to be the equation (3.2). 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

[2]But see section 3.2.1.

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 $0(0+1)/2 = 0$, which is true by definition. The second statement is more complicated. But remember the basic plan for proving the validity of any implication: *assume* the statement on the left and then *prove* the statement on the right. In this case, we assume $P(n)$ in order to prove $P(n+1)$, which is the equation

$$1 + 2 + 3 + \cdots + n + (n+1) = \frac{(n+1)(n+2)}{2}. \tag{3.3}$$

These two equations are quite similar; in fact, adding $(n+1)$ to both sides of equation (3.2) and simplifying the right side gives the equation (3.3):

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

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. Therefore, the induction principle says that the predicate $P(m)$ is true for all nonnegative integers, m, so the theorem is proved.

A Template for Induction Proofs

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

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

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

pothesis. Sometimes the induction hypothesis will involve several variables, in which case you should indicate which variable serves as n.

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

4. **Prove that $P(n)$ implies $P(n+1)$ for every nonnegative integer n.** This is called the *inductive step*. The basic plan is always the same: assume that $P(n)$ is true and then use this assumption to prove that $P(n+1)$ is true. These two statements should be fairly similar, but bridging the gap may require some ingenuity. Whatever argument you give must be valid for every nonnegative integer n, since the goal is to prove the implications $P(0) \to P(1)$, $P(1) \to P(2)$, $P(2) \to P(3)$, 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 it is so standard that it's usual not to mention it explicitly,

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

A Clean Writeup

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

Proof. We use induction. The induction hypothesis, $P(n)$, will be equation (3.2).

Base case: $P(0)$ is true, because both sides of equation (3.2) equal zero when $n = 0$.

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

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

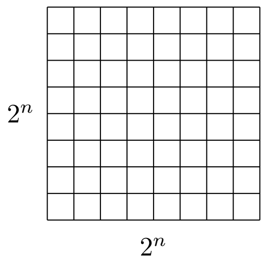
which proves $P(n+1)$.

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

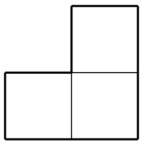
Induction was helpful for *proving the correctness* of this summation formula, but not helpful for *discovering* it in the first place. Tricks and methods for finding such formulas will appear in a later chapter.

Courtyard Tiling

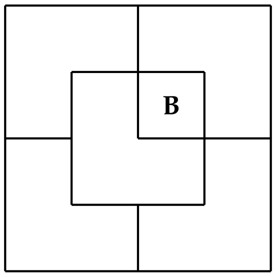
During the development of MIT's famous Stata Center, costs rose further and further over budget, and there were some radical fundraising ideas. One rumored plan was to install a big courtyard with dimensions $2^n \times 2^n$:



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



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



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

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

Proof. (doomed attempt) The proof is by induction. Let $P(n)$ be the proposition that there exists a tiling of a $2^n \times 2^n$ courtyard with Bill in the center.

Base case: $P(0)$ is true because Bill fills the whole courtyard.

Inductive step: Assume that there is a tiling of a $2^n \times 2^n$ courtyard with Bill in the center for some $n \geq 0$. We must prove that there is a way to tile a $2^{n+1} \times 2^{n+1}$ courtyard with Bill in the center .... ■

Now we're in trouble! The ability to tile a smaller courtyard with Bill in the center isn't much help in tiling a larger courtyard with Bill in the center. We haven't figured out how to bridge the gap between $P(n)$ and $P(n+1)$.

So if we're going to prove Theorem 3.2.2 by induction, we're going to need some *other* induction hypothesis than simply the statement about n that we're trying to prove.

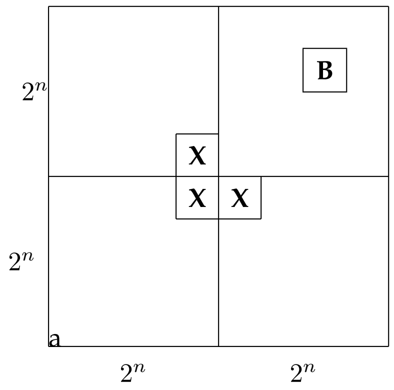
When this happens, your first fallback should be to look for a *stronger* induction hypothesis; that is, one which implies your previous hypothesis. For example, we could make $P(n)$ the proposition that for *every* location of Bill in a $2^n \times 2^n$ courtyard, there exists a tiling of the remainder.

This advice may sound bizarre: "If you can't prove something, try to prove something grander!" But for induction arguments, this makes sense. In the inductive step, where you have to prove $P(n)$ IMPLIES $P(n+1)$, you're in better shape because you can *assume* $P(n)$, which is now a more powerful statement. Let's see how this plays out in the case of courtyard tiling.

Proof. (successful attempt) The proof is by induction. Let $P(n)$ be the proposition that for every location of Bill in a $2^n \times 2^n$ courtyard, there exists a tiling of the remainder.

Base case: $P(0)$ is true because Bill fills the whole courtyard.

Inductive step: Assume that $P(n)$ is true for some $n \geq 0$; that is, for every location of Bill in a $2^n \times 2^n$ courtyard, there exists a tiling of the remainder. Divide the $2^{n+1} \times 2^{n+1}$ courtyard into four quadrants, each $2^n \times 2^n$. One quadrant contains Bill (**B** in the diagram below). Place a temporary Bill (**X** in the diagram) in each of the three central squares lying outside this quadrant:



Now we can tile each of the four quadrants by the induction assumption. Replacing the three temporary Bills with a single L-shaped tile completes the job. This proves that $P(n)$ implies $P(n+1)$ for all $n \geq 0$. The theorem follows as a special case. ■

This proof has two nice properties. First, not only does the argument guarantee that a tiling exists, but also it gives an algorithm for finding such a tiling. Second, we have a stronger result: if Bill wanted a statue on the edge of the courtyard, away from the pigeons, we could accommodate him!

Strengthening the induction hypothesis is often a good move when an induction proof won't go through. But keep in mind that the stronger assertion must actually be *true*; otherwise, there isn't much hope of constructing a valid proof! Sometimes finding just the right induction hypothesis requires trial, error, and insight. For example, mathematicians spent almost twenty years trying to prove or disprove the conjecture that "Every planar graph is 5-choosable"[3]. Then, in 1994, Carsten Thomassen gave an induction proof simple enough to explain on a napkin. The key turned out to be finding an extremely clever induction hypothesis; with that in hand, completing the argument is easy!

A Faulty Induction Proof

False Theorem. *All horses are the same color.*

Notice that no n is mentioned in this assertion, so we're going to have to reformulate it in a way that makes an n explicit. In particular, we'll (falsely) prove that

[3]5-choosability is a slight generalization of 5-colorability. Although every planar graph is 4-colorable and therefore 5-colorable, not every planar graph is 4-choosable. If this all sounds like nonsense, don't panic. We'll discuss graphs, planarity, and coloring in a later chapter.

False Theorem 3.2.3. *In every set of $n \geq 1$ horses, all the horses are the same color.*

This a statement about all integers $n \geq 1$ rather ≥ 0, so it's natural to use a slight variation on induction: prove $P(1)$ in the base case and then prove that $P(n)$ implies $P(n+1)$ for all $n \geq 1$ in the inductive step. This is a perfectly valid variant of induction and is *not* the problem with the proof below.

False proof. The proof is by induction on n. The induction hypothesis, $P(n)$, will be

$$\text{In every set of } n \text{ horses, all are the same color.} \tag{3.4}$$

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 "..." notation creates the impression that there are some remaining horses besides h_1 and h_{n+1}. However, this is not true when $n = 1$. In that case, the first set is just h_1 and the second is h_2, and there are no remaining horses besides them. So h_1 and h_2 need not be the same color!

This mistake knocks a critical link out of our induction argument. We proved $P(1)$ and we *correctly* proved $P(2) \longrightarrow P(3)$, $P(3) \longrightarrow P(4)$, etc. But we failed to prove $P(1) \longrightarrow P(2)$, and so everything falls apart: we can not conclude that $P(2)$, $P(3)$, etc., are true. And, of course, these propositions are all false; there are horses of a different color.

Students sometimes claim that the mistake in the proof is because $P(n)$ is false for $n \geq 2$, and the proof assumes something false, namely, $P(n)$, in order to prove $P(n+1)$. You should think about how to explain to such a student why this claim would get no credit on a Math for Computer Science exam.

Induction versus Well Ordering

The Induction Axiom looks nothing like the Well Ordering Principle, but these two proof methods are closely related. In fact, as the examples above suggest, we can take any Well Ordering proof and reformat it into an Induction proof. Conversely, it's equally easy to take any Induction proof and reformat it into a Well Ordering proof.

EDITING NOTE: Here's how to reformat an induction proof and into a Well Ordering proof : suppose that we have a proof by induction with hypothesis $P(n)$. Then we start a Well Ordering proof by assuming the set of counterexamples to P is nonempty. Then by Well Ordering there is a smallest counterexample, s, that is, a smallest s such that $P(s)$ is false.

Now we use the proof of $P(0)$ that was part of the Induction proof to conclude that s must be greater than 0. Also since s is the smallest counterexample, we can conclude that $P(s-1)$ must be true. At this point we reuse the proof of the inductive step in the Induction proof, which shows that since $P(s-1)$ 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 $n \in \mathbb{N}$. ■

So what's the difference? Well, sometimes induction proofs are clearer because they resemble recursive procedures that reduce handling an input of size $n + 1$ to handling one of size n. On the other hand, Well Ordering proofs sometimes seem more natural, and also come out slightly shorter. The choice of method is really a matter of style—but style does matter.

3.2.2 Strong Induction

A useful variant of induction is called *strong induction*. Strong Induction and Ordinary Induction are used for exactly the same thing: proving that a predicate $P(n)$ is true for all $n \in \mathbb{N}$.

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

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

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

Rule. ***Strong Induction Rule***

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

The only change from the ordinary induction principle is that strong induction allows you to assume more stuff in the inductive step of your proof! In an ordinary induction argument, you assume that $P(n)$ is true and try to prove that $P(n+1)$ is also true. In a strong induction argument, you may assume that $P(0)$, $P(1)$, $\ldots$, and $P(n)$ are *all* true when you go to prove $P(n+1)$. These extra assumptions can only make your job easier.

Products of Primes

As a first example, we'll use strong induction to re-prove Theorem 3.1.1 which we previously proved using Well Ordering.

Lemma 3.2.4. *Every integer greater than 1 is a product of primes.*

Proof. We will prove Lemma 3.2.4 by strong induction, letting the induction hypothesis, $P(n)$, be

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

So Lemma 3.2.4 will follow if we prove that $P(n)$ holds for all $n \geq 2$.

Base Case: ($n = 2$) $P(2)$ is true because 2 is prime, and so it is a length one product of primes by convention.

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

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

Otherwise, $n + 1$ is not prime, which by definition means $n + 1 = km$ for some integers k, m such that $2 \leq k, m < n + 1$. Now by strong induction hypothesis, we know that k is a product of primes. Likewise, m is a product of primes. it follows immediately that $km = n$ is also a product of primes. Therefore, $P(n + 1)$ holds in this case as well.

So $P(n + 1)$ holds in any case, which completes the proof by strong induction that $P(n)$ holds for all nonnegative integers, n.

■

EDITING NOTE: Here's a fallacious argument: every number can be factored uniquely into primes. Apply the same proof as before, adding "uniquely" to the inductive hypothesis. The problem is that even if $n = ab$ and a, b have unique factorizations, it is still possible that $n = cd$ for different c and d, producing a different factorization of n.

The argument is false, but the claim is true and is known as the fundamental theorem of arithmetic.

■

Making Change

The country Inductia, whose unit of currency is the Strong, has coins worth 3Sg (3 Strongs) and 5Sg. Although the Inductians have some trouble making small change like 4Sg or 7Sg, it turns out that they can collect coins to make change for any number that is at least 8 Strongs.

Strong induction makes this easy to prove for $n+1 \geq 11$, because then $(n+1) - 3 \geq 8$, so by strong induction the Inductians can make change for exactly $(n+1)-3$ Strongs, and then they can add a 3Sg coin to get $(n+1)$Sg. So the only thing to do is check that they can make change for all the amounts from 8 to 10Sg, which is not too hard to do.

Here's a detailed writeup using the official format:

Proof. We prove by strong induction that the Inductians can make change for any amount of at least 8Sg. The induction hypothesis, $P(n)$ will be:

There is a collection of coins whose value is $n + 8$ Strongs.

Base case: $P(0)$ is true because a 3Sg coin together with 5Sgcoin makes 8Sg.

Inductive step: We assume $P(m)$ holds for all $m \leq n$, and prove that $P(n + 1)$ holds. We argue by cases:

Case ($n+1$ = 1): We have to make $(n+1)+8 = 9$Sg. We can do this using three 3Sg coins.

Case ($n + 1$ = 2): We have to make $(n + 1) + 8 = 10$Sg. Use two 5Sg coins.

Case ($n + 1 \geq 3$): Then $0 \leq n - 2 \leq n$, so by the strong induction hypothesis, the Inductians can make change for $n - 2$ Strong. Now by adding a 3Sg coin, they can make change for $(n + 1)$Sg.

So in any case, $P(n + 1)$ is true, and we conclude by strong induction that for all $n = 0, 1, \ldots$, the Inductians can make change for $n + 8$ Strong. That is, they can make change for any number of eight or more Strong.

■

The Stacking Game

Here is another exciting game that's surely about to sweep the nation `:-)` !

You begin with a stack of n boxes. Then you make a sequence of moves. In each move, you divide one stack of boxes into two nonempty stacks. The game ends when you have n stacks, each containing a single box. You earn points for each move; in particular, if you divide one stack of height $a + b$ into two stacks with heights a and b, then you score ab points for that move. Your overall score is the sum of the points that you earn for each move. What strategy should you use to maximize your total score?

As an example, suppose that we begin with a stack of $n = 10$ boxes. Then the game might proceed as follows:

Stack Heights										**Score**
$\underline{10}$										
5	$\underline{5}$									25 points
$\underline{5}$	3	2								6
$\underline{4}$	3	2	1							4
2	$\underline{3}$	2	1	2						4
$\underline{2}$	2	2	1	2	1					2
1	$\underline{2}$	2	1	2	1	1				1
1	1	$\underline{2}$	1	2	1	1	1			1
1	1	1	1	$\underline{2}$	1	1	1	1		1
1	1	1	1	1	1	1	1	1	1	1
							Total Score		$=$	45 points

On each line, the underlined stack is divided in the next step. Can you find a better strategy?

Analyzing the Game

Let's use strong induction to analyze the unstacking game. We'll prove that your score is determined entirely by the number of boxes —your strategy is irrelevant!

Theorem 3.2.5. *Every way of unstacking n blocks gives a score of $n(n-1)/2$ points.*

There are a couple technical points to notice in the proof:

- The template for a strong induction proof is exactly the same as for ordinary induction.
- As with ordinary induction, we have some freedom to adjust indices. In this case, we prove $P(1)$ in the base case and prove that $P(1), \ldots, P(n)$ imply $P(n+1)$ for all $n \geq 1$ in the inductive step.

Proof. The proof is by strong induction. Let $P(n)$ be the proposition that every way of unstacking n blocks gives a score of $n(n-1)/2$.

Base case: If $n = 1$, then there is only one block. No moves are possible, and so the total score for the game is $1(1-1)/2 = 0$. Therefore, $P(1)$ is true.

Inductive step: Now we must show that $P(1), \ldots, P(n)$ imply $P(n+1)$ for all $n \geq 1$. So assume that $P(1), \ldots, P(n)$ are all true and that we have a stack of $n+1$ blocks. The first move must split this stack into substacks with positive sizes a and b where $a + b = n + 1$ and $0 < a, b \leq n$. Now the total score for the game is the sum of points for this first move plus points obtained by unstacking the two resulting substacks:

$$\begin{aligned}
\text{total score} &= (\text{score for 1st move}) \\
&\quad + (\text{score for unstacking } a \text{ blocks}) \\
&\quad + (\text{score for unstacking } b \text{ blocks}) \\
&= ab + \frac{a(a-1)}{2} + \frac{b(b-1)}{2} && \text{by } P(a) \text{ and } P(b) \\
&= \frac{(a+b)^2 - (a+b)}{2} = \frac{(a+b)((a+b)-1)}{2} \\
&= \frac{(n+1)n}{2}
\end{aligned}$$

This shows that $P(1), P(2), \ldots, P(n)$ imply $P(n+1)$.

Therefore, the claim is true by strong induction. ■

3.2.3 Strong Induction versus Induction

Is strong induction really "stronger" than ordinary induction? You can assume a lot more when proving the induction step, so it may seem that strong induction is much more powerful, but it's not. Strong induction may make it easier to prove a proposition, but any proof by strong induction can be reformatted to prove the same thing by ordinary induction (using a slightly more complicated induction hypothesis). Again, the choice of method is a matter of style.

When you're doing a proof by strong induction, you should say so: it will help your reader to know that $P(n + 1)$ may not follow directly from just $P(n)$.

3.2.4 Problems

Practice Problems

Class Problems

Homework Problems

EDITING NOTE:

Problem 3.1.

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.

■

Chapter 4

Number Theory

Number theory is the study of the integers. *Why* anyone would want to study the integers is not immediately obvious. First of all, what's to know? There's 0, there's 1, 2, 3, and so on, and, oh yeah, -1, -2, ... Which one don't you understand? Second, what practical value is there in it? The mathematician G. H. Hardy expressed pleasure in its impracticality when he wrote:

> [Number theorists] may be justified in rejoicing that there is one science, at any rate, and that their own, whose very remoteness from or-

dinary human activities should keep it gentle and clean.

Hardy was specially concerned that number theory not be used in warfare; he was a pacifist. You may applaud his sentiments, but he got it wrong: Number Theory underlies modern cryptography, which is what makes secure online communication possible. Secure communication is of course crucial in war—which may leave poor Hardy spinning in his grave. It's also central to online commerce. Every time you buy a book from Amazon, check your grades on WebSIS, or use a PayPal account, you are relying on number theoretic algorithms.

Number theory also provides an excellent environment for us to practice and apply the proof techniques that we developed in Chapters 2 and 3.

Since we'll be focusing on properties of the integers, we'll adopt the default convention in this chapter that *variables range over the set of integers*, $\mathbb{Z}$.

4.1 Divisibility

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

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

The notation, $a \mid b$, is an abbreviation for "a divides b." If $a \mid b$, then we also say that b is a *multiple* of a. A consequence of this definition is that every number divides zero.

This seems simple enough, but let's play with this definition. The Pythagoreans, an ancient sect of mathematical mystics, said that a number is *perfect* if it equals the sum of its positive integral divisors, excluding itself. For example, $6 = 1+2+3$ and $28 = 1 + 2 + 4 + 7 + 14$ are perfect numbers. On the other hand, 10 is not perfect because $1 + 2 + 5 = 8$, and 12 is not perfect because $1 + 2 + 3 + 4 + 6 = 16$. Euclid characterized all the *even* perfect numbers around 300 BC. But is there an *odd* perfect number? More than two thousand years later, we still don't know! All numbers up to about 10^{300} have been ruled out, but no one has proved that there isn't an odd perfect number waiting just over the horizon.

So a half-page into number theory, we've strayed past the outer limits of human knowledge! This is pretty typical; number theory is full of questions that are easy to pose, but incredibly difficult to answer.[1] For example, several such problems are shown in the box on the following page.

Interestingly, we'll see that computer scientists have found ways to turn some of these difficulties to their advantage.

4.1.1 Facts about Divisibility

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

Lemma 4.1.1. *The following statements about divisibility hold.*

1. *If $a \mid b$, then $a \mid bc$ for all c.*

2. *If $a \mid b$ and $b \mid c$, then $a \mid c$.*

3. *If $a \mid b$ and $a \mid c$, then $a \mid sb + tc$ for all s and t.*

[1] *Don't Panic*—we're going to stick to some relatively benign parts of number theory. These super-hard unsolved problems rarely get put on problem sets.

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: Assume $a \mid b$ and $b \mid c$. Since $a \mid b$, there exists an integer k_1 such that $ak_1 = b$. Since $b \mid c$, there exists an integer k_2 such that $bk_2 = c$. Substituting ak_1 for b in the second equation gives $(ak_1)k_2 = c$. So $a(k_1k_2) = c$, which implies that $a \mid c$. ■

EDITING NOTE:

Proof of (4): We must show that $a \mid b$ implies $ca \mid cb$ and vice-versa.

- First, suppose $a \mid b$. This means $ak = b$ for some k. Multiplying both sides by c gives $cak = cb$ for some k. This implies $ca \mid cb$.

- Now, suppose $ca \mid cb$. Then $cak = cb$ for some k. We can divide both sides by c since c is nonzero, so $ak = b$ for some k. This means $a \mid b$.

■

Famous Conjectures in Number Theory

Fermat's Last Theorem There are no positive integers x, y, and z such that

$$x^n + y^n = z^n$$

for some integer $n > 2$. In a book he was reading around 1630, Fermat claimed to have a proof but not enough space in the margin to write it down. Wiles finally gave a proof of the theorem in 1994, after seven years of working in secrecy and isolation in his attic. His proof did not fit in any margin.

Goldbach Conjecture Every even integer greater than two is equal to the sum of two primes[a]. For example, $4 = 2 + 2$, $6 = 3 + 3$, $8 = 3 + 5$, etc. The conjecture holds for all numbers up to 10^{16}. In 1939 Schnirelman proved that every even number can be written as the sum of not more than 300,000 primes, which was a start. Today, we know that every even number is the sum of at most 6 primes.

Twin Prime Conjecture There are infinitely many primes p such that $p + 2$ is also

a prime. In 1966 Chen showed that there are infinitely many primes p such

4.1.2 When Divisibility Goes Bad

As you learned in elementary school, if one number does *not* evenly divide another, you get a "quotient" and a "remainder" left over. More precisely:

Theorem 4.1.2 (Division Theorem). [2] *Let n and d be integers such that $d > 0$. Then there exists a unique pair of integers q and r, such that*

$$n = q \cdot d + r \text{ AND } 0 \leq r < d. \tag{4.1}$$

The number q is called the *quotient* and the number r is called the *remainder* of n divided by d. We use the notation $\text{qcnt}(n, d)$ for the quotient and $\text{rem}(n, d)$ for the remainder.

For example, $\text{qcnt}(2716, 10) = 271$ and $\text{rem}(2716, 10) = 6$, since $2716 = 271 \cdot 10 + 6$. 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

[2] This theorem is often called the "Division Algorithm," even though it is not what we would call an algorithm. We will take this familiar result for granted without proof.

negative numbers strangely.

EDITING NOTE:

but it's worth emphasizing that it is an "existence and uniqueness" theorem: it asserts that q and r *exist* and also that these values are *unique*. Thus, the Division Theorem is one example of an "existence and uniqueness" theorem; there are many others.

Not surprisingly, the proof of such a theorem always has two parts:

- A proof that something exists, such as the quotient q and remainder r.
- A proof that nothing else fits the bill; that is, there is no other quotient q' and remainder r'.

We'll prove a famous "existence and uniqueness" theorem in this way shortly.

■

4.1.3 Die Hard

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?

The preceding script is from the movie *Die Hard 3: With a Vengeance*. In the movie, Samuel L. Jackson and Bruce Willis have to disarm a bomb planted by the diabolical Simon Gruber. Fortunately, they find a solution in the nick of time. (No doubt reading the script helped.) On the surface, *Die Hard 3* is just a B-grade action movie; however, we think the inner message of the film is that everyone should learn at least a little number theory.

Unfortunately, Hollywood never lets go of a gimmick. Although there were no water jug tests in *Die Hard 4*, rumor has it that the jugs will return in future sequels:

Die Hard 5: Die Hardest Bruce goes on vacation and—shockingly—happens into a terrorist plot. To save the day, he must make 3 gallons using 21- and 26-gallon jugs.

Die Hard 6: Die of Old Age Bruce must save his assisted living facility from a criminal mastermind by forming 2 gallons with 899- and 1147-gallon jugs.

Die Hard 7: Die Once and For All Bruce has to make 4 gallons using 3- and 6-gallon jugs.

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?

Finding an Invariant Property

Suppose that we have water jugs with capacities a and b with $b \geq a$. The state of the system is described below with a pair of numbers (x, y), where x is the amount of water in the jug with capacity a and y is the amount in the jug with capacity b. Let's carry out sample operations and see what happens, assuming the b-jug is big enough:

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

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

$$s \cdot a + t \cdot b \tag{4.2}$$

for some integers s and t. An expression of the form (4.2) is called an *integer linear combination* of a and b, but in this chapter we'll just call it a *linear combination*, since we're only talking integers. So we're suggesting:

Lemma 4.1.3. *Suppose that we have water jugs with capacities a and b. Then the amount of water in each jug is always a linear combination of a and b.*

Lemma 4.1.3 is easy to prove by induction on the number of pourings.

Proof. The induction hypothesis, $P(n)$, is the proposition that after n steps, the amount of water in each jug is a linear combination of a and b.

Base case: ($n = 0$). $P(0)$ is true, because both jugs are initially empty, and $0 \cdot a + 0 \cdot b = 0$.

Inductive step. We assume by induction hypothesis that after n steps the amount of water in each jug is a linear combination of a and b. There are two cases:

- If we fill a jug from the fountain or empty a jug into the fountain, then that jug is empty or full. The amount in the other jug remains a linear combination of a and b. So $P(n + 1)$ holds.

- Otherwise, we pour water from one jug to another until one is empty or the other is full. By our assumption, the amount in each jug is a linear combination of a and b before we begin pouring:

$$j_1 = s_1 \cdot a + t_1 \cdot b$$

$$j_2 = s_2 \cdot a + t_2 \cdot b$$

After pouring, one jug is either empty (contains 0 gallons) or full (contains a or b gallons). Thus, the other jug contains either $j_1 + j_2$ gallons, $j_1 + j_2 - a$, or $j_1 + j_2 - b$ gallons, all of which are linear combinations of a and b. So $P(n+1)$ holds in this case as well.

So in any case, $P(n + 1)$ follows, completing the proof by induction. ∎

This theorem has an important corollary:

Corollary 4.1.4. *Bruce dies.*

Proof. In Die Hard 7 , Bruce has water jugs with capacities 3 and 6 and must form 4 gallons of water. However, the amount in each jug is always of the form $3s + 6t$ by Lemma 4.1.3. This is always a multiple of 3 by Lemma 4.1.1.3, so he cannot measure out 4 gallons. ∎

But Lemma 4.1.3 isn't very satisfying. We've just managed to recast a pretty understandable question about water jugs into a complicated question about linear combinations. This might not seem like progress. Fortunately, linear combinations are closely related to something more familiar, namely greatest common divisors, and these will help us solve the water jug problem.

4.2 The Greatest Common Divisor

The *greatest common divisor* of a and b is exactly what you'd guess: the largest number that is a divisor of both a and b. It is denoted by $\gcd(a, b)$. For example, $\gcd(18, 24) = 6$. The greatest common divisor turns out to be a very valuable piece of information about the relationship between a and b and for reasoning about integers in general. So we'll be making lots of arguments about greatest common divisors in what follows.

4.2.1 Linear Combinations and the GCD

The theorem below relates the greatest common divisor to linear combinations. This theorem is *very* useful; take the time to understand it and then remember it!

Theorem 4.2.1. *The greatest common divisor of a and b is equal to the smallest positive linear combination of a and b.*

For example, the greatest common divisor of 52 and 44 is 4. And, sure enough, 4 is a linear combination of 52 and 44:

$$6 \cdot 52 + (-7) \cdot 44 = 4$$

Furthermore, no linear combination of 52 and 44 is equal to a smaller positive integer.

Proof of Theorem 4.2.1. 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 $sa + tb$ for any s and t. The $\gcd(a, b)$ is by definition a common divisor of a and b, so

$$\gcd(a, b) \mid sa + tb \tag{4.3}$$

every s and t. In particular, $\gcd(a, b) \mid m$, which implies that $\gcd(a, b) \leq m$.

Now, we show that $m \leq \gcd(a, b)$. We do this by showing that $m \mid a$. A symmetric argument shows that $m \mid b$, which means that m is a common divisor of a and b. Thus, m must be less than or equal to the *greatest* common divisor of a and b.

All that remains is to show that $m \mid a$. By the Division Algorithm, there exists a quotient q and remainder r such that:

$$a = q \cdot m + r \qquad \text{(where } 0 \leq r < m\text{)}$$

Recall that $m = sa + tb$ for some integers s and t. Substituting in for m gives:

$$a = q \cdot (sa + tb) + r, \qquad \text{so}$$

$$r = (1 - qs)a + (-qt)b.$$

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 $m \mid a$. ■

Corollary 4.2.2. *An integer is linear combination of a and b iff it is a multiple of* $\gcd(a, b)$.

Proof. By (4.3), every linear combination of a and b is a multiple of $\gcd(a, b)$. Conversely, since $\gcd(a, b)$ is a linear combination of a and b, every multiple of $\gcd(a, b)$ is as well. ■

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

Corollary 4.2.3. *Suppose that we have water jugs with capacities a and b. Then the amount of water in each jug is always a multiple of* $\gcd(a, b)$.

For example, there is no way to form 4 gallons using 3- and 6-gallon jugs, because 4 is not a multiple of $\gcd(3, 6) = 3$.

4.2.2 Properties of the Greatest Common Divisor

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

Lemma 4.2.4. *The following statements about the greatest common divisor hold:*

1. *Every common divisor of a and b divides $\gcd(a, b)$.*
2. $\gcd(ka, kb) = k \cdot \gcd(a, b)$ *for all $k > 0$.*
3. *If $\gcd(a, b) = 1$ and $\gcd(a, c) = 1$, then $\gcd(a, bc) = 1$.*
4. *If $a \mid bc$ and $\gcd(a, b) = 1$, then $a \mid c$.*
5. $\gcd(a, b) = \gcd(b, \operatorname{rem}(a, b))$.

Here's the trick to proving these statements: translate the $\gcd$ world to the linear combination world using Theorem 4.2.1, argue about linear combinations, and then translate back using Theorem 4.2.1 again.

Proof. We prove only parts 3. and 4.

Proof of 3. The assumptions together with Theorem 4.2.1 imply that there exist integers s, t, u, and v such that:

$$sa + tb = 1$$

$$ua + vc = 1$$

Multiplying these two equations gives:

$$(sa + tb)(ua + vc) = 1$$

The left side can be rewritten as $a \cdot (asu + btu + csv) + bc(tv)$. This is a linear combination of a and bc that is equal to 1, so $\gcd(a, bc) = 1$ by Theorem 4.2.1.

Proof of 4. Theorem 4.2.1 says that $\gcd(ac, bc)$ is equal to a linear combination of ac and bc. Now $a \mid ac$ trivially and $a \mid bc$ by assumption. Therefore, a divides *every* linear combination of ac and bc. In particular, a divides $\gcd(ac, bc) = c \cdot \gcd(a, b) = c \cdot 1 = c$. The first equality uses part 2. of this lemma, and the second uses the assumption that $\gcd(a, b) = 1$. ■

4.2.3 Euclid's Algorithm

Part (5) of Lemma 4.2.4 is useful for quickly computing the greatest common divisor of two numbers. For example, we could compute the greatest common divisor of 1147 and 899 by repeatedly applying part (5):

$$\begin{aligned}
\gcd(1247, 899) &= \gcd\big(899, \underbrace{\text{rem}(1247, 899)}_{=248}\big) \\
&= \gcd\big(248, \underbrace{\text{rem}(899, 248)}_{=155}\big) \\
&= \gcd\big(155, \underbrace{\text{rem}(248, 155)}_{=93}\big) \\
&= \gcd\big(93, \underbrace{\text{rem}(155, 93)}_{=62}\big) \\
&= \gcd\big(62, \underbrace{\text{rem}(93, 62)}_{=31}\big) \\
&= \gcd\big(31, \underbrace{\text{rem}(62, 31)}_{=0}\big) \\
&= \gcd(31, 0) \\
&= 31
\end{aligned}$$

The last equation might look wrong, but 31 is a divisor of both 31 and 0 since every integer divides 0.

This process is called *Euclid's algorithm* and it was discovered by the Greeks over 3000 years ago.

But what about Die Hard 5. Is it possible for Bruce 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 if it can be done either. To resolve the matter, we will need more number theory.

4.2.4 One Solution for All Water Jug Problems

Corollary 4.2.2 says that 3 can be written as a linear combination of 21 and 26, since 3 is a multiple of $\gcd(21, 26) = 1$. In other words, there exist integers s and t such that:

$$3 = s \cdot 21 + t \cdot 26$$

We don't know what the coefficients s and t are, but we do know that they exist.

Now the coefficient s could be either positive or negative. However, we can readily transform this linear combination into an equivalent linear combination

$$3 = s' \cdot 21 + t' \cdot 26 \tag{4.4}$$

where the coefficient s' is positive. The trick is to notice that if we increase s by 26 in the original equation and decrease t by 21, then the value of the expression $s \cdot 21 + t \cdot 26$ is unchanged overall. Thus, by repeatedly increasing the value of s (by 26 at a time) and decreasing the value of t (by 21 at a time), we get a linear combination $s' \cdot 21 + t' \cdot 26 = 3$ where the coefficient s' is positive. Notice that then t' must be negative; otherwise, this expression would be much greater than 3.

Now we can form 3 gallons using jugs with capacities 21 and 26: We simply repeat the following steps s' times:

1. Fill the 21-gallon jug.

2. Pour all the water in the 21-gallon jug into the 26-gallon jug. If at any time the 26-gallon jug becomes full, empty it out, and continue pouring the 21-gallon jug into the 26-gallon jug.

At the end of this process, we must have have emptied the 26-gallon jug exactly $|t'|$ times. Here's why: we've taken $s' \cdot 21$ gallons of water from the fountain, and we've poured out some multiple of 26 gallons. If we emptied fewer than $|t'|$ times, then by (4.4), the big jug would be left with at least $3 + 26$ gallons, which is more than it can hold; if we emptied it more times, the big jug would be left containing at most $3 - 26$ gallons, which is nonsense. But once we have emptied the 26-gallon jug exactly $|t'|$ times, equation (4.4) implies that there are exactly 3 gallons left.

Remarkably, we don't even need to know the coefficients s' and t' in order to use this strategy! Instead of repeating the outer loop s' times, we could just repeat *until we obtain 3 gallons*, since that must happen eventually. Of course, we have to keep track of the amounts in the two jugs so we know when we're done. Here's the solution that approach gives:

$$
\begin{array}{lllllllll}
(0,0) & \xrightarrow{\text{fill 21}} & (21,0) & \xrightarrow{\text{pour 21 into 26}} & (0,21) & & & & \\
& \xrightarrow{\text{fill 21}} & (21,21) & \xrightarrow{\text{pour 21 into 26}} & (16,26) & \xrightarrow{\text{empty 26}} & (16,0) & \xrightarrow{\text{pour 21 into 26}} & (0,16) \\
& \xrightarrow{\text{fill 21}} & (21,16) & \xrightarrow{\text{pour 21 into 26}} & (11,26) & \xrightarrow{\text{empty 26}} & (11,0) & \xrightarrow{\text{pour 21 into 26}} & (0,11) \\
& \xrightarrow{\text{fill 21}} & (21,11) & \xrightarrow{\text{pour 21 into 26}} & (6,26) & \xrightarrow{\text{empty 26}} & (6,0) & \xrightarrow{\text{pour 21 into 26}} & (0,6) \\
& \xrightarrow{\text{fill 21}} & (21,6) & \xrightarrow{\text{pour 21 into 26}} & (1,26) & \xrightarrow{\text{empty 26}} & (1,0) & \xrightarrow{\text{pour 21 into 26}} & (0,1) \\
& \xrightarrow{\text{fill 21}} & (21,1) & \xrightarrow{\text{pour 21 into 26}} & (0,22) & & & & \\
& \xrightarrow{\text{fill 21}} & (21,22) & \xrightarrow{\text{pour 21 into 26}} & (17,26) & \xrightarrow{\text{empty 26}} & (17,0) & \xrightarrow{\text{pour 21 into 26}} & (0,17) \\
& \xrightarrow{\text{fill 21}} & (21,17) & \xrightarrow{\text{pour 21 into 26}} & (12,26) & \xrightarrow{\text{empty 26}} & (12,0) & \xrightarrow{\text{pour 21 into 26}} & (0,12) \\
& \xrightarrow{\text{fill 21}} & (21,12) & \xrightarrow{\text{pour 21 into 26}} & (7,26) & \xrightarrow{\text{empty 26}} & (7,0) & \xrightarrow{\text{pour 21 into 26}} & (0,7) \\
& \xrightarrow{\text{fill 21}} & (21,7) & \xrightarrow{\text{pour 21 into 26}} & (2,26) & \xrightarrow{\text{empty 26}} & (2,0) & \xrightarrow{\text{pour 21 into 26}} & (0,2) \\
& \xrightarrow{\text{fill 21}} & (21,2) & \xrightarrow{\text{pour 21 into 26}} & (0,23) & & & & \\
& \xrightarrow{\text{fill 21}} & (21,23) & \xrightarrow{\text{pour 21 into 26}} & (18,26) & \xrightarrow{\text{empty 26}} & (18,0) & \xrightarrow{\text{pour 21 into 26}} & (0,18) \\
& \xrightarrow{\text{fill 21}} & (21,18) & \xrightarrow{\text{pour 21 into 26}} & (13,26) & \xrightarrow{\text{empty 26}} & (13,0) & \xrightarrow{\text{pour 21 into 26}} & (0,13) \\
& \xrightarrow{\text{fill 21}} & (21,13) & \xrightarrow{\text{pour 21 into 26}} & (8,26) & \xrightarrow{\text{empty 26}} & (8,0) & \xrightarrow{\text{pour 21 into 26}} & (0,8) \\
& \xrightarrow{\text{fill 21}} & (21,8) & \xrightarrow{\text{pour 21 into 26}} & (3,26) & \xrightarrow{\text{empty 26}} & (3,0) & \xrightarrow{\text{pour 21 into 26}} & (0,3)
\end{array}
$$

The same approach works regardless of the jug capacities and even regardless the amount we're trying to produce! Simply repeat these two steps until the desired amount of water is obtained:

1. Fill the smaller jug.

2. Pour all the water in the smaller jug into the larger jug. If at any time the larger jug becomes full, empty it out, and continue pouring the smaller jug into the larger jug.

By the same reasoning as before, this method eventually generates every multiple of the greatest common divisor of the jug capacities—all the quantities we can possibly produce. No ingenuity is needed at all!

4.2.5 The Pulverizer

We have shown that no matter which pair of numbers a and b we are given, there is always a pair of integer coefficients s and t such that

$$\gcd(a, b) = sa + tb.$$

Unfortunately, the proof was *nonconstructive*: it didn't suggest a way for finding such s and t. That job is tackled by a mathematical tool that dates to sixth-century India, where it was called *kuttak*, which means "The Pulverizer". Today, the Pulverizer is more commonly known as "the extended Euclidean GCD algorithm", because it is so close to Euclid's Algorithm.

Euclid's Algorithm for finding the GCD of two numbers relies on repeated application of the equation:

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

For example, we can compute the GCD of 259 and 70 as follows:

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

The Pulverizer goes through the same steps, but requires some extra bookkeeping along the way: as we compute $\gcd(a, b)$, we keep track of how to write each of the remainders (49, 21, and 7, in the example) as a linear combination of a and b (this is worthwhile, because our objective is to write the last nonzero remainder, which is the GCD, as such a linear combination). For our example, here is this extra bookkeeping:

x	y	$(\text{rem}(x,y))$	$=$	$x - q \cdot y$
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 $\text{rem}(x, y)$, which can be written in the form $x - q \cdot y$. (Remember that the Division Algorithm says $x = q \cdot y + r$, where r is the remainder. We get $r = x - q \cdot y$ by rearranging terms.) Then we replaced x and y in this equation with equivalent linear combinations of a and b, which we already had computed. After simplifying, we were left with a linear combination of a and b that was equal to the remainder as desired. The final solution is boxed.

4.2.6 Problems

Class Problems

4.3 The Fundamental Theorem of Arithmetic

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

Theorem 4.3.1 (Fundamental Theorem of Arithmetic)**.** *Every positive integer n can be written in a unique way as a product of primes:*

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

Notice that the theorem would be false if 1 were considered a prime; for example, 15 could be written as $3 \cdot 5$ or $1 \cdot 3 \cdot 5$ or $1^2 \cdot 3 \cdot 5$. Also, we're relying on a standard convention: the product of an empty set of numbers is defined to be 1, much as the 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 4.3.2. *If p is a prime and $p \mid ab$, then $p \mid a$ or $p \mid b$.*

Proof. The greatest common divisor of a and p must be either 1 or p, since these are the only positive divisors of p. If $\gcd(a, p) = p$, then the claim holds, because a is a multiple of p. Otherwise, $\gcd(a, p) = 1$ and so $p \mid b$ by part (4) of Lemma 4.2.4. ■

A routine induction argument extends this statement to:

Lemma 4.3.3. *Let p be a prime. If $p \mid a_1 a_2 \cdots a_n$, then p divides some a_i.*

The Prime Number Theorem

Let $\pi(x)$ denote the number of primes less than or equal to x. For example, $\pi(10) = 4$ because 2, 3, 5, and 7 are the primes less than or equal to 10. Primes are very irregularly distributed, so the growth of π is similarly erratic. However, the Prime Number Theorem gives an approximate answer:

$$\lim_{x \to \infty} \frac{\pi(x)}{x / \ln x} = 1$$

Thus, primes gradually taper off. As a rule of thumb, about 1 integer out of every $\ln x$ in the vicinity of x is a prime.

The Prime Number Theorem was conjectured by Legendre in 1798 and proved a century later by de la Vallee Poussin and Hadamard in 1896. However, after his death, a notebook of Gauss was found to contain the same conjecture, which he apparently made in 1791 at age 15. (You sort of have to feel sorry for all the otherwise "great" mathematicians who had the misfortune of being contemporaries of Gauss.)

Now we're ready to prove the Fundamental Theorem of Arithmetic.

Proof. Theorem 3.1.1 showed, using the Well Ordering Principle, that every positive integer can be expressed as a product of primes. So we just have to prove this expression is unique. We will use Well Ordering to prove this too.

The proof is by contradiction: assume, contrary to the claim, that there exist positive integers that can be written as products of primes in more than one way. By the Well Ordering Principle, there is a smallest integer with this property. Call this integer n, and let

$$\begin{aligned} n &= p_1 \cdot p_2 \cdots p_j \\ &= q_1 \cdot q_2 \cdots q_k \end{aligned}$$

be two of the (possibly many) ways to write n as a product of primes. Then $p_1 \mid n$ and so $p_1 \mid q_1 q_2 \cdots q_k$. Lemma 4.3.3 implies that p_1 divides one of the primes q_i. But since q_i is a prime, it must be that $p_1 = q_i$. Deleting p_1 from the first product and q_i from the second, we find that n/p_1 is a positive integer *smaller* than n that can also be written as a product of primes in two distinct ways. But this contradicts

Figure 4.1: Alan Turing

the definition of n as the smallest such positive integer. ■

4.3.1 Problems

Class Problems

4.4 Alan Turing

The man pictured in Figure 4.1 is Alan Turing, the most important figure in the history of computer science. For decades, his fascinating life story was shrouded by government secrecy, societal taboo, and even his own deceptions.

At age 24, Turing wrote a paper entitled *On Computable Numbers, with an Application to the Entscheidungsproblem*. The crux of the paper was an elegant way to model a computer in mathematical terms. This was a breakthrough, because it allowed the tools of mathematics to be brought to bear on questions of computation. For example, with his model in hand, Turing immediately proved that there exist problems that no computer can solve—no matter how ingenious the programmer. Turing's paper is all the more remarkable because he wrote it in 1936, a full decade before any electronic computer actually existed.

The word "Entscheidungsproblem" in the title refers to one of the 28 mathematical problems posed by David Hilbert in 1900 as challenges to mathematicians of the 20th century. Turing knocked that one off in the same paper. And perhaps you've heard of the "Church-Turing thesis"? Same paper. So Turing was obviously a brilliant guy who generated lots of amazing ideas. But this lecture is about one of Turing's less-amazing ideas. It involved codes. It involved number theory. And it was sort of stupid.

Let's look back to the fall of 1937. Nazi Germany was rearming under Adolf Hitler, world-shattering war looked imminent, and—like us—Alan Turing was pondering the usefulness of number theory. He foresaw that preserving military secrets would be vital in the coming conflict and proposed a way *to encrypt communications using number theory*. This is an idea that has ricocheted up to our own time. Today, number theory is the basis for numerous public-key cryptosystems, digital signature schemes, cryptographic hash functions, and electronic payment systems. Furthermore, military funding agencies are among the biggest investors in cryptographic research. Sorry Hardy!

Soon after devising his code, Turing disappeared from public view, and half a century would pass before the world learned the full story of where he'd gone and what he did there. We'll come back to Turing's life in a little while; for now, let's investigate the code Turing left behind. The details are uncertain, since he never formally published the idea, so we'll consider a couple of possibilities.

4.4.1 Turing's Code (Version 1.0)

The first challenge is to translate a text message into an integer so we can perform mathematical operations on it. This step is not intended to make a message harder to read, so the details are not too important. Here is one approach: replace each letter of the message with two digits ($A = 01$, $B = 02$, $C = 03$, etc.) and string all the digits together to form one huge number. For example, the message "victory" could be translated this way:

$$\begin{array}{cccccccc} & \text{“v} & \text{i} & \text{c} & \text{t} & \text{o} & \text{r} & \text{y”} \\ \rightarrow & 22 & 09 & 03 & 20 & 15 & 18 & 25 \end{array}$$

Turing's code requires the message to be a prime number, so we may need to pad the result with a few more digits to make a prime. In this case, appending the digits 13 gives the number 2209032015182513, which is prime.

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 that further improvements will lead to a procedure that is useful in practice. But the truth is, there's no practical need to improve it, since very efficient *probabilistic* procedures for prime-testing have been known since the early 1970's. These procedures have some probability of giving a wrong answer, but their probability of being wrong is so tiny that relying on their answers is the best bet you'll ever make.

2. Is Turing's code secure?

   The Nazis see only the encrypted message $m^* = m \cdot k$, so recovering the original message m requires factoring m^*. Despite immense efforts, no really efficient factoring algorithm has ever been found. It appears to be a fundamentally difficult problem, though a breakthrough someday is not impossible. In effect, Turing's code puts to practical use his discovery that there are limits to the power of computation. Thus, provided m and k are sufficiently large, the Nazis seem to be out of luck!

This all sounds promising, but there is a major flaw in Turing's code.

4.4.2 Breaking Turing's Code

Let's consider what happens when the sender transmits a *second* message using Turing's code and the same key. This gives the Nazis two encrypted messages to look at:

$$m_1^* = m_1 \cdot k \qquad \text{and} \qquad m_2^* = m_2 \cdot k$$

The greatest common divisor of the two encrypted messages, m_1^* and m_2^*, is the secret key k. And, as we've seen, the GCD of two numbers can be computed very efficiently. So after the second message is sent, the Nazis can recover the secret key and read *every* message!

It is difficult to believe a mathematician as brilliant as Turing could overlook such a glaring problem. One possible explanation is that he had a slightly different system in mind, one based on *modular* arithmetic.

4.5 Modular Arithmetic

On page 1 of his masterpiece on number theory, *Disquisitiones Arithmeticae*, Gauss introduced the notion of "congruence". Now, Gauss is another guy who managed to cough up a half-decent idea every now and then, so let's take a look at this one. Gauss said that a is *congruent* to b *modulo* n iff $n \mid (a-b)$. This is written

$$a \equiv b \pmod{n}.$$

For example:

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

There is a close connection between congruences and remainders:

Lemma 4.5.1 (Congruences and Remainders)**.**

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

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

such that:

$$a = q_1 n + r_1 \qquad \text{where } 0 \leq r_1 < n,$$

$$b = q_2 n + r_2 \qquad \text{where } 0 \leq r_2 < n.$$

Subtracting the second equation from the first gives:

$$a - b = (q_1 - q_2)n + (r_1 - r_2) \qquad \text{where } -n < r_1 - r_2 < n.$$

Now $a \equiv b \pmod{n}$ if and only if n divides the left side. This is true if and only if n divides the right side, which holds if and only if $r_1 - r_2$ is a multiple of n. Given the bounds on $r_1 - r_2$, this happens precisely when $r_1 = r_2$, that is, when $\text{rem}(a, n) = \text{rem}(b, n)$. ■

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 isn't any more strongly associated with the 15 than with the 29. It would really be clearer to write $29 \equiv_{\bmod 7} 15$ for example, but the notation with the modulus at the end is firmly entrenched and we'll stick to it.

We'll make frequent use of the following immediate Corollary of Lemma 4.5.1:

Corollary 4.5.2.

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

Still another way to think about congruence modulo n is that it *defines a partition of the integers into n sets so that congruent numbers are all in the same set*. For example, suppose that we're working modulo 3. Then we can partition the integers into 3 sets as follows:

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

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 4.5.3 (Facts About Congruences)**.** *The following hold for $n \geq 1$:*

1. $a \equiv a \pmod{n}$
2. $a \equiv b \pmod{n}$ *implies* $b \equiv a \pmod{n}$
3. $a \equiv b \pmod{n}$ *and* $b \equiv c \pmod{n}$ *implies* $a \equiv c \pmod{n}$
4. $a \equiv b \pmod{n}$ *implies* $a + c \equiv b + c \pmod{n}$
5. $a \equiv b \pmod{n}$ *implies* $ac \equiv bc \pmod{n}$
6. $a \equiv b \pmod{n}$ *and* $c \equiv d \pmod{n}$ *imply* $a + c \equiv b + d \pmod{n}$
7. $a \equiv b \pmod{n}$ *and* $c \equiv d \pmod{n}$ *imply* $ac \equiv bd \pmod{n}$

Proof. Parts 1–3. follow immediately from Lemma 4.5.1. Part 4. follows immediately from the definition that $a \equiv b \pmod{n}$ iff $n \mid (a - b)$. Likewise, part 5. follows because if $n \mid (a-b)$ then it divides $(a-b)c = ac - bc$. To prove part 6., assume

$$a \equiv b \pmod{n} \tag{4.5}$$

and

$$c \equiv d \pmod{n}. \tag{4.6}$$

Then

$$\begin{aligned} a + c &\equiv b + c \pmod{n} && \text{(by part 4. and (4.5))},\\ c + b &\equiv d + b \pmod{n} && \text{(by part 4. and (4.6)), so}\\ b + c &\equiv b + d \pmod{n} && \text{and therefore}\\ a + c &\equiv b + d \pmod{n} && \text{(by part 3.)} \end{aligned}$$

Part 7 has a similar proof. ■

EDITING NOTE:

There is a close connection between modular arithmetic and the remainder operation, which we looked at last time. To clarify this link, let's reconsider the partition of the integers defined by congruence modulo 3:

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

Notice that two numbers are in the same set if and only if they leave the same remainder when divided by 3. The numbers in the first set all leave a remainder of 0 when divided by 3, the numbers in the second set leave a remainder of 1, and the numbers in the third leave a remainder of 2. Furthermore, notice that each number is in the same set as its own remainder. For example, 11 and $\text{rem}(11, 3) = 2$ are both in the same set. Let's bundle all this happy goodness into a lemma.

■

4.5.1 Turing's Code (Version 2.0)

In 1940, France had fallen before Hitler's army, and Britain stood alone against the Nazis in western Europe. British resistance depended on a steady flow of supplies brought across the north Atlantic from the United States by convoys of ships. These convoys were engaged in a cat-and-mouse game with German "U-boats"—

submarines—which prowled the Atlantic, trying to sink supply ships and starve Britain into submission. The outcome of this struggle pivoted on a balance of information: could the Germans locate convoys better than the Allies could locate U-boats or vice versa?

Germany lost.

But a critical reason behind Germany's loss was made public only in 1974: Germany's naval code, *Enigma*, had been broken by the Polish Cipher Bureau (see `http://en.wikipedia.org/wiki/Polish_Cipher_Bureau`) and the secret had been turned over to the British a few weeks before the Nazi invasion of Poland in 1939. Throughout much of the war, the Allies were able to route convoys around German submarines by listening in to German communications. The British government didn't explain *how* Enigma was broken until 1996. When it was finally released (by the US), the story revealed that Alan Turing had joined the secret British codebreaking effort at Bletchley Park in 1939, where he became the lead developer of methods for rapid, bulk decryption of German Enigma messages.

Turing's Enigma deciphering was an invaluable contribution to the Allied victory over Hitler.

Governments are always tight-lipped about cryptography, but the half-century of official silence about Turing's role in breaking Enigma and saving Britain may be related to some disturbing events after the war. More on that later. Let's get back to number theory and consider an alternative interpretation of Turing's code. Perhaps we had the basic idea right (multiply the message by the key), but erred in using *conventional* arithmetic instead of *modular* arithmetic. Maybe this is what Turing meant:

Beforehand The sender and receiver agree on a large prime p, which may be made public. (This will be the modulus for all our arithmetic.) They also agree on a secret key $k \in \{1, 2, \ldots, p-1\}$.

Encryption The message m can be any integer in the set $\{0, 1, 2, \ldots, p-1\}$; in particular, the message is no longer required to be a prime. The sender encrypts the message m to produce m^* by computing:

$$m^* = \text{rem}(mk, p) \tag{4.7}$$

Decryption (Uh-oh.)

The decryption step is a problem. We might hope to decrypt in the same way as before: by dividing the encrypted message m^* by the key k. The difficulty is that m^* is the *remainder* when mk is divided by p. So dividing m^* by k might not even give us an integer!

This decoding difficulty can be overcome with a better understanding of arithmetic modulo a prime.

4.5.2 Problems

Class Problems

4.6 Arithmetic with a Prime Modulus

4.6.1 Multiplicative Inverses

The *multiplicative inverse* of a number x is another number x^{-1} such that:

$$x \cdot x^{-1} = 1$$

Generally, multiplicative inverses exist over the real numbers. For example, the multiplicative inverse of 3 is $1/3$ since:

$$3 \cdot \frac{1}{3} = 1$$

The sole exception is that 0 does not have an inverse.

On the other hand, inverses generally do not exist over the integers. For example, 7 can not be multiplied by another integer to give 1.

Surprisingly, multiplicative inverses do exist when we're working *modulo a* *prime number*. For example, if we're working modulo 5, then 3 is a multiplicative inverse of 7, since:

$$7 \cdot 3 \equiv 1 \pmod 5$$

(All numbers congruent to 3 modulo 5 are also multiplicative inverses of 7; for example, $7 \cdot 8 \equiv 1 \pmod 5$ as well.) The only exception is that numbers congruent to 0 modulo 5 (that is, the multiples of 5) do not have inverses, much as 0 does not have an inverse over the real numbers. Let's prove this.

Lemma 4.6.1. *If p is prime and k is not a multiple of p, then k has a multiplicative inverse modulo p.*

Proof. Since p is prime, it has only two divisors: 1 and p. And since k is not a multiple of p, we must have $\gcd(p, k) = 1$. Therefore, there is a linear combination of p and k equal to 1:

$$sp + tk = 1$$

Rearranging terms gives:

$$sp = 1 - tk$$

This implies that $p \mid (1 - tk)$ by the definition of divisibility, and therefore $tk \equiv 1 \pmod{p}$ by the definition of congruence. Thus, t is a multiplicative inverse of k. ■

Multiplicative inverses are the key to decryption in Turing's code. Specifically, we can recover the original message by multiplying the encoded message by the *inverse* of the key:

$$\begin{aligned}
m^* \cdot k^{-1} &= \text{rem}(mk, p) \cdot k^{-1} && \text{(the def. (4.7) of } m^*\text{)} \\
&\equiv (mk)k^{-1} \pmod{p} && \text{(by Cor. 4.5.2)} \\
&\equiv m \pmod{p}.
\end{aligned}$$

This shows that $m^* k^{-1}$ is congruent to the original message m. Since m was in the range $0, 1, \ldots, p - 1$, we can recover it exactly by taking a remainder:

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

So all we need to decrypt the message is to find a value of k^{-1}. From the proof of Lemma 4.6.1, we know that t is such a value, where $sp + tk = 1$. Finding t is easy using the Pulverizer.

4.6.2 Cancellation

Another sense in which real numbers are nice is that one can cancel multiplicative terms. In other words, if we know that $m_1 k = m_2 k$, then we can cancel the k's and conclude that $m_1 = m_2$, provided $k \neq 0$. In general, cancellation is *not* valid in modular arithmetic. For example,

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

but canceling the 3's leads to the *false* conclusion that $2 \equiv 4 \pmod{6}$. The fact that multiplicative terms can not be canceled is the most significant sense in which congruences differ from ordinary equations. However, this difference goes away if we're working modulo a *prime*; then cancellation is valid.

Lemma 4.6.2. *Suppose p is a prime and k is not a multiple of p. Then*

$$ak \equiv bk \pmod{p} \quad \text{IMPLIES} \quad a \equiv b \pmod{p}.$$

Proof. Multiply both sides of the congruence by k^{-1}. ∎

We can use this lemma to get a bit more insight into how Turing's code works. In particular, the encryption operation in Turing's code *permutes the set of possible messages*. This is stated more precisely in the following corollary.

Corollary 4.6.3. *Suppose p is a prime and k is not a multiple of p. Then the sequence:*

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

is a permutation[3] *of the sequence:*

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

Proof. The sequence of remainders contains $p-1$ numbers. Since $i \cdot k$ is not divisible by p for $i = 1, \ldots p-1$, all these remainders are in the range 1 to $p-1$ by the definition of remainder. Furthermore, the remainders are all different: no two numbers in the range 1 to $p-1$ are congruent modulo p, and by Lemma 4.6.2, $i \cdot k \equiv j \cdot k \pmod{p}$ if and only if $i \equiv j \pmod{p}$. Thus, the sequence of remainders must contain *all* of the numbers from 1 to $p-1$ in some order. ■

[3] A *permutation* of a sequence of elements is a reordering of the elements.

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

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

is a permutation of 1, 2, 3, 4. As long as the Nazis don't know the secret key k, they don't know how the set of possible messages are permuted by the process of encryption and thus they can't read encoded messages.

4.6.3 Fermat's Little Theorem

An alternative approach to finding the inverse of the secret key k in Turing's code (about equally efficient and probably more memorable) is to rely on Fermat's Little Theorem, which is much easier than his famous Last Theorem.

Theorem 4.6.4 (Fermat's Little Theorem)**.** *Suppose p is a prime and k is not a multiple of p. Then:*

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

Proof. We reason as follows:

$$\begin{aligned}
(p-1)! &::= 1 \cdot 2 \cdots (p-1) \\
&= \text{rem}(k,p) \cdot \text{rem}(2k,p) \cdots \text{rem}((p-1)k,p) && \text{(by Cor 4.6.3)} \\
&\equiv k \cdot 2k \cdots (p-1)k \pmod{p} && \text{(by Cor 4.5.2)} \\
&\equiv (p-1)! \cdot k^{p-1} \pmod{p} && \text{(rearranging terms)}
\end{aligned}$$

Now $(p-1)!$ is not a multiple of p because the prime factorizations of $1, 2, \ldots, (p-1)$ contain only primes smaller than p. So by Lemma 4.6.2, we can cancel $(p-1)!$ from the first and last expressions, which proves the claim. ■

Here is how we can find inverses using Fermat's Theorem. Suppose p is a prime and k is not a multiple of p. Then, by Fermat's Theorem, we know that:

$$k^{p-2} \cdot k \equiv 1 \pmod{p}$$

Therefore, k^{p-2} must be a multiplicative inverse of k. For example, suppose that we want the multiplicative inverse of 6 modulo 17. Then we need to compute

$\text{rem}(6^{15}, 17)$, which we can do by successive squaring. All the congruences below hold modulo 17.

$$
\begin{aligned}
6^2 &\equiv 36 \equiv 2 \\
6^4 &\equiv (6^2)^2 \equiv 2^2 \equiv 4 \\
6^8 &\equiv (6^4)^2 \equiv 4^2 \equiv 16 \\
6^{15} &\equiv 6^8 \cdot 6^4 \cdot 6^2 \cdot 6 \equiv 16 \cdot 4 \cdot 2 \cdot 6 \equiv 3
\end{aligned}
$$

Therefore, $\text{rem}(6^{15}, 17) = 3$. Sure enough, 3 is the multiplicative inverse of 6 modulo 17, since:

$$3 \cdot 6 \equiv 1 \pmod{17}$$

In general, if we were working modulo a prime p, finding a multiplicative inverse by trying every value between 1 and $p - 1$ would require about p operations. However, the approach above requires only about **[Illegible]**2(?) $\log p$ operations, which is far better when p is large.

4.6.4 Breaking Turing's Code—Again

The Germans didn't bother to encrypt their weather reports with the highly-secure Enigma system. After all, so what if the Allies learned that there was rain off the south coast of Iceland? But, amazingly, this practice provided the British with a critical edge in the Atlantic naval battle during 1941.

The problem was that some of those weather reports had originally been transmitted using Enigma from U-boats out in the Atlantic. Thus, the British obtained both unencrypted reports and the same reports encrypted with Enigma. By comparing the two, the British were able to determine which key the Germans were using that day and could read all other Enigma-encoded traffic. Today, this would be called a *known-plaintext attack*.

Let's see how a known-plaintext attack would work against Turing's code. Suppose that the Nazis know both m and m^* where:

$$m^* \equiv mk \pmod{p}$$

Now they can compute:

$$\begin{aligned}
m^{p-2} \cdot m^* &= m^{p-2} \cdot \text{rem}(mk, p) && \text{(def. (4.7) of } m^*\text{)} \\
&\equiv m^{p-2} \cdot mk \pmod{p} && \text{(by Cor 4.5.2)} \\
&\equiv m^{p-1} \cdot k \pmod{p} \\
&\equiv k \pmod{p} && \text{(Fermat's Theorem)}
\end{aligned}$$

Now the Nazis have the secret key k and can decrypt any message!

This is a huge vulnerability, so Turing's code has no practical value. Fortunately, Turing got better at cryptography after devising this code; his subsequent deciphering of Enigma messages surely saved thousands of lives, if not the whole of Britain.

4.6.5 Turing Postscript

A few years after the war, Turing's home was robbed. Detectives soon determined that a former homosexual lover of Turing's had conspired in the robbery. So they arrested him—that is, they arrested Alan Turing—because homosexuality was a

British crime punishable by up to two years in prison at that time. Turing was sentenced to a hormonal “treatment” for his homosexuality: he was given estrogen injections. He began to develop breasts.

Three years later, Alan Turing, the founder of computer science, was dead. His mother explained what happened in a biography of her own son. Despite her repeated warnings, Turing carried out chemistry experiments in his own home. Apparently, her worst fear was realized: by working with potassium cyanide while eating an apple, he poisoned himself.

However, Turing remained a puzzle to the very end. His mother was a devoutly religious woman who considered suicide a sin. And, other biographers have pointed out, Turing had previously discussed committing suicide by eating a poisoned apple. Evidently, Alan Turing, who founded computer science and saved his country, took his own life in the end, and in just such a way that his mother could believe it was an accident.

Turing’s last project before he disappeared from public view in 1939 involved the construction of an elaborate mechanical device to test a mathematical conjecture called the Riemann Hypothesis. This conjecture first appeared in a sketchy paper by Bernhard Riemann in 1859 and is now one of the most famous unsolved problem in mathematics.

4.6.6 Problems

Class Problems

Homework Problems

4.7 Arithmetic with an Arbitrary Modulus

Turing's code did not work as he hoped. However, his essential idea—using number theory as the basis for cryptography—succeeded spectacularly in the decades after his death.

In 1977, Ronald Rivest, Adi Shamir, and Leonard Adleman at MIT proposed a highly secure cryptosystem (called **RSA**) based on number theory. Despite decades

The Riemann Hypothesis

The formula for the sum of an infinite geometric series says:

$$1 + x + x^2 + x^3 + \cdots = \frac{1}{1-x}$$

Substituting $x = \frac{1}{2^s}$, $x = \frac{1}{3^s}$, $x = \frac{1}{5^s}$, and so on for each prime number gives a sequence of equations:

$$1 + \frac{1}{2^s} + \frac{1}{2^{2s}} + \frac{1}{2^{3s}} + \cdots = \frac{1}{1-1/2^s}$$

$$1 + \frac{1}{3^s} + \frac{1}{3^{2s}} + \frac{1}{3^{3s}} + \cdots = \frac{1}{1-1/3^s}$$

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

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 of attack, no significant weakness has been found. Moreover, RSA has a major advantage over traditional codes: the sender and receiver of an encrypted message need not meet beforehand to agree on a secret key. Rather, the receiver has both a *secret key*, which she guards closely, and a *public key*, which she distributes as widely as possible. The sender then encrypts his message using her widely-distributed public key. Then she decrypts the received message using her closely-held private key. The use of such a *public key cryptography* system allows you and Amazon, for example, to engage in a secure transaction without meeting up beforehand in a dark alley to exchange a key.

Interestingly, RSA does not operate modulo a prime, as Turing's scheme may have, but rather modulo the product of *two* large primes. Thus, we'll need to know a bit about how arithmetic works modulo a composite number in order to understand RSA. Arithmetic modulo an arbitrary positive integer is really only a little more painful than working modulo a prime—though you may think this is like the doctor saying, "This is only going to hurt a little," before he jams a big needle in your arm.

4.7.1 Relative Primality

First, we need a new definition. Integers a and b are *relatively prime* iff $\gcd(a, b) = 1$. For example, 8 and 15 are relatively prime, since $\gcd(8, 15) = 1$. Note that, except for multiples of p, every integer is relatively prime to a prime number p.

Next we'll need to generalize what we know about arithmetic modulo a prime to work modulo an arbitrary positive integer n. The basic theme is that arithmetic modulo n may be complicated, but the integers *relatively prime* to n remain fairly well-behaved. For example, the proof of Lemma 4.6.1 of an inverse for k modulo p extends to an inverse for k relatively prime to n:

Lemma 4.7.1. *Let n be a positive integer. If k is* relatively prime *to n, then there exists an integer k^{-1} such that:*

$$k \cdot k^{-1} \equiv 1 \pmod{n}$$

As a consequence of this lemma, we can cancel a multiplicative term from both sides of a congruence if that term is relatively prime to the modulus:

Corollary 4.7.2. *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.

The following lemma is the natural generalization of Corollary 4.7.2.

Lemma 4.7.3. *Suppose n is a positive integer and k is relatively prime to n. Let $k_1, \ldots, k_r$ denote all the integers relatively prime to n in the range 1 to $n-1$. Then the sequence:*

$$\operatorname{rem}(k_1 \cdot k, n), \quad \operatorname{rem}(k_2 \cdot k, n), \quad \operatorname{rem}(k_3 \cdot k, n), \qquad \ldots \quad , \operatorname{rem}(k_r \cdot k, n)$$

is a permutation of the sequence:

$$k_1, \quad k_2, \quad \ldots \quad , k_r.$$

Proof. We will show that the remainders in the first sequence are all distinct and are equal to some member of the sequence of k_j's. Since the two sequences have the same length, the first must be a permutation of the second.

First, we show that the remainders in the first sequence are all distinct. Suppose that $\text{rem}(k_i k, n) = \text{rem}(k_j k, n)$. This is equivalent to $k_i k \equiv k_j k \pmod{n}$, which implies $k_i \equiv k_j \pmod{n}$ by Corollary 4.7.2. This, in turn, means that $k_i = k_j$ since both are between 1 and $n - 1$. Thus, none of the remainder terms in the first sequence is equal to any other remainder term.

Next, we show that each remainder in the first sequence equals one of the k_i. By assumption, $\gcd(k_i, n) = 1$ and $\gcd(k, n) = 1$, which means that

$$\begin{aligned}
\gcd(n, \text{rem}(k_i k, n)) &= \gcd(k_i k, n) && \text{(by part (5) of Lemma 4.2.4)} \\
&= 1 && \text{(by part (3) of Lemma 4.2.4).}
\end{aligned}$$

Since $\text{rem}(k_i k, n)$ is in the range from 0 to $n - 1$ by the definition of remainder, and since it is relatively prime to n, it must (by definition of the k_j's) be equal to some k_j. ■

4.7.2 Euler's Theorem

RSA relies heavily on a generalization of Fermat's Theorem known as Euler's Theorem. For both theorems, the exponent of k needed to produce an inverse of k modulo n depends on the number of integers in the set $\{1, 2, \ldots, n-1\}$ that are relatively prime to n. This value is known as *Euler's ϕ function* (a.k.a. *Euler's totient function*) and it is denoted as $\phi(n)$. For example, $\phi(7) = 6$ since 1, 2, 3, 4, 5, and 6 are all relatively prime to 7. Similarly, $\phi(12) = 4$ since 1, 5, 7, and 11 are the only numbers less than 12 that are relatively prime to 12.

If n is prime, then $\phi(n) = n - 1$ since every number less than a prime number is relatively prime to that prime. When n is composite, however, the ϕ function gets a little complicated. The following theorem characterizes the ϕ function for composite n. We won't prove the theorem in its full generality, although we will give a proof for the special case when n is the product of two primes since that is the case that matters for RSA.

Theorem 4.7.4. *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 $k \geq 1$.

Corollary 4.7.5. *Let $p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_j^{\alpha_j}$ be the unique prime factorization of an integer n where $p_1 < p_2 < \cdots < p_j$ for $j \geq 1$. Then*

$$\phi(n) = (p_1^{\alpha_1} - p_1^{\alpha_1 - 1})(p_2^{\alpha_2} - p_2^{\alpha_2 - 1}) \ldots (p_j^{\alpha_j} - p_j^{\alpha_j - 1})$$

For example,

$$\begin{aligned} \phi(300) &= \phi(2^2 \cdot 3 \cdot 5^2) \\ &= (2^2 - 2^1)(3^1 - 3^0)(5^2 - 5^1) \\ &= 2 \cdot 2 \cdot 20 \\ &= 80. \end{aligned}$$

Corollary 4.7.6. *Let $n = pq$ where p and q are different primes. Then $\phi(n) = (p-1)(q-1)$.*

Corollary 4.7.6 follows easily from Corollary 4.7.5 and Theorem 4.7.4, but since Corollary 4.7.6 is important to RSA and we have not provided a proof of Theorem 4.7.4, we will give a direct proof of Corollary 4.7.6 in what follows.

Proof of 4.7.6. Since p and q are prime, any number that is not relatively prime to $n = pq$ must be a multiple of p or a multiple of q. Among the numbers 1, 2, $\ldots$, $pq-1$, there are precisely $q-1$ multiples of p and $p-1$ multiples of q. Since p and q are relatively prime and since the numbers under consideration are less than pq, the $q-1$ multiples of p are different than the $p-1$ multiples of q. Hence,

$$\begin{aligned}\phi(n) &= (pq-1)-(q-1)-(p-1)\\ &= pq-q-p+1\\ &= (p-1)(q-1),\end{aligned}$$

as claimed. ■

We can now prove Euler's Theorem:

Theorem 4.7.7 (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 $0 \le k_i < n$. Then $r = \phi(n)$, by the definition of the function ϕ. The remainder of the proof mirrors the proof of Fermat's Theorem. In particular,

$$\begin{aligned}
&k_1 \cdot k_2 \cdots k_r \\
&\qquad = \text{rem}(k_1 \cdot k, n) \cdot \text{rem}(k_2 \cdot k, n) \cdots \text{rem}(k_r \cdot k, n) && \text{(by Lemma 4.7.3)} \\
&\qquad \equiv (k_1 \cdot k) \cdot (k_2 \cdot k) \cdot \cdots \cdot (k_r \cdot k) \pmod{n} && \text{(by Cor 4.5.2)} \\
&\qquad \equiv (k_1 \cdot k_2 \cdots k_r) \cdot k^r \pmod{n} && \text{(rearranging terms)}
\end{aligned}$$

Part (3) of Lemma 4.2.4. implies that $k_1 \cdot k_2 \cdots k_r$ is relatively prime to n. So by Corollary 4.7.2, we can cancel this product from the first and last expressions. This proves the claim. ■

We can find multiplicative inverses using Euler's theorem as we did with Fermat's theorem: if k is relatively prime to n, then $k^{\phi(n)-1}$ is a multiplicative inverse of k modulo n. However, this approach requires computing $\phi(n)$. Computing $\phi(n)$ (using Corollary 4.7.5) if we know the prime factorization of n. Unfortunately, finding the factors of n can be hard to do when n is large and so the Pulverizer is often the best approach to computing inverses mod n.

4.8 The RSA Algorithm

Here, then, is the *RSA public key encryption scheme*:

Beforehand The receiver creates a public key and a secret key as follows.

1. Generate two distinct primes, p and q. Since they can be used to generate the secret key, they must be kept hidden.

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 $de \equiv 1 \pmod{(p-1)(q-1)}$. This can be done using the Pulverizer.

   The *secret key* is the pair (d, n). This should be kept hidden!

Encoding Given a message m, the sender first checks that $\gcd(m, n) = 1$.[4] The

[4]It would be very bad if $\gcd(m, n)$ equals p or q since then it would be easy for someone to use the

sender then encrypts message m to produce m' using the public key:

$$m' = \text{rem}(m^e, n).$$

Decoding The receiver decrypts message m' back to message m using the secret key:

$$m = \text{rem}((m')^d, n).$$

Why does decoding work? We need to show that the decryption $\text{rem}((m')^d, n)$ is indeed equal to the sender's message m. Since $m' = \text{rem}(m^e, n)$, m' is congruent to m^e modulo n by Corollary 4.5.1. That is,

$$m' \equiv m^e \pmod{n}.$$

By raising both sides to the power d, we obtain the congruence

$$(m')^d \equiv m^{ed} \pmod{n}. \tag{4.8}$$

encoded message to compute the secret key If $\text{gcd}(m, n) = n$, then the encoded message would be 0, which is fairly useless.

The encryption exponent e and the decryption exponent d are chosen such that $de \equiv 1 \pmod{(p-1)(q-1)}$. So, there exists an integer r such that $ed = 1 + r(p-1)(q-1)$. By substituting $1 + r(p-1)(q-1)$ for ed in4.8, we obtain

$$(m')^d \equiv m \cdot m^{r(p-1)(q-1)} \pmod{n}. \tag{4.9}$$

By Euler's Theorem and the assumption that $\gcd(m, n) = 1$, we know that

$$m^{\phi(n)} \equiv 1 \pmod{n}.$$

From Corollary 4.7.6, we know that $\phi(n) = (p-1)(q-1)$. Hence,

$$\begin{aligned}(m')^d &= m \cdot m^{r(p-1)(q-1)} \pmod{n} \\ &= m \cdot 1^r \pmod{n} \\ &= m \pmod{n}.\end{aligned}$$

Hence, the decryption process indeed reproduces the original message m.

Is it hard for someone without the secret key to decrypt th message? No one knows for sure but it is generally believed that if n is a very large number (say, with a thousand digits), then it is difficult to reverse engineer d from e and n. Of course, it is easy to compute d if you know p and q (by using the Pulverizer) but it is not known how to quickly factor n into p and q when n is very large. Maybe with a little more studying of number theory, you will be the first to figure out how to do it. Although, we should warn you that Gauss worked on it for years without a lot to show for his efforts. And if you do figure it out, you might wind up meeting some serious-looking fellows in black suits. . . .

4.8.1 Problems

Practice Problems

Class Problems

Homework Problems

Exam Problems

Part II

Mathematical Data Types

Chapter 5

Sets and Relations

5.1 Sets

EDITING NOTE: 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. ■

Propositions of the sort we've considered so far are good for reasoning about individual statements, but not so good for reasoning about a collection of objects.

Let's first review a couple mathematical tools for grouping objects and then extend our logical language to cope with such collections.

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

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

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

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

The order of elements is not significant, so $\{x, y\}$ and $\{y, x\}$ are the same set written two different ways. Also, any object is, or is not, an element of a given set —there is no notion of an element appearing more than once in a set.[1] So writing $\{x, x\}$ is just indicating the same thing twice, namely, that x is in the set. In particular, $\{x, x\} = \{x\}$.

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

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

5.1.1 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.

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

5.1.2 Comparing and Combining Sets

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

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

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 *set difference* of X and Y (denoted $X - Y$) consists of all elements that are in X, but not in Y. Therefore, $X - Y = \{1\}$ and $Y - X = \{4\}$.

5.1.3 Complement of a Set

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

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

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

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

5.1.4 Power Set

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

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

5.1.5 Set Builder Notation

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

$$A ::= \{n \in \mathbb{N} \mid n \text{ is a prime and } n = 4k + 1 \text{ for some integer } k\}$$

$$B ::= \left\{x \in \mathbb{R} \mid x^3 - 3x + 1 > 0\right\}$$

$$C ::= \left\{a + bi \in \mathbb{C} \mid a^2 + 2b^2 \leq 1\right\}$$

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

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

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

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

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

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

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

$a + bi$ such that:

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

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

5.1.6 Proving Set Equalities

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

Theorem 5.1.1 (*Distributive Law* for Sets)**.** *Let A, B, and C be sets. Then:*

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C) \tag{5.1}$$

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

$$z \in A \cap (B \cup C) \quad \text{iff} \quad z \in (A \cap B) \cup (A \cap C) \tag{5.2}$$

for all z. Now we'll prove (5.2) by a chain of iff's.

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

Lemma 5.1.2. *The propositional formulas*

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

and

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

are equivalent.

Now we have

$$\begin{aligned}
&z \in A \cap (B \cup C) \\
&\quad \text{iff} \quad (z \in A) \text{ AND } (z \in B \cup C) && \text{(def of } \cap\text{)} \\
&\quad \text{iff} \quad (z \in A) \text{ AND } (z \in B \text{ OR } z \in C) && \text{(def of } \cup\text{)} \\
&\quad \text{iff} \quad (z \in A \text{ AND } z \in B) \text{ OR } (z \in A \text{ AND } z \in C) && \text{(Lemma 5.1.2)} \\
&\quad \text{iff} \quad (z \in A \cap B) \text{ OR } (z \in A \cap C) && \text{(def of } \cap\text{)} \\
&\quad \text{iff} \quad z \in (A \cap B) \cup (A \cap C) && \text{(def of } \cup\text{)}
\end{aligned}$$

■

5.1.7 Glossary of Symbols

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

5.1.8 Problems

Homework Problems

5.2 The Logic of Sets

5.2.1 Russell's Paradox

Reasoning naively about sets turns out to be risky. In fact, one of the earliest attempts to come up with precise axioms for sets by a late nineteenth century logican named Gotlob *Frege* was shot down by a three line argument known as *Russell's Paradox*:[2] This was an astonishing blow to efforts to provide an axiomatic foundation for mathematics.

[2]Bertrand *Russell* was a mathematician/logician at Cambridge University at the turn of the Twentieth Century. He reported that when he felt too old to do mathematics, he began to study and write about philosophy, and when he was no longer smart enough to do philosophy, he began writing about politics. He was jailed as a conscientious objector during World War I. For his extensive philosophical and political writing, he won a Nobel Prize for Literature.

Let S be a variable ranging over all sets, and define

$$W ::= \{S \mid S \notin S\}.$$

So by definition,

$$S \in W \text{ iff } S \notin S,$$

for every set S. In particular, we can let S be W, and obtain the contradictory result that

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

A way out of the paradox was clear to Russell and others at the time: *it's unjustified to assume that W is a set*. So the step in the proof where we let S be W has no justification, because S ranges over sets, and W may not be a set. In fact, the paradox implies that W had better not be a set!

But denying that W is a set means we must *reject* the very natural axiom that every mathematically well-defined collection of elements is actually a set. So the problem faced by Frege, Russell and their colleagues was how to specify *which* well-defined collections are sets. Russell and his fellow Cambridge University colleague Whitehead immediately went to work on this problem. They spent a dozen years developing a huge new axiom system in an even huger monograph called *Principia Mathematica*.

5.2.2 The ZFC Axioms for Sets

It's generally agreed that, using some simple logical deduction rules, essentially all of mathematics can be derived from some axioms about sets called the Axioms of Zermelo-Frankel Set Theory with Choice (ZFC).

We're *not* going to be working with these axioms in this course, but we thought you might like to see them –and while you're at it, get some practice reading quantified formulas:

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

$$(\forall z.\ (z \in x \text{ IFF } z \in y)) \text{ IMPLIES } x = y.$$

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

$$\forall x, y.\ \exists u.\ \forall z.\ [z \in u \text{ IFF } (z = x \text{ OR } z = y)]$$

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

$$\forall z.\, \exists u \forall x.\ (\exists y.\ x \in y \text{ AND } y \in z) \text{ IFF } x \in u.$$

Infinity. There is an infinite set. Specifically, there is a nonempty set, x, such that for any set $y \in x$, the set $\{y\}$ is also a member of x.

EDITING NOTE:

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:

$$\forall x.\ \exists p.\ \forall u.\ u \subseteq x \text{ IFF } u \in p.$$

Replacement. Suppose a formula, ϕ, of set theory defines the graph of a function, that is,

$$\forall x, y, z. [\phi(x, y) \text{ AND } \phi(x, z)] \text{ IMPLIES } y = z.$$

Then the image of any set, s, under that function is also a set, t. Namely,

$$\forall s\, \exists t\, \forall y. [\exists x. \phi(x, y) \text{ IFF } y \in t].$$

Foundation. There cannot be an infinite sequence

$$\cdots \in x_n \in \cdots \in x_1 \in x_0$$

of sets each of which is a member of the previous one. This is equivalent to saying every nonempty set has a "member-minimal" element. Namely, define

$$\text{member-minimal}(m, x) ::= [m \in x \text{ AND } \forall y \in x. y \notin m].$$

Then the Foundation axiom is

$$\forall x. x \neq \emptyset \text{ IMPLIES } \exists m. \text{member-minimal}(m, x).$$

EDITING NOTE: If well-founded posets are defined, then rephrase Foundation as *The $\in$ relation on sets is well-founded.* ■

Choice. Given a set, s, whose members are nonempty sets no two of which have any element in common, then there is a set, c, consisting of exactly one element from each set in s.

EDITING NOTE:

$$\begin{aligned}
\exists y \forall z \forall w \quad & ((z \in w \text{ AND } w \in x) \text{IMPLIES} \\
& \exists v \exists u (\exists t ((u \in w \text{AND} \quad w \in t) \quad \text{AND} (u \in t \text{ AND } t \in y)) \\
& \qquad \text{IFF} u = v))
\end{aligned}$$

■

5.2.3 Avoiding Russell's Paradox

These modern ZFC axioms for set theory are much simpler than the system Russell and Whitehead first came up with to avoid paradox. In fact, the ZFC axioms are as simple and intuitive as Frege's original axioms, with one technical addition: the

Foundation axiom. Foundation captures the intuitive idea that sets must be built up from "simpler" sets in certain standard ways. And in particular, Foundation implies that no set is ever a member of itself. So the modern resolution of Russell's paradox goes as follows: since $S \notin S$ for all sets S, it follows that W, defined above, contains every set. This means W can't be a set —or it would be a member of itself.

5.2.4 Does All This Really Work?

So this is where mainstream mathematics stands today: there is a handful of ZFC axioms from which virtually everything else in mathematics can be logically derived. This sounds like a rosy situation, but there are several dark clouds, suggesting that the essence of truth in mathematics is not completely resolved.

- The ZFC axioms weren't etched in stone by God. Instead, they were mostly made up by some guy named Zermelo. Probably some days he forgot his house keys.

So maybe Zermelo, just like Frege, didn't get his axioms right and will be shot down by some successor to Russell who will use Zermelo's axioms to prove a proposition P and its negation NOT P. Then math would be broken. This sounds crazy, but after all, it has happened before.

In fact, while there is broad agreement that the ZFC axioms are capable of proving all of standard mathematics, the axioms have some further consequences that sound paradoxical. For example, the Banach-Tarski Theorem says that, as a consequence of the Axiom of Choice, a solid ball can be divided into six pieces and then the pieces can be rigidly rearranged to give *two* solid balls, each the same size as the original!

- Georg *Cantor* was a contemporary of Frege and Russell who first developed the theory of infinite sizes (because he thought he needed it in his study of Fourier series). Cantor raised the question whether there is a set whose size is strictly between the "smallest[3]" infinite set, $\mathbb{N}$, and $\mathcal{P}(\mathbb{N})$; he guessed not:

[3]See Problem **??**

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 modern Set Theory — discovered in part by Gödel in the 1930's and Paul Cohen in the 1960's — namely, the ZFC axioms are not sufficient to settle the Continuum Hypothesis: there are two collections of sets, each obeying the laws of ZFC, and in one collection the Continuum Hypothesis is true, and in the other it is false. So settling the Continuum Hypothesis requires a new understanding of what Sets should be to arrive at persuasive new axioms that extend ZFC and are strong enough to determine the truth of the Continuum Hypothesis one way or the other.

- But even if we use more or different axioms about sets, there are some unavoidable problems. In the 1930's, Gödel proved that, assuming that an axiom system like ZFC is consistent —meaning you can't prove both P and

NOT(P) for any proposition, P —then the very proposition that the system is consistent (which is not too hard to express as a logical formula) cannot be proved in the system. In other words, no consistent system is strong enough to verify itself.

5.3 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.
- Texts differ on notation for the *empty sequence*; we use λ for the empty sequence.

The product operation is one link between sets and sequences. A *product of sets*, $S_1 \times S_2 \times \cdots \times S_n$, is a new set consisting of all sequences where the first component is drawn from S_1, the second from S_2, and so forth. For example, $\mathbb{N} \times \{a, b\}$ is the set of all pairs whose first element is a nonnegative integer and whose second element is an a or a b:

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

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:

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

5.4 Functions

A *function* assigns an element of one set, called the *domain*, to elements of another set, called the *codomain*. The notation

$$f : A \to B$$

indicates that f is a function with domain, A, and codomain, B. The familiar notation "$f(a) = b$" indicates that f assigns the element $b \in B$ to a. Here b would be called the *value* of f at *argument* a.

Functions are often defined by formulas as in:

$$f_1(x) ::= \frac{1}{x^2}$$

where x is a real-valued variable, or

$$f_2(y, z) ::= y\texttt{10}yz$$

where y and z range over binary strings, or

$$f_3(x, n) ::= \text{ the pair } (n, x)$$

where n ranges over the nonnegative integers.

A function with a finite domain could be specified by a table that shows the value of the function at each element of the domain. For example, a function $f_4(P, Q)$ where P and Q are propositional variables is specified by:

P	Q	$f_4(P, Q)$
T	**T**	**T**
T	**F**	**F**
F	**T**	**T**
F	**F**	**T**

Notice that f_4 could also have been described by a formula:

$$f_4(P, Q) ::= [P \text{ IMPLIES } Q].$$

A function might also be defined by a procedure for computing its value at any element of its domain, or by some other kind of specification. For example, define $f_5(y)$ to be the length of a left to right search of the bits in the binary string y until a `1` appears, so

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

Notice that f_5 does not assign a value to any string of just `0`'s. This illustrates an important fact about functions: they need not assign a value to every element in the domain. In fact this came up in our first example $f_1(x) = 1/x^2$, which does not assign a value to 0. So in general, functions may be *partial functions*, meaning that there may be domain elements for which the function is not defined. If a function is defined on every element of its domain, it is called a *total function*.

It's often useful to find the set of values a function takes when applied to the elements in *a set* of arguments. So if $f : A \to B$, and S is a subset of A, we define $f(S)$ to be the set of all the values that f takes when it is applied to elements of S. That is,

$$f(S) ::= \{b \in B \mid f(s) = b \text{ for some } s \in S\}.$$

For example, if we let $[r, s]$ denote the interval from r to s on the real line, then $f_1([1, 2]) = [1/4, 1]$.

For another example, let's take the "search for a `1`" function, f_5. If we let X be the set of binary words which start with an even number of `0`'s followed by a `1`, then $f_5(X)$ would be the odd nonnegative integers.

Applying f to a set, S, of arguments is referred to as "applying f pointwise to S", and the set $f(S)$ is referred to as the *image* of S under f.[4] The set of values that arise from applying f to all possible arguments is called the *range* of f. That is,

$$\text{range}(f) ::= f(\text{domain}(f)).$$

Some authors refer to the codomain as the range of a function, but they shouldn't. The distinction between the range and codomain will be important in Sections 5.5.4 and 5.6 when we relate sizes of sets to properties of functions between them.

5.4.1 Function Composition

Doing things step by step is a universal idea. Taking a walk is a literal example, but so is cooking from a recipe, executing a computer program, evaluating a formula, and recovering from substance abuse.

[4]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.

Abstractly, taking a step amounts to applying a function, and going step by step corresponds to applying functions one after the other. This is captured by the operation of *composing* functions. Composing the functions f and g means that first f applied is to some argument, x, to produce $f(x)$, and then g is applied to that result to produce $g(f(x))$.

Definition 5.4.1. For functions $f : A \to B$ and $g : B \to C$, the *composition*, $g \circ f$, of g with f is defined to be the function $h : A \to C$ defined by the rule:

$$h(x) ::= (g \circ f)(x) ::= g(f(x)),$$

for all $x \in A$.

Function composition is familiar as a basic concept from elementary calculus, and it plays an equally basic role in discrete mathematics.

5.5 Relations

Relations are another fundamental mathematical data type. Equality and "less-than" are very familiar examples of mathematical relations. These are called *binary* *relations* because they apply to a pair (a, b) of objects; the equality relation holds for the pair when $a = b$, and less-than holds when a and b are real numbers and $a < b$.

In this section we'll define some basic vocabulary and properties of binary relations.

5.5.1 Binary Relations and Functions

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

Definition 5.5.1. A *binary relation*, R, consists of a set, A, called the *domain* of R, a set, B, called the *codomain* of R, and a subset of $A \times B$ called the *graph of* R.

Notice that Definition 5.5.1 is exactly the same as the definition in Section 5.4 of a *function*, except that it doesn't require the functional condition that, for each domain element, a, there is *at most* one pair in the graph whose first coordinate is a. So a function is a special case of a binary relation.

A relation whose domain is A and codomain is B is said to be "between A and

B", or "from A to B." When the domain and codomain are the same set, A, we simply say the relation is "on A." It's common to use infix notation "$a \ R \ b$" to mean that the pair (a, b) is in the graph of R.

For example, we can define an "in-charge of" relation, T, for MIT in Spring '10 to have domain equal to the set, F, of names of the faculty and codomain equal to all the set, N, of subject numbers in the current catalogue. The graph of T contains precisely the pairs of the form

$$(\langle\text{instructor-name}\rangle, \langle\text{subject-num}\rangle)$$

such that the faculty member named ⟨instructor-name⟩ is in charge of the subject with number ⟨subject-num⟩ in Spring '10. So graph (T) contains pairs like

```
(A. R. Meyer,  6.042),
(A. R. Meyer,  18.062),
(A. R. Meyer,  6.844),
(T. Leighton,  6.042),
(T. Leighton,  18.062),
(G, Freeman,   6.011),
(G. Freeman,   6.881)
(G. Freeman,   6.882)
(G. Freeman,   6.UAT)
(T. Eng,       6.UAT)
(J. Guttag,    6.00)
     ⋮
```

This is a surprisingly complicated relation: Meyer is in charge of subjects with three numbers. Leighton is also in charge of subjects with two of these three numbers —because the same subject, Mathematics for Computer Science, has two numbers: 6.042 and 18.062, and Meyer and Leighton are co-in-charge of the subject. Freeman is in-charge of even more subjects numbers (around 20), since as Department Education Officer, he is in charge of whole blocks of special subject numbers. Some subjects, like 6.844 and 6.00 have only one person in-charge. Some faculty, like Guttag, are in charge of only one subject number, and no one else is co-in-charge of his subject, 6.00.

Some subjects in the codomain, N, do not appear in the list —that is, they are not an element of any of the pairs in the graph of T; these are the Fall term only subjects. Similarly, there are faculty in the domain, F, who do not appear in the list because all their in-charge subjects are Fall term only.

5.5.2 Relational Images

The idea of the image of a set under a function extends directly to relations.

Definition 5.5.2. The *image* of a set, Y, under a relation, R, written $R(Y)$, is the set of elements of the codomain, B, of R that are related to some element in Y, namely,

$$R(Y) ::= \{b \in B \mid yRb \text{ for some } y \in Y\}.$$

For example, to find the subject numbers that Meyer is in charge of in Spring '09, we can look for all the pairs of the form

$$(\text{A. Meyer}, \langle\text{subject-number}\rangle)$$

in the graph of the teaching relation, T, and then just list the right hand sides of these pairs. These righthand sides are exactly the image T(A. Meyer), which happens to be $\{6.042, 18.062, 6.844\}$. Similarly, to find the subject numbers that either Freeman or Eng are in charge of, we can collect all the pairs in T of the form

$$(\text{G. Freeman}, \langle\text{subject-number}\rangle)$$

or

$$(\text{T. Eng}, \langle\text{subject-number}\rangle);$$

and list their right hand sides. These right hand sides are exactly the image $T(\{\text{G. Freeman, T. Eng}\}$.

So the partial list of pairs in T given above implies that

$$\{\texttt{6.011, 6.881, 6.882, 6.UAT}\} \subseteq T(\{\text{G. Freeman, T. Eng}\}\,.$$

Finally, since the domain, F, is the set of all in-charge faculty, $T(F)$ is exactly the set of *all* Spring '09 subjects being taught.

5.5.3 Inverse Relations and Images

Definition 5.5.3. The *inverse*, R^{-1} of a relation $R : A \to B$ is the relation from B to A defined by the rule

$$bR^{-1}a \text{ IFF } a\ R\ B.$$

The image of a set under the relation, R^{-1}, is called the *inverse image* of the set. That is, the inverse image of a set, X, under the relation, R, is $R^{-1}(X)$.

Continuing with the in-charge example above, we can find the faculty in charge of 6.UAT in Spring '10 can be found by taking the pairs of the form

$$(\langle\text{instructor-name}\rangle\,, 6.UAT)$$

in the graph of the teaching relation, T, and then just listing the left hand sides of these pairs; these turn out to be just Eng and Freeman. These left hand sides are exactly the inverse image of {6.UAT} under T.

Now let D be the set of introductory course 6 subject numbers. These are the subject numbers that start with *6.0*. Now we can likewise find out all the instructors who were in-charge of introductory course 6 subjects in Spring '09, by taking all the pairs of the form $(\langle\text{instructor-name}\rangle, 6.0\dots)$ and list the left hand sides of these pairs. These left hand sides are exactly the inverse image of of D under T. From the part of the graph of T shown above, we can see that

$$\{\text{Meyer, Leighton, Freeman, Guttag}\} \subseteq T^{-1}(D).$$

That is, Meyer, Leighton, Freeman, and Guttag were among the instructors in charge of introductory subjects in Spring '10. Finally, the inverse image under T of the set, N, of all subject numbers is the set of all instructors who were in charge of a Spring '09 subject.

It gets interesting when we write composite expressions mixing images, inverse images and set operations. For example, $T(T^{-1}(D))$ is the set of Spring '09 subjects that have an instructor in charge who also is in in charge of an introductory subject. So $T(T^{-1}(D)) - D$ are the advanced subjects with someone in-charge who is also in-charge of an introductory subject. Similarly, $T^{-1}(D) \cap T^{-1}(N - D)$ is the set of faculty in charge of both an introductory *and* an advanced subject in Spring '09.

5.5.4 Surjective and Injective Relations

There are a few properties of relations that will be useful when we take up the topic of counting because they imply certain relations between the *sizes* of domains and codomains. We say a binary relation $R : A \to B$ is:

- *surjective* when every element of B is mapped to *at least once*; more concisely, R is surjective iff $R(A) = B$.
- *total* when every element of A is assigned to some element of B; more concisely, R is total iff $A = R^{-1}(B)$.
- *injective* if every element of B is mapped to *at most once*, and

- *bijective* if R is total, surjective, and injective *function*.[5]

Note that this definition of R being total agrees with the definition in Section 5.4 when R is a function.

If R is a binary relation from A to B, we define $R(A)$ to to be the *range* of R. So a relation is surjective iff its range equals its codomain. Again, in the case that R is a function, these definitions of "range" and "total" agree with the definitions in Section 5.4.

5.5.5 Relation Diagrams

We can explain all these properties of a relation $R : A \to B$ in terms of a diagram where all the elements of the domain, A, appear in one column (a very long one if A is infinite) and all the elements of the codomain, B, appear in another column, and we draw an arrow from a point a in the first column to a point b in the sec-

[5]These words "surjective," "injective," and "bijective" are not very memorable. Some authors use the possibly more memorable phrases *onto* for surjective, *one-to-one* for injective, and *exact correspondence* for bijective.

ond column when a is related to b by R. For example, here are diagrams for two functions:

$$\begin{tikzcd}
A & B \\
\text{a} \arrow[r] & 1 \\
\text{b} \arrow[dr] & 2 \\
\text{c} \arrow[dr] & 3 \\
\text{d} \arrow[uur] & 4 \\
\text{e} \arrow[uur] &
\end{tikzcd}
\qquad
\begin{tikzcd}
A & B \\
\text{a} \arrow[r] & 1 \\
\text{b} \arrow[dr] & 2 \\
\text{c} \arrow[ddr] & 3 \\
\text{d} \arrow[uur] & 4 \\
 & 5
\end{tikzcd}$$

Here is what the definitions say about such pictures:

- "R is a function" means that every point in the domain column, A, has *at most one arrow out of it*.

- "R is total" means that *every* point in the A column has *at least one arrow out of it*. So if R is a function, being total really means every point in the A column has *exactly one arrow out of it*.

- "R is surjective" means that *every* point in the codomain column, B, has *at least one arrow into it*.

- "R is injective" means that every point in the codomain column, B, has *at most one arrow into it*.

- "R is bijective" means that *every* point in the A column has exactly one arrow out of it, and *every* point in the B column has exactly one arrow into it.

So in the diagrams above, the relation on the left is a total, surjective function (every element in the A column has exactly one arrow out, and every element in the B column has at least one arrow in), but not injective (element 3 has two arrows going into it). The relation on the right is a total, injective function (every element in the A column has exactly one arrow out, and every element in the B column has at most one arrow in), but not surjective (element 4 has no arrow going into it).

Notice that the arrows in a diagram for R precisely correspond to the pairs in the graph of R. But graph (R) does not determine by itself whether R is total or surjective; we also need to know what the domain is to determine if R is total, and we need to know the codomain to tell if it's surjective.

Example 5.5.4. The function defined by the formula $1/x^2$ is total if its domain is $\mathbb{R}^+$ but partial if its domain is some set of real numbers including 0. It is bijective if its domain and codomain are both $\mathbb{R}^+$, but neither injective nor surjective if its domain and codomain are both $\mathbb{R}$.

5.6 Cardinality

5.6.1 Mappings and Cardinality

The relational properties in Section 5.5 are useful in figuring out the relative sizes of domains and codomains.

If A is a finite set, we let $|A|$ be the number of elements in A. A finite set may have no elements (the empty set), or one element, or two elements,...or any non-negative integer number of elements.

Now suppose $R : A \to B$ is a function. Then every arrow in the diagram for R comes from exactly one element of A, so the number of arrows is at most the number of elements in A. That is, if R is a function, then

$$|A| \geq \#\text{arrows}.$$

Similarly, if R is surjective, then every element of B has an arrow into it, so there must be at least as many arrows in the diagram as the size of B. That is,

$$\#\text{arrows} \geq |B|.$$

Combining these inequalities implies that if R is a surjective function, then $|A| \geq |B|$. In short, if we write $A \text{ surj } B$ to mean that there is a surjective function from A to B, then we've just proved a lemma: if $A \text{ surj } B$, then $|A| \geq |B|$. The following definition and lemma lists this statement and three similar rules relating domain and codomain size to relational properties.

Definition 5.6.1. Let A, B be (not necessarily finite) sets. Then

1. $A \text{ surj } B$ iff there is a surjective *function* from A to B.
2. $A \text{ inj } B$ iff there is a total injective *relation* from A to B.
3. $A \text{ bij } B$ iff there is a bijection from A to B.
4. $A \text{ strict } B$ iff $A \text{ surj } B$, but not $B \text{ surj } A$.

Lemma 5.6.2. *[Mapping Rules] Let A and B be finite sets.*

1. *If $A \text{ surj } B$, then $|A| \geq |B|$.*

2. *If* A inj B*, then* $|A| \leq |B|$.

3. *If* R bij B*, then* $|A| = |B|$.

4. *If* R strict B*, then* $|A| > |B|$.

Mapping rule 2. can be explained by the same kind of "arrow reasoning" we used for rule 1. Rules 3. and 4. are immediate consequences of these first two mapping rules.

5.6.2 The sizes of infinite sets

Mapping Rule 1 has a converse: if the size of a finite set, A, is greater than or equal to the size of another finite set, B, then it's always possible to define a surjective function from A to B. In fact, the surjection can be a total function. To see how this works, suppose for example that

$$A = \{a_0, a_1, a_2, a_3, a_4, a_5\}$$

$$B = \{b_0, b_1, b_2, b_3\}.$$

Then define a total function $f : A \to B$ by the rules

$$f(a_0) ::= b_0,\ f(a_1) ::= b_1,\ f(a_2) ::= b_2,\ f(a_3) = f(a_4) = f(a_5) ::= b_3.$$

EDITING NOTE:

$$f(a_i) ::= b_{\min(i,3)},$$

for $i = 0, \dots, 5$. Since $5 \geq 3$, this f is a surjection. ■

In fact, if A and B are finite sets of the same size, then we could also define a bijection from A to B by this method.

In short, we have figured out if A and B are finite sets, then $|A| \geq |B|$ *if and only if* $A \text{ surj } B$, and similar iff's hold for all the other Mapping Rules:

Lemma 5.6.3. *For* finite *sets, A, B,*

$$\begin{aligned}
|A| \geq |B| &\quad \textit{iff} \quad A \text{ surj } B, \\
|A| \leq |B| &\quad \textit{iff} \quad A \text{ inj } B, \\
|A| = |B| &\quad \textit{iff} \quad A \text{ bij } B, \\
|A| > |B| &\quad \textit{iff} \quad A \text{ strict } B.
\end{aligned}$$

This lemma suggests a way to generalize size comparisons to infinite sets, namely, we can think of the relation surj as an "*at least as big as*" relation between sets, even if they are infinite. Similarly, the relation bij can be regarded as a "*same size*" relation between (possibly infinite) sets, and strict can be thought of as a "*strictly bigger* than" relation between sets.

Warning: We haven't, and won't, define what the "size" of an infinite is. The definition of infinite "sizes" is cumbersome and technical, and we can get by just fine without it. All we need are the "as big as" and "same size" relations, surj and bij, between sets.

But there's something else to watch out for. We've referred to surj as an "as big as" relation and bij as a "same size" relation on sets. Of course most of the "as big as" and "same size" properties of surj and bij on finite sets do carry over to infinite sets, but *some important ones don't* —as we're about to show. So you have to be careful: don't assume that surj has any particular "as big as" property on *infinite* sets until it's been proved.

Let's begin with some familiar properties of the "as big as" and "same size" relations on finite sets that do carry over exactly to infinite sets:

Lemma 5.6.4. *For any sets, A, B, C,*

1. $A \text{ surj } B$ *and* $B \text{ surj } C$, *implies* $A \text{ surj } C$.

2. $A \text{ bij } B$ *and* $B \text{ bij } C$, *implies* $A \text{ bij } C$.

3. $A \text{ bij } B$ *implies* $B \text{ bij } A$.

Lemma 5.6.4.1 and 5.6.4.2 follow immediately from the fact that compositions of surjections are surjections, and likewise for bijections, and Lemma 5.6.4.3 follows from the fact that the inverse of a bijection is a bijection. We'll leave a proof of these facts to Problem **??**.

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

Theorem 5.6.5 (Schröder-Bernstein)**.** *For any sets A, B, if A surj B and B surj A, then A bij B.*

That is, the Schröder-Bernstein Theorem says that if A is at least as big as B and conversely, B is at least as big as A, then A is the same size as B. Phrased this way, you might be tempted to take this theorem for granted, but that would be a mistake. For infinite sets A and B, the Schröder-Bernstein Theorem is actually pretty technical. Just because there is a surjective function $f : A \to B$ —which need not be a bijection —and a surjective function $g : B \to A$ —which also need not be a bijection —it's not at all clear that there must be a bijection $e : A \to B$. The idea is to construct e from parts of both f and g. We'll leave the actual construction to Problem **??**.

5.6.3 Infinity is different

A basic property of finite sets that does *not* carry over to infinite sets is that adding something new makes a set bigger. That is, if A is a finite set and $b \notin A$, then $|A \cup \{b\}| = |A| + 1$, and so A and $A \cup \{b\}$ are not the same size. But if A is infinite, then these two sets *are* the same size!

Lemma 5.6.6. *Let A be a set and $b \notin A$. Then A is infinite iff A bij $A \cup \{b\}$.*

Proof. Since A is *not* the same size as $A \cup \{b\}$ when A is finite, we only have to show that $A \cup \{b\}$ *is* the same size as A when A is infinite.

That is, we have to find a bijection between $A \cup \{b\}$ and A when A is infinite. Here's how: since A is infinite, it certainly has at least one element; call it a_0. But since A is infinite, it has at least two elements, and one of them must not be equal to a_0; call this new element a_1. But since A is infinite, it has at least three elements, one of which must not equal a_0 or a_1; call this new element a_2. Continuing in the way, we conclude that there is an infinite sequence $a_0, a_1, a_2, \ldots, a_n, \ldots$ of different elements of A. Now it's easy to define a bijection $e : A \cup \{b\} \to A$:

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

■

A set, C, is *countable* iff its elements can be listed in order, that is, the distinct elements is A are precisely

$$c_0, c_1, \dots, c_n, \dots.$$

This means that if we defined a function, f, on the nonnegative integers by the rule that $f(i) ::= c_i$, then f would be a bijection from $\mathbb{N}$ to C. More formally,

Definition 5.6.7. A set, C, is *countably infinite* iff $\mathbb{N}$ bij C. A set is *countable* iff it is finite or countably infinite.

A small modification[6] of the proof of Lemma 5.6.6 shows that countably infinite sets are the "smallest" infinite sets, namely, if A is a countably infinite set, then

[6]See Problem **??**

A surj $\mathbb{N}$.

Since adding one new element to an infinite set doesn't change its size, it's obvious that neither will adding any *finite* number of elements. It's a common mistake to think that this proves that you can throw in countably infinitely many new elements. But just because it's ok to do something any finite number of times doesn't make it OK to do an infinite number of times. For example, starting from 3, you can add 1 any finite number of times and the result will be some integer greater than or equal to 3. But if you add add 1 a countably infinite number of times, you don't get an integer at all.

It turns out you really can add a countably infinite number of new elements to a countable set and still wind up with just a countably infinite set, but another argument is needed to prove this:

Lemma 5.6.8. *If A and B are countable sets, then so is $A \cup B$.*

Proof. Suppose the list of distinct elements of A is $a_0, a_1, \ldots$ and the list of B is

$b_0, b_1, \ldots$. Then a list of all the elements in $A \cup B$ is just

$$a_0, b_0, a_1, b_1, \ldots a_n, b_n, \ldots. \tag{5.3}$$

Of course this list will contain duplicates if A and B have elements in common, but then deleting all but the first occurrences of each element in list (5.3) leaves a list of all the distinct elements of A and B. ■

5.6.4 Power sets are strictly bigger

It turns out that the ideas behind Russell's Paradox, which caused so much trouble for the early efforts to formulate Set Theory, also lead to a correct and astonishing fact discovered by Georg Cantor in the late nineteenth century: infinite sets are *not all the same size*.

In particular,

Theorem 5.6.9. *For any set, A, the power set, $\mathcal{P}(A)$, is strictly bigger than A.*

Proof. First of all, $\mathcal{P}(A)$ is as big as A: for example, the partial function $f : \mathcal{P}(A) \to A$, where $f(\{a\}) ::= a$ for $a \in A$ and f is only defined on one-element sets, is a surjection.

To show that $\mathcal{P}(A)$ is strictly bigger than A, we have to show that if g is a function from A to $\mathcal{P}(A)$, then g is not a surjection. So, mimicking Russell's Paradox, define

$$A_g ::= \{a \in A \mid a \notin g(a)\}.$$

Now A_g is a well-defined subset of A, which means it is a member of $\mathcal{P}(A)$. But A_g can't be in the range of g, because if it were, we would have

$$A_g = g(a_0)$$

for some $a_0 \in A$, so by definition of A_g,

$$a \in g(a_0) \quad \text{iff} \quad a \in A_g \quad \text{iff} \quad a \notin g(a)$$

for all $a \in A$. Now letting $a = a_0$ yields the contradiction

$$a_0 \in g(a_0) \quad \text{iff} \quad a_0 \notin g(a_0).$$

So g is not a surjection, because there is an element in the power set of A, namely the set A_g, that is not in the range of g. ■

Larger Infinities

There are lots of different sizes of infinite sets. For example, starting with the infinite set, $\mathbb{N}$, of nonnegative integers, we can build the infinite sequence of sets

$$\mathbb{N},\ \mathcal{P}(\mathbb{N}),\ \mathcal{P}(\mathcal{P}(\mathbb{N})),\ \mathcal{P}(\mathcal{P}(\mathcal{P}(\mathbb{N}))),\ \ldots.$$

By Theorem 5.6.9, each of these sets is strictly bigger than all the preceding ones. But that's not all: the union of all the sets in the sequence is strictly bigger than each set in the sequence (see Problem **??**). In this way you can keep going, building still bigger infinities.

So there is an endless variety of different size infinities.

5.7 Infinities in Computer Science

We've run into a lot of computer science students who wonder why they should care about infinite sets. They point out that any data set in a computer memory is limited by the size of memory, and there is a finite limit on the possible size of computer memory for the simple reason that the universe is (or at least appears to be) finite.

The problem with this argument is that universe-size bounds on data items are so big and uncertain (the universe seems to be getting bigger all the time), that it's simply not helpful to make use of such bounds. For example, by this argument the physical sciences shouldn't assume that measurements might yield arbitrary real numbers, because there can only be a finite number of finite measurements in a universe with a finite lifetime. What do you think scientific theories would look like without using the infinite set of real numbers?

Similarly, in computer science it simply isn't plausible that writing a program to add nonnegative integers with up to as many digits as, say, the stars in the sky (billions of galaxies each with billions of stars), would be any different than writing a program that would add any two integers no matter how many digits they had.

That's why basic programming data types like integers or strings, for example, can be defined without imposing any bound on the sizes of data items. Each datum of type `string` has only a finite number of letters, but there are an infinite number of data items of type `string`. When we then consider string procedures of type `string-->string`, not only are there an infinite number of such procedures, but each procedure generally behaves differently on different inputs, so that a single `string-->string` procedure may embody an infinite number of behaviors. In short, an educated computer scientist can't get around having to cope with infinite sets.

On the other hand, the more exotic theory of different size infinities and continuum hypotheses rarely comes up in mainstream mathematics, and it hardly comes up at all in computer science, where the focus is mainly on finite sets, and occasionallly on countable sets. In practice, only logicians and set theorists have to worry about collections that are too big to be sets. In fact, at the end of the 19th century, the general mathematical community doubted the relevance of what they called "Cantor's paradise" of unfamiliar sets of arbitrary infinite size. So if the romance of really big infinities doesn't appeal to you, be assured that not knowing about them won't lower your professional abilities as a computer scientist.

Yet the idea behind Russell's paradox and Cantor's proof embodies the simplest form of what is known as a "diagonal argument." Diagonal arguments are used to prove many fundamental results about the limitations of computation, such as the undecidability of the Halting Problem for programs (see Problem **??**) and the inherent, unavoidable, inefficiency (exponential time or worse) of procedures for other computational problems. So computer scientists do need to study diagonal arguments in order to understand the logical limits of computation.

5.7.1 Problems

Practice Problems

Class Problems

EDITING NOTE: Add problem that the 4^n time-bounded halting problem requires time 2^n. ■

Homework Problems

Chapter 6

Recursive Data Types

Recursive data types play a central role in programming. From a mathematical point of view, recursive data types are what induction is about. Recursive data types are specified by *recursive definitions* that say how to build something from its parts. These definitions have two parts:

- **Base case(s)** that don't depend on anything else.
- **Constructor case(s)** that depend on previous cases.

6.1 Strings of Brackets

Let brkts be the set of all strings of square brackets. For example, the following two strings are in brkts:

$$[\,]\,]\,[\,[\,[\,[\,[\,]\,] \quad \text{and} \quad [\,[\,[\,]\,]\,[\,]\,]\,[\,] \tag{6.1}$$

Since we're just starting to study recursive data, just for practice we'll formulate brkts as a recursive data type,

Definition 6.1.1. The data type, brkts, of strings of brackets is defined recursively:

- **Base case:** The *empty string*, λ, is in brkts.
- **Constructor case:** If $s \in$ brkts, then $s]$ and $s[$ are in brkts.

Here we're writing $s]$ to indicate the string that is the sequence of brackets (if any) in the string s, followed by a right bracket; similarly for $s[$.

A string, $s \in$ brkts, is called a *matched string* if its brackets "match up" in the usual way. For example, the left hand string above is not matched because its second right bracket does not have a matching left bracket. The string on the right is matched.

We're going to examine several different ways to define and prove properties of matched strings using recursively defined sets and functions. These properties are pretty straighforward, and you might wonder whether they have any particular relevance in computer scientist —other than as a nonnumerical example of recursion. The honest answer is "not much relevance, *any more*." The reason for this is one of the great successes of computer science.

Expression Parsing

During the early development of computer science in the 1950's and 60's, creation of effective programming language compilers was a central concern. A key aspect in processing a program for compilation was expression parsing. The problem was to take in an expression like

$$x + y * z^2 \div y + 7$$

and *put in* the brackets that determined how it should be evaluated —should it be

$$[[x + y] * z^2 \div y] + 7, \text{ or,}$$

$$x + [y * z^2 \div [y + 7]], \text{ or,}$$

$$[x + [y * z^2]] \div [y + 7],$$

or ...?

The Turing award (the "Nobel Prize" of computer science) was ultimately bestowed on Robert Floyd, for, among other things, being discoverer of a simple program that would insert the brackets properly.

One precise way to determine if a string is matched is to start with 0 and read the string from left to right, adding 1 to the count for each left bracket and subtracting 1 from the count for each right bracket. For example, here are the counts for the two strings above

$$\begin{array}{ccccccccccccc}
 & [&] &] & [& [& [& [& [&] &] &] &] \\
0 & 1 & 0 & -1 & 0 & 1 & 2 & 3 & 4 & 3 & 2 & 1 & 0
\end{array}$$

$$\begin{array}{ccccccccccc}
 & [& [& [&] &] & [&] &] & [&] \\
0 & 1 & 2 & 3 & 2 & 1 & 2 & 1 & 0 & 1 & 0
\end{array}$$

A string has a *good count* if its running count never goes negative and ends with 0. So the second string above has a good count, but the first one does not because its count went negative at the third step.

Definition 6.1.2. Let

$$\text{GoodCount} ::= \{s \in \texttt{brkts} \mid s \text{ has a good count}\}\,.$$

The matched strings can now be characterized precisely as this set of strings with good counts. But it turns out to be really useful to characterize the matched strings in another way as well, namely, as a recursive data type:

Definition 6.1.3. Recursively define the set, RecMatch, of strings as follows:

- **Base case:** $\lambda \in$ RecMatch.

- **Constructor case:** If $s, t \in$ RecMatch, then

$$[\, s \,]\, t \in \text{RecMatch}.$$

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

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

$$[\,\lambda]\,\lambda = [\,] \in \text{RecMatch}$$

by the Constructor case. So now,

$$[\,\lambda]\,[\,] = [\,]\,[\,] \in \text{RecMatch} \qquad (\text{letting } s = \lambda, t = [\,])$$

$$[\,[\,]\,]\,\lambda = [\,[\,]\,] \in \text{RecMatch} \qquad (\text{letting } s = [\,], t = \lambda)$$

$$[\,[\,]\,]\,[\,] \in \text{RecMatch} \qquad (\text{letting } s = [\,], t = [\,])$$

are also strings in RecMatch by repeated applications of the Constructor case. If you haven't seen this kind of definition before, you should try continuing this example to verify that [[[ ]] [ ]] [ ] $\in$ RecMatch

Given the way this section is set up, you might guess that RecMatch $=$ GoodCount, and you'd be right, but it's not completely obvious. The proof is worked out in Problem **??**.

6.2 Arithmetic Expressions

Expression evaluation is a key feature of programming languages, and recognition of expressions as a recursive data type is a key to understanding how they can be processed.

To illustrate this approach we'll work with a toy example: arithmetic expressions like $3x^2 + 2x + 1$ involving only one variable, "x." We'll refer to the data type of such expressions as Aexp. Here is its definition:

Definition 6.2.1. • **Base cases:**

1. The variable, x, is in Aexp.

2. The arabic numeral, `k`, for any nonnegative integer, k, is in Aexp.

- **Constructor cases:** If $e, f \in$ Aexp, then

  3. $(e + f) \in$ Aexp. The expression $(e + f)$ is called a *sum*. The Aexp's e and f are called the *components* of the sum; they're also called the *summands*.

  4. $(e * f) \in$ Aexp. The expression $(e * f)$ is called a *product*. The Aexp's e and f are called the *components* of the product; they're also called the *multiplier* and *multiplicand*.

  5. $-(e) \in$ Aexp. The expression $-(e)$ is called a *negative*.

Notice that Aexp's are fully parenthesized, and exponents aren't allowed. So the Aexp version of the polynomial expression $3x^2 + 2x + 1$ would officially be written as

$$((3 * (x * x)) + ((2 * x) + 1)). \tag{6.2}$$

These parentheses and $*$'s clutter up examples, so we'll often use simpler expressions like "$3x^2 + 2x + 1$" instead of (6.2). But it's important to recognize that $3x^2 + 2x + 1$ is not an Aexp; it's an *abbreviation* for an Aexp.

EDITING NOTE:

Example 6.2.2. The set, List, of *pure lists* is defined recursively by:

1. The 0-tuple is in List.

2. If ℓ_1 and ℓ_2 are in List, then the pair (ℓ_1, ℓ_2) is in List.

In Lisp-like programming languages, the pairing operation is called `cons` and the 0-tuple is called `nil`.

■

6.3 Structural Induction on Recursive Data Types

Structural induction is a method for proving some property, P, of all the elements of a recursively-defined data type. The proof consists of two steps:

- Prove P for the base cases of the definition.

- Prove P for the constructor cases of the definition, assuming that it is true for the component data items.

A very simple application of structural induction proves that the recursively defined matched strings always have an equal number of left and right brackets. To do this, define a predicate, P, on strings $s \in \texttt{brkts}$:

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

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

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

Constructor case: For $r = \textbf{[}\ s\ \textbf{]}\ t$, we must show that $P(r)$ holds, given that $P(s)$ and $P(t)$ holds. So let n_s, n_t be, respectively, the number of left brackets in s and t. So the number of left brackets in r is $1 + n_s + n_t$.

Now from the respective hypotheses $P(s)$ and $P(t)$, we know that the number of right brackets in s is n_s, and likewise, the number of right brackets in t is n_t. So the number of right brackets in r is $1 + n_s + n_t$, which is the same as the number of left brackets. This proves $P(r)$. We conclude by structural induction that $P(s)$

holds for all $s \in$ RecMatch. ∎

6.3.1 Functions on Recursively-defined Data Types

Functions on recursively-defined data types can be defined recursively using the same cases as the data type definition. Namely, to define a function, f, on a recursive data type, define the value of f for the base cases of the data type definition, and then define the value of f in each constructor case in terms of the values of f on the component data items.

For example, from the recursive definition of the set, RecMatch, of strings of matched brackets, we define:

Definition 6.3.1. The *depth*, $d(s)$, of a string, $s \in$ RecMatch, is defined recursively by the rules:

- $d(\lambda) ::= 0.$
- $d([\, s\,]\, t) ::= \max\{d(s)+1, d(t)\}$

Warning: When a recursive definition of a data type allows the same element to be constructed in more than one way, the definition is said to be *ambiguous*. A function defined recursively from an ambiguous definition of a data type will not be well-defined unless the values specified for the different ways of constructing the element agree.

We were careful to choose an *un*ambiguous definition of RecMatch to ensure that functions defined recursively on the definition would always be well-defined. As an example of the trouble an ambiguous definition can cause, let's consider yet another definition of the matched strings.

Example 6.3.2. Define the set, $M \subseteq$ `brkts` recursively as follows:

- **Base case:** $\lambda \in M$,
- **Constructor cases:** if $s, t \in M$, then the strings **[** s **]** and st are also in M.

Quick Exercise: Give an easy proof by structural induction that $M =$ RecMatch.

Since $M =$ RecMatch, and the definition of M seems more straightforward, why didn't we use it? Because the definition of M is ambiguous, while the trickier definition of RecMatch is unambiguous. Does this ambiguity matter? Yes it does. For suppose we defined

$$\begin{aligned} f(\lambda) &::= 1, \\ f(\,[\,s\,]\,) &::= 1 + f(s), \\ f(st) &::= (f(s)+1)\cdot(f(t)+1) && \text{for } st \neq \lambda. \end{aligned}$$

Let a be the string $[\,[\,]\,]$ $\in M$ built by two successive applications of the first M constructor starting with λ. Next let $b ::= aa$ and $c ::= bb$, each built by successive applications of the second M constructor starting with a.

Alternatively, we can build ba from the second constructor with $s = b$ and $t = a$, and then get to c using the second constructor with $s = ba$ and $t = a$.

Now by these rules, $f(a) = 2$, and $f(b) = (2+1)(2+1) = 9$. This means that $f(c) = f(bb) = (9+1)(9+1) = 100$.

But also $f(ba) = (9+1)(2+1) = 27$, so that $f(c) = f(ba\,a) = (27+1)(2+1) = 84$.

The outcome is that $f(c)$ is defined to be both 100 and 84, which shows that the rules defining f are inconsistent.

On the other hand, structural induction remains a sound proof method even for ambiguous recursive definitions, which is why it was easy to prove that $M =$ RecMatch.

6.3.2 Recursive Functions on Nonnegative Integers

The nonnegative integers can be understood as a recursive data type.

Definition 6.3.3. The set, $\mathbb{N}$, is a data type defined recursivly as:

- $0 \in \mathbb{N}$.
- If $n \in \mathbb{N}$, then the *successor*, $n + 1$, of n is in $\mathbb{N}$.

This of course makes it clear that ordinary induction is simply the special case of structural induction on the recursive Definition 6.3.3, This also justifies the familiar recursive definitions of functions on the nonnegative integers. Here are some examples.

The Factorial function. This function is often written "$n!$." You will see a lot of it later in the term. Here we'll use the notation $\text{fac}(n)$:

- $\text{fac}(0) ::= 1$.
- $\text{fac}(n+1) ::= (n+1) \cdot \text{fac}(n)$ for $n \geq 0$.

The Fibonacci numbers. Fibonacci numbers arose out of an effort 800 years ago to model population growth. They have a continuing fan club of people captivated by their extraordinary properties. The nth Fibonacci number, fib, can be defined recursively by:

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

Here the recursive step starts at $n = 2$ with base cases for 0 and 1. This is needed since the recursion relies on two previous values.

What is $\text{fib}(4)$? Well, $\text{fib}(2) = \text{fib}(1) + \text{fib}(0) = 1$, $\text{fib}(3) = \text{fib}(2) + \text{fib}(1) = 2$, so $\text{fib}(4) = 3$. The sequence starts out $0, 1, 1, 2, 3, 5, 8, 13, 21, \ldots$.

Sum-notation. Let "$S(n)$" abbreviate the expression "$\sum_{i=1}^{n} f(i)$." We can recursively define $S(n)$ with the rules

- $S(0) ::= 0$.
- $S(n+1) ::= f(n+1) + S(n)$ for $n \geq 0$.

EDITING NOTE:

Simultaneous recursive definitions: You can define several things at the same time, in terms of each other. For example, we may define two functions f and g from $\mathbb{N}$ to $\mathbb{N}$, recursively, by:

- $f(0) ::= 1$,
- $g(0) ::= 1$,
- $f(n+1) ::= f(n) + g(n)$, for $n \geq 0$,
- $g(n+1) ::= f(n) \times g(n)$, for $n \geq 0$.

■

EDITING NOTE:

Induction on Fibonacci Numbers

We can use the recursive definition of a function to establish its properties by structural induction.

As an illustration, we'll prove a cute identity involving Fibonacci numbers. Fibonacci numbers provide lots of fun for mathematicians because they satisfy many such identities.

Proposition 6.3.4. $\forall n \geq 0 (\Sigma_{i=0}^{n} F_i^2 = F_n F_{n+1})$.

Example: $n = 4$:

$$0^2 + 1^2 + 1^2 + 2^2 + 3^2 = 15 = 3 \cdot 5.$$

Let's try a proof by (standard, not strong) induction. The theorem statement suggests trying it with $P(n)$ defined as:

$$\sum_{i=0}^{n} F_i^2 = F_n F_{n+1}.$$

Base case ($n = 0$). $\Sigma_{i=0}^{0} F_i^2 ::= (F_0)^2 = 0 = F_0 F_1$ because $F_0 ::= 0$.

Inductive step ($n \geq 0$). Now we stare at the gap between $P(n)$ and $P(n+1)$.

$P(n+1)$ is given by a summation that's obtained from that for $P(n)$ by adding one term; this suggests that, once again, we subtract. The difference is just the term F_{n+1}^2. Now, we are assuming that the original $P(n)$ summation totals $F_n F_{n+1}$ and want to show that the new $P(n+1)$ summation totals $F_{n+1}F_{n+2}$. So we would *like* the difference to be

$$F_{n+1}F_{n+2} - F_n F_{n+1}.$$

So, the actual difference is F_{n+1}^2 and the difference we want is $F_{n+1}F_{n+2} - F_n F_{n+1}$. Are these the same? We want to check that:

$$F_{n+1}^2 = F_{n+1}F_{n+2} - F_n F_{n+1}.$$

But this is true, because it is really the Fibonacci definition in disguise: to see this, divide by F_{n+1}.

■

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.

$$f_1(n) ::= 2 + f_1(n-1). \tag{6.3}$$

This "definition" has no base case. If some function, f_1, satisfied (6.3), so would a function obtained by adding a constant to the value of f_1. So equation (6.3) does not uniquely define an f_1.

$$f_2(n) ::= \begin{cases} 0, & \text{if } n = 0, \\ f_2(n+1) & \text{otherwise.} \end{cases} \tag{6.4}$$

This "definition" has a base case, but still doesn't uniquely determine f_2. Any function that is 0 at 0 and constant everywhere else would satisfy the specification, so (6.4) also does not uniquely define anything.

In a typical programming language, evaluation of $f_2(1)$ would begin with a recursive call of $f_2(2)$, which would lead to a recursive call of $f_2(3)$, ... with recursive calls continuing without end. This "operational" approach interprets (6.4) as defining a *partial* function, f_2, that is undefined everywhere but 0.

$$f_3(n) ::= \begin{cases} 0, & \text{if } n \text{ is divisible by 2,} \\ 1, & \text{if } n \text{ is divisible by 3,} \\ 2, & \text{otherwise.} \end{cases} \tag{6.5}$$

This "definition" is inconsistent: it requires $f_3(6) = 0$ and $f_3(6) = 1$, so (6.5) doesn't define anything.

A Mysterious Function

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

$$f_4(n) ::= \begin{cases} 1, & \text{if } n \leq 1, \\ f_4(n/2) & \text{if } n > 1 \text{ is even,} \\ f_4(3n+1) & \text{if } n > 1 \text{ is odd.} \end{cases} \tag{6.6}$$

For example, $f_4(3) = 1$ because

$$f_4(3) ::= f_4(10) ::= f_4(5) ::= f_4(16) ::= f_4(8) ::= f_4(4) ::= f_4(2) ::= f_4(1) ::= 1.$$

The constant function equal to 1 will satisfy (6.6), but it's not known if another function does too. The problem is that the third case specifies $f_4(n)$ in terms of

f_4 at arguments larger than n, and so cannot be justified by induction on $\mathbb{N}$. It's known that any f_4 satisfying (6.6) equals 1 for all n up to over a billion.

Quick exercise: Why does the constant function 1 satisfy (6.6)?

EDITING NOTE:

Tagged data

Labelling a recursively defined data item with a tag that uniquely determines the rule used to construct it is a standard approach to avoiding ambiguous recursive definitions in programming. This amounts to working with data items that are already *parsed*, that is, represented as *parse trees*.

For example, the parse tree for the arithmetic expression

$$-(a(x \cdot x) + bx) + 1 \tag{6.7}$$

is shown in Figure 6.1.

In a computer, such a tree would be represented by pairs or triples that begin with a *tag* equal to the label of the top node of the parse tree. The general definition

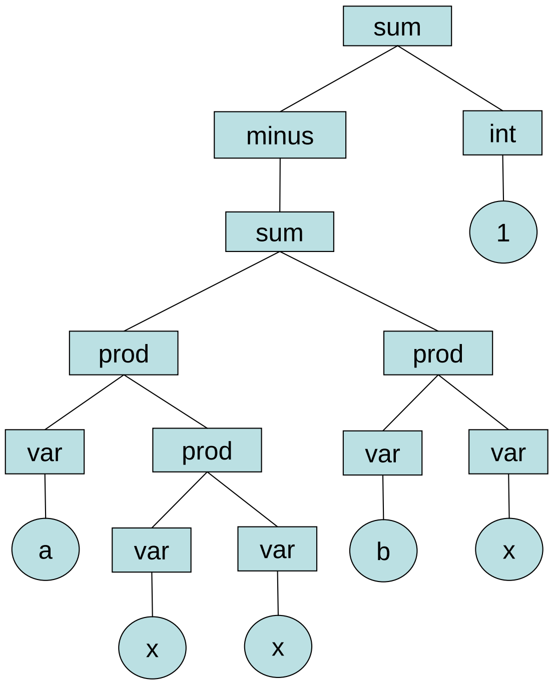


Figure 6.1: Parse tree for $-(a(x \cdot x) + bx) + 1$.

of parse trees for Aexp's would be:

Definition 6.3.5. The set, Aexp-parse-tree, of *parse trees for arithmetic expressions* over a set of *variables*, V, is defined recursively as follows:

- **Base cases:**
  1. If $n \in \mathbb{Z}$, then $\langle \texttt{int}, n \rangle \in$ Aexp-parse-tree.
  2. If $v \in V$, then $\langle \texttt{var}, v \rangle \in$ Aexp-parse-tree.
- **Constructor cases:** if $e, e' \in$ Aexp-parse-tree, then
  1. $\langle \texttt{sum}, e, e' \rangle \in$ Aexp-parse-tree,
  2. $\langle \texttt{prod}, e, e' \rangle \in$ Aexp-parse-tree, and
  3. $\langle \texttt{minus}, e \rangle \in$ Aexp-parse-tree.

So the Aexp-parse-tree corresponding to formula 6.7 would be:

$$\begin{array}{l} \langle\, \texttt{sum},\ \ \langle\, \texttt{minus},\ \langle\, \texttt{sum},\ \ \langle \texttt{prod},\ \langle \texttt{var},\ a\rangle, \langle \texttt{prod},\ \langle \texttt{var},\ x\rangle,\ \langle \texttt{var},\ x\rangle\rangle\rangle, \\ \qquad\qquad\qquad\qquad\qquad\qquad \langle \texttt{prod},\ \langle \texttt{var},\ b\rangle,\ \langle \texttt{var},\ x\rangle\rangle\rangle\rangle, \\ \qquad\quad \langle \texttt{int},\ 1\rangle\rangle\rangle \end{array} \tag{6.8}$$

Now the expression 6.7 is certainly a lot more humanly intelligible than 6.8, but 6.8 is in the representation best suited and commonly used in compiling and processing computer programs.

■

6.3.3 Evaluation and Substitution with Aexp's

Evaluating Aexp's

Since the only variable in an Aexp is x, the value of an Aexp is determined by the value of x. For example, if the value of x is 3, then the value of $3x^2 + 2x + 1$ is obviously 34. In general, given any Aexp, e, and an integer value, n, for the variable, x, we can evaluate e to finds its value, $\text{eval}(e, n)$. It's easy, and useful, to specify this evaluation process with a recursive definition.

Definition 6.3.6. The *evaluation function*, $\text{eval} : \text{Aexp} \times \mathbb{Z} \to \mathbb{Z}$, is defined recursively on expressions, $e \in \text{Aexp}$, as follows. Let n be any integer.

- **Base cases:**

1. Case[e is x]

$$\text{eval}(x, n) ::= n.$$

(The value of the variable, x, is given to be n.)

2. Case[e is k]

$$\text{eval}(\texttt{k}, n) ::= k.$$

(The value of the numeral `k` is the integer k, no matter what value x has.)

- **Constructor cases:**

3. Case[e is $(e_1 + e_2)$]

$$\text{eval}((e_1 + e_2), n) ::= \text{eval}(e_1, n) + \text{eval}(e_2, n).$$

4. Case[e is $(e_1 * e_2)$]

$$\text{eval}((e_1 * e_2), n) ::= \text{eval}(e_1, n) \cdot \text{eval}(e_2, n).$$

5. Case[e is $-(e_1)$]

$$\text{eval}(-(e_1), n) ::= -\text{eval}(e_1, n).$$

For example, here's how the recursive definition of eval would arrive at the value of $3 + x^2$ when x is 2:

$$\begin{aligned}
\text{eval}((\texttt{3} + (x * x)), 2) &= \text{eval}(\texttt{3}, 2) + \text{eval}((x * x), 2) && \text{(by Def 6.3.6.3)}\\
&= 3 + \text{eval}((x * x), 2) && \text{(by Def 6.3.6.2)}\\
&= 3 + (\text{eval}(x, 2) \cdot \text{eval}(x, 2)) && \text{(by Def 6.3.6.4)}\\
&= 3 + (2 \cdot 2) && \text{(by Def 6.3.6.1)}\\
&= 3 + 4 = 7.
\end{aligned}$$

Substituting into Aexp's

Substituting expressions for variables is a standard, important operation. For example the result of substituting the expression $3x$ for x in the $(x(x - 1))$ would be $(3x(3x - 1)$. We'll use the general notation $\text{subst}(f, e)$ for the result of substituting an Aexp, f, for each of the x's in an Aexp, e. For instance,

$$\text{subst}(3x, x(x - 1)) = 3x(3x - 1).$$

This substitution function has a simple recursive definition:

Definition 6.3.7. The *substitution function* from Aexp $\times$ Aexp to Aexp is defined recursively on expressions, $e \in$ Aexp, as follows. Let f be any Aexp.

- **Base cases:**

  1. Case[e is x]

  $$\text{subst}(f, x) ::= f.$$

  (The result of substituting f for the variable, x, is just f.)

  2. Case[e is $\texttt{k}$]

  $$\text{subst}(f, \texttt{k}) ::= \texttt{k}.$$

  (The numeral, `k`, has no x's in it to substitute for.)

- **Constructor cases:**

  3. Case[e is $(e_1 + e_2)$]

  $$\text{subst}(f, (e_1 + e_2))) ::= (\text{subst}(f, e_1) + \text{subst}(f, e_2)).$$

  4. Case[e is $(e_1 * e_2)$]

  $$\text{subst}(f, (e_1 * e_2))) ::= (\text{subst}(f, e_1) * \text{subst}(f, e_2)).$$

5. Case[e is $-(e_1)$]

$$\text{subst}(f, -(e_1)) ::= -(\text{subst}(f, e_1)).$$

Here's how the recursive definition of the substitution function would find the result of substituting $3x$ for x in the $x(x-1)$:

$$\begin{aligned}
\text{subst}(3x, (x(x-1))) &= \text{subst}(3x, (x * (x + -(1)))) && \text{(unabbreviating)}\\
&= (\text{subst}(3x, x) * \text{subst}(3x, (x + -(1)))) && \text{(by Def 6.3.7 4)}\\
&= (3x * \text{subst}(3x, (x + -(1)))) && \text{(by Def 6.3.7 1)}\\
&= (3x * (\text{subst}(3x, x) + \text{subst}(3x, -(1)))) && \text{(by Def 6.3.7 3)}\\
&= (3x * (3x + -(\text{subst}(3x, 1)))) && \text{(by Def 6.3.7 1 \& 5)}\\
&= (3x * (3x + -(1))) && \text{(by Def 6.3.7 2)}\\
&= 3x(3x-1) && \text{(abbreviation)}
\end{aligned}$$

Now suppose we have to find the value of $\text{subst}(3x, (x(x-1)))$ when $x = 2$. There are two approaches.

First, we could actually do the substitution above to get $3x(3x-1)$, and then we could evaluate $3x(3x - 1)$ when $x = 2$, that is, we could recursively calculate $\mathrm{eval}(3x(3x - 1), 2)$ to get the final value 30. In programming jargon, this would be called evaluation using the *Substitution Model*. Tracing through the steps in the evaluation, we find that the Substitution Model requires two substitutions for occurrences of x and 5 integer operations: 3 integer multiplications, 1 integer addition, and 1 integer negative operation. Note that in this Substitution Model the multiplication $3 \cdot 2$ was performed twice to get the value of 6 for each of the two occurrences of $3x$.

The other approach is called evaluation using the *Environment Model*. Namely, we evaluate $3x$ when $x = 2$ using just 1 multiplication to get the value 6. Then we evaluate $x(x - 1)$ when x has this value 6 to arrive at the value $6 \cdot 5 = 30$. So the Environment Model requires 2 variable lookups and only 4 integer operations: 1 multiplication to find the value of $3x$, another multiplication to find the value $6 \cdot 5$, along with 1 integer addition and 1 integer negative operation.

So the Environment Model approach of calculating

$$\text{eval}(x(x-1), \text{eval}(3x, 2))$$

instead of the Substitution Model approach of calculating

$$\text{eval}(\text{subst}(3x, x(x-1)), 2)$$

is faster. But how do we know that these final values reached by these two approaches always agree? We can prove this easily by structural induction on the definitions of the two approaches. More precisely, what we want to prove is

Theorem 6.3.8. *For all expressions $e, f \in$ Aexp and $n \in \mathbb{Z}$,*

$$\text{eval}(\text{subst}(f, e), n) = \text{eval}(e, \text{eval}(f, n)). \tag{6.9}$$

Proof. The proof is by structural induction on e.[1]

Base cases:

- Case[e is x]

[1]This is an example of why it's useful to notify the reader what the induction variable is—in this case it isn't n.

The left hand side of equation (6.9) equals $\text{eval}(f, n)$ by this base case in Definition 6.3.7 of the substitution function, and the right hand side also equals $\text{eval}(f, n)$ by this base case in Definition 6.3.6 of eval.

- Case[e is `k`].

  The left hand side of equation (6.9) equals `k` by this base case in Definitions 6.3.7 and 6.3.6 of the substitution and evaluation functions. Likewise, the right hand side equals `k` by two applications of this base case in the Definition 6.3.6 of eval.

Constructor cases:

- Case[e is $(e_1 + e_2)$]

  By the structural induction hypothesis (6.9), we may assume that for all $f \in$ Aexp and $n \in \mathbb{Z}$,

$$\text{eval}(\text{subst}(f, e_i), n) = \text{eval}(e_i, \text{eval}(f, n)) \tag{6.10}$$

for $i = 1, 2$. We wish to prove that

$$\mathrm{eval}(\mathrm{subst}(f, (e_1 + e_2)), n) = \mathrm{eval}((e_1 + e_2), \mathrm{eval}(f, n)) \tag{6.11}$$

But the left hand side of (6.11) equals

$$\mathrm{eval}(\ (\mathrm{subst}(f, e_1) + \mathrm{subst}(f, e_2)),\ n)$$

by Definition 6.3.7.3 of substitution into a sum expression. But this equals

$$\mathrm{eval}(\mathrm{subst}(f, e_1), n) + \mathrm{eval}(\mathrm{subst}(f, e_2), n)$$

by Definition 6.3.6.3 of eval for a sum expression. By induction hypothesis (6.10), this in turn equals

$$\mathrm{eval}(e_1, \mathrm{eval}(f, n)) + \mathrm{eval}(e_2, \mathrm{eval}(f, n)).$$

Finally, this last expression equals the right hand side of (6.11) by Definition 6.3.6.3 of eval for a sum expression. This proves (6.11) in this case.

- e is $(e_1 * e_2)$. Similar.
- e is $-(e_1)$. Even easier.

This covers all the constructor cases, and so completes the proof by structural induction.

■

EDITING NOTE:

A String Theorem

Here is a more complex proof, illustrating a combination of structural induction and strengthening the hypothesis.

Theorem 6.3.9. *In a string of* 0s *and* 1s*, the number of occurrences of the pattern* 01 *is less than or equal to the number of occurrences of* 10*, plus one.*

Let's try to prove this by structural induction. First we must define $P(s)$. Let's write $\text{num}(pat, s)$ as the number of occurrences of the pattern string pat in s. Now our inductive hypothesis is

$$P(s) : \text{num}(01, s) \leq \text{num}(10, s) + 1.$$

If you try to prove this by structural induction, you will get stuck. Why? Consider what happens when you add 1 at the end. This could increase the number of 01s without increasing the number of 10s.

So, to prove by structural induction on strings, let's strengthen the hypothesis by adding another clause. If a string ends in 0 then the number of 01s is less than or equal to the number of 10s. That solves the problem by weakening what we have to show when the string ends in 1. But maybe it causes another problem somewhere else. Let's give it a try:

Redefine $P(s)$::=

$$\text{num}(01, s) \quad \leq \quad \text{num}(10, s) + 1, \text{ and}$$

$$\text{If } s \text{ ends in } 0 \text{ then}$$

$$\text{num}(01, s) \quad \leq \quad \text{num}(10, s).$$

This means that, for each inductive step have two things to show.

Structured proof display commented out here

First let's consider $s1$. This is the case that looks dangerous, because it might increase the number of 01s. We have to prove two statements. The second is easy, because the new string doesn't end in 0. We say it's "vacuously true".

The first statement now takes some work. We might be adding to the number of 01s. However, if we do, the previous string must have ended with 0. Then the inductive hypothesis says that the previous string had to satisfy the stronger inequality in the second statement. Adding one to the LHS of the stronger inequality yields the weaker inequality we want.

The following proof fragment considers cases based on whether s ends in 0 or not. If not, it might end in 1, or might be empty (don't forget this possibility).

Structured proof display commented out here

Of course, you could also expand the step for s ending in 1 into a careful series of inequalities.

Now consider $s0$. We hope that what we did to make the $s1$ case work doesn't mess up the $s0$ case. But we have to check.

The first statement is easy. It follows from the first statement of the inductive hypothesis for s, because we are not increasing the number of 01s. But now the second statement takes more work. The difficulty is that the new string ends in 0, which means that we have to show the stronger inequality in the second statement. But to do this, we might only have the weaker inequality for the previous string. The argument again depends on what the previous string s ended with. So again, we consider cases, based on whether s ends in 0 or 1, or is empty. If s ends in 0 we rely on the second statement of the inductive hypothesis for s (with the stronger inequality), whereas if s ends in 1 we rely on the first statement (with the weaker inequality). In this case, we have to "turn the weaker inequality into the stronger inequality".

Structured proof display commented out here

If you actually write out all these cases in the proof, you will notice that some facts are stated repeatedly, e.g., that when you add a 0 to the end of a string you are not increasing the number of 01s. To avoid having to state these facts several times, you can move them earlier in the proof.

■

EDITING NOTE:

Example.

Definition 6.3.10. 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.

A very simple application of structural induction proves that the set E given by Definition 6.3.10 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 6.3.10 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 $n \in E$. ■

defining the set, E, of even numbers as in Definitions 6.3.10, but without the conditions 1. and 2. that restrict application of the rules. Namely,

Definition 6.3.11. 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 6.3.11 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 6.3.11 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 6.3.11:

$$s(0) ::= 1,$$

$$s(n+2) ::= 1 + s(n),$$

$$s(-n) ::= 1 + s(n).$$

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 6.3.10 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 6.3.10 of E.

■

6.3.4 Problems

Practice Problems

Class Problems

Homework Problems

6.4 Games as a Recursive Data Type

Chess, Checkers, and Tic-Tac-Toe are examples of *two-person terminating games of perfect information*, —2PTG's for short. These are games in which two players alternate moves that depend only on the visible board position or state of the game. "Perfect information" means that the players know the complete state of the game at each move. (Most card games are *not* games of perfect information because neither player can see the other's hand.) "Terminating" means that play cannot go on forever —it must end after a finite number of moves.[2]

We will define 2PTG's as a recursive data type. To see how this will work, let's use the game of Tic-Tac-Toe as an example.

6.4.1 Tic-Tac-Toe

Tic-Tac-Toe is a game for young children. There are two players who alternately write the letters "X" and "O" in the empty boxes of a 3×3 grid. Three copies of the same letter filling a row, column, or diagonal of the grid is called a *tic-tac-toe*, and the first player who gets a tic-tac-toe of their letter wins the game.

EDITING NOTE: Children generally don't take long to figure out an optimal strategy for playing the game. ■

We're now going give a precise mathematical definition of the Tic-Tac-Toe *game tree* as a recursive data type.

[2]Since board positions can repeat in chess and checkers, termination is enforced by rules that prevent any position from being repeated more than a fixed number of times. So the "state" of these games is the board position *plus* a record of how many times positions have been reached.

EDITING NOTE: Children of course have no need for such a definition, and it would be too complicated for them anyway. But if we had to write a Tic-Tac-Toe playing *computer program*, we'd need this kind of picky precision. ■

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 6.2.

Definition 6.4.1. A Tic-Tac-Toe *pattern* is a 3×3 grid each of whose 9 cells contains either the single letter, X, the single letter, O, or is empty.

EDITING NOTE:

Moreover, there must be either

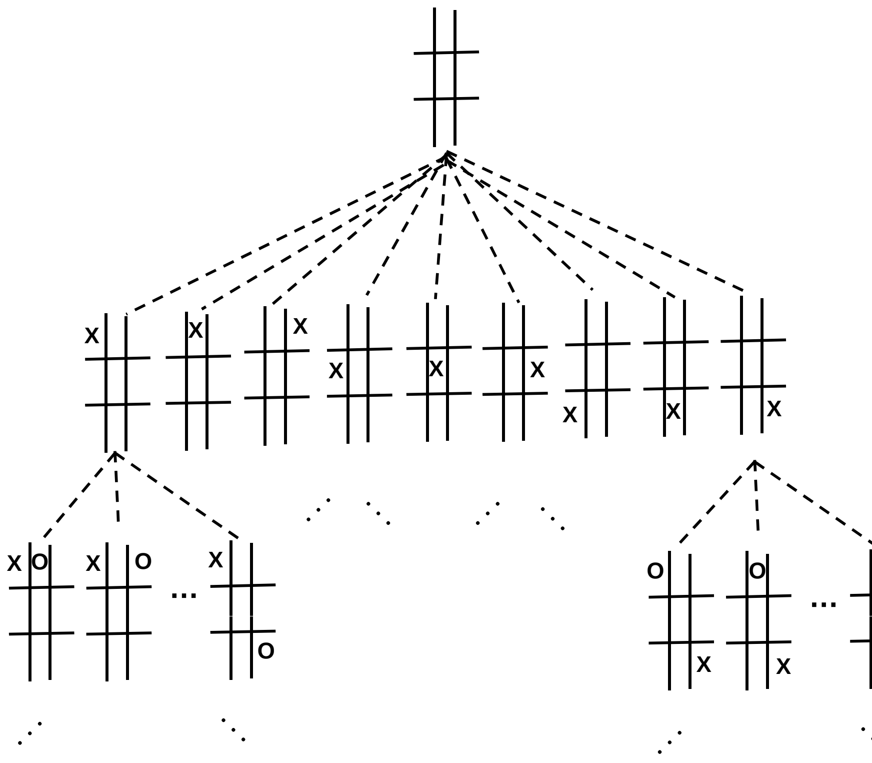


Figure 6.2: The Top of the Game Tree for Tic-Tac-Toe.

- one more X than O's, with at most two tic-tac-toes of X's, and no tic-tac-toe of O's, or
- an equal number of X's and O's, with at most one tic-tac-toes of O's, and no tic-tac-toe of X's.

■

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 game trees starting at possible next patterns after P.

For example, if

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

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

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

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

the game tree starting at P_0 is pictured in Figure 6.3.

EDITING NOTE:

Then,

$$\langle P, \{\langle Q_1, \langle \texttt{lose}\rangle\rangle, \langle Q_2, \{\langle R, \langle \texttt{tie}\rangle\rangle\}\rangle\}\rangle \tag{6.12}$$

is the tagged recursive datum that corresponds to a Tic-Tac-Toe "end game" that starts with P. This game is easier to understand by looking at its game tree in Figure 6.3. Notice that the game tree —which so far we haven't actually defined —is simply the parse tree of the tagged datum.

■

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

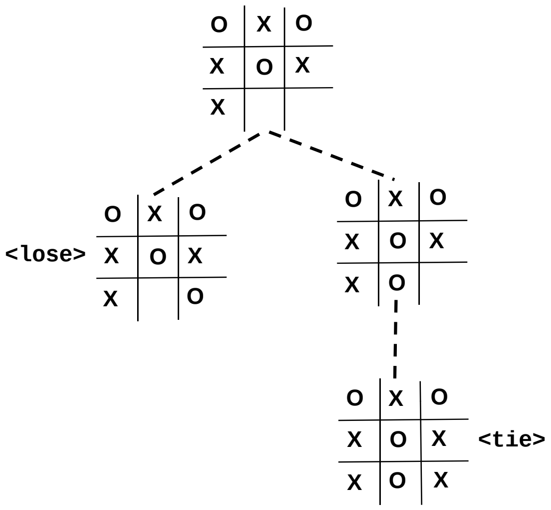


Figure 6.3: Game Tree for the Tic-Tac-Toe game starting at P_0.

trees, and the second usage refers to a "play.")

6.4.2 Infinite Tic-Tac-Toe Games

At any point in a Tic-Tac-Toe game, there are at most nine possible next patterns, and no play can continue for more than nine moves. But we can expand Tic-Tac-Toe into a larger game by running a 5-game tournament: play Tic-Tac-Toe five times and the tournament winner is the player who wins the most individual games. A 5-game tournament can run for as many as 45 moves.

It's not much of generalization to have an *n-game Tic-Tac-Toe tournament*. But then comes a generalization that sounds simple but can be mind-boggling: consolidate all these different size tournaments into a single game we can call *Tournament-Tic-Tac-Toe (T^4)*. The first player in a game of T^4 chooses any integer $n > 0$. Then the players play an n-game tournament. Now we can no longer say how long a T^4 play can take. In fact, there are T^4 plays that last as long as you might like: if you want a game that has a play with, say, nine billion moves, just have the first player choose n equal to one billion. This should make it clear the game tree for

T^4 is infinite.

But still, it's obvious that every possible T^4 play will stop. That's because after the first player chooses a value for n, the game can't continue for more than $9n$ moves. So it's not possible to keep playing forever even though the game tree is infinite.

This isn't very hard to understand, but there is an important difference between any given n-game tournament and T^4: even though every play of T^4 must come to an end, there is no longer any initial bound on how many moves it might be before the game ends —a play might end after 9 moves, or $9(2001)$ moves, or $9(10^{10} + 1)$ moves. It just can't continue forever.

EDITING NOTE:

While there is no bound on how long to play, at least after the first move to an $n \times n$ board in meta-Tic-Tac-Toe, we know the game will end with n^2 moves.

■

Now that we recognize T^4 as a 2PTG, we can go on to a *meta-*T^4 game, where the first player chooses a number, $m > 0$, of T^4 games to play, and then the second player gets the first move in each of the individual T^4 games to be played.

Then, of course, there's meta-meta-T^4....

EDITING NOTE: Every play of the meta-meta game must still end, but now even after the first move, there is no bound on how long a game might continue. ■

6.4.3 Two Person Terminating Games

Familiar games like Tic-Tac-Toe, Checkers, and Chess can all end in ties, but for simplicity we'll only consider win/lose games —no "everybody wins"-type games at MIT. :-) But everything we show about win/lose games will extend easily to games with ties, and more generally to games with outcomes that have different payoffs.

EDITING NOTE:

Of course Tic-Tac-Toe and the other games will fit this set up if we treat a game that ends in a tie as a loss for the usual first player —White in Chess, Red in Checkers, the X-player in Tic-Tac-Toe.

■

Like Tic-Tac-Toe, or Tournament-Tic-Tac-Toe, the idea behind the definition of 2PTG's as a recursive data type is that making a move in a 2PTG leads to the start of a subgame. In other words, given any set of games, we can make a new game whose first move is to pick a game to play from the set.

So what defines a game? For Tic-Tac-Toe, we used the patterns and the rules of Tic-Tac-Toe to determine the next patterns. But once we have a complete game tree, we don't really need the pattern labels: the root of a game tree itself can play the role of a "board position" with its possible "next positions" determined by the roots of its subtrees. So any game is defined by its game tree. This leads to the following very simple —perhaps deceptively simple —general definition.

Definition 6.4.2. The 2PTG, *game trees for two-person terminating games of perfect information* are defined recursively as follows:

- **Base cases:**

$$\begin{aligned} \langle \texttt{leaf}, \texttt{win} \rangle &\in \text{2PTG, and} \\ \langle \texttt{leaf}, \texttt{lose} \rangle &\in \text{2PTG.} \end{aligned}$$

- **Constructor case:** If $\mathcal{G}$ is a nonempty set of 2PTG's, then G is a 2PTG, where

$$G ::= \langle \texttt{tree}, \mathcal{G} \rangle .$$

  The game trees in $\mathcal{G}$ are called the possible *next moves* from G.

These games are called "terminating" because, even though a 2PTG may be a (very) infinite datum like Tournament^2-Tic-Tac-Toe, every play of a 2PTG must terminate. This is something we can now prove, after we give a precise definition of "play":

Definition 6.4.3. A *play* of a 2PTG, G, is a (potentially infinite) sequence of 2PTG's starting with G and such that if G_1 and G_2 are consecutive 2PTG's in the play, then G_2 is a possible next move of G_1.

If a 2PTG has no infinite play, it is called a *terminating* game.

Theorem 6.4.4. *Every 2PTG is terminating.*

Proof. By structural induction on the definition of a 2PTG, G, with induction hypothesis

$$G \text{ is terminating.}$$

Base case: If $G = \langle \texttt{leaf}, \texttt{win} \rangle$ or $G = \langle \texttt{leaf}, \texttt{lose} \rangle$ then the only possible play of G is the length one sequence consisting of G. Hence G terminates.

Constructor case: For $G = \langle \texttt{tree}, \mathcal{G} \rangle$, we must show that G is terminating, given the Induction Hypothesis that *every* $G' \in \mathcal{G}$ is terminating.

But any play of G is, by definition, a sequence starting with G and followed by a play starting with some $G_0 \in \mathcal{G}$. But G_0 is terminating, so the play starting at G_0 is finite, and hence so is the play starting at G.

This completes the structural induction, proving that every 2PTG, G, is terminating. ■

6.4.4 Game Strategies

A key question about a game is whether a player has a winning strategy. A *strategy* for a player in a game specifies which move the player should make at any point in the game. A *winning* strategy ensures that the player will win no matter what moves the other player makes.

In Tic-Tac-Toe for example, most elementary school children figure out strategies for both players that each ensure that the game ends with no tic-tac-toes, that is, it ends in a tie. Of course the first player can win if his opponent plays childishly, but not if the second player follows the proper strategy. In more complicated games like Checkers or Chess, it's not immediately clear that anyone has a winning strategy, even if we agreed to count ties as wins for the second player.

But structural induction makes it easy to prove that in any 2PTG, *somebody* has the winning strategy!

Theorem 6.4.5. ***Fundamental Theorem for Two-Person Games:*** *For every two-person terminating game of perfect information, there is a winning strategy for one of the players.*

Proof. The proof is by structural induction on the definition of a 2PTG, G. The induction hypothesis is that there is a winning strategy for G.

Base cases:

1. $G = \langle \texttt{leaf}, \texttt{win} \rangle$. Then the first player has the winning strategy: "make the winning move."

2. $G = \langle \texttt{leaf}, \texttt{lose} \rangle$. Then the second player has a winning strategy: "Let the first player make the losing move."

Constructor case: Suppose $G = \langle \texttt{tree}, \mathcal{G} \rangle$. By structural induction, we may assume that some player has a winning strategy for each $G' \in \mathcal{G}$. There are two cases to consider:

- some $G_0 \in \mathcal{G}$ has a winning strategy for its second player. Then the first player in G has a winning strategy: make the move to G_0 and then follow the second player's winning strategy in G_0.

- every $G' \in \mathcal{G}$ has a winning strategy for its first player. Then the second player in G has a winning strategy: if the first player's move in G is to $G_0 \in \mathcal{G}$, then follow the winning strategy for the first player in G_0.

So in any case, one of the players has a winning strategy for G, which completes the proof of the constructor case.

It follows by structural induction that there is a winning strategy for every 2PTG, G. ■

Notice that although Theorem 6.4.5 guarantees a winning strategy, its proof gives no clue which player has it. For most familiar 2PTG's like Chess, Go, ..., no one knows which player has a winning strategy.[3]

[3]Checkers used to be in this list, but there has been a recent announcement that each player has a strategy that forces a tie. (reference TBA)

6.4.5 Problems

Homework Problems

6.5 Induction in Computer Science

Induction is a powerful and widely applicable proof technique, which is why we've devoted two entire chapters to it. Strong induction and its special case of ordinary induction are applicable to any kind of thing with nonnegative integer sizes –which is a awful lot of things, including all step-by-step computational processes.

Structural induction then goes beyond natural number counting by offering a simple, natural approach to proving things about recursive computation and recursive data types. This makes it a technique every computer scientist should embrace.

EDITING NOTE:

In many cases a nonnegative integer size can be defined for a recursively defined datum, such as the length of a string, or the number of operations in an Aexp. It is then possible to prove properties of data by ordinary induction on their size. But this approach often produces more cumbersome proofs than structural induction.

In fact, structural induction is theoretically more powerful than ordinary induction. However, it's only more powerful when it comes to reasoning about infinite data types —like infinite trees, for example —so this greater power doesn't matter in practice. What does matter is that for recursively defined data types, structural induction is a simple and natural approach.

■

Chapter 7

Simple Graphs

Graphs in which edges are *not* directed are called *simple graphs*. They come up in all sorts of applications, including scheduling, optimization, communications, and the design and analysis of algorithms. Two Stanford students even used graph theory to become multibillionaires!

But we'll start with an application designed to get your attention: we are going to make a professional inquiry into sexual behavior. Namely, we'll look at some data about who, on average, has more opposite-gender partners, men or women.

Sexual demographics have been the subject of many studies. In one of the largest, researchers from the University of Chicago interviewed a random sample of 2500 people over several years to try to get an answer to this question. Their study, published in 1994, and entitled *The Social Organization of Sexuality* found that on average men have 74% more opposite-gender partners than women.

Other studies have found that the disparity is even larger. In particular, ABC News claimed that the average man has 20 partners over his lifetime, and the average woman has 6, for a percentage disparity of 233%. The ABC News study, aired on Primetime Live in 2004, purported to be one of the most scientific ever done, with only a 2.5% margin of error. It was called ”American Sex Survey: A peek between the sheets,” —which raises some question about the seriousness of their reporting.

EDITING NOTE: 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.

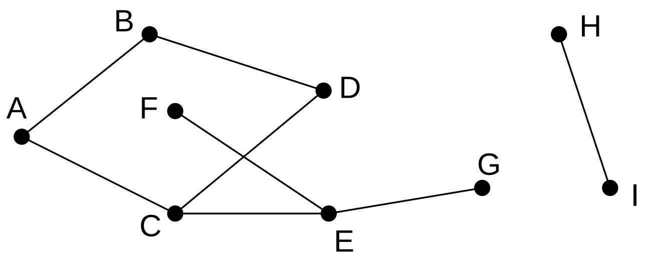
Probably that last part about going where few scientific surveys have gone before is pretty accurate! ■

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.

7.1 Degrees & Isomorphism

7.1.1 Definition of Simple Graph

Informally, a graph is a bunch of dots with lines connecting some of them. Here is an example:



For many mathematical purposes, we don't really care how the points and lines are laid out —only which points are connected by lines. The definition of *simple graphs* aims to capture just this connection data.

Definition 7.1.1. A *simple graph*, G, consists of a nonempty set, V, called the *vertices* of G, and a collection, E, of two-element subsets of V. The members of E are called the *edges* of G.

The vertices correspond to the dots in the picture, and the edges correspond to the lines. For example, the connection data given in the diagram above can also be given by listing the vertices and edges according to the official definition of simple graph:

$$V = \{A, B, C, D, E, F, G, H, I\}$$

$$E = \{\{A, B\}, \{A, C\}, \{B, D\}, \{C, D\}, \{C, E\}, \{E, F\}, \{E, G\}, \{H, I\}\}.$$

It will be helpful to use the notation A—B for the edge $\{A, B\}$. Note that A—B and B—A are different descriptions of the same edge, since sets are unordered.

So the definition of simple graphs is the same as for directed graphs, except that instead of a directed edge $v \to w$ which starts at vertex v and ends at vertex w, a simple graph only has an undirected edge, v—w, that connects v and w.

Definition 7.1.2. Two vertices in a simple graph are said to be *adjacent* if they are joined by an edge, and an edge is said to be *incident* to the vertices it joins. The number of edges incident to a vertex is called the *degree* of the vertex; equivalently, the degree of a vertex is equals the number of vertices adjacent to it.

For example, in the simple graph above, A is adjacent to B and B is adjacent to

D, and the edge A—C is incident to vertices A and C. Vertex H has degree 1, D has degree 2, and E has degree 3.

Graph Synonyms

A synonym for "vertices" is "*nodes*," and we'll use these words interchangeably. Simple graphs are sometimes called *networks*, edges are sometimes called *arcs*. We mention this as a "heads up" in case you look at other graph theory literature; we won't use these words.

Some technical consequences of Definition 7.1.1 are worth noting right from the start:

1. Simple graphs do *not* have self-loops ($\{a, a\}$ is not an undirected edge because an undirected edge is defined to be a set of *two* vertices.)

2. There is at most one edge between two vertices of a simple graph.

3. Simple graphs have at least one vertex, though they might not have any edges.

There's no harm in relaxing these conditions, and some authors do, but we don't need self-loops, multiple edges between the same two vertices, or graphs with no vertices, and it's simpler not to have them around.

For the rest of this Chapter we'll only be considering simple graphs, so we'll just call them "graphs" from now on.

7.1.2 Sex in America

Let's model the question of heterosexual partners in graph theoretic terms. To do this, we'll let G be the graph whose vertices, V, are all the people in America. Then we split V into two separate subsets: M, which contains all the males, and F, which contains all the females.[1] We'll put an edge between a male and a female iff they have been sexual partners. This graph is pictured in Figure 7.1 with males on the left and females on the right.

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,

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

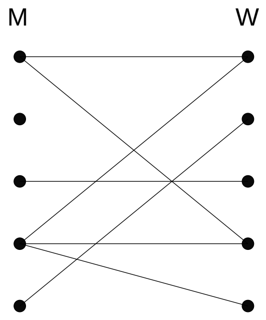


Figure 7.1: The sex partners graph

approximately 50.8% are female and 49.2% are male, so $|M| \approx 147.6M$, and $|F| \approx 152.4M$. And we don't even have trustworthy estimates of how many edges there are, let alone exactly which couples are adjacent. But it turns out that we don't need to know any of this —we just need to figure out the relationship between the average number of partners per male and partners per female. To do this, we note that every edge is incident to exactly one M vertex (remember, we're only considering male-female relationships); so the sum of the degrees of the M vertices equals the number of edges. For the same reason, the sum of the degrees of the F

vertices equals the number of edges. So these sums are equal:

$$\sum_{x \in M} \deg(x) = \sum_{y \in F} \deg(y).$$

Now suppose we divide both sides of this equation by the product of the sizes of the two sets, $|M| \cdot |F|$:

$$\left(\frac{\sum_{x \in M} \deg(x)}{|M|} \right) \cdot \frac{1}{|F|} = \left(\frac{\sum_{y \in F} \deg(y)}{|F|} \right) \cdot \frac{1}{|M|}$$

The terms above in parentheses are the *average degree of an M vertex* and the *average degree of a F* vertex. So we know:

$$\text{Avg. deg in } M = \frac{|F|}{|M|} \cdot \text{Avg. deg in } F$$

In other words, we've proved that the average number of female partners of males in the population compared to the average number of males per female is *determined solely by the relative number of males and females in the population*.

Now the Census Bureau reports that there are slightly more females than males in America; in particular $|F| / |M|$ is about 1.035. So we know that on average, males have 3.5% more opposite-gender partners than females, and this tells us nothing about any sex's promiscuity or selectivity. Rather, it just has to do with the relative number of males and females. Collectively, males and females have the same number of opposite gender partners, since it takes one of each set for every partnership, but there are fewer males, so they have a higher ratio. This means that the University of Chicago, ABC, and the Federal government studies are way off. After a huge effort, they gave a totally wrong answer.

There's no definite explanation for why such surveys are consistently wrong. One hypothesis is that males exaggerate their number of partners —or maybe females downplay theirs —but these explanations are speculative. Interestingly, the principal author of the National Center for Health Statistics study reported that she knew the results had to be wrong, but that was the data collected, and her job was to report it.

The same underlying issue has led to serious misinterpretations of other survey data. For example, a couple of years ago, the Boston Globe ran a story on a survey of the study habits of students on Boston area campuses. Their survey showed that on average, minority students tended to study with non-minority students more than the other way around. They went on at great length to explain why this "remarkable phenomenon" might be true. But it's not remarkable at all —using our graph theory formulation, we can see that all it says is that there are fewer minority students than non-minority students, which is, of course what "minority" means.

7.1.3 Handshaking Lemma

The previous argument hinged on the connection between a sum of degrees and the number edges. There is a simple connection between these in any graph:

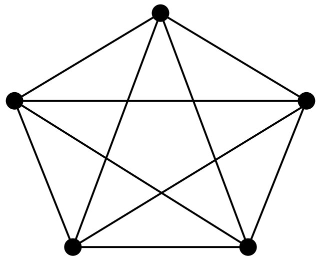
Lemma 7.1.3. *The sum of the degrees of the vertices in a graph equals twice the number of edges.*

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

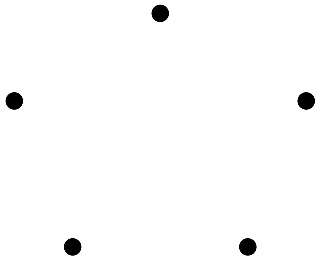
Lemma 7.1.3 is sometimes called the *Handshake Lemma*: if we total up the number of people each person at a party shakes hands with, the total will be twice the number of handshakes that occurred.

7.1.4 Some Common Graphs

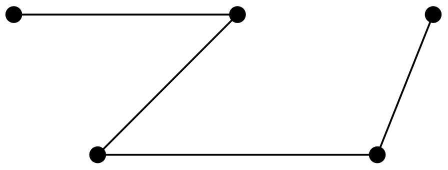
Some graphs come up so frequently that they have names. The *complete graph* on n vertices, also called K_n, has an edge between every two vertices. Here is K_5:



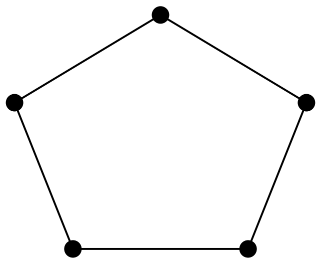
The *empty graph* has no edges at all. Here is the empty graph on 5 vertices:



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

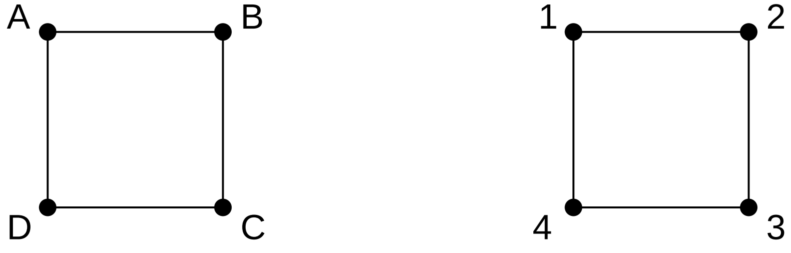


And here is C_5, a *simple cycle* with 5 vertices:



7.1.5 Isomorphism

Two graphs that look the same might actually be different in a formal sense. For example, the two graphs below are both simple cycles with 4 vertices:



But one graph has vertex set $\{A, B, C, D\}$ while the other has vertex set $\{1, 2, 3, 4\}$. If so, then the graphs are different mathematical objects, strictly speaking. But this is a frustrating distinction; the graphs *look the same*!

Fortunately, we can neatly capture the idea of "looks the same" by adapting Definition 9.2.1 of isomorphism of digraphs to handle simple graphs.

Definition 7.1.4. If G_1 is a graph with vertices, V_1, and edges, E_1, and likewise for G_2, then G_1 is *isomorphic* to G_2 iff there exists a **bijection**, $f : V_1 \to V_2$, such that for every pair of vertices $u, v \in V_1$:

$$u—v \in E_1 \quad \text{iff} \quad f(u)—f(v) \in E_2.$$

The function f is called an *isomorphism* between G_1 and G_2.

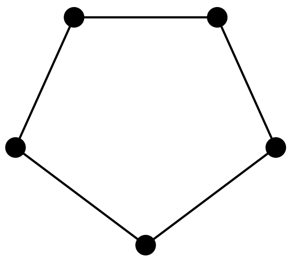
EDITING NOTE: Graphs G_1 and G_2 are *isomorphic* if there exists a bijection between the vertices in G_1 and the vertices in G_2 such that there is an edge between two vertices in G_1 if and only if there is an edge between the two corresponding vertices in G_2. ■

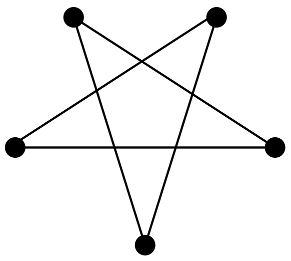
For example, here is an isomorphism between vertices in the two graphs above:

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

You can check that there is an edge between two vertices in the graph on the left if and only if there is an edge between the two corresponding vertices in the graph on the right.

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





Isomorphism preserves the connection properties of a graph, abstracting out what the vertices are called, what they are made out of, or where they appear in a drawing of the graph. More precisely, a property of a graph is said to be *preserved under isomorphism* if whenever G has that property, every graph isomorphic to G

also has that property. For example, since an isomorphism is a bijection between sets of vertices, isomorphic graphs must have the same number of vertices. What's more, if f is a graph isomorphism that maps a vertex, v, of one graph to the vertex, $f(v)$, of an isomorphic graph, then by definition of isomorphism, every vertex adjacent to v in the first graph will be mapped by f to a vertex adjacent to $f(v)$ in the isomorphic graph. That is, v and $f(v)$ will have the same degree. So if one graph has a vertex of degree 4 and another does not, then they can't be isomorphic. In fact, they can't be isomorphic if the number of degree 4 vertices in each of the graphs is not the same.

Looking for preserved properties can make it easy to determine that two graphs are not isomorphic, or to actually find an isomorphism between them, if there is one. In practice, it's frequently easy to decide whether two graphs are isomorphic. However, no one has yet found a *general* procedure for determining whether two graphs are isomorphic that is *guaranteed* to run much faster than an exhaustive (and exhausting) search through all possible bijections between their vertices.

Having an efficient procedure to detect isomorphic graphs would, for example, make it easy to search for a particular molecule in a database given the molecular bonds. On the other hand, knowing there is no such efficient procedure would also be valuable: secure protocols for encryption and remote authentication can be built on the hypothesis that graph isomorphism is computationally exhausting.

7.1.6 Problems

Class Problems

Homework Problems

Exam Problems

7.2 The Stable Marriage Problem

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

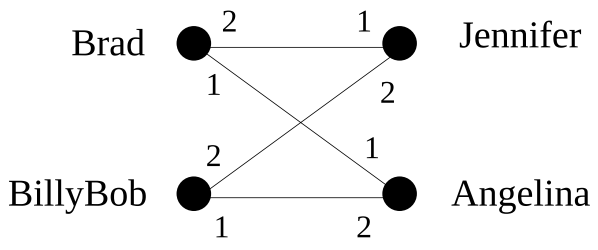
7.2.1 The Problem

Suppose there are a bunch of boys and an equal number of girls that we want to marry off. Each boy has his personal preferences about the girls —in fact, we assume he has a complete list of all the girls ranked according to his preferences, with no ties. Likewise, each girl has a ranked list of all of the boys.

The preferences don't have to be symmetric. That is, Jennifer might like Brad best, but Brad doesn't necessarily like Jennifer best. The goal is to marry off boys and girls: every boy must marry exactly one girl and vice-versa—no polygamy. In mathematical terms, we want the mapping from boys to their wives to be a bijection or *perfect matching*. We'll just call this a "matching," for short.

Here's the difficulty: suppose *every* boy likes Angelina best, and every girl likes Brad best, but Brad and Angelina are married to other people, say Jennifer and Billy Bob. Now *Brad and Angelina prefer each other to their spouses*, which puts their marriages at risk: pretty soon, they're likely to start spending late nights is study sessions together `:-)` .

This situation is illustrated in the following diagram where the digits "1" and "2" near a boy shows which of the two girls he ranks first and which second, and similarly for the girls:



More generally, in any matching, a boy and girl who are not married to each other and who like each other better than their spouses, is called a *rogue couple*. In the situation above, Brad and Angelina would be a rogue couple.

Having a rogue couple is not a good thing, since it threatens the stability of the marriages. On the other hand, if there are no rogue couples, then for any boy and girl who are not married to each other, at least one likes their spouse better than the other, and so won't be tempted to start an affair.

Definition 7.2.1. A *stable matching* is a matching with no rogue couples.

The question is, given everybody's preferences, how do you find a stable set of marriages? In the example consisting solely of the four people above, we could let Brad and Angelina both have their first choices by marrying each other. Now neither Brad nor Angelina prefers anybody else to their spouse, so neither will be in a rogue couple. This leaves Jen not-so-happily married to Billy Bob, but neither Jen nor Billy Bob can entice somebody else to marry them.

It is something of a surprise that there always is a stable matching among a group of boys and girls, but there is, and we'll shortly explain why. The surprise springs in part from considering the apparently similar "buddy" matching problem. That is, if people can be paired off as buddies, regardless of gender, then a stable matching *may not* be possible. For example, Figure 7.2 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:

Lemma. *There is no stable buddy matching among the four people in Figure 7.2.*

Proof. We'll prove this by contradiction.

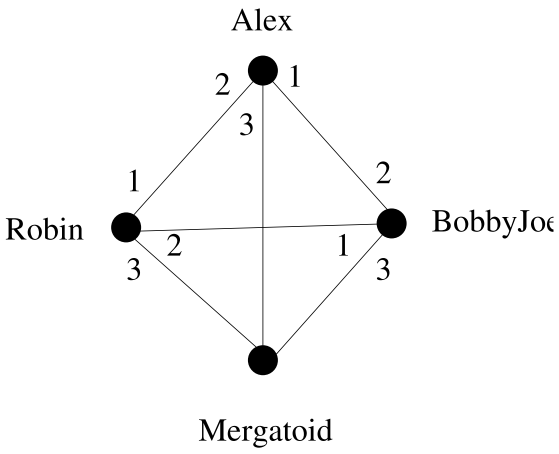


Figure 7.2: Some preferences with no stable buddy matching.

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.

7.2.2 The Mating Ritual

The procedure for finding a stable matching involves a *Mating Ritual* that takes place over several days. The following events happen each day:

Morning: Each girl stands on her balcony. Each boy stands under the balcony of his favorite among the girls on his list, and he serenades her. If a boy has no girls left on his list, he stays home and does his 6.042 homework.

Afternoon: Each girl who has one or more suitors serenading her, says to her favorite among them, "We might get engaged. Come back tomorrow." To the other suitors, she says, "No. I will never marry you! Take a hike!"

Evening: Any boy who is told by a girl to take a hike, crosses that girl off his list.

Termination condition: When every girl has at most one suitor, the ritual ends with each girl marrying her suitor, if she has one.

There are a number of facts about this Mating Ritual that we would like to prove:

- The Ritual has a last day.
- Everybody ends up married.
- The resulting marriages are stable.

7.2.3 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 7.2.2. A *Marriage Problem* consists of two disjoint sets of the same finite size, called the-Boys and the-Girls. The members of the-Boys are called *boys*, and members of the-Girls are called *girls*. For each boy, B, there is a strict total order, $<_B$, on the-Girls, and for each girl, G, there is a strict total order, $<_G$, on the-Boys. If $G_1 <_B G_2$ we say B *prefers* girl G_2 to girl G_1. Similarly, if $B_1 <_G B_2$ we say G

prefers boy B_2 to boy B_1.

A *marriage assignment* or *perfect matching* is a bijection, w : the-Boys $\rightarrow$ the-Girls. If $B \in$ the-Boys, then $w(B)$ is called B's *wife* in the assignment, and if $G \in$ the-Girls, then $w^{-1}(G)$ is called G's *husband*. A *rogue couple* is a boy, B, and a girl, G, such that B prefers G to his wife, and G prefers B to her husband. An assignment is *stable* if it has no rogue couples. A *solution* to a marriage problem is a stable perfect matching.

To model the Mating Ritual with a state machine, we make a key observation: to determine what happens on any day of the Ritual, all we need to know is which girls are still on which boys' lists on the morning of that day. So we define a state to be some mathematical data structure providing this information. For example, we could define a state to be the "still-has-on-his-list" relation, R, between boys and girls, where $B \ R \ G$ means girl G is still on boy B's list.

We start the Mating Ritual with no girls crossed off. That is, the start state is the *complete bipartite* relation in which every boy is related to every girl.

According to the Mating Ritual, on any given morning, a boy will *serenade* the girl he most prefers among those he has not as yet crossed out. Mathematically, the girl he is serenading is just the maximum among the girls on B's list, ordered by $<_B$. (If the list is empty, he's not serenading anybody.) A girl's *favorite* is just the maximum, under her preference ordering, among the boys serenading her.

Continuing in this way, we could mathematically specify a precise Mating Ritual state machine, but we won't bother. The intended behavior of the Mating Ritual is clear enough that we don't gain much by giving a formal state machine, so we stick to a more memorable description in terms of boys, girls, and their preferences. The point is, though, that it's not hard to define everything using basic mathematical data structures like sets, functions, and relations, if need be.

7.2.4 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.

7.2.5 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 7.2.3. *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 7.2.4. *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. ■

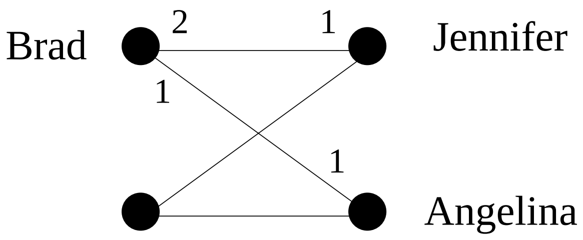
7.2.6 ...Especially the Boys

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 7.2.5. 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 7.2.6. *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 7.2.7. *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 7.2.6, 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. ■

7.2.7 Applications

Not surprisingly, a stable matching procedure is used by at least one large dating agency. But although "boy-girl-marriage" terminology is traditional and makes some of the definitions easier to remember (we hope without offending anyone), solutions to the Stable Marriage Problem are widely useful.

The Mating Ritual was first announced in a paper by D. Gale and L.S. Shapley in 1962, but ten years before the Gale-Shapley paper was appeared, and unknown by them, the Ritual was being used to assign residents to hospitals by the National Resident Matching Program (NRMP). The NRMP has, since the turn of the twentieth century, assigned each year's pool of medical school graduates to hospital residencies (formerly called "internships") with hospitals and graduates playing the roles of boys and girls. (In this case there may be multiple boys married to one girl, but there's an easy way to use the Ritual in this situation (see Problem **??**). Before the Ritual was adopted, there were chronic disruptions and awkward countermeasures taken to preserve assignments of graduates to residencies. The Ritual resolved these problems so successfully, that it was used essentially without change at least through 1989.[2]

The internet infrastructure company, Akamai, also uses a variation of the Gale-Shapley procedure to assign web traffic to servers. In the early days, Akamai used other combinatorial optimization algorithms that got to be too slow as the number of servers (over 20,000 in 2010) reference needed and traffic increased. Akamai switched to Gale-Shapley since it is fast and can be run in a distributed manner. In this case, the web traffic corresponds to the boys and the web servers to the

[2]Much more about the Stable Marriage Problem can be found in the very readable mathematical monograph by Dan Gusfield and Robert W. Irving, The Stable Marriage Problem: Structure and Algorithms, MIT Press, Cambridge, Massachusetts, 1989, 240 pp.

girls. The servers have preferences based on latency and packet loss; the traffic has preferences based on the cost of bandwidth.

7.2.8 Problems

Practice Problems

Class Problems

Homework Problems

EDITING NOTE: Add problem proving that the Mating Ritual need not proceed in morning/afternoon/evening lock step: a girl can reject nonfavorite suitors one at a time and at any time, and a rejected boy can change the girl he serenades without waiting for the other boys to change. The proof uses the fact that single actions commute, so induction proves that all executions are confluent —which implies all executions end with the same boy-optimal matching. This lemma can be cited in the planar graphs section to prove that the edges in an embedding can be added in any order. ■

7.3 Connectedness

7.3.1 Paths and Simple Cycles

Paths in simple graphs are esentially the same as paths in digraphs. We just modify the digraph definitions using undirected edges instead of directed ones. For example, the formal definition of a path in a simple graph is a virtually that same as Definition 8.1.1 of paths in digraphs:

Definition 7.3.1. A *path* in a graph, G, is a sequence of $k \geq 0$ vertices

$$v_0, \ldots, v_k$$

such that v_i—v_{i+1} is an edge of G for all i where $0 \leq i < k$. The path is said to *start* at v_0, to *end* at v_k, and the *length* of the path is defined to be k.

An edge, u—v, is *traversed* n times by the path if there are n different values of i such that v_i—$v_{i+1} = u$—v. The path is *simple*[3] iff all the v_i's are different, that is,

[3]Heads up: what we call "paths" are commonly referred to in graph theory texts as "walks," and simple paths are referred to as just "paths". Likewise, what we will call *cycles* and *simple cycles* are commonly called "closed walks" and just "cycles".

if $i \neq j$ implies $v_i \neq v_j$.

For example, the graph in Figure 7.3 has a length 6 simple path A,B,C,D,E,F,G. This is the longest simple path in the graph.

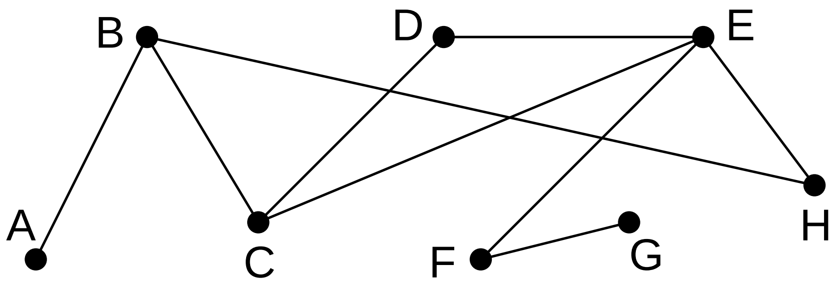


Figure 7.3: *A graph with 3 simple cycles.*

As in digraphs, the length of a path is the total number of times it traverses edges, which is *one less* than its length as a sequence of vertices. For example, the length 6 path A,B,C,D,E,F,G is actually a sequence of seven vertices.

A *cycle* can be described by a path that begins and ends with the same vertex. For example, B,C,D,E,C,B is a cycle in the graph in Figure 7.3. 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 7.3 has three simple cycles B,H,E,C,B and C,D,E,C and B,C,D,E,H,B. More precisely, a simple cycle is a cycle that can be described by a path of length at least three whose vertices are all different except for the beginning and end vertices. So in contrast to simple *paths*, the length of a simple *cycle* is the *same* as the number of distinct vertices that appear in it.

From now on we'll stop being picky about distinguishing a cycle from a path that describes it, and we'll just refer to the path as a cycle. [4]

Simple cycles are especially important, so we will give a proper definition of them. Namely, we'll define a simple cycle in G to be a *subgraph* of G that looks like a cycle that doesn't cross itself. Formally:

Definition 7.3.2. A *subgraph*, G', of a graph, G, is a graph whose vertices, V', are a subset of the vertices of G and whose edges are a subset of the edges of G.

Notice that since a subgraph is itself a graph, the endpoints of every edge of G' must be vertices in V'.

Definition 7.3.3. For $n \geq 3$, let C_n be the graph with vertices $1, \ldots, n$ and edges

$$1\text{—}2, \;\; 2\text{—}3, \;\; \ldots, \;\; (n-1)\text{—}n, \;\; n\text{—}1.$$

A graph is a *simple cycle* of length n iff it is isomorphic to C_n for some $n \geq 3$. A *simple cycle of a graph*, G, is a subgraph of G that is a simple cycle.

[4]Technically speaking, we haven't ever defined what a cycle *is*, only how to describe it with paths. But we won't need an abstract definition of cycle, since all that matters about a cycle is which paths describe it.

This definition formally captures the idea that simple cycles don't have direction or beginnings or ends.

7.3.2 Connected Components

Definition 7.3.4. Two vertices in a graph are said to be *connected* when there is a path that begins at one and ends at the other. By convention, every vertex is considered to be connected to itself by a path of length zero.

EDITING NOTE:

Now if there is a path from vertex u to vertex v, then v is connected to u by the reverse path, so connectedness is a symmetric relation. Also, if there is a path from u to v, and also a path from v to w, then these two paths can be combined to form a path from u to w. So the connectedness relation is transitive. It is also reflexive, since every vertex is by definition connected to itself by a path of length zero.

■

The diagram in Figure 7.5 looks like a picture of three graphs, but is intended to be a picture of *one* graph. This graph consists of three pieces (subgraphs). Each piece by itself is connected, but there are no paths between vertices in different pieces.

EDITING NOTE:

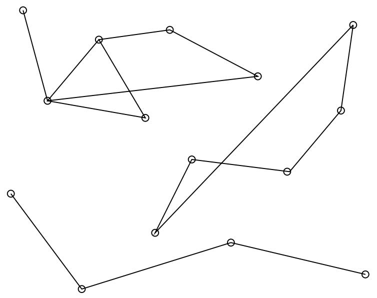


Figure 7.4: *One graph with 3 connected components.*

■

Definition 7.3.5. A graph is said to be *connected* when every pair of vertices are connected.

These connected pieces of a graph are called its *connected components*. A rigorous definition is easy: a connected component is the set of all the vertices connected

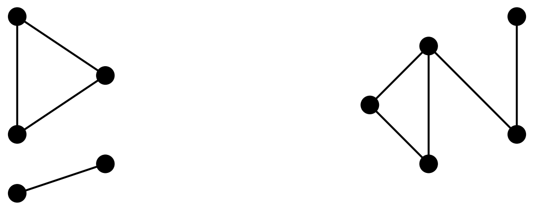


Figure 7.5: *One graph with 3 connected components.*

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.

7.3.3 How Well Connected?

If we think of a graph as modelling cables in a telephone network, or oil pipelines, or electrical power lines, then we not only want connectivity, but we want connectivity that survives component failure. A graph is called *k-edge connected* if it takes at least k "edge-failures" to disconnect it. More precisely:

Definition 7.3.6. Two vertices in a graph are *k-edge connected* if they remain connected in every subgraph obtained by deleting $k - 1$ edges. A graph with at least two vertices is k-edge connected[5] if every two of its vertices are k-edge connected.

So 1-edge connected is the same as connected for both vertices and graphs. Another way to say that a graph is k-edge connected is that every subgraph obtained from it by deleting at most $k - 1$ edges is connected. For example, in the graph in Figure 7.3, 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 even takes some ingenuity to prove this for the

[5]The corresponding definition of connectedness based on deleting vertices rather than edges is common in Graph Theory texts and is usually simply called "k-connected" rather than "k-vertex connected."

case $k = 2$.

7.3.4 Connection by Simple Path

Where there's a path, there's a simple path. This is sort of obvious, but it's easy enough to prove rigorously using the Well Ordering Principle.

Lemma 7.3.7. *If vertex u is connected to vertex v in a graph, then there is a simple path from u to v.*

Proof. Since there is a path from u to v, there must, by the Well-ordering Principle, be a minimum length path from u to v. If the minimum length is zero or one, this minimum length path is itself a simple path from u to v. Otherwise, there is a minimum length path

$$v_0, v_1, \ldots, v_k$$

from $u = v_0$ to $v = v_k$ where $k \geq 2$. We claim this path must be simple. To prove the claim, suppose to the contrary that the path is not simple, that is, some vertex on the path occurs twice. This means that there are integers i, j such that

$0 \le i < j \le k$ 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 7.3.8. *For any path of length k in a graph, there is a simple path of length* at most *k with the same endpoints.*

7.3.5 The Minimum Number of Edges in a Connected Graph

The following theorem says that a graph with few edges must have many connected components.

Theorem 7.3.9. *Every graph with v vertices and e edges has at least $v - e$ connected components.*

Of course for Theorem 7.3.9 to be of any use, there must be fewer edges than vertices.

Proof. We use induction on the number of edges, e. Let $P(e)$ be the proposition that

> for every v, every graph with v vertices and e edges has at least $v - e$ connected components.

Base case:($e = 0$). In a graph with 0 edges and v vertices, each vertex is itself a connected component, and so there are exactly $v = v - 0$ connected components. So $P(e)$ holds.

Inductive step: Now we assume that the induction hypothesis holds for every e-edge graph in order to prove that it holds for every $(e+1)$-edge graph, where $e \geq 0$. Consider a graph, G, with $e + 1$ edges and k vertices. We want to prove that G has at least $v - (e + 1)$ connected components. To do this, remove an arbitrary edge a—b and call the resulting graph G'. By the induction assumption, G' has at least $v - e$ connected components. Now add back the edge a—b to obtain the original graph G. If a and b were in the same connected component of G', then G has the same connected components as G', so G has at least $v - e > v - (e + 1)$ components. Otherwise, if a and b were in different connected components of G', then these two components are merged into one in G, but all other components remain unchanged, reducing the number of components by 1. Therefore, G has at least $(v-e)-1 = v-(e+1)$ connected components. So in either case, $P(e+1)$ holds. This completes the Inductive step. The theorem now follows by induction. ■

Corollary 7.3.10. *Every connected graph with v vertices has at least $v - 1$ edges.*

A couple of points about the proof of Theorem 7.3.9 are worth noticing. First, we used induction on the number of edges in the graph. This is very common in proofs involving graphs, and so is induction on the number of vertices. When you're presented with a graph problem, these two approaches should be among the first you consider. The second point is more subtle. Notice that in the inductive step, we took an arbitrary $(n + 1)$-edge graph, threw out an edge so that we could apply the induction assumption, and then put the edge back. You'll see this shrink-down, grow-back process very often in the inductive steps of proofs related to graphs. This might seem like needless effort; why not start with an n-edge graph and add one more to get an $(n + 1)$-edge graph? That would work fine in this case, but opens the door to a nasty logical error called *buildup error*, illustrated in Problems **??** and **??**. Always use shrink-down, grow-back arguments, and you'll never fall into this trap.

7.3.6 Problems

Class Problems

Homework Problems

Homework Problems

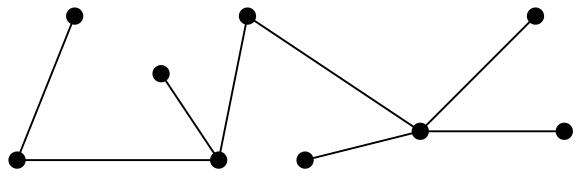
7.4 Trees

Trees are a fundamental data structure in computer science, and there are many kinds, such as rooted, ordered, and binary trees. In this section we focus on the purest kind of tree. Namely, we use the term *tree* to mean a connected graph without simple cycles.

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

7.4.1 Tree Properties

Here is an example of a tree:



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

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

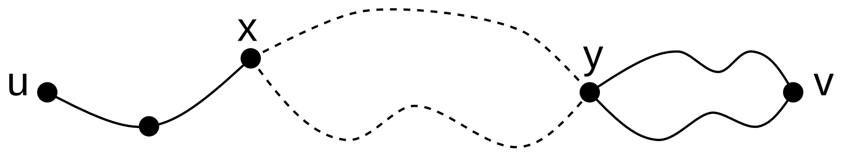
Theorem 7.4.1. *Every tree has the following properties:*

1. *Any connected subgraph is a tree.*
2. *There is a unique simple path between every pair of vertices.*
3. *Adding an edge between two vertices creates a cycle.*
4. *Removing any edge disconnects the graph.*
5. *If it has at least two vertices, then it has at least two leaves.*
6. *The number of vertices is one larger than the number of edges.*

Proof.

1. A simple cycle in a subgraph is also a simple cycle in the whole graph, so any subgraph of an acyclic graph must also be acyclic. If the subgraph is also connected, then by definition, it is a tree.

2. There is at least one path, and hence one simple path, between every pair of vertices, because the graph is connected. Suppose that there are two different simple paths between vertices u and v. Beginning at u, let x be the first vertex where the paths diverge, and let y be the next vertex they share. Then there are two simple paths from x to y with no common edges, which defines a simple cycle. This is a contradiction, since trees are acyclic. Therefore, there is exactly one simple path between every pair of vertices.



3. An additional edge u—v together with the unique path between u and v forms a simple cycle.

4. Suppose that we remove edge u—v. Since the tree contained a unique path between u and v, that path must have been u—v. Therefore, when that edge is removed, no path remains, and so the graph is not connected.

5. Let $v_1, \ldots, v_m$ be the sequence of vertices on a longest simple path in the tree. Then $m \geq 2$, since a tree with two vertices must contain at least one edge. There cannot be an edge v_1—v_i for $2 < i \leq m$; otherwise, vertices $v_1, \ldots, v_i$ would from a simple cycle. Furthermore, there cannot be an edge u—v_1 where u is not on the path; otherwise, we could make the path longer. Therefore, the only edge incident to v_1 is v_1—v_2, which means that v_1 is a leaf. By a symmetric argument, v_m is a second leaf.

6. We use induction on the number of vertices. For a tree with a single vertex, the claim holds since it has no edges and $0 + 1 = 1$ vertex. Now suppose that the claim holds for all n-vertex trees and consider an $(n+1)$-vertex tree, T. Let v be a leaf of the tree. You can verify that deleting a vertex of degree 1 (and its incident edge) from any connected graph leaves a connected subgraph. So by 1., deleting v and its incident edge gives a smaller tree, and this smaller tree has one more vertex than edge by induction. If we re-attach the vertex, v, and its incident edge, then the equation still holds because the number of

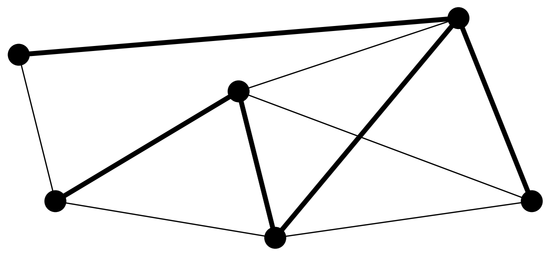
vertices and number of edges both increase by 1. Thus, the claim holds for T and, by induction, for all trees.

■

Various subsets of these properties provide alternative characterizations of trees, though we won't prove this. For example, a *connected* graph with a number of vertices one larger than the number of edges is necessarily a tree. Also, a graph with unique paths between every pair of vertices is necessarily a tree.

7.4.2 Spanning Trees

Trees are everywhere. In fact, every connected graph contains a subgraph that is a tree with the same vertices as the graph. This is a called a *spanning tree* for the graph. For example, here is a connected graph with a spanning tree highlighted.



Theorem 7.4.2. *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:

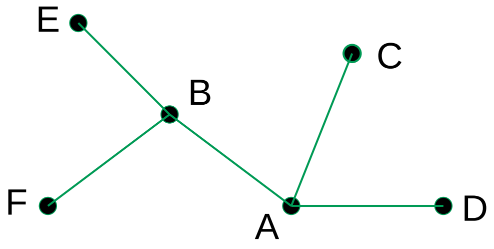
$$v_0\text{—}v_1, v_1\text{—}v_2, \ldots, v_n\text{—}v_0$$

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. ■

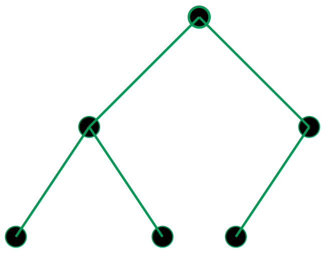
EDITING NOTE:

Tree Variations

Trees come up often in computer science. For example, information is often stored in tree-like data structures and the execution of many recursive programs can be regarded as a traversal of a tree. There are many varieties of trees. For example, a *rooted tree* is a tree with one vertex identified as the *root*. Let u—v be an edge in a rooted tree such that u is closer to the root than v. Then u is the *parent* of v, and v is a *child* of u.

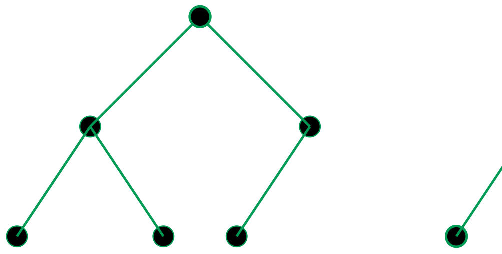


In the tree above, suppose that we regard vertex A as the root. Then E and F are the children of B, and A is the parent of B, C, and D. A *binary* tree is a rooted tree in which every vertex has at most two children. Here is an example, where the top-



most vertex is the root.

In an *ordered, binary* tree, the children of a vertex v are distinguished. One is called the *left child* of v, and the other is called the *right child*. For example, if we regard the two binary trees below as unordered, then they are equivalent. However, if we regard these trees as ordered, then they are different.



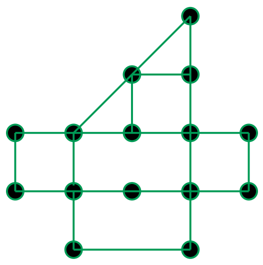
■

EDITING NOTE:

Problem **??** presents most of this as a homework problem.

Traversing a Graph

Can you walk every hallway in the Museum of Fine Arts *exactly once*? If we represent hallways and intersections with edges and vertices, then this reduces to a question about graphs. For example, could you visit every hallway exactly once in a museum with this floorplan?



Euler Tours and Hamiltonian Cycles

The entire field of graph theory began when Euler asked whether the seven bridges of Königsberg could all be traversed exactly once— essentially the same question we asked about the Museum of Fine Arts. In his honor, an *Euler walk* is a defined to be a path that traverses every edge in a graph exactly once. Similarly, an *Euler tour* is an Euler walk that starts and finishes at the same vertex, that is a cycle that traverses every edge exactly once. Graphs with Euler tours and Euler walks both have simple characterizations.

Theorem 7.4.3. *A graph has an Euler tour iff it is connected and every vertex has even degree.*

Proof. Suppose a graph has an Euler tour. Every pair of vertices must appear in the tour, so the graph is connected. Moreover, a vertex that appears k times in the tour must have degree $2k$, so every vertex of the graph has even degree.

Unconvincing!

Conversely, suppose every vertex in a graph, G, has even degree. Let $W = (v_0, \dots, v_n)$ be the longest path in G that traverses every edge *at most* once. Now W must traverse every edge incident to v_n; otherwise, the path could be extended. In particular, the W traverses two of these edges each time it passes through v_n, and it traverses v_{n-1}—v_n at the end. This accounts for an odd number of edges, but the degree of v_n is even by assumption. Therefore, the W must also begin at v_n; that is, $v_0 = v_n$. Suppose that W is not an Euler tour. Because G is a connected graph, we can find an edge not in W but incident to some vertex in W. Call this edge u—v_i. But then we can construct a longer walk:

$$u, u\text{—}v_i, v_i, v_i\text{—}v_{i+1}, \ldots, v_{n-1}\text{—}v_n, v_n, v_0\text{—}v_1, \ldots, v_{i-1}\text{—}v_i, v_i$$

This contradicts the definition of W, so W must be an Euler tour after all. ■

Corollary 7.4.4. *A connected graph has an Euler walk if and only if either 0 or 2 vertices have odd degree.*

Hamiltonian cycles are the unruly cousins of Euler tours. A *Hamiltonian cycle* is walk that starts and ends at the same vertex and visits every *vertex* in a graph exactly once. There is no simple characterization of all graphs with a Hamiltonian cycle. In fact, determining whether a given graph has a Hamiltonian cycle is is the same category of problem as the SAT problem of Section 1.5: you get a million dollars for finding an efficient way to determine when a graph has a Hamiltonian cycle —or for proving that no procedure works efficiently on all graphs.

■

7.4.3 Problems

Class Problems

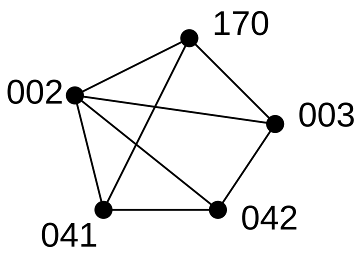
Homework Problems

7.5 Coloring Graphs

In section 7.1.2, we used edges to indicate an affinity between two nodes, but having an edge represent a *conflict* between two nodes also turns out to be really useful.

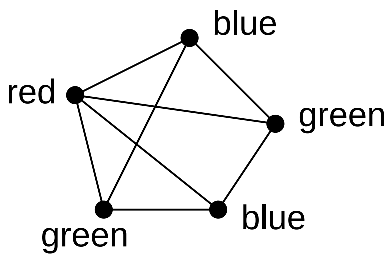
7.6 Modelling Scheduling Conflicts

Each term the MIT Schedules Office must assign a time slot for each final exam. This is not easy, because some students are taking several classes with finals, and a student can take only one test during a particular time slot. The Schedules Office wants to avoid all conflicts. Of course, you can make such a schedule by having every exam in a different slot, but then you would need hundreds of slots for the hundreds of courses, and exam period would run all year! So, the Schedules Office would also like to keep exam period short. The Schedules Office's problem is easy to describe as a graph. There will be a vertex for each course with a final exam, and two vertices will be adjacent exactly when some student is taking both courses. For example, suppose we need to schedule exams for 6.041, 6.042, 6.002, 6.003 and 6.170. The scheduling graph might look like this:



6.002 and 6.042 cannot have an exam at the same time since there are students in both courses, so there is an edge between their nodes. On the other hand, 6.042 and 6.170 can have an exam at the same time if they're taught at the same time (which they sometimes are), since no student can be enrolled in both (that is, no student *should* be enrolled in both when they have a timing conflict). Next, identify each time slot with a color. For example, Monday morning is red, Monday afternoon is blue, Tuesday morning is green, etc.

Assigning an exam to a time slot is now equivalent to coloring the corresponding vertex. The main constraint is that *adjacent vertices must get different colors* — otherwise, some student has two exams at the same time. Furthermore, in order to keep the exam period short, we should try to color all the vertices using as *few different colors as possible*. For our example graph, three colors suffice:



This coloring corresponds to giving one final on Monday morning (red), two Monday afternoon (blue), and two Tuesday morning (green). Can we use fewer than three colors? No! We can't use only two colors since there is a triangle in the graph, and three vertices in a triangle must all have different colors.

This is an example of what is a called a *graph coloring problem*: given a graph G, assign colors to each node such that adjacent nodes have different colors. A color assignment with this property is called a *valid coloring* of the graph —a "*coloring*," for short. A graph G is *k-colorable* if it has a coloring that uses at most k colors.

Definition 7.6.1. The minimum value of k for which a graph, G, has a valid coloring is called its *chromatic number*, $\chi(G)$.

In general, trying to figure out if you can color a graph with a fixed number of colors can take a long time. It's a classic example of a problem for which no fast algorithms are known. In fact, it is easy to check if a coloring works, but it seems really hard to find it (if you figure out how, then you can get a $1 million Clay prize).

7.6.1 Degree-bounded Coloring

There are some simple graph properties that give useful upper bounds on colorings. For example, if we have a bound on the degrees of all the vertices in a graph, then we can easily find a coloring with only one more color than the degree bound.

Theorem 7.6.2. *A graph with maximum degree at most k is $(k+1)$-colorable.*

Unfortunately, if you try induction on k, it will lead to disaster. It is not that it is impossible, just that it is extremely painful and would ruin you if you tried it on an exam. Another option, especially with graphs, is to change what you are inducting on. In graphs, some good choices are n, the number of nodes, or e, the number of edges.

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

Base case: $(n = 1)$ A 1-vertex graph has maximum degree 0 and is 1-colorable, so $P(1)$ is true.

Inductive step: Now assume that $P(n)$ is true, and let G be an $(n+1)$-vertex graph with maximum degree at most k. Remove a vertex v (and all edges incident to it), leaving an n-vertex subgraph, H. The maximum degree of H is at most k, and so H is $(k+1)$-colorable by our assumption $P(n)$. Now add back vertex v. We can assign v a color different from all its adjacent vertices, since there are at most k adjacent vertices and $k+1$ colors are available. Therefore, G is $(k+1)$-colorable.

This completes the inductive step, and the theorem follows by induction. ■

Sometimes $k + 1$ colors is the best you can do. For example, in the complete graph, K_n, every one of its n vertices is adjacent to all the others, so all n must be assigned different colors. Of course n colors is also enough, so $\chi(K_n) = n$. So K_{k+1} is an example where Theorem 7.6.2 gives the best possible bound. This means that Theorem 7.6.2 also gives the best possible bound for *any* graph with degree bounded by k that has K_{k+1} as a subgraph.

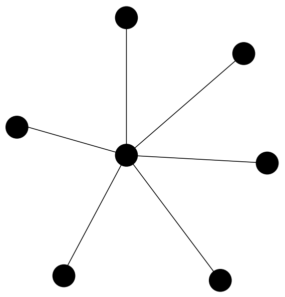
EDITING NOTE:

The complete graph, K_n, is also called a size n *clique*, just like a clique of friends, where nodes represent the people and an edge represents the friendship relationship.[6] ■

But sometimes $k+1$ colors is far from the best that you can do. Here's an exam-

[6] When speaking of friends, clique is usually pronounced similar to click. However, for some reason, graph theorists think that the word clique rhymes with geek.

ple 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!

7.6.2 Why coloring?

One reason coloring problems come all the time is because scheduling conflicts are so common. For example, at Akamai, a new version of software is deployed over each of 20,000 servers every few days. The updates cannot be done at the same time since the servers need to be taken down in order to deploy the software. Also, the servers cannot be handled one at a time, since it would take forever to update them all (each one takes about an hour). Moreover, certain pairs of servers cannot be taken down at the same time since they have common critical functions.

This problem was eventually solved by making a 20,000 node conflict graph and coloring it with 8 colors – so only 8 waves of install are needed! Another example comes from the need to assign frequencies to radio stations. If two stations have an overlap in their broadcast area, they can't be given the same frequency. Frequencies are precious and expensive, so you want to minimize the number handed out. This amounts to finding the minimum coloring for a graph whose vertices are the stations and whose edges are between stations with overlapping areas.

Coloring also comes up in allocating registers for program variables. While a variable is in use, its value needs to be saved in a register, but registers can often be reused for different variables. But two variables need different registers if they are referenced during overlapping intervals of program execution. So register allocation is the coloring problem for a graph whose vertices are the variables; vertices are adjacent if their intervals overlap, and the colors are registers.

Finally, there's the famous map coloring problem stated in Propostion 1.3.4. The question is how many colors are needed to color a map so that adjacent territories get different colors? This is the same as the number of colors needed to color a graph that can be drawn in the plane without edges crossing. A proof that four colors are enough for the *planar* graphs was acclaimed when it was discovered about thirty years ago. Implicit in that proof was a 4-coloring procedure that takes time proportional to the number of vertices in the graph (countries in the map). On the other hand, it's another of those million dollar prize questions to find an efficient procedure to tell if a planar graph really *needs* four colors or if three will actually do the job. But it's always easy to tell if an *arbitrary* graph is 2-colorable, as we show in section 7.7. Finally, in section 7.8, we'll develop enough planar graph theory to present an easy proof at least that planar graphs are 5-colorable.

7.6.3 Problems

Class Problems

Homework Problems

Exam Problems

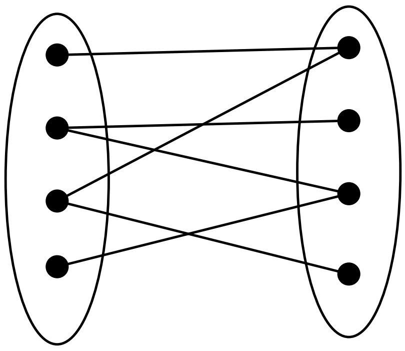
7.7 Bipartite Matchings

7.7.1 Bipartite Graphs

There were two kinds of vertices in the "Sex in America" graph —males and females, and edges only went between the two kinds. Graphs like this come up so frequently they have earned a special name —they are called *bipartite graphs*.

Definition 7.7.1. A *bipartite graph* is a graph together with a partition of its vertices into two sets, L and R, such that every edge is incident to a vertex in L and to a vertex in R.

So every bipartite graph looks something like this:



Now we can immediately see how to color a bipartite graph using only two colors: let all the L vertices be black and all the R vertices be white. Conversely, if a graph is 2-colorable, then it is bipartite with L being the vertices of one color and R the vertices of the other color. In other words,

> "bipartite" is a synonym for "*2-colorable*."

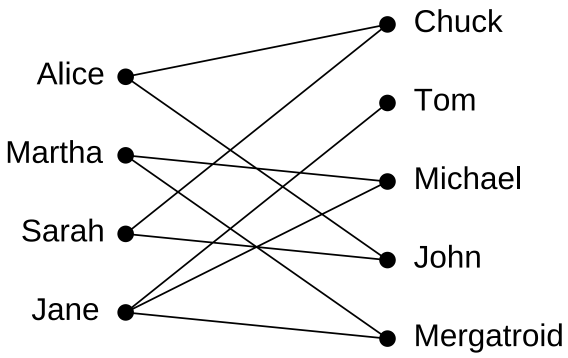
The following Lemma gives another useful characterization of bipartite graphs.

Theorem 7.7.2. *A graph is bipartite iff it has no odd-length cycle.*

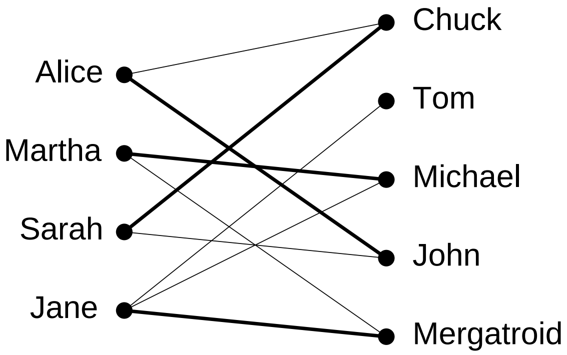
The proof of Theorem 7.7.2 is left to Problem **??**.

7.7.2 Bipartite Matchings

The *bipartite matching* problem resembles the stable Marriage Problem in that it concerns a set of girls and a set of at least as many boys. There are no preference lists, but each girl does have some boys she likes and others she does not like. In the bipartite matching problem, we ask whether every girl can be paired up with a boy that she likes. Any particular matching problem can be specified by a bipartite graph with a vertex for each girl, a vertex for each boy, and an edge between a boy and a girl iff the girl likes the boy. For example, we might obtain the following graph:



Now a *matching* will mean a way of assigning every girl to a boy so that different girls are assigned to different boys, and a girl is always assigned to a boy she likes. For example, here is one possible matching for the girls:



Hall's Matching Theorem states necessary and sufficient conditions for the existence of a matching in a bipartite graph. It turns out to be a remarkably useful mathematical tool.

7.7.3 The Matching Condition

We'll state and prove Hall's Theorem using girl-likes-boy terminology. Define *the set of boys liked by a given set of girls* to consist of all boys liked by at least one of those girls. For example, the set of boys liked by Martha and Jane consists of Tom, Michael, and Mergatroid. For us to have any chance at all of matching up the girls, the following *matching condition* must hold:

Every subset of girls likes at least as large a set of boys.

For example, we can not find a matching if some 4 girls like only 3 boys. Hall's Theorem says that this necessary condition is actually sufficient; if the matching condition holds, then a matching exists.

Theorem 7.7.3. *A matching for a set of girls G with a set of boys B can be found if and only if the matching condition holds.*

Proof. First, let's suppose that a matching exists and show that the matching condition holds. Consider an arbitrary subset of girls. Each girl likes at least the boy she is matched with. Therefore, every subset of girls likes at least as large a set of boys. Thus, the matching condition holds.

Next, let's suppose that the matching condition holds and show that a matching exists. We use strong induction on $|G|$, the number of girls.

Base Case: ($|G| = 1$) If $|G| = 1$, then the matching condition implies that the lone girl likes at least one boy, and so a matching exists.

Inductive Step: Now suppose that $|G| \geq 2$. There are two cases:

Case 1: Every proper subset of girls likes a *strictly larger* set of boys. In this case, we have some latitude: we pair an arbitrary girl with a boy she likes and send them both away. The matching condition still holds for the remaining boys and girls, so we can match the rest of the girls by induction.

Case 2: Some proper subset of girls $X \subset G$ likes an *equal-size* set of boys $Y \subset B$. We match the girls in X with the boys in Y by induction and send them all away. We can also match the rest of the girls by induction if we show that the matching condition holds for the remaining boys and girls. To check the matching condition for the remaining people, consider an arbitrary subset of the remaining girls $X' \subseteq (G - X)$, and let Y' be the set of remaining boys that they like. We must show that $|X'| \leq |Y'|$. Originally, the combined set of girls $X \cup X'$ liked the set of boys $Y \cup Y'$. So, by the matching condition, we know:

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

We sent away $|X|$ girls from the set on the left (leaving X') and sent away an equal number of boys from the set on the right (leaving Y'). Therefore, it must be that $|X'| \leq |Y'|$ as claimed.

So there is in any case a matching for the girls, which completes the proof of the Inductive step. The theorem follows by induction. ■

The proof of this theorem gives an algorithm for finding a matching in a bipartite graph, albeit not a very efficient one. However, efficient algorithms for finding a matching in a bipartite graph do exist. Thus, if a problem can be reduced to finding a matching, the problem is essentially solved from a computational perspective.

7.7.4 A Formal Statement

Let's restate Hall's Theorem in abstract terms so that you'll not always be condemned to saying, "Now this group of little girls likes at least as many little boys..."

A *matching* in a graph, G, is a set of edges such that no two edges in the set share a vertex. A matching is said to *cover* a set, L, of vertices iff each vertex in L

has an edge of the matching incident to it. In any graph, the set $N(S)$, of *neighbors*[7] of some set, S, of vertices is the set of all vertices adjacent to some vertex in S. That is,

$$N(S) ::= \{r \mid s\text{—}r \text{ is an edge for some } s \in S\}.$$

S is called a *bottleneck* if

$$|S| > |N(S)|.$$

Theorem 7.7.4 (Hall's Theorem). *Let G be a bipartite graph with vertex partition L, R. There is matching in G that covers L iff no subset of L is a bottleneck.*

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

[7]An equivalent definition of $N(S)$ uses relational notation: $N(S)$ is simply the image, SR, of S under the adjacency relation, R, on vertices of the graph.

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 7.7.5. A bipartite graph G with vertex partition L, R is *degree-constrained* if deg $(l) \geq$ deg (r) for every $l \in L$ and $r \in R$.

Now we can always find a matching in a degree-constrained bipartite graph.

Lemma 7.7.6. *Every degree-constrained bipartite graph satisifies the matching condition.*

Proof. Let S be any set of vertices in L. The number of edges incident to vertices in S is exactly the sum of the degrees of the vertices in S. Each of these edges is incident to a vertex in $N(S)$ by definition of $N(S)$. So the sum of the degrees of the vertices in $N(S)$ is at least as large as the sum for S. But since the degree of every vertex in $N(S)$ is at most as large as the degree of every vertex in S, there would have to be at least as many terms in the sum for $N(S)$ as in the sum for S. So there have to be at least as many vertices in $N(S)$ as in S, proving that S is not a bottleneck. So there are no bottlenecks, proving that the degree-constrained graph satisifies the matching condition. ■

Of course being degree-constrained is a very strong property, and lots of graphs that aren't degree-constrained have matchings. But we'll see examples of degree-constrained graphs come up naturally in some later applications.

7.7.5 Problems

Class Problems

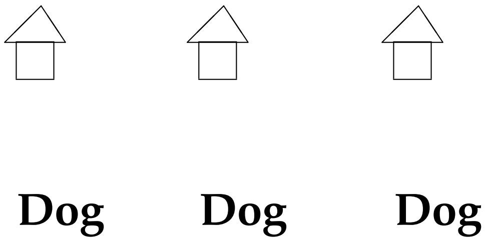
Exam Problems

Homework Problems

7.8 Planar Graphs

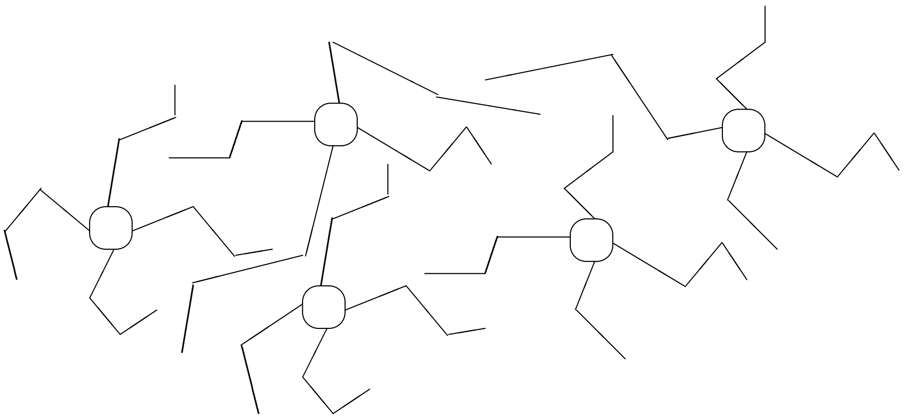
Drawing Graphs in the Plane

Here are three dogs and three houses.



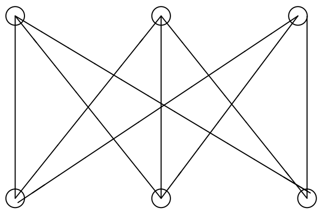
Can you find a path from each dog to each house such that no two paths intersect?

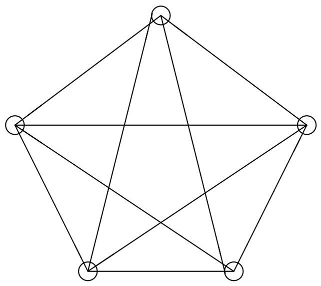
A *quadapus* is a little-known animal similar to an octopus, but with four arms. Here are five quadapi resting on the seafloor:



Can each quadapus simultaneously shake hands with every other in such a way that no arms cross?

Informally, a *planar graph* is a graph that can be drawn in the plane so that no edges cross, as in a map of showing the borders of countries or states. Thus, these two puzzles are asking whether the graphs below are planar; that is, whether they can be redrawn so that no edges cross. The first graph is called the *complete bipartite graph*, $K_{3,3}$, and the second is K_5.





In each case, the answer is, "No— but almost!" In fact, each drawing *would* be possible if any single edge were removed.

Planar graphs have applications in circuit layout and are helpful in displaying graphical data, for example, program flow charts, organizational charts, and scheduling conflicts. We will treat them as a recursive data type and use structural induction to establish their basic properties. Then we'll be able to describe a simple recursive procedure to color any planar graph with *five* colors, and also prove that there is no uniform way to place n satellites around the globe unless $n = 4, 6, 8, 12$, or 20.

EDITING NOTE: We will use them to prove a wonderful mathematical fact that was first proved by the ancient Greeks. ■

EDITING NOTE: One is rooted in human pyschology: many kinds of information can be presented as a graph (family relations, chemical structures, computer data structures, contact data for study of disease spread, flow of cash in money laundering trials, etc.). Big graphs are typically incomprehensible messes, but planar graphs are relatively easy for humans to grasp since there are no crisscrossing edges. ■

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.

EDITING NOTE: Finally, as we'll see shortly, planar graphs reveal a fundamental truth about the structure of our three-dimensional world. ■

7.8.1 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 7.6 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

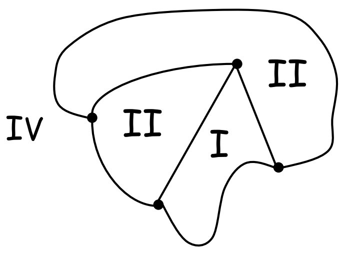


Figure 7.6: A Planar Drawing with Four Faces.

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 7.6 form a simple cycle. For example, labeling the vertices as in Figure 7.7, the simple cycles for the face boundaries are

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

Since every edge in the drawing appears on the boundaries of exactly two continuous faces, every edge of the simple graph appears on exactly two of the simple cycles.

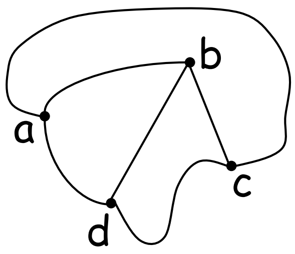


Figure 7.7: The Drawing with Labelled Vertices.

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 7.8. Now the cycle around the outer face is

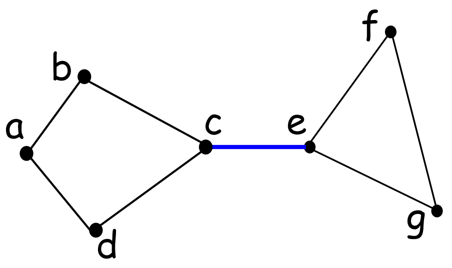


Figure 7.8: A Planar Drawing with a *Bridge*.

`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 7.9. Now the cycle around

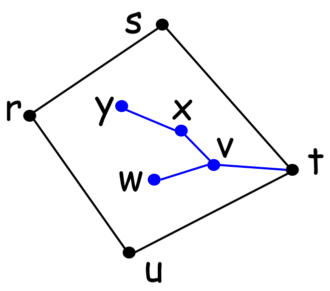


Figure 7.9: A Planar Drawing with a *Dongle*.

the inner face is

`rstvxyxvwvtur`,

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

But bridges and dongles are really the only complications, which leads us to the discrete data type of *planar embeddings* that we can use in place of continuous planar drawings. Namely, we'll define a planar embedding recursively to be the set of boundary-tracing cycles we could get drawing one edge after another.

7.8.2 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 7.8.1. 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[8]

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

as illustrated in Figure 7.10.

- **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

$$a \dots a.$$

Similarly, let b be a vertex on a discrete face, δ, in the embedding of H, so δ is

[8] 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.

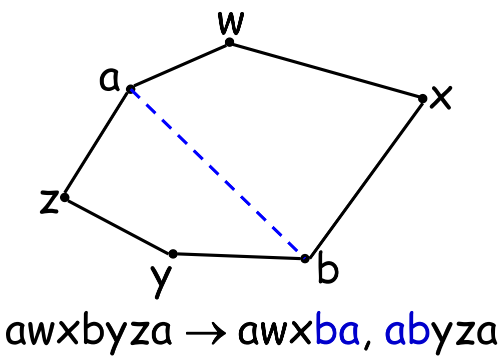


Figure 7.10: The Split a Face Case.

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 7.11, where the faces of G and H are:

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

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

{axyzabtuvwba, axya, ayza, btvwb, tuvt}.

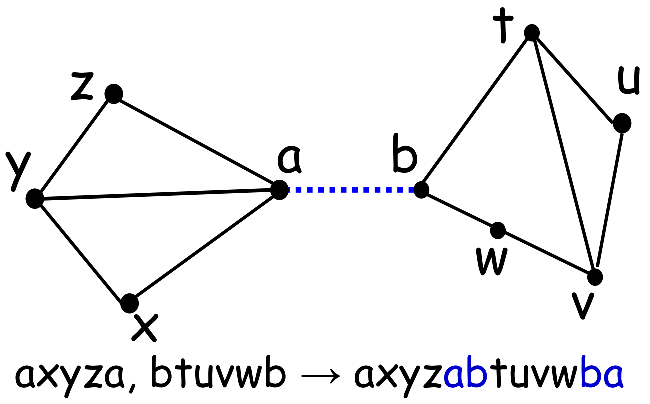


Figure 7.11: The Add Bridge Case.

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

7.8.3 What outer face?

Notice that the definition of planar embedding does not distinguish an “outer” face. There really isn’t any need to distinguish one.

In fact, a planar embedding could be drawn with any given face on the outside. An intuitive explanation of this is to think of drawing the embedding on a *sphere* instead of the plane. Then any face can be made the outside face by “puncturing” that face of the sphere, stretching the puncture hole to a circle around the rest of the faces, and flattening the circular drawing onto the plane.

So pictures that show different “outside” boundaries may actually be illustrations of the same planar embedding.

This is what justifies the “add bridge” case in a planar embedding: whatever face is chosen in the embeddings of each of the disjoint planar graphs, we can draw a bridge between them without needing to cross any other edges in the drawing, because we can assume the bridge connects two “outer” faces.

7.8.4 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 7.8.2 (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 7.6, $|V| = 4$, $|E| = 6$, 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, $\gamma = a \dots b \cdots a$, of the planar embedding.

Then the graph obtained by adding the edge a—b to the edges of G has a planar embedding with one more face and one more edge than G. So the quantity $v - e + f$ will remain the same for both graphs, and since by structural induction this quantity is 2 for G's embedding, it's also 2 for the embedding of G with the added edge. So P holds for the constructed embedding.

Constructor case: (add bridge) Suppose G and H are connected graphs with planar embeddings and disjoint sets of vertices. Then connecting these two graphs with a bridge merges the two bridged faces into a single face, and leaves all other faces unchanged. So the bridge operation yields a planar embedding of a connected graph with $v_G + v_H$ vertices, $e_G + e_H + 1$ edges, and $f_G + f_H - 1$ faces.

But

$$
\begin{aligned}
&(v_G + v_H) - (e_G + e_H + 1) + (f_G + f_H - 1) \\
&= (v_G - e_G + f_G) + (v_H - e_H + f_H) - 2 \\
&= (2) + (2) - 2 && \text{(by structural induction hypothesis)} \\
&= 2.
\end{aligned}
$$

So $v - e + f$ remains equal to 2 for the constructed embedding. That is, P also holds in this case.

This completes the proof of the constructor cases, and the theorem follows by structural induction. ■

7.8.5 Number of Edges versus Vertices

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

Lemma 7.8.3. *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 7.8.4. *In a planar embedding of a connected graph with at least three vertices, each face is of length at least three.*

Corollary 7.8.5. *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 7.8.3, 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 7.8.4, when $v \geq 3$, each face boundary is of length at least three, so this sum is at least $3f$. This implies that

$$3f \leq 2e. \tag{7.1}$$

But $f = e - v + 2$ by Euler's formula, and substituting into (7.1) gives

$$3(e - v + 2) \leq 2e$$

$$e - 3v + 6 \leq 0$$

$$e \leq 3v - 6$$

■

Corollary 7.8.5 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 7.8.5 required for K_5 to be planar, which proves

Lemma 7.8.6. *K_5 is not planar.*

Another consequence is

Lemma 7.8.7. *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 7.8.5. ■

7.8.6 Planar Subgraphs

If you draw a graph in the plane by repeatedly adding edges that don't cross, you clearly could add the edges in any other order and still wind up with the same drawing. This is so basic that we might presume that our recursively defined planar embeddings have this property. But that wouldn't be fair: we really need to prove it. After all, the recursive definition of planar embedding was pretty technical —maybe we got it a little bit wrong, with the result that our embeddings don't have this basic draw-in-any-order property.

Now any ordering of edges can be obtained just by repeatedly switching the order of successive edges, and if you think about the recursive definition of embedding for a minute, you should realize that you can switch *any* pair of successive edges if you can just switch the last two. So it all comes down to the following lemma.

Lemma 7.8.8. *Suppose that, starting from some embeddings of planar graphs with disjoint sets of vertices, it is possible by two successive applications of constructor operations to add edges e and then f to obtain a planar embedding, $\mathcal{F}$. Then starting from the same embeddings, it is also possible to obtain $\mathcal{F}$ by adding f and then e with two successive applications of constructor operations.*

We'll leave the proof of Lemma 7.8.8 to Problem **??**.

Corollary 7.8.9. *Suppose that, starting from some embeddings of planar graphs with disjoint sets of vertices, it is possible to add a sequence of edges $e_0, e_1, \ldots, e_n$ by successive applications of constructor operations to obtain a planar embedding, $\mathcal{F}$. Then starting from the same embeddings, it is also possible to obtain $\mathcal{F}$ by applications of constructor operations that successively add any permutation*[9] *of the edges $e_0, e_1, \ldots, e_n$.*

Corollary 7.8.10. *Deleting an edge from a planar graph leaves a planar graph.*

[9]If $\pi : \{0, 1, \ldots, n\} \to \{0, 1, \ldots, n\}$ is a bijection, then the sequence $e_{\pi(0)}, e_{\pi(1)}, \ldots, e_{\pi(n)}$ is called a *permutation* of the sequence $e_0, e_1, \ldots, e_n$.

Proof. By Corollary 7.8.9, we may assume the deleted edge was the last one added in constructing an embedding of the graph. So the embedding to which this last edge was added must be an embedding of the graph without that edge. ■

Since we can delete a vertex by deleting all its incident edges, Corollary 7.8.10 immediately implies

Corollary 7.8.11. *Deleting a vertex from a planar graph, along with all its incident edges of course, leaves another planar graph.*

A *subgraph* of a graph, G, is any graph whose set of vertices is a subset of the vertices of G and whose set of edges is a subset of the set of edges of G. So we can summarize Corollaries 7.8.10 and 7.8.11 and their consequences in a Theorem.

Theorem 7.8.12. *Any subgraph of a planar graph is planar.*

7.8.7 Planar 5-Colorability

We need to know one more property of planar graphs in order to prove that planar graphs are 5-colorable.

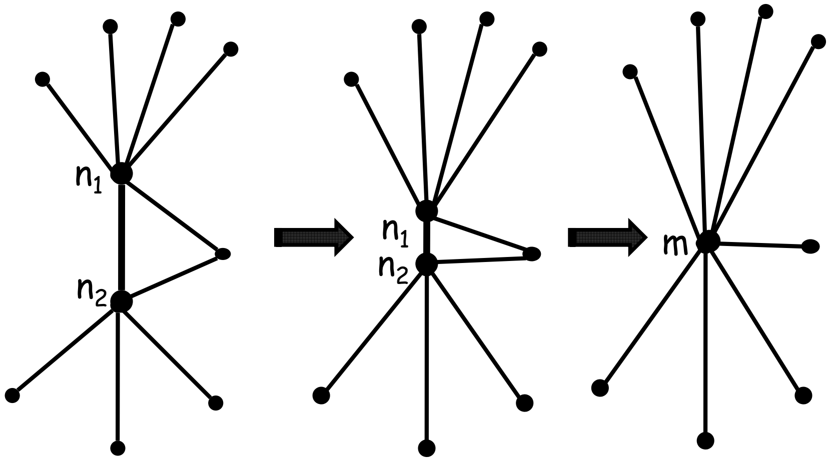


Figure 7.12: Merging adjacent vertices n_1 and n_2 into new vertex, m.

Lemma 7.8.13. *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 7.12.

Lemma 7.8.13 can be proved by structural induction, but the proof is kind of boring, and we hope you'll be relieved that we're going to omit it. (If you insist, we can add it to the next problem set.)

Now we've got all the simple facts we need to prove 5-colorability.

Theorem 7.8.14. *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 7.8.7 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 7.8.11, 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 nonplanar subgraph isomorphic to K_5, contradicting Theorem 7.8.12. 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 7.12. In this new graph, n_2 is adjacent to m, and the graph is planar by Lemma 7.8.13. So we can then merge m and n_2 into a another new vertex, m', resulting in a new graph, G', which by Lemma 7.8.13 is also planar. Now G' has $v - 1$ vertices and so is five-colorable by the induction hypothesis.

Now define a five coloring of G as follows: use the five-coloring of G' for all the vertices besides g, n_1 and n_2. Next assign the color of m' in G' to be the color of the neighbors n_1 and n_2. Since n_1 and n_2 are not adjacent in G, this defines a proper five-coloring of G except for vertex g. But since these two neighbors of g have the same color, the neighbors of g have been colored using fewer than five colors altogether. So complete the five-coloring of G by assigning one of the five colors to g that is not the same as any of the colors assigned to its neighbors.

■

A graph obtained from a graph, G, be repeatedly deleting vertices, deleting edges, and merging adjacent vertices is called a *minor* of G. Since K_5 and $K_{3,3}$ are not planar, Lemmas 7.8.10, 7.8.11, and 7.8.13 immediately imply:

Corollary 7.8.15. *A graph which has K_5 or $K_{3,3}$ as a minor is not planar.*

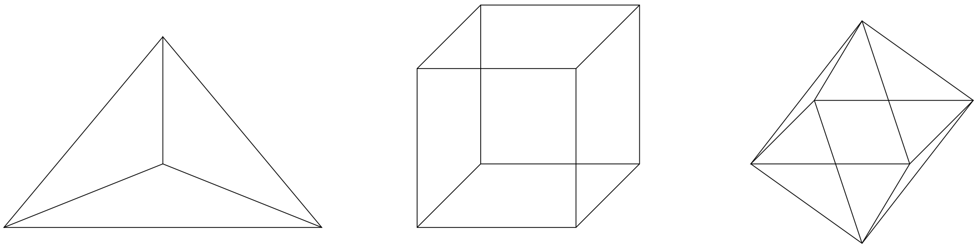
We don't have time to prove it, but the converse of Corollary 7.8.15 is also true. This gives the following famous, very elegant, and purely discrete characterization of planar graphs:

Theorem 7.8.16 (Kuratowksi)**.** *A graph is not planar iff it has K_5 or $K_{3,3}$ as a minor.*

7.8.8 Classifying Polyhedra

The Pythagoreans had two great mathematical secrets, the irrationality of $\sqrt{2}$ and a geometric construct that we're about to rediscover!

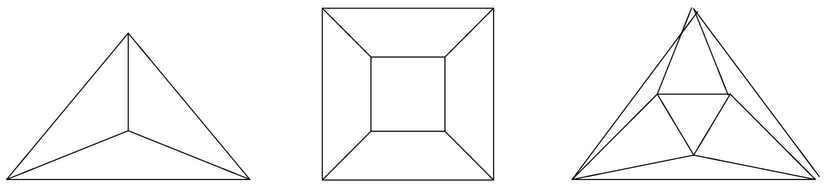
A *polyhedron* is a convex, three-dimensional region bounded by a finite number of polygonal faces. If the faces are identical regular polygons and an equal number of polygons meet at each corner, then the polyhedron is *regular*. Three examples of regular polyhedra are shown below: the tetrahedron, the cube, and the octahedron.



We can determine how many more regular polyhedra there are by thinking about planarity. Suppose we took *any* polyhedron and placed a sphere inside it. Then we could project the polyhedron face boundaries onto the sphere, which would give an image that was a planar graph embedded on the sphere, with the images of the corners of the polyhedron corresponding to vertices of the graph.

But we've already observed that embeddings on a sphere are the same as embeddings on the plane, so Euler's formula for planar graphs can help guide our search for regular polyhedra.

For example, planar embeddings of the three polyhedra above look like this:



Let m be the number of faces that meet at each corner of a polyhedron, and let n be the number of sides on each face. In the corresponding planar graph, there are m edges incident to each of the v vertices. Since each edge is incident to two vertices, we know:

$$mv = 2e$$

Also, each face is bounded by n edges. Since each edge is on the boundary of two faces, we have:

$$nf = 2e$$

Solving for v and f in these equations and then substituting into Euler's formula gives:

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

which simplifies to

$$\frac{1}{m} + \frac{1}{n} = \frac{1}{e} + \frac{1}{2} \tag{7.2}$$

This last equation (7.2) places strong restrictions on the structure of a polyhedron. Every nondegenerate polygon has at least 3 sides, so $n \geq 3$. And at least 3 polygons must meet to form a corner, so $m \geq 3$. On the other hand, if either n or m were 6 or more, then the left side of the equation could be at most $1/3 + 1/6 = 1/2$, which is less than the right side. Checking the finitely-many cases that remain turns up only five solutions. For each valid combination of n and m, we can compute the associated number of vertices v, edges e, and faces f. And polyhedra with these properties do actually exist:

n	m	v	e	f	polyhedron
3	3	4	6	4	tetrahedron
4	3	8	12	6	cube
3	4	6	12	8	octahedron
3	5	12	30	20	icosahedron
5	3	20	30	12	dodecahedron

The last polyhedron in this list, the dodecahedron, was the other great mathematical secret of the Pythagorean sect. These five, then, are the only possible regular polyhedra.

So if you want to put more than 20 geocentric satellites in orbit so that they *uniformly* blanket the globe —tough luck!

7.8.9 Problems

Exam Problems

Class Problems

Homework Problems

Chapter 8

Directed graphs

8.1 Digraphs

A *directed graph* (*digraph* for short) is formally the same as a binary relation, R, on a set, A —that is, a relation whose domain and codomain are the same set, A. But we describe digraphs as though they were diagrams, with elements of A pictured as points on the plane and arrows drawn between related points. The elements of A are referred to as the *vertices* of the digraph, and the pairs $(a, b) \in \text{graph}(R)$

are *directed edges*. Writing $a \to b$ is a more suggestive alternative for the pair (a, b). Directed edges are also called *arrows*.

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

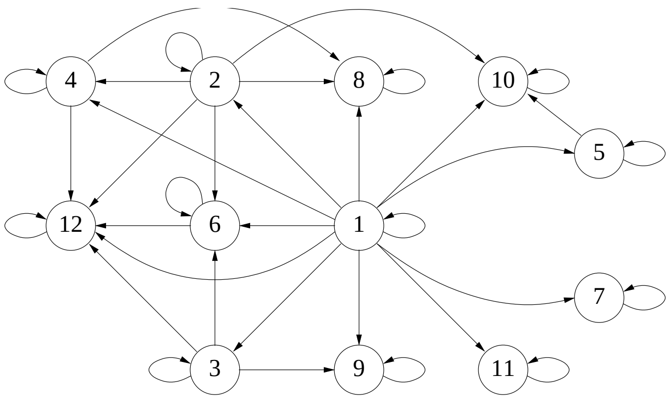


Figure 8.1: The Digraph for Divisibility on $\{1, 2, \ldots, 12\}$.

8.1.1 Paths in Digraphs

Picturing digraphs with points and arrows makes it natural to talk about following a *path* of successive edges through the graph. For example, in the digraph of Figure 8.1, a path might start at vertex 1, successively follow the edges from vertex 1 to vertex 2, from 2 to 4, from 4 to 12, and then from 12 to 12 twice (or as many times as you like). We can represent the path with the sequence of sucessive vertices it went through, in this case:

$$1, 2, 4, 12, 12, 12.$$

So a path is just a sequence of vertices, with consecutive vertices on the path connected by directed edges. Here is a formal definition:

Definition 8.1.1. A *path in a digraph* is a sequence of vertices $a_0, \ldots, a_k$ with $k \geq 0$ such that $a_i \to a_{i+1}$ is an edge of the digraph for $i = 0, 1, \ldots, k-1$. The path is said to *start* at a_0, to *end* at a_k, and the *length* of the path is defined to be k. The path is *simple* iff all the a_i's are different, that is, if $i \neq j$, then $a_i \neq a_j$.

Note that a single vertex counts as length zero path that begins and ends at itself.

It's pretty natural to talk about the edges in a path, but technically, paths only have points, not edges. So to instead, we'll say a path *traverses* an edge $a \to b$ when a and b are consecutive vertices in the path.

For any digraph, R, we can define some new relations on vertices based on paths, namely, the *path relation*, R^*, and the *positive-length path relation*, R^+:

$$a\ R^*\ b ::= \text{there is a path in } R \text{ from } a \text{ to } b,$$

$$a\ R^+\ b ::= \text{there is a positive length path in } R \text{ from } a \text{ to } b.$$

By the definition of path, both R^* and R^+ are transitive. Since edges count as length one paths, the edges of R^+ include all the edges of R. The edges of R^* in turn include all the edges of R^+ and, in addition include an edge (self-loop) from each vertex to itself. The self-loops get included in R^* because of the a length zero paths in R. So R^* is reflexive. [1]

8.2 Picturing Relational Properties

Many of the relational properties we've discussed have natural descriptions in terms of paths. For example:

Reflexivity: All vertices have self-loops (a *self-loop* at a vertex is an arrow going from the vertex back to itself).

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

Irreflexivity: No vertices have self-loops.

Antisymmetry: At most one (directed) edge between different vertices.

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

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

Symmetry: A binary relation R is *symmetric* iff aRb implies bRa for all a, b in the domain of R. That is, if there is an edge from a to b, there is also one in the reverse direction.

EDITING NOTE: The pair of directed edges between two vertices may be represented by a single undirected edge may as well be represented without arrows, indicating that they can be followed in either direction. ■

8.3 Composition of Relations

There is a simple way to extend composition of functions to composition of relations, and this gives another way to talk about paths in digraphs.

Let $R : B \to C$ and $S : A \to B$ be relations. Then the composition of R with S is the binary relation $(R \circ S) : A \to C$ defined by the rule

$$a \; (R \circ S) \; c ::= \; \exists b \in B. \, (b \; R \; c) \text{ AND } (a \; S \; b).$$

This agrees with the Definition 5.4.1 of composition in the special case when R and S are functions.

EDITING NOTE:

[2]

■

Now when R is a digraph, it makes sense to compose R with itself. Then if we let R^n denote the composition of R with itself n times, it's easy to check that R^n is

[2]Some texts define $R \circ S$ the other way around, that is, with S applied to the result of applying R first.

the length-n path relation:

$$a \ R^n \ b \quad \text{iff} \quad \text{there is a length } n \text{ path in } R \text{ from } a \text{ to } b.$$

This even works for $n = 0$, if we adopt the convention that R^0 is the identity relation Id_A on the set, A, of vertices. That is, $(a \ \text{Id}_A \ b)$ iff $a = b$.

8.4 Directed Acyclic Graphs

Definition 8.4.1. A *cycle* in a digraph is defined by a path that begins and ends at the same vertex. This includes the cycle of length zero that begins and ends at the vertex. A *directed acyclic graph (DAG)* is a directed graph with no *positive* length cycles.

A *simple cycle* in a digraph is a cycle whose vertices are distinct except for the beginning and end vertices.

EDITING NOTE:

In contrast to undirected graphs, a single vertex *is* considered to be a simple cycle.

■

DAG's can be an economical way to represent partial orders. For example, in Section 9.1 the *direct prerequisite* relation between MIT subjects was used to determine the partial order of indirect prerequisites on subjects. This indirect prerequisite partial order is precisely the positive length path relation of the direct prerequisites.

Lemma 8.4.2. *If D is a DAG, then D^+ is a strict partial order.*

Proof. We know that D^+ is transitive. Also, a positive length path from a vertex to itself would be a cycle, so there are no such paths. This means D^+ is irreflexive, which implies it is a strict partial order (see problem **??**). ■

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

The divisibility partial order can also be more economically represented by the path relation in a DAG. A DAG whose *path* relation is divisibility on $\{1, 2, \ldots, 12\}$ is shown in Figure 8.2; the arrowheads are omitted in the Figure, and edges are understood to point upwards.

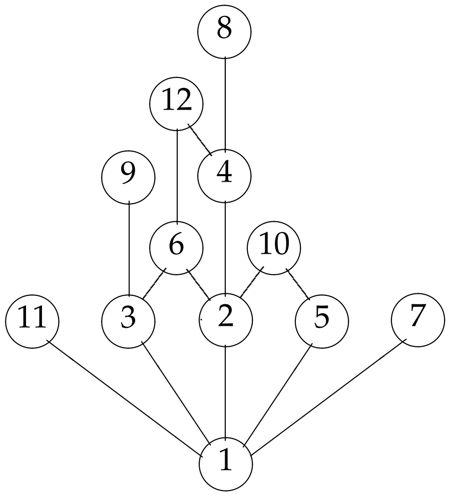


Figure 8.2: A DAG whose Path Relation is Divisibility on $\{1, 2, \ldots, 12\}$.

If we're using a DAG to represent a partial order —so all we care about is the the path relation of the DAG —we could replace the DAG with any other DAG with the same path relation. This raises the question of finding a DAG with the same path relation but the *smallest* number of edges. This DAG turns out to be unique and easy to find (see Problem **??**).

8.4.1 Problems

Practice Problems

Class Problems

Homework Problems

EDITING NOTE: * add problem using matrix operations to compute transitive closure

* add section on shortest paths?

* add section on directed tours and walks? ■

8.5 Communication Networks

EDITING NOTE: ADD DISCUSSION of routing problems in general, with defs of

*"IO-assignments" = specification of sources & destinations of packets,

* "routing" = paths that achieve the IO-assignment,

* congestion and latency of a routing.

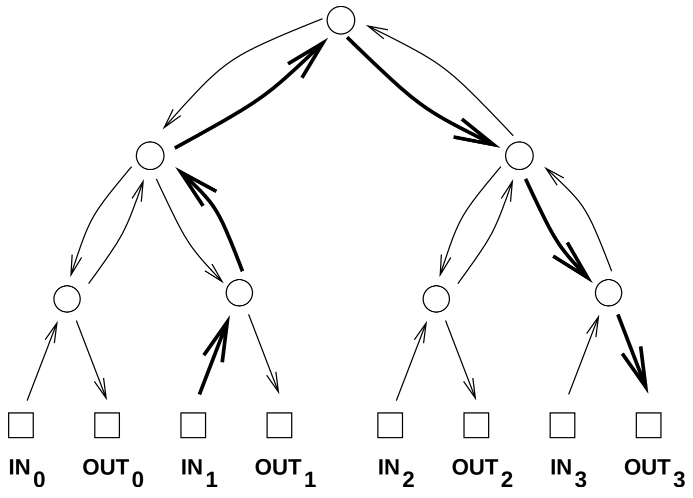
* Possible latency/congestion tradeoffs

* congestion and latency of a net. 🟥

Modeling communication networks is an important application of digraphs in computer science. In this such models, vertices represent computers, processors, and switches; edges will represent wires, fiber, or other transmission lines through which data flows. For some communication networks, like the internet, the corresponding graph is enormous and largely chaotic. Highly structured networks, by contrast, find application in telephone switching systems and the communication hardware inside parallel computers. In this chapter, we'll look at some of the nicest and most commonly used structured networks.

8.6 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.

8.7 Routing Problems

Communication networks are supposed to get packets from inputs to outputs, with each packet entering the network at its own input switch and arriving at its own output switch. We're going to consider several different communication network designs, where each network has N inputs and N outputs; for convenience, we'll assume N is a power of two.

Which input is supposed to go where is specified by a permutation of $\{0, 1, \ldots, N-1\}$. So a permutation, π, defines a *routing problem*: get a packet that starts at input i to output $\pi(i)$. A *routing*, P, that *solves* a routing problem, π, is a set of paths from each input to its specified output. That is, P is a set of n paths, P_i, for $i = 0 \ldots, N-1$, where P_i goes from input i to output $\pi(i)$.

8.8 Network Diameter

The delay between the time that a packets arrives at an input and arrives at its designated output is a critical issue in communication networks. Generally this delay is proportional to the length of the path a packet follows. Assuming it takes one time unit to travel across a wire,

EDITING NOTE: and that there are no additional delays at switches, ■

the delay of a packet will be the number of wires it crosses going from input to output.

EDITING NOTE:

[3]

[3]Latency is often measured as the number of switches that a packet must pass through when traveling between the most distant input and output, since switches usually have the biggest impact on network speed. For example, in the complete binary tree example, the packet traveling from input 1 to output 3 crosses 5 switches.

■

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[4] is the maximum length of any shortest path between an input and an output. For example, in the complete binary tree above, the distance from input 1 to output 3 is six. No input and output are farther apart than this, so the diameter of this tree is also six.

More generally, the diameter of a complete binary tree with N inputs and outputs is $2 \log N + 2$. (All logarithms in this lecture— and in most of computer science —are base 2.) This is quite good, because the logarithm function grows very slowly. We could connect up $2^{10} = 1024$ inputs and outputs using a complete binary tree and the worst input-output delay for any packet would be this diameter, namely,

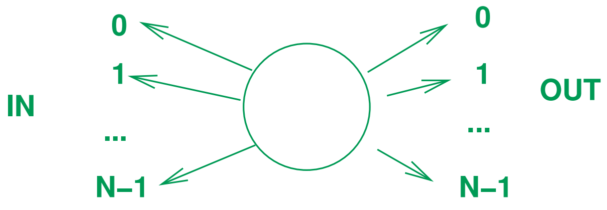
[4]The usual definition of *diameter* for a general *graph* (simple or directed) is the largest distance between *any* two vertices, but in the context of a communication network we're only interested in the distance between inputs and outputs, not between arbitrary pairs of vertices.

$2\log(2^{10}) + 2 = 22$.

8.8.1 Switch Size

One way to reduce the diameter of a network is to use larger switches. For example, in the complete binary tree, most of the switches have three incoming edges and three outgoing edges, which makes them 3×3 switches. If we had 4×4 switches, then we could construct a complete *ternary* tree with an even smaller diameter. In principle, we could even connect up all the inputs and outputs via a single monster $N \times N$ switch.

EDITING NOTE:



■

This isn't very productive, however, since we've just concealed the original network design problem inside this abstract switch. Eventually, we'll have to design the internals of the monster switch using simpler components, and then we're right back where we started. So the challenge in designing a communication network is figuring out how to get the functionality of an $N \times N$ switch using fixed size, elementary devices, like 3×3 switches.

8.9 Switch Count

Another goal in designing a communication network is to use as few switches as possible. The number of switches in a complete binary tree is $1+2+4+8+\cdots+N$, since there is 1 switch at the top (the "root switch"), 2 below it, 4 below those, and so forth. By the formula (**??**) for geometric sums, the total number of switches is $2N-1$, which is nearly the best possible with 3×3 switches.

8.10 Network Latency

We'll sometimes be choosing routings through a network that optimize some quantity besides delay. For example, in the next section we'll be trying to minimize packet congestion. When we're not minimizing delay, shortest routings are not always the best, and in general, the delay of a packet will depend on how it is routed. For any routing, the most delayed packet will be the one that follows the longest path in the routing. The length of the longest path in a routing is called its *latency*.

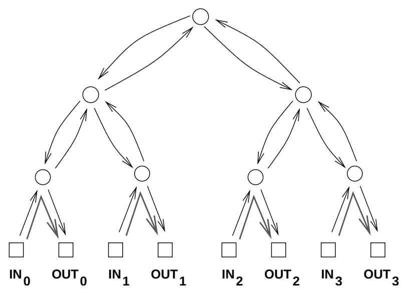
The latency of a *network* depends on what's being optimized. It is measured by assuming that optimal routings are always chosen in getting inputs to their specified outputs. That is, for each routing problem, π, we choose an optimal routing that solves π. Then *network latency* is defined to be the largest routing latency among these optimal routings. Network latency will equal network diameter if routings are always chosen to optimize delay, but it may be significantly larger if routings are chosen to optimize something else.

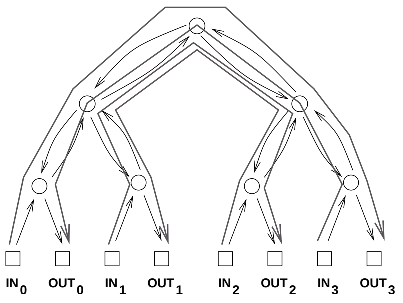
For the networks we consider below, paths from input to output are uniquely determined (in the case of the tree) or all paths are the same length, so network latency will always equal network diameter.

8.11 Congestion

The complete binary tree has a fatal drawback: the root switch is a bottleneck. At best, this switch must handle an enormous amount of traffic: every packet traveling from the left side of the network to the right or vice-versa. Passing all these packets through a single switch could take a long time. At worst, if this switch fails, the network is broken into two equal-sized pieces.

For example, if the routing problem is given by the identity permutation, $\mathrm{Id}(i) ::= i$, then there is an easy routing, P, that solves the problem: let P_i be the path from input i up through one switch and back down to output i. On the other hand, if the problem was given by $\pi(i) ::= (N-1) - i$, then in *any* solution, Q, for π, each path Q_i beginning at input i must eventually loop all the way up through the root switch and then travel back down to output $(N-1) - i$. These two situations are illustrated below.





We can distinguish between a "good" set of paths and a "bad" set based on congestion. The *congestion* of a routing, P, is equal to the largest number of paths in P that pass through a single switch. For example, the congestion of the routing on the left is 1, since at most 1 path passes through each switch. However, the congestion of the routing on the right is 4, since 4 paths pass through the root switch (and the two switches directly below the root). Generally, lower congestion is better since packets can be delayed at an overloaded switch.

By extending the notion of congestion to networks, we can also distinguish between "good" and "bad" networks with respect to bottleneck problems. For each routing problem, π, for the network, we assume a routing is chosen that optimizes congestion, that is, that has the minimum congestion among all routings that solve

π. Then the largest congestion that will ever be suffered by a switch will be the maximum congestion among these optimal routings. This "maximin" congestion is called the *congestion of the network*.

EDITING NOTE:

You may find it helpful to think about max congestion in terms of a value game. You design your spiffy, new communication network; this defines the game. Your opponent makes the first move in the game: she inspects your network and specifies a permutation routing problem that will strain your network.You move second: given her specification, you choose the precise paths that the packets should take through your network; you're trying to avoid overloading any one switch. Then her next move is to pick a switch with as large as possible a number of packets passing through it; this number is her score in the competition. The max congestion of your network is the largest score she can ensure; in other words, it is precisely the max-value of this game.

For example, if your enemy were trying to defeat the complete binary tree, she would choose a permutation like $\pi(i) = (N-1) - i$. Then for *every* packet i, you would be forced to select a path $P_{i,\pi(i)}$ passing through the root switch. Thus, the max congestion of the complete binary tree is N— which is horrible!

■

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

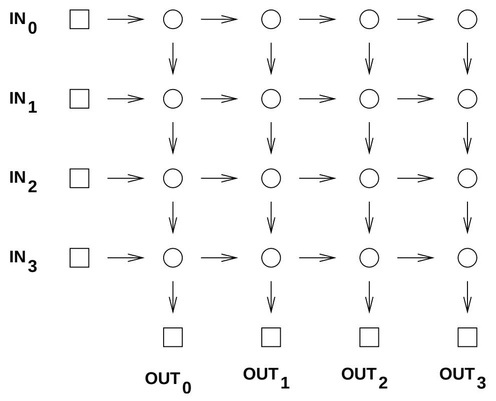
Let's tally the results of our analysis so far:

network	**diameter**	**switch size**	**# switches**	**congestion**
complete binary tree	$2\log N + 2$	3×3	$2N - 1$	N

8.12 2-D Array

Let's look at an another communication network. This one is called a *2-dimensional array* or *grid*.

EDITING NOTE: or *crossbar*. ■



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 8.12.1. *The congestion of an N-input array is 2.*

Proof. First, we show that the congestion is at most 2. Let π be any permutation. Define a solution, P, for π to be the set of paths, P_i, where P_i goes to the right from input i to column $\pi(i)$ and then goes down to output $\pi(i)$. Thus, the switch in row

i and column j transmits at most two packets: the packet originating at input i and the packet destined for output j.

Next, we show that the congestion is at least 2. This follows because in any routing problem, π, where $\pi(0) = 0$ and $\pi(N-1) = N-1$, two packets must pass through the lower left switch. ■

As with the tree, the network latency when minimizing congestion is the same as the diameter. That's because all the paths between a given input and output are the same length.

Now we can record the characteristics of the 2-D array.

network	**diameter**	**switch size**	**# switches**	**congestion**
complete binary tree	$2\log N + 2$	3×3	$2N - 1$	N
2-D array	$2N$	2×2	N^2	2

The crucial entry here is the number of switches, which is N^2. This is a major defect of the 2-D array; a network of size $N = 1000$ would require a *million* 2×2 switches! Still, for applications where N is small, the simplicity and low congestion of the array make it an attractive choice.

8.13 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 8.3.

EDITING NOTE:

The butterfly of size $2N$ consists of two butterflies of size N, which are shown in dashed boxes below, and one additional level of switches. Each switch in the new level has directed edges to a pair of corresponding switches in the smaller butterflies; one example is dashed in the figure.

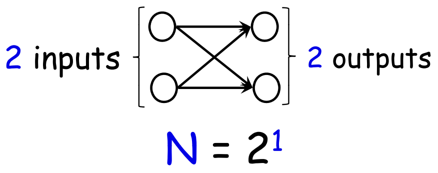
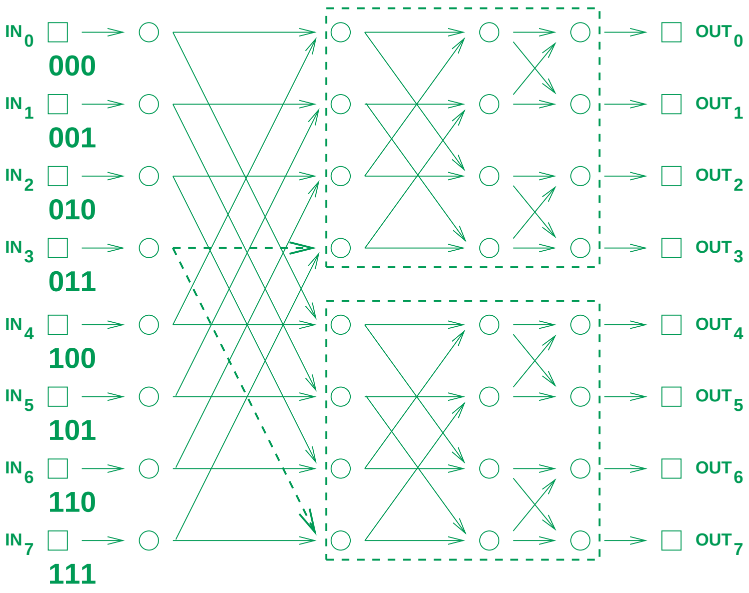


Figure 8.3: F_1, the Butterfly Net switches with $N = 2^1$.



Despite the relatively complicated structure of the butterfly, there is a simple way to route packets. In particular, suppose that we want to send a packet from input $x_1x_2\dots x_{\log N}$ to output $y_1y_2\dots y_{\log N}$. (Here we are specifying the input and output numbers in binary.) Roughly, the plan is to "correct" the first bit by level 1, correct the second bit by level 2, and so forth. Thus, the sequence of switches visited by the packet is:

$$\begin{aligned}
(x_1, x_2, x_3, \dots, x_{\log N}, 0) &\to (y_1, x_2, x_3, \dots, x_{\log N}, 1)\\
&\to (y_1, y_2, x_3, \dots, x_{\log N}, 2)\\
&\to (y_1, y_2, y_3, \dots, x_{\log N}, 3)\\
&\to \qquad \dots\\
&\to (y_1, y_2, y_3, \dots, y_{\log N}, \log N)
\end{aligned}$$

In fact, this is the *only* path from the input to the output!

■

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 8.4. That is, the ith and $2^n + i$th new input switches are each connected to the same two switches, namely, to the ith input switches of each of two F_n components for $i = 1, \ldots, 2^n$. The output switches of F_{n+1} are simply the output switches of each of the F_n copies.

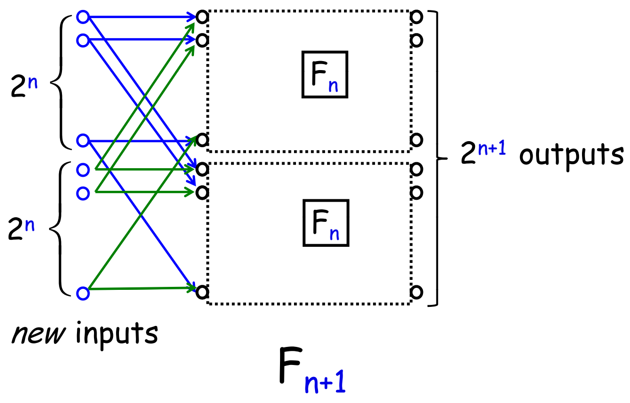


Figure 8.4: F_{n+1}, the Butterfly Net switches with 2^{n+1} inputs and outputs.

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, which implies that the network latency when minimizing congestion is the same as the diameter.

The congestion of the butterfly network is about $\sqrt{N}$, more precisely, the congestion is $\sqrt{N}$ if N is an even power of 2 and $\sqrt{N/2}$ if N is an odd power of 2. A simple proof of this appears in Problem**??**.

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.

8.14 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 8.3.

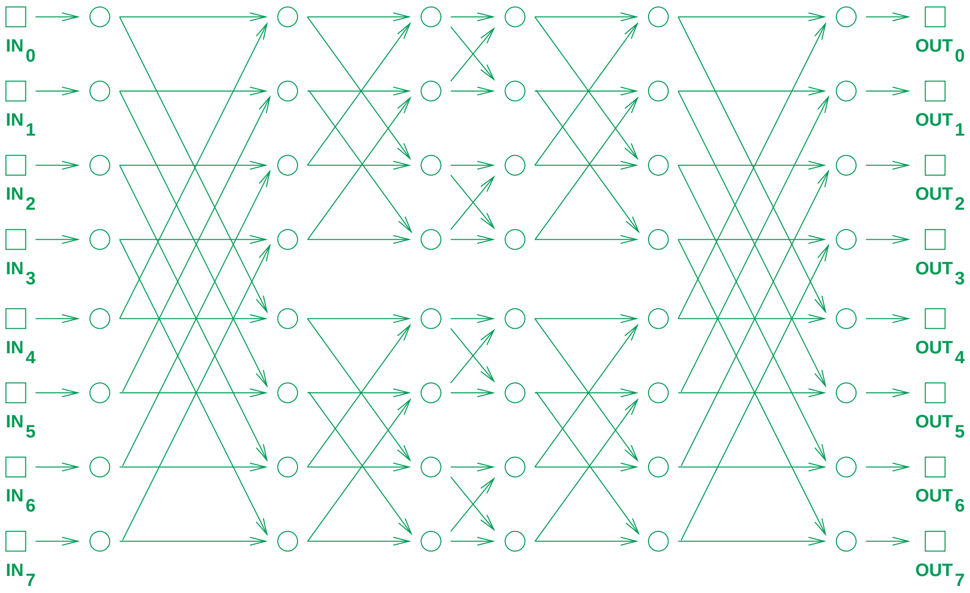
In the constructor step, we construct B_{n+1} out of two B_n nets connected to a new set of 2^{n+1} input switches *and also* a new set of 2^{n+1} output switches. This is illustrated in Figure 8.5.

Namely, the ith and $2^n + i$th new input switches are each connected to the same two switches, namely, to the ith input switches of each of two B_n components for $i = 1, \dots, 2^n$, exactly as in the Butterfly net. In addition, the ith and $2^n + i$th new *output* switches are connected to the same two switches, namely, to the ith output switches of each of two B_n components.

Now B_{n+1} is laid out in columns of height 2^{n+1} by adding two more columns of switches to the columns in B_n. So the B_{n+1} switches are arrayed in $2(n+1)$ columns. The total number of switches is the number of columns times the height of the columns, namely, $2(n+1)2^{n+1}$.

All paths in B_{n+1} from an input switch to an output are the same length, namely, $2(n+1) - 1$, and the diameter of the Beneš net with 2^{n+1} inputs is this length plus two because of the two edges connecting to the terminals.

EDITING NOTE:



This network now has levels labeled $0, \ldots, 2 \log N + 1$. For $1 \leq k \leq \log N$, the connections from level $k - 1$ to level k are just as in the Butterfly network, the connections based on bit k. The conections from level $2 \log N - k + 1$ to level $2 \log N - k + 2$ are also the ones based on bit k. (Informally, to make the connections from level 0 to level $2 \log N + 1$ one level at a time, use the connections based on bits $1, 2, 3, \ldots, \log N - 1, \log N, \log N - 1, \log N - 2, \ldots, 3, 2, 1$ in that order.)

■

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

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

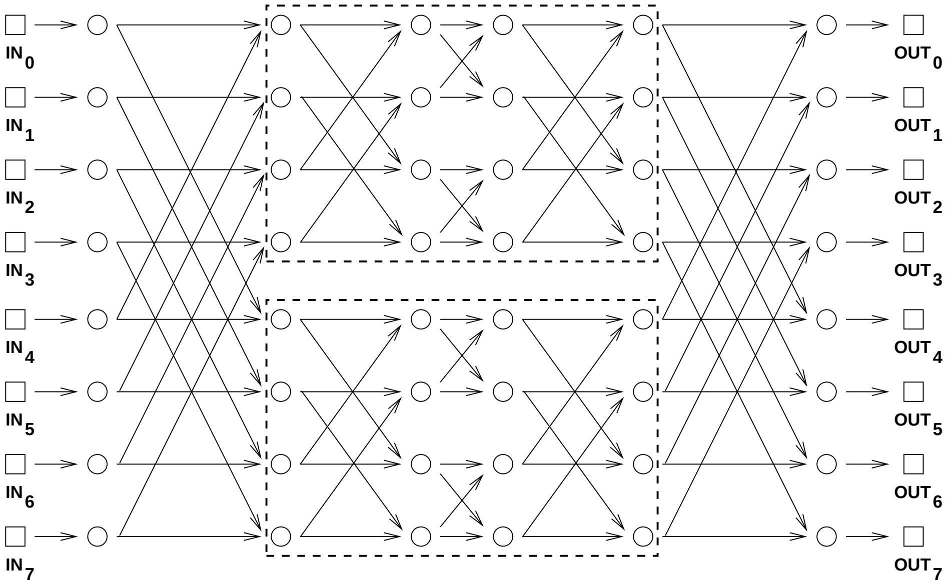
$$P(n) ::= \text{the congestion of } B_n \text{ is } 1.$$

Base case ($n = 1$): $B_1 = F_1$ and the unique routings in F_1 have congestion 1.

Inductive step: We assume that the congestion of an $N = 2^n$-input Beneš network is 1 and prove that the congestion of a $2N$-input Beneš network is also 1.

Digression. Time out! Let's work through an example, develop some intuition, and then complete the proof. In the Beneš network shown below with $N = 8$

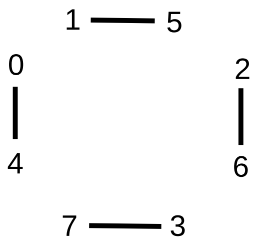
inputs and outputs, the two 4-input/output subnetworks are in dashed boxes.



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

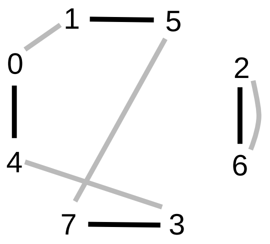
$$\begin{aligned} \pi(0) &= 1 & \pi(4) &= 3 \\ \pi(1) &= 5 & \pi(5) &= 6 \\ \pi(2) &= 4 & \pi(6) &= 0 \\ \pi(3) &= 7 & \pi(7) &= 2 \end{aligned}$$

We can route each packet to its destination through either the upper subnetwork or the lower subnetwork. However, the choice for one packet may constrain the choice for another. For example, we can not route both packet 0 *and* packet 4 through the same network since that would cause two packets to collide at a single switch, resulting in congestion. So one packet must go through the upper network and the other through the lower network. Similarly, packets 1 and 5, 2 and 6, and 3 and 7 must be routed through different networks. Let's record these constraints in a graph. The vertices are the 8 packets. If two packets must pass through different networks, then there is an edge between them. Thus, our constraint graph looks like this:



Notice that at most one edge is incident to each vertex.

The output side of the network imposes some further constraints. For example, the packet destined for output 0 (which is packet 6) and the packet destined for output 4 (which is packet 2) can not both pass through the same network; that would require both packets to arrive from the same switch. Similarly, the packets destined for outputs 1 and 5, 2 and 6, and 3 and 7 must also pass through different switches. We can record these additional constraints in our graph with gray edges:



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

Now here's the key insight: *a 2-coloring of the graph corresponds to a solution to the routing problem*. In particular, suppose that we could color each vertex either red or blue so that adjacent vertices are colored differently. Then all constraints are satisfied if we send the red packets through the upper network and the blue packets through the lower network.

The only remaining question is whether the constraint graph is 2-colorable, which is easy to verify:

Lemma 8.14.2. *Prove that if the edges of a graph can be grouped into two sets such that every vertex has at most 1 edge from each set incident to it, then the graph is 2-colorable.*

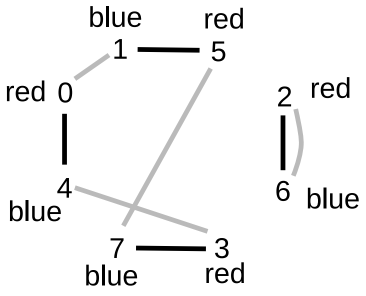
Proof. Since the two sets of edges may overlap, let's call an edge that is in both sets a *doubled edge*.

We know from Theorem 7.7.2 that all we have to do is show that every cycle has even length. There are two cases:

Case 1: [The cycle contains a doubled edge.] No other edge can be incident to either of the endpoints of a doubled edge, since that endpoint would then be incident to two edges from the same set. So a cycle traversing a doubled edge has nowhere to go but back and forth along the edge an even number of times.

Case 2: [No edge on the cycle is doubled.] Since each vertex is incident to at most one edge from each set, any path with no doubled edges must traverse successive edges that alternate from one set to the other. In particular, a cycle must traverse a path of alternating edges that begins and ends with edges from different sets. This means the cycle has to be of even length. ■

For example, here is a 2-coloring of the constraint graph:



The solution to this graph-coloring problem provides a start on the packet routing problem:

We can complete the routing in the two smaller Beneš networks by induction! Back to the proof. **End of Digression.**

Let π be an arbitrary permutation of $\{0, 1, \dots, N-1\}$. Let G be the graph whose vertices are packet numbers $0, 1, \dots, N-1$ and whose edges come from the union of these two sets:

$$E_1 ::= \{u\text{—}v \mid |u - v| = N/2\}, \text{ and}$$

$$E_2 ::= \{u\text{—}w \mid |\pi(u) - \pi(w)| = N/2\}.$$

Now any vertex, u, is incident to at most two edges: a unique edge $u\text{—}v \in E_1$ and a unique edge $u\text{—}w \in E_2$. So according to Lemma 8.14.2, there is a 2-coloring for the vertices of G. Now route packets of one color through the upper subnetwork and packets of the other color through the lower subnetwork. Since for each edge in E_1, one vertex goes to the upper subnetwork and the other to the lower subnetwork, there will not be any conflicts in the first level. Since for each edge in E_2, one vertex comes from the upper subnetwork and the other from the lower subnetwork, there will not be any conflicts in the last level. We can complete the routing within each subnetwork by the induction hypothesis $P(n)$. ■

8.14.1 Problems

Exam Problems

Class Problems

Homework Problems

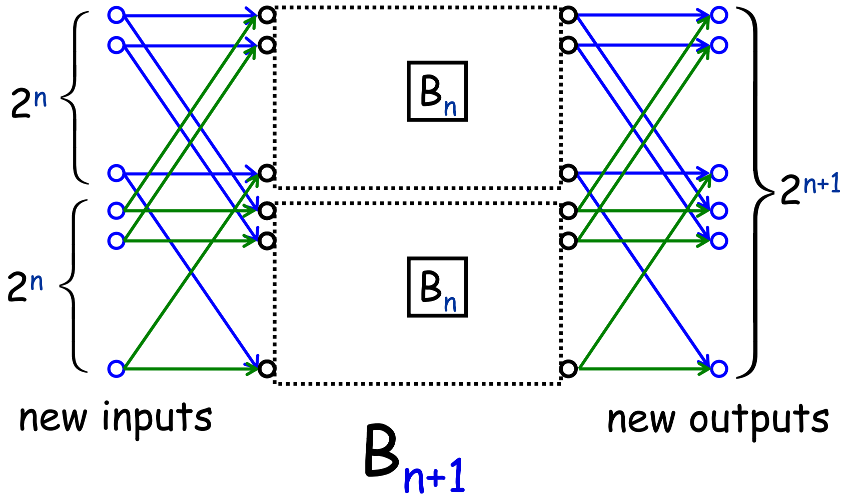


Figure 8.5: B_{n+1}, the Beneš Net switches with 2^{n+1} inputs and outputs.

Chapter 9

Partial Orders and Scheduling

EDITING NOTE:

EQUIVALENCE RELATIONS to be added, including SYMMETRY axiom, representation by functions mapping elements to properties, and partitions

ALL THE MATERIAL BELOW BEFORE the SCHEDULING section WILL MOVE into sectionRelations of Ch.5 (sets & relations). ■

Partial orders are a kind of binary relation that come up a lot. The familiar $\leq$

order on numbers is a partial order, but so is the containment relation on sets and the divisibility relation on integers.

Partial orders have particular importance in computer science because they capture key concepts used, for example, in solving task scheduling problems, analyzing concurrency control, and proving program termination.

9.1 Axioms for Partial Orders

The prerequisite structure among MIT subjects provides a nice illustration of partial orders. Here is a table indicating some of the prerequisites of subjects in the the Course 6 program of Spring '07:

Direct Prerequisites	Subject
18.01	6.042
18.01	18.02
18.01	18.03
8.01	8.02
6.001	6.034
6.042	6.046
18.03, 8.02	6.002
6.001, 6.002	6.004
6.001, 6.002	6.003
6.004	6.033
6.033	6.857
6.046	6.840

Since 18.01 is a direct prerequisite for 6.042, a student must take 18.01 before 6.042. Also, 6.042 is a direct prerequisite for 6.046, so in fact, a student has to take *both* 18.01 and 6.042 before taking 6.046. So 18.01 is also really a prerequisite for 6.046, though an implicit or indirect one; we'll indicate this by writing

$$18.01 \to 6.046.$$

This prerequisite relation has a basic property known as *transitivity*: if subject a is an indirect prerequisite of subject b, and b is an indirect prerequisite of subject c, then a is also an indirect prerequisite of c.

In this table, a longest sequence of prerequisites is

$$18.01 \to 18.03 \to 6.002 \to 6.004 \to 6.033 \to 6.857$$

so a student would need at least six terms to work through this sequence of subjects. But it would take a lot longer to complete a Course 6 major if the direct prerequisites led to a situation[1] where two subjects turned out to be prerequisites

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

of *each other*! So another crucial property of the prerequisite relation is that if $a \to b$, then it is not the case that $b \to a$. This property is called *asymmetry*.

Another basic example of a partial order is the subset relation, $\subseteq$, on sets. In fact, we'll see that every partial order can be represented by the subset relation.

Definition 9.1.1. A binary relation, R, on a set A is:

- *transitive* iff $[a \mathrel{R} b \text{ and } b \mathrel{R} c]$ IMPLIES $a \mathrel{R} c$ for every $a, b, c \in A$,
- *asymmetric* iff $a \mathrel{R} b$ IMPLIES NOT$(b \mathrel{R} a)$ for all $a, b \in A$,
- a *strict partial order* iff it is transitive and asymmetric.

So the prerequisite relation, $\to$, on subjects in the MIT catalogue is a strict partial order. More familiar examples of strict partial orders are the relation, $<$, on real numbers, and the proper subset relation, $\subset$, on sets.

The subset relation, $\subseteq$, on sets and $\leq$ relation on numbers are examples of *reflexive* relations in which each element is related to itself. Reflexive partial orders are called *weak* partial orders. Since asymmetry is incompatible with reflexivity, the asymmetry property in weak partial orders is relaxed so it applies only to two different elements. This relaxation of the asymmetry is called antisymmetry:

Definition 9.1.2. A binary relation, R, on a set A, is

- *reflexive* iff $a \; R \; a$ for all $a \in A$,
- *antisymmetric* iff $a \; R \; b$ IMPLIES NOT$(b \; R \; a)$ for all $a \neq b \in A$,
- a *weak partial order* iff it is transitive, reflexive and antisymmetric.

Some authors define partial orders to be what we call weak partial orders, but we'll use the phrase "partial order" to mean either a weak or strict one.

For weak partial orders in general, we often write an ordering-style symbol like $\preceq$ or $\sqsubseteq$ instead of a letter symbol like R. (General relations are usually denoted by a letter like R instead of a cryptic squiggly symbol, so $\preceq$ is kind of like the musical performer/composer Prince, who redefined the spelling of his name to be his own squiggly symbol. A few years ago he gave up and went back to the spelling "Prince.") Likewise, we generally use $\prec$ or $\sqsubset$ to indicate a strict partial order.

Two more examples of partial orders are worth mentioning:

Example 9.1.3. Let A be some family of sets and define $a \; R \; b$ iff $a \supset b$. Then R is a strict partial order.

For integers, m, n we write $m \mid n$ to mean that m *divides* n, namely, there is an integer, k, such that $n = km$.

Example 9.1.4. The divides relation is a weak partial order on the nonnegative integers.

9.2 Representing Partial Orders by Set Containment

Axioms can be a great way to abstract and reason about important properties of objects, but it helps to have a clear picture of the things that satisfy the axioms. We'll show that every partial order can be pictured as a collection of sets related by containment. That is, every partial order has the "same shape" as such a collection. The technical word for "same shape" is "isomorphic."

Definition 9.2.1. A binary relation, R, on a set, A, is *isomorphic* to a relation, S, on a set D iff there is a relation-preserving bijection from A to D. That is, there is bijection $f : A \to D$, such that for all $a, a' \in A$,

$$a \mathrel{R} a' \quad \text{iff} \quad f(a) \mathrel{S} f(a').$$

Theorem 9.2.2. *Every weak partial order, $\preceq$, is isomorphic to the subset relation, on a collection of sets.*

To picture a partial order, $\preceq$, on a set, A, as a collection of sets, we simply represent each element A by the set of elements that are $\preceq$ to that element, that is,

$$a \longleftrightarrow \{b \in A \mid b \preceq a\}.$$

For example, if $\preceq$ is the divisibility relation on the set of integers, $\{1, 3, 4, 6, 8, 12\}$, then we represent each of these integers by the set of integers in A that divides it.

So

$$\begin{aligned}
1 &\longleftrightarrow \{1\} \\
3 &\longleftrightarrow \{1,3\} \\
4 &\longleftrightarrow \{1,4\} \\
6 &\longleftrightarrow \{1,3,6\} \\
8 &\longleftrightarrow \{1,4,8\} \\
12 &\longleftrightarrow \{1,3,4,6,12\}
\end{aligned}$$

So, the fact that $3 \mid 12$ corresponds to the fact that $\{1,3\} \subseteq \{1,3,4,6,12\}$.

In this way we have completely captured the weak partial order $\preceq$ by the subset relation on the corresponding sets. Formally, we have

Lemma 9.2.3. *Let $\preceq$ be a weak partial order on a set, A. Then $\preceq$ is isomorphic to the subset relation on the collection of inverse images of elements $a \in A$ under the $\preceq$ relation.*

We leave the proof to Problem **??**. Essentially the same construction shows that strict partial orders can be represented by set under the proper subset relation, $\subset$.

9.2.1 Problems

Class Problems

Homework Problems

9.3 Total Orders

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

Definition 9.3.1. Let R be a binary relation on a set, A, and let a, b be elements of A. Then a and b are *comparable* with respect to R iff $[a \; R \; b \;\; \text{OR} \;\; b \; R \; a]$. A partial order for which every two different elements are comparable is called a *total order*.

So $<$ and $\leq$ are total orders on $\mathbb{R}$. On the other hand, the subset relation is

[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.

not total, since, for example, any two different finite sets of the same size will be incomparable under $\subseteq$. The prerequisite relation on Course 6 required subjects is also not total because, for example, neither 8.01 nor 6.001 is a prerequisite of the other.

9.3.1 Problems

Practice Problems

Class Problems

Homework Problems

Exam Problems

9.4 Product Orders

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

Definition 9.4.1. The product, $R_1 \times R_2$, of relations R_1 and R_2 is defined to be the relation with

$$\begin{aligned}
\text{domain}(R_1 \times R_2) &::= \text{domain}(R_1) \times \text{domain}(R_2), \\
\text{codomain}(R_1 \times R_2) &::= \text{codomain}(R_1) \times \text{codomain}(R_2), \\
(a_1, a_2)\,(R_1 \times R_2)\,(b_1, b_2) &\quad \text{iff} \quad [a_1\, R_1\, b_1 \text{ and } a_2\, R_2\, b_2].
\end{aligned}$$

Example 9.4.2. Define a relation, Y, on age-height pairs of being younger *and* shorter. This is the relation on the set of pairs (y, h) where y is a nonnegative integer ≤ 2400 which we interpret as an age in months, and h is a nonnegative integer ≤ 120 describing height in inches. We define Y by the rule

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

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

It follows directly from the definitions that products preserve the properties of transitivity, reflexivity, irreflexivity, and antisymmetry, as shown in Problem **??**. That is, if R_1 and R_2 both have one of these properties, then so does $R_1 \times R_2$. This implies that if R_1 and R_2 are both partial orders, then so is $R_1 \times R_2$.

On the other hand, the property of being a total order is not preserved. For example, the age-height relation Y is the product of two total orders, but it is not total: the age 240 months, height 68 inches pair, (240,68), and the pair (228,72) are incomparable under Y.

9.4.1 Problems

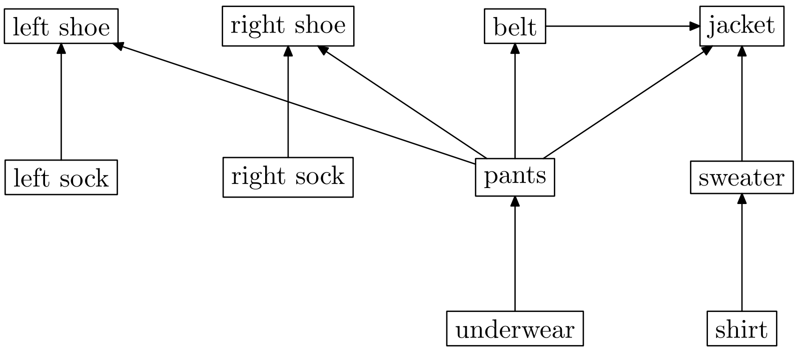
Class Problems

9.5 Scheduling

9.5.1 Scheduling with Constraints

Scheduling problems are a common source of partial orders: there is a set, A, of tasks and a set of constraints specifying that starting a certain task depends on other tasks being completed beforehand. We can picture the constraints by drawing labelled boxes corresponding to different tasks, with an arrow from one box to another if the first box corresponds to a task that must be completed before starting the second one.

Example 9.5.1. Here is a drawing describing the order in which you could put on clothes. The tasks are the clothes to be put on, and the arrows indicate what should be put on directly before what.



When we have a partial order of tasks to be performed, it can be useful to have an order in which to perform all the tasks, one at a time, while respecting the dependency constraints. This amounts to finding a total order that is consistent with the partial order. This task of finding a total ordering that is consistent with a partial order is known as *topological sorting*.

Definition 9.5.2. A *topological sort* of a partial order, $\prec$, on a set, A, is a total ordering, $\sqsubset$, on A such that

$$a \prec b \text{ IMPLIES } a \sqsubset b.$$

For example,

$$\begin{aligned} \text{shirt} \sqsubset \text{sweater} \sqsubset \text{underwear} \sqsubset \text{leftsock} \sqsubset \text{rightsock} \sqsubset \text{pants} \\ \sqsubset \text{leftshoe} \sqsubset \text{rightshoe} \sqsubset \text{belt} \sqsubset \text{jacket}, \end{aligned}$$

is one topological sort of the partial order of dressing tasks given by Example 9.5.1; there are several other possible sorts as well.

Topological sorts for partial orders on finite sets are easy to construct by starting from *minimal* elements:

Definition 9.5.3. Let $\preceq$ be a partial order on a set, A. An element $a_0 \in A$ is *minim<u>um</u>* iff it is $\preceq$ every other element of A, that is, $a_0 \preceq b$ for all $b \neq a_0$.

The element a_0 is *minima<u>l</u>* iff no other element is $\preceq a_0$, that is, NOT$(b \preceq a_0)$ for all $b \neq a_0$.

There are corresponding definitions for *maximum* and *maximal*. Alternatively, a maximum(al) element for a relation, R, could be defined to be as a minimum(al) element for R^{-1}.

In a total order, minimum and minimal elements are the same thing. But a partial order may have no minimum element but lots of minimal elements. There are four minimal elements in the clothes example: leftsock, rightsock, underwear, and shirt.

To construct a total ordering for getting dressed, we pick one of these minimal elements, say shirt. Next we pick a minimal element among the remaining ones. For example, once we have removed shirt, sweater becomes minimal. We continue in this way removing successive minimal elements until all elements have been picked. The sequence of elements in the order they were picked will be a topological sort. This is how the topological sort above for getting dressed was constructed.

EDITING NOTE: pedantic lemma

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 it depends crucially on the partial order being finite —there is no mimimal element among the negative integers for example. So we'll practice doing an elementary proof about partial orders by proving that minimal elements exist.

Lemma 9.5.4. *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 by well ordering, 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 9.5.5. *Every partial order on a finite set has a topological sort.*

There are many other ways of constructing topological sorts. For example, instead of starting "from the bottom" with minimal elements, we could build a total starting *anywhere* and simply keep putting additional elements into the total order wherever they will fit. In fact, the domain of the partial order need not even be finite: we won't prove it, but *all* partial orders, even infinite ones, have topological sorts.

9.5.2 Parallel Task Scheduling

For a partial order of task dependencies, topological sorting provides a way to execute tasks one after another while respecting the dependencies. But what if we have the ability to execute more than one task at the same time? For example, say tasks are programs, the partial order indicates data dependence, and we have a parallel machine with lots of processors instead of a sequential machine with only one. How should we schedule the tasks? Our goal should be to minimize the total *time* to complete all the tasks. For simplicity, let's say all the tasks take one unit of time, and all the processors are identical.

So, given a finite partially ordered set of tasks, how long does it take to do them all, in an optimal parallel schedule? We can also use partial order concepts to analyze this problem.

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

In general, a *schedule* for performing tasks specifies which tasks to do at successive steps. Every task, a, has be scheduled at some step, and all the tasks that have to be completed before task a must be scheduled for an earlier step.

Definition 9.5.6. A *parallel schedule* for a strict partial order, $\prec$, on a set, A, is a partition[3] of A into sets $A_0, A_1, \dots$, such that for all $a, b \in A$, $k \in \mathbb{N}$,

$$[a \in A_k \text{ AND } b \prec a] \quad \text{IMPLIES} \quad b \in A_j \text{ for some } j < k.$$

The set A_k is called the set of elements *scheduled at step k*, and the *length* of the schedule is the number of sets A_k in the partition. The maximum number of elements scheduled at any step is called the *number of processors* required by the schedule.

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

So the schedule we chose above for clothes has four steps

$$\begin{aligned}
A_0 &= \{\text{leftsock, rightsock, underwear, shirt}\}, \\
A_1 &= \{\text{pants, sweater}\}, \\
A_2 &= \{\text{leftshoe, rightshoe, belt}\}, \\
A_3 &= \{\text{jacket}\}.
\end{aligned}$$

and requires four processors (to complete the first step).

Because you have to put on your underwear before your pants, your pants before your belt, and your belt before your jacket, at least four steps are needed in *every* schedule for getting dressed —if we used fewer than four steps, two of these tasks would have to be scheduled at the same time. A set of tasks that must be done in sequence like this is called a *chain*.

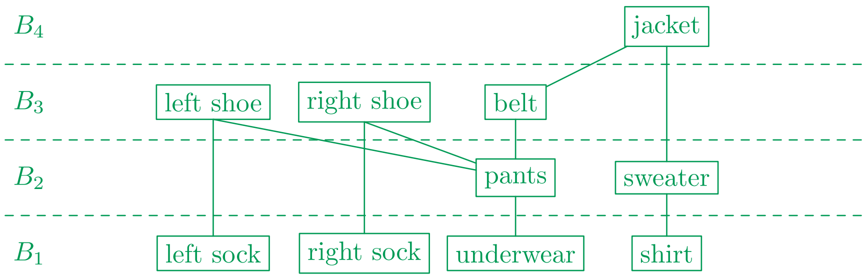
Definition 9.5.7. A *chain* in a partial order is a set of elements such that any two different elements in the set are comparable. A chain is said to *end at* an its maximum element.

In general, the earliest step at which a task, a, can ever be scheduled must be at least as large as any chain that ends at a. A *largest* chain ending at a is called a *critical path* to a, and the size of the critical path is called the *depth* of a. So in any possible parallel schedule, it takes at least depth (a) steps to complete task a.

There is a very simple schedule that completes every task in this minimum number of steps. Just use a "greedy" strategy of performing tasks as soon as possible. Namely, schedule all the elements of depth k at step k. That's how we found the schedule for getting dressed given above.

EDITING NOTE:

For getting dressed, here is a picture of the schedule obtained in this way:



■

Theorem 9.5.8. *Let $\prec$ be a strict partial order on a set, A. A minimum length schedule* *for $\prec$ consists of the sets $A_0, A_1, \ldots$, where*

$$A_k ::= \{a \mid \text{depth}\,(a) = k\}\,.$$

We'll leave to Problem **??** the proof that the sets A_k are a parallel schedule according to Definition 9.5.6.

The minimum number of steps needed to schedule a partial order, $\prec$, is called the *parallel time* required by $\prec$, and a largest possible chain in $\prec$ is called a *critical path* for $\prec$. So we can summarize the story above by this way: with an unlimited number of processors, the minimum parallel time to complete all tasks is simply the size of a critical path:

Corollary 9.5.9. *Parallel time = length of critical path.*

9.6 Dilworth's Lemma

Definition 9.6.1. An *antichain* in a partial order is a set of elements such that any two elements in the set are incomparable.

For example, it's easy to verify that each set A_k is an antichain (see Problem **??**).

So our conclusions about scheduling also tell us something about antichains.

Corollary 9.6.2. *If the largest chain in a partial order on a set, A, is of size t, then A can be partitioned into t antichains.*

Proof. Let the antichains be the sets $A_2, A_2, \ldots, A_t$. ■

Corollary 9.6.2 implies a famous result[4] about partially ordered sets:

Lemma 9.6.3 (Dilworth)**.** *For all $t > 0$, every partially ordered set with n elements must have either a chain of size greater than t or an antichain of size at least n/t.*

Proof. Suppose the largest chain is of size $\leq t$. Then by Corollary 9.6.2, the n elements can be partitioned into at most t antichains. Let ℓ be the size of the largest antichain. Since every element belongs to exactly one antichain, and there are at most t antichains, there can't be more than ℓt elements, namely, $\ell t \geq n$. So there is an antichain with at least $\ell \geq n/t$ elements. ■

[4]Lemma 9.6.3 also follows from a more general result known as Dilworth's Theorem which we will not discuss.

Corollary 9.6.4. *Every partially ordered set with n elements has a chain of size greater than $\sqrt{n}$ or an antichain of size at least $\sqrt{n}$.*

Proof. Set $t = \sqrt{n}$ in Lemma 9.6.3. ■

Example 9.6.5. In the dressing partially ordered set, $n = 10$.

Try $t = 3$. There is a chain of size 4.

Try $t = 4$. There is no chain of size 5, but there is an antichain of size $4 \geq 10/4$.

Example 9.6.6. Suppose we have a class of 101 students. Then using the product partial order, Y, from Example 9.4.2, we can apply Dilworth's Lemma to conclude that there is a chain of 11 students who get taller as they get older, or an antichain of 11 students who get taller as they get younger, which makes for an amusing in-class demo.

9.6.1 Problems

Practice Problems

Class Problems

Homework Problems

Chapter 10

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.

10.1 Basic definitions

A state machine is really nothing more than a binary relation on a set, except that the elements of the set are called "states," the relation is called the *transition relation*, and a pair (p, q) in the graph of the transition relation is called a *transition*. The transition from state p to state q will be written $p \longrightarrow q$. The transition relation is also called the *state graph* of the machine. A state machine also comes equipped with a designated *start state*.

State machines used in digital logic and compilers usually have only a finite number of states, but machines that model continuing computations typically have an infinite number of states. In many applications, the states, and/or the transitions have labels indicating input or output values, costs, capacities, or probabilities, but for our purposes, unlabelled states and transitions are all we need.[1]

Example 10.1.1. A bounded counter, which counts from 0 to 99 and overflows at

[1]We do name states, as in Figure 10.1, so we can talk about them, but the names aren't part of the state machine.

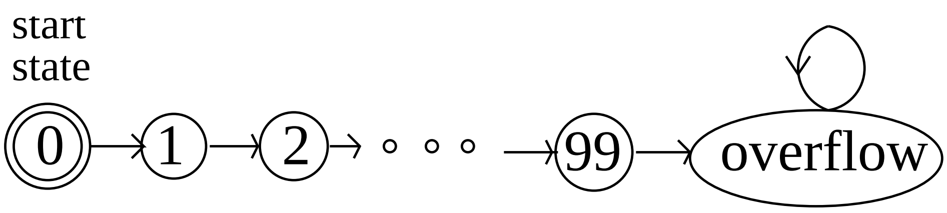


Figure 10.1: *State transitions for the 99-bounded counter.*

100. The transitions are pictured in Figure 10.1, with start state zero. This machine isn't much use once it overflows, since it has no way to get out of its overflow state.

Example 10.1.2. An unbounded counter is similar, but has an infinite state set. This is harder to draw `:-)` .

Example 10.1.3. **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?

Fortunately, they find a solution in the nick of time. We'll let the reader work out how.

The *Die Hard* series is getting tired, so we propose a final *Die Hard Once and For All*. Here Simon's brother returns to avenge him, and he poses the same challenge, but with the 5 gallon jug replaced by a 9 gallon one.

We can model jug-filling scenarios with a state machine. In the scenario with a 3 and a 5 gallon water jug, the states will be pairs, (b, l) of real numbers such that $0 \leq b \leq 5, 0 \leq l \leq 3$. We let b and l be arbitrary real numbers. (We can prove that the values of b and l will only be nonnegative integers, but we won't assume this.) The start state is $(0, 0)$, since both jugs start empty.

Since the amount of water in the jug must be known exactly, we will only consider moves in which a jug gets completely filled or completely emptied. There are several kinds of transitions:

1. Fill the little jug: $(b, l) \longrightarrow (b, 3)$ for $l < 3$.

2. Fill the big jug: $(b, l) \longrightarrow (5, l)$ for $b < 5$.

3. Empty the little jug: $(b, l) \longrightarrow (b, 0)$ for $l > 0$.

4. Empty the big jug: $(b, l) \longrightarrow (0, l)$ for $b > 0$.

5. Pour from the little jug into the big jug: for $l > 0$,

$$(b, l) \longrightarrow \begin{cases} (b + l, 0) & \text{if } b + l \leq 5, \\ (5, l - (5 - b)) & \text{otherwise.} \end{cases}$$

6. Pour from big jug into little jug: for $b > 0$,

$$(b, l) \longrightarrow \begin{cases} (0, b + l) & \text{if } b + l \leq 3, \\ (b - (3 - l), 3) & \text{otherwise.} \end{cases}$$

Note that in contrast to the 99-counter state machine, there is more than one possible transition out of states in the Die Hard machine. Machines like the 99-counter with at most one transition out of each state are called *deterministic*. The Die Hard machine is *nondeterministic* because some states have transitions to several different states.

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

10.2 Reachability and Preserved Invariants

The Die Hard 3 machine models every possible way of pouring water among the jugs according to the rules. Die Hard properties that we want to verify can now be expressed and proved using the state machine model. For example, Bruce's character will disarm the bomb if he can get to some state of the form $(4, l)$.

A (possibly infinite) path through the state graph beginning at the start state corresponds to a possible system behavior; such a path is called an *execution* of the state machine. A state is called *reachable* if it appears in some execution. The bomb in Die Hard 3 gets disarmed successfully because the state (4,3) is reachable.

A useful approach in analyzing state machine is to identify properties of states that are preserved by transitions.

Definition 10.2.1. A *preserved invariant* of a state machine is a predicate, P, on states, such that whenever $P(q)$ is true of a state, q, and $q \longrightarrow r$ for some state, r, then $P(r)$ holds.

The Invariant Principle

If a preserved invariant of a state machine is true for the start state, then it is true for all reachable states.

The Invariant Principle is nothing more than the Induction Principle reformulated in a convenient form for state machines. Showing that a predicate is true in the start state is the base case of the induction, and showing that a predicate is a preserved invariant is the inductive step.[2]

[2]Preserved invariants are commonly just called "invariants" in the literature on program correctness, but we decided to throw in the extra adjective to avoid confusion with other definitions. For example, another subject at MIT uses "invariant" to mean "predicate true of all reachable states." Let's call this definition "invariant-2." Now invariant-2 seems like a reasonable definition, since unreachable states by definition don't matter, and all we want to show is that a desired property is invariant-2. But this confuses the *objective* of demonstrating that a property is invariant-2 with the *method* for showing that it is. After all, if we already knew that a property was invariant-2, we'd have no need for an Invariant Principle to demonstrate it.

10.2.1 Die Hard Once and For All

Now back to Die Hard Once and For All. This time there is a 9 gallon jug instead of the 5 gallon jug. We can model this with a state machine whose states and transitions are specified the same way as for the Die Hard 3 machine, with all occurrences of "5" replaced by "9."

Now reaching any state of the form $(4, l)$ is impossible. We prove this using the Invariant Principle. Namely, we define the preserved invariant predicate, $P(b, l)$, to be that b and l are nonnegative integer multiples of 3. So P obviously holds for the state state $(0, 0)$.

To prove that P is a preserved invariant, we assume $P(b, l)$ holds for some state (b, l) and show that if $(b, l) \longrightarrow (b', l')$, then $P(b', l')$. The proof divides into cases, according to which transition rule is used. For example, suppose the transition followed from the "fill the little jug" rule. This means $(b, l) \longrightarrow (b, 3)$. But $P(b, l)$ implies that b is an integer multiple of 3, and of course 3 is an integer multiple of 3, so P still holds for the new state $(b, 3)$. Another example is when the transition rule used is "pour from big jug into little jug" for the subcase that $b + l > 3$. Then state is $(b, l) \longrightarrow (b - (3 - l), 3)$. But since b and l are integer multiples of 3, so is $b - (3 - l)$. So in this case too, P holds after the transition.

We won't bother to crank out the remaining cases, which can all be checked just as easily. Now by the Invariant Principle, we conclude that every reachable state satisifies P. But since no state of the form $(4, l)$ satisifies P, we have proved rigorously that Bruce dies once and for all!

By the way, notice that the state (1,0), which satisfies NOT(P), has a transition to (0,0), which satisfies P. So it's wrong to assume that the complement of a preserved invariant is also a preserved invariant.

10.2.2 A Robot on a Grid

There is a robot. It walks around on a grid, and at every step it moves diagonally in a way that changes its position by one unit up or down *and* one unit left or right. The robot starts at position $(0, 0)$. Can the robot reach position $(1, 0)$?

To get some intuition, we can simulate some robot moves. For example, starting at (0,0) the robot could move northeast to (1,1), then southeast to (2,0), then southwest to (1,-1), then southwest again to (0,-2).

Let's model the problem as a state machine and then find a suitable invariant. A state will be a pair of integers corresponding to the coordinates of the robot's position. State (i, j) has transitions to four different states: $(i \pm 1, j \pm 1)$.

The problem is now to choose an appropriate preserved invariant, P, that is true for the start state $(0, 0)$ and false for $(1, 0)$. The Invariant Theorem then will imply that the robot can never reach $(1, 0)$. A direct attempt for a preserved invariant is the predicate $P(q)$ that $q \neq (1, 0)$.

Unfortunately, this is not going to work. Consider the state $(2, 1)$. Clearly $P(2, 1)$ holds because $(2, 1) \neq (1, 0)$. And of course $P(1, 0)$ does not hold. But $(2, 1) \longrightarrow (1, 0)$, so this choice of P will not yield a preserved invariant.

We need a stronger predicate. Looking at our example execution you might be able to guess a proper one, namely, that the sum of the coordinates is even! If we can prove that this is a preserved invariant, then we have proven that the robot never reaches $(1,0)$ —because the sum $1+0$ of its coordinates is odd, while the sum $0+0$ of the coordinates of the start state is even.

Theorem 10.2.2. *The sum of the robot's coordinates is always even.*

Proof. The proof uses the Invariant Principle.

Let $P(i,j)$ be the predicate that $i+j$ is even.

First, we must show that the predicate holds for the start state $(0,0)$. Clearly, $P(0,0)$ is true because $0+0$ is even.

Next, we must show that P is a preserved invariant. That is, we must show that for each transition $(i,j) \longrightarrow (i',j')$, if $i+j$ is even, then $i'+j'$ is even. But $i' = i \pm 1$ and $j' = j \pm 1$ by definition of the transitions. Therefore, $i'+j'$ is equal to $i+j$ or $i+j \pm 2$, all of which are even. ■

Corollary 10.2.3. *The robot cannot reach* $(1,0)$.

Robert W. Floyd

The Invariant Principle was formulated by Robert Floyd at Carnegie Tech[a] in 1967. Floyd was already famous for work on formal grammars which transformed the field of programming language parsing; that was how he got to be a professor even though he never got a Ph.D. (He was admitted to a PhD program as a teenage prodigy, but flunked out and never went back.)

In that same year, Albert R. Meyer was appointed Assistant Professor in the Carnegie Tech Computer Science Department where he first met Floyd. Floyd and Meyer were the only theoreticians in the department, and they were both delighted to talk about their shared interests. After just a few conversations, Floyd's new junior colleague decided that Floyd was the smartest person he had ever met.

Naturally, one of the first things Floyd wanted to tell Meyer about was his new, as yet unpublished, Invariant Principle. Floyd explained the result to Meyer, and Meyer wondered (privately) how someone as brilliant as Floyd could be excited by such a trivial observation. Floyd had to show Meyer a bunch of examples be-

10.3 Sequential algorithm examples

10.3.1 Proving Correctness

Robert Floyd, who pioneered modern approaches to program verification, distinguished two aspects of state machine or process correctness:

1. The property that the final results, if any, of the process satisfy system requirements. This is called *partial correctness*.

   You might suppose that if a result was only partially correct, then it might also be partially incorrect, but that's not what he meant. The word "partial" comes from viewing a process that might not terminate as computing a *partial function*. So partial correctness means that when there is a result, it is correct, but the process might not always produce a result, perhaps because it gets stuck in a loop.

2. The property that the process always finishes, or is guaranteed to produce some legitimate final output. This is called *termination*.

Partial correctness can commonly be proved using the Invariant Principle. Termination can commonly be proved using the Well Ordering Principle. We'll illustrate Floyd's ideas by verifying the Euclidean Greatest Common Divisor (GCD) Algorithm.

10.3.2 The Euclidean Algorithm

The *Euclidean algorithm* is a three-thousand-year-old procedure to compute the greatest common divisor, $\gcd(a, b)$ of integers a and b. We can represent this algorithm as a state machine. A state will be a pair of integers (x, y) which we can think of as integer registers in a register program. The state transitions are defined by the rule

$$(x, y) \longrightarrow (y, \text{remainder}(x, y))$$

for $y \neq 0$. The algorithm terminates when no further transition is possible, namely when $y = 0$. The final answer is in x.

We want to prove:

1. Starting from the state with $x = a$ and $y = b > 0$, if we ever finish, then we have the right answer. That is, at termination, $x = \gcd(a, b)$. This is a *partial correctness* claim.

2. We do actually finish. This is a process *termination* claim.

Partial Correctness of GCD First let's prove that if GCD gives an answer, it is a correct answer. Specifically, let $d ::= \gcd(a, b)$. We want to prove that *if* the procedure finishes in a state (x, y), then $x = d$.

Proof. Define the state predicate

$$P(x, y) ::= \quad [\gcd(x, y) = d \text{ and } (x > 0 \text{ or } y > 0)].$$

P holds for the start state (a, b), by definition of d and the requirement that b is positive. Also, the preserved invariance of P follows immediately from

Lemma 10.3.1. *For all $m, n \in \mathbb{N}$ such that $n \neq 0$,*

$$\gcd(m, n) = \gcd(n, \text{remainder}(m, n)). \tag{10.1}$$

Lemma 10.3.1 is easy to prove: let q be the quotient and r be the remainder of m divided by n. Then $m = qn + r$ by definition. So any factor of both r and n will be a factor of m, and similarly any factor of both m and n will be a factor of r. So r, n and m, n have the same common factors and therefore the same gcd. Now by the Invariant Principle, P holds for all reachable states.

Since the only rule for termination is that $y = 0$, it follows that if (x, y) is a terminal state, then $y = 0$. If this terminal state is reachable, then the preserved invariant holds for (x, y). This implies that $\gcd(x, 0) = d$ and that $x > 0$. We conclude that $x = \gcd(x, 0) = d$. ■

Termination of GCD Now we turn to the second property, that the procedure must terminate. To prove this, notice that y gets strictly smaller after any one transition. That's because the value of y after the transition is the remainder of x divided by y, and this remainder is smaller than y by definition. But the value of y is always a nonnegative integer, so by the Well Ordering Principle, it reaches a minimum value among all its values at reachable states. But there can't be a transition from a state where y has its minimum value, because the transition would decrease y still further. So the reachable state where y has its minimum value is a state at which no further step is possible, that is, at which the procedure terminates.

Note that this argument does not prove that the minimum value of y is zero, only that the minimum value occurs at termination. But we already noted that the only rule for termination is that $y = 0$, so it follows that the minimum value of y must indeed be zero.

EDITING NOTE:

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,

$$\gcd(a, b) = sa + tb \tag{10.2}$$

for some $s, t \in \mathbb{Z}$. We'll see some nice proofs of (10.2) later when we study Number Theory, but now we'll look at an extension of the Euclidean Algorithm that efficiently, if obscurely, produces the desired s and t. It is 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).

Don't worry if you find this Extended Euclidean Algorithm hard to follow, and you can't imagine where it came from. In fact, that's good, because this will illustrate an important point: given the right preserved invariant, you can verify programs you don't understand.

In particular, given nonnegative integers x and y, with $y > 0$, we claim the following procedure[3] halts with registers `S` and `T` containing integers s and t satisfying (10.2).

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;
```

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

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

The following properties are preserved invariants that imply partial correctness:

$$\gcd(X,Y) = \gcd(a,b), \tag{10.3}$$

$$Sa+Tb = Y, \text{ and} \tag{10.4}$$

$$Ua+Vb = X. \tag{10.5}$$

To verify that these are preserved invariants, note that (10.3) is the same one we observed for the Euclidean algorithm. To check the other two properties, 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 properties hold for these values. We must prove that (10.4) and (10.5) hold (we already know (10.3) 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 (10.5) holds for

u', v', x' because of (10.4) for s, t, y. Also,

$$s' = u - qs, \quad t' = v - qt, \quad y' = x - qy$$

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 (10.4) holds for s', t', y'.

Also, it's easy to check that all three preserved invariants are true just before the first time around the loop. Namely, at the start:

$$X = a, Y = b, S = 0, T = 1 \qquad \text{so}$$

$$Sa + Tb = 0a + 1b = b = Y \qquad \text{confirming (10.4).}$$

Also,

$$U = 1, V = 0, \qquad \text{so}$$

$$Ua + Vb = 1a + 0b = a = X \qquad \text{confirming (10.5).}$$

Now 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 preserved invariants (10.3) and (10.4) 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 by repeating that you're not expected to understand this program. ■

We've already observed that a preserved invariant is really just an induction hypothesis. As with induction, finding the right hypothesis is usually the hard part. We repeat:

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

We expect that the Extended Euclidean Algorithm presented above illustrates this point.

10.4 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 $\rightarrow \mathbb{R}$, 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-coord$((i,j)) ::= i$.

We can formulate our general termination method as follows:

Definition 10.4.1. Let $\prec$ be a strict partial order on a set, A. A derived variable f : states $\rightarrow A$ 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 `Y`, respectively, that were nonnegative integer-valued and strictly decreasing. We can summarize this approach to proving termination as follows:

Theorem 10.4.2. *If f is a strictly decreasing $\mathbb{N}$-valued derived variable of a state machine, then the length of any execution starting at state q is at most $f(q)$.*

Of course we could prove Theorem 10.4.2 by induction on the value of $f(q)$, but think about what it says: "If you start counting down at some nonnegative integer $f(q)$, then you can't count down more than $f(q)$ times." Put this way, it's obvious.

10.4.1 Weakly Decreasing Variables

In addition being strictly decreasing, it will be useful to have derived variables with some other, related properties.

Definition 10.4.3. Let $\preceq$ be a weak partial order on a set, A. A derived variable $f : Q \to A$ is *weakly decreasing* iff

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

Strictly increasing and *weakly increasing* derived variables are defined similarly.[4]

[4] 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.

EDITING NOTE:

Well-founded termination

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 whenever there is a derived variable that strictly decreases with respect to any well-founded partial order.

We now define some other useful flavors of derived variables taking values over partial ordered sets. We'll use the notational convention that when $\prec$ denotes a strict partial order on some set, then $\preceq$ is the corresponding *weak* partial order

$$a \preceq a' \quad ::= \quad a \prec a' \vee a = a'.$$

Definition 10.4.4. 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.[5]

Theorem 10.4.5. *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 10.4.5 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

[5] 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.

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 10.4.6. *The robot will always get stuck at the origin.*

If we think of the robot as a nondeterministic state machine, then Claim 10.4.6 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 10.4.2.

So does Claim 10.4.6 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 10.4.5 proves that the robot always get stuck as claimed. ■

10.4.2 Problems

Homework Problems

Class Problems

10.5 The Alternating Bit Protocol

EDITING NOTE: Lynch Notes S07.H11-sm ■

The Alternating Bit Protocol is a well-known two-process communication protocol that achieves reliable FIFO communication over unreliable channels. The unreliable channels may lose or duplicate messages, but are assumed not to reorder them. We'll use the Invariant Method to verify that the Protocol

The Protocol allows a **Sender** process to send a sequence of messages from a message alphabet, M, to a **Receiver** process. It works as follows.

Sender repeatedly sends the rightmost message in its **outgoing-queue** of messages, tagged with a **tagbit** that is initially 1. When **Receiver** receives this tagged message, it sets its **ackbit** to be the message tag 1, and adds the message to the lefthand end of its **received-msgs** list. Then as an acknowledgement, **Receiver** sends back **ackbit** 1 repeatedly. When **Sender** gets this acknowledgement bit, it deletes the rightmost outgoing message in its queue, sets its **tagbit** to 0, and begins sending the new rightmost outgoing message, tagged with **tagbit**.

Receiver, having already accepted the message tagged with **ackbit** 1, ignores subsequent messages with tag 1, and waits until it sees the first message with tag 0; it adds this message to the lefthand side of its **received-msgs** list, sets **ackbit** to 0 and acknowledges repeatedly with with **ackbit** 0. **Sender** now waits till it gets acknowledgement bit 0, then goes on to send the next outgoing message with tag 1. In this way, it alternates use of the tags 1 and 0 for successive messages.

We claim that this causes **Sender** to receive *suffix* original **outgoing-msgs** queue. That is, at any stage in the process when the the **outgoing-msgs**

(The fact that **Sender** actually outputs the entire outgoing queuee is a *liveness* claim —liveness properties are a generalization of termination properties. We'll ignore this issue for now.)

We formalize the description above as a state whose states consist of:

outgoing-msgs, a finite sequence of M, whose initial value is called **all-msgs**

tagbit $\in \{0, 1\}$, initially 1

received-msgs, a finite sequence of M, initially empty

ackbit$(\in \{0, 1\}$, initially 0

msg-channel, a finite sequence of $M \times \{0, 1\}$, initially empty,

ack-channel, a finite sequence of $\{0, 1\}$, initially empty

The transitions are:

SEND: (a) **action**: **send-msg**(m, b)

precondition: $m =$ **rightend**(**outgoing-msgs**) AND $b =$ **tagbit**

effect: add (m, b) to the lefthand end of **msg-channel**, any number ≥ 0 of times

(b) **action**: **send-ack**(b)

precondition: $b =$ **ackbit**

effect: add b to the righthand end of **ack-channel**, any number ≥ 0 of times

RECEIVE: (a) **action**: **receive-msg**(m, b)

precondition: $(m, b) =$ **rightend**(**msg-channel**)

effect: remove **rightend** of **msg-channel**;

if $b \neq$ **ackbit**, then [add m to the lefthand end of **receive-msgs**; **ackbit** $:=$ b.]

(b) **action**: **receive-ack**(b)

precondition: $b =$ **leftend**(**ack-channel**()

effect: remove **leftend** of **ack-channel**.

if $b =$ **tagbit**, then [remove **rightend** of **outgoing-msgs** (if nonempty); **tagbit** $:=$ $\overline{\textbf{tagbit}}$]

Our goal is to show that when **tagbit** $\neq$ **ackbit**, then

$$\textbf{outgoing-queue} \cdot \textbf{received-msgs} = \textbf{all-msgs}. \qquad (10.6)$$

This requires three auxiliary invariants. For the first of these, we need a definition.

Let **tag-sequence** be the sequence consisting of bits in **ack-channel**, in right-to-left order, followed by **tagbit**, followed by the tag components of the elements of **msg-channel**, in left-to-right order, followed by **ackbit**.

Property 2: tag-sequence consists of one of the following:

1. All 0's.
2. All 1's.
3. A positive number of 0's followed by a positive number of 1's.
4. A positive number of 1's followed by a positive number of 0's.

What is being ruled out by these four cases is the situation where the sequence contains more than one switch of tag value.

The fact that Property 2 is an invariant can be proved easily by induction. We also need:

Property 3: If $(m, \textbf{tag})$ is in **msg-channel** then $m = \textbf{rightend}(\textbf{outgoing-queue})$.

Proof. (That Property 3 is an invariant)

By induction, using Property 2.

Base: Obvious, since no message is in the channel initially.

Inductive step: It is easy to see that the property is preserved by $\textbf{send}_{m,b}$, which adds new messages to $\textbf{channel}_{1,2}$. The only other case that could cause a problem is $\textbf{receive}(b)_{2,1}$, which could cause $\textbf{tag}_1$ to change when there is another message already in $\textbf{channel}_{1,2}$ with the same tag. But this can't happen, by Property 2 applied before the step – since the incoming tag g must be equal to $\textbf{tag}_1$ in this case, all the tags in **tag-sequence** must be the same. ■

Finally, we need that the following counterpart to (10.6): when **tagbit** $=$ **ackbit**, then

$$\textbf{lefttail}(\textbf{outgoing-queue}) \cdot \textbf{received-msgs} = \textbf{all-msgs}, \tag{10.7}$$

where **lefttail**(**outgoing-queue**) all but the rightmost message, if any, in **outgoing-queue**.

Property 4, part 2, easily implies the goal Property 1. It also implies that **work-buf**$_2$ is always nonempty when **receive**$(b)_{2,1}$ occurs with equal tags; therefore, the parenthetical check in the code always works out to be true.

Proof. (That Property 4 is an invariant)

By induction. Base: In an initial state, the tags are unequal, **work-buf**$_1$ $=$ **buf**$_1$ and **buf**$_2$ is empty. This suffices to show part 1. part 2 is vacuous.

Inductive step: When a **send** occurs, the tags and buffers are unchanged, so the truth of the invariants must be preserved. It remains to consider **receive** events.

receive$(m, b)_{1,2}$:

If $b =$ **tag**$_2$, nothing happens, so the invariants are preserved. So suppose that

$b \neq \mathbf{tag}_2$. Then Property 2 implies that $b = \mathbf{tag}_1$, and then Property 3 implies that m is the first message on $\mathbf{work\text{-}buf}_1$. The effect of the transition is to change $\mathbf{tag}_2$ to make it equal to $\mathbf{tag}_1$, and to replicate the first element of $\mathbf{work\text{-}buf}22_1$ at the end of $\mathbf{buf}_2$.

The inductive hypothesis implies that, before the step, $\mathbf{buf}_2 \cdot \mathbf{work\text{-}buf}_1 = \mathbf{buf}_1$. The changes caused by the step imply that, after the step, $\mathbf{tag}_1 = \mathbf{tag}_2$, $\mathbf{work\text{-}buf}_1$ and $\mathbf{buf}_2$ are nonempty, $\mathbf{head}(\mathbf{work\text{-}buf}_1) = \mathbf{last}(\mathbf{buf}_2)$, and $\mathbf{buf}_2 \cdot \mathbf{tail}(\mathbf{work\text{-}buf}_1) = \mathbf{buf}_1$. This is as needed.

$\mathbf{receive}(b)_{2,1}$:

The argument is similar to the one for $\mathbf{receive}(m, b)_{1,2}$. If $b \neq \mathbf{tag}_1$, nothing happens so the invariants are preserved. So suppose that $b = \mathbf{tag}_1$. Then Property 2 implies that $b = \mathbf{tag}_2$, and the step changes $\mathbf{tag}_1$ to make it unequal to $\mathbf{tag}44_2$. The step also removes the first element of $\mathbf{work\text{-}buf}_1$. The inductive hypothesis implies that, before the step, $\mathbf{work\text{-}buf}_1$ and $\mathbf{buf}_2$ are nonempty, $\mathbf{head}(\mathbf{work\text{-}buf}_1) = \mathbf{last}(\mathbf{buf}_2)$, and $\mathbf{buf}_2 \cdot \mathbf{tail}(\mathbf{work\text{-}buf}_1) = \mathbf{buf}_1$. The changes caused by the step imply that, after the step, $\mathbf{tag}_1 \neq \mathbf{tag}_2$ and $\mathbf{buf}_2 \cdot \textbf{work-buf}_1 = \mathbf{buf}_1$. This is as needed. ■

10.6 Reasoning About While Programs

Real programs and programming languages are often huge and complicated, making them hard to model and even harder to reason about. Still, making programs "reasonable" is a crucial aspect of software engineering. In this section we'll illustrate what it means to have a clean mathematical model of a simple programming language and reasoning principles that go with it —if only real programming languages allowed for such simple, accurate modeling.

10.6.1 While Programs

The programs we'll study are called "***while** programs*." We can define them as a recursive data type:

Definition 10.6.1.

base cases:

- $x := e$ is a **while** program, called an *assignment statement*, where x is a variable and e is an expression.
- ***Done*** is a **while** program.

constructor cases: If C and D are **while** programs, and T is a test, then the following are also **while** programs:

- $C;D$ —called the *sequencing* of C and D,
- **if** T **then** C **else** D —called a *conditional* with *test*, T, and *branches*, C and D,
- **while** T **do** C **od** —called a *while loop* with *test*, T, and *body*, C.

For simplicity we'll stick to **while** programs operating on integers. So by expressions we'll mean any of the familiar integer valued expressions involving integer constants and operations such as addition, multiplication, exponentiation, quotient or remainder. As *tests*, we'll allow propositional formulas built from basic formulas of the form $e \leq f$ where e and f are expressions. For example, here is the Euclidean algorithm for $\gcd(a, b)$ expressed as a **while** program.

```
x := $a$;
y := $b$;
while y $\neq 0$ do
    t := y;
    y := $\text{rem}$(x, y);
    x := t od
```

10.6.2 The While Program State Machine

A **while** program acts as a pure command: it is run solely for its side effects on stored data and it doesn't return a value. The data consists of integers stored as the values of variables, namely environments:

Definition 10.6.2. An *environment* is a total function from variables to integers. Let Env be set of all environments.

So if ρ is an environment and x is a variable, then $\rho(x)$ is an integer. More generally, the environment determines the integer value of each expression, e, and the truth value of each test, T. We can think of an expression, e as defining a function $[\![e]\!] : \text{Env} \to \mathbb{Z}$, and refer to this function, $[\![e]\!]$ as the *meaning* of e, and likewise for tests.

It's standard in programming language theory to write $[\![e]\!]\rho$ as shorthand for $[\![e]\!](\rho)$, that is, applying the *meaning*, $[\![e]\!]$, of e to ρ. For example, if $\rho(\mathtt{x}) = 4$, and $\rho(\mathtt{y}) = -2$, then

$$[\![\mathtt{x}^2 + \mathtt{y} - \mathtt{3}]\!]\rho = \rho(\mathtt{x})^2 + \rho(y) - 3 = 11. \tag{10.8}$$

Executing a program causes a succession of changes to the environment[6] which may continue until the program halts. Actually the only command which immediately alters the environment is an assignment command. Namely, the effect of the command

$$x := e$$

[6]More sophisticated programming models distinguish the environment from a *store* which is affected by commands, but this distinction is unnecessary for our purposes.

on an environment is that the value assigned to the variable x is changed to the value of e in the original environment. We can say this precisely and concisely using the following notation: $f[a \leftarrow b]$ is a function that is the same as the function, f, except that when applied to element a its value is b. Namely,

Definition 10.6.3. If $f : A \to B$ is a function and a, b are arbitrary elements, define

$$f[a \leftarrow b]$$

to be the function g such that

$$g(u) = \begin{cases} b & \text{if } u = a. \\ f(u) & \text{otherwise.} \end{cases}$$

Now we can specify the step-by-step execution of a **while** program as a state machine, where the states of the machine consist of a **while** program paired with an environment. The transitions of this state machine are defined recursively on the definition of **while** programs.

Definition 10.6.4. The transitions $\langle C, \rho \rangle \longrightarrow \langle D, \rho' \rangle$ of the **while** program state machine are defined as follows:

base cases:

$$\langle x := e,\ \rho\rangle \longrightarrow \langle \textbf{Done},\ \rho[x \leftarrow [\![e]\!]\rho]\rangle$$

constructor cases: If C and D are **while** programs, and T is a test, then:

- if $\langle C,\ \rho\rangle \longrightarrow \langle C',\ \rho'\rangle$, then

  $$\langle C;D,\ \rho\rangle \longrightarrow \langle C';D,\ \rho'\rangle\ .$$

  Also,

  $$\langle \textbf{Done};D,\ \rho\rangle \longrightarrow \langle D,\ \rho\rangle\ .$$

- if $[\![T]\!]\rho = \textbf{T}$, then

  $$\langle \textbf{if}\ T\ \textbf{then}\ C\ \textbf{else}\ D,\ \rho\rangle \longrightarrow \langle C,\ \rho\rangle\ ,$$

  or if $[\![T]\!]\rho = \textbf{F}$, then

  $$\langle \textbf{if}\ T\ \textbf{then}\ C\ \textbf{else}\ D,\ \rho\rangle \longrightarrow \langle D,\ \rho\rangle\ .$$

- if $[\![T]\!]\rho = \textbf{T}$, then

  $$\langle \textbf{while}\ T\ \textbf{do}\ C\ \textbf{od},\ \rho\rangle \longrightarrow \langle C;\textbf{while}\ T\ \textbf{do}\ C\ \textbf{od},\ \rho\rangle$$

or if $[\![T]\!]\rho = \mathbf{F}$, then

$$\langle \textbf{while}\ T\ \textbf{do}\ C\ \textbf{od},\ \rho\rangle \longrightarrow \langle \textbf{Done},\ \rho\rangle\,.$$

Now **while** programs are probably going to be the simplest kind of programs you will ever see, but being condescending about them would be a mistake. It turns that *every function on nonnegative integers that can be computed by any program on any machine whatsoever can also be computed by* **while** programs (maybe more slowly). We can't take the time to explain how such a sweeping claim can be justified, but you can find out by taking a course in computability theory such as 6.045 or 6.840.

10.6.3 Denotational Semantics

The net effect of starting a **while** program in some environment is reflected in the final environment when the program halts. So we can think of a **while** program, C, aas defining a function, $[\![C]\!] : \text{Env} \to \text{Env}$, from initial environments to environments at halting. The function $[\![C]\!]$ is called the *meaning* of C.

$[\![C]\!]$ of a **while** program, C to be a partial function from Env to Env mapping an initial environment to the final halting environment.

We'll need one bit of notation first. For any function $f : S \to S$, let $f^{(n)}$ be the composition of f with itself n times where $n \in \mathbb{N}$. Namely,

$$f^{(0)} ::= \mathrm{Id}_S$$

$$f^{(n+1)} ::= f \circ f^{(n)},$$

where "$\circ$" denotes functional composition.

The recursive definition of the meaning of a program follows the definition of the **while** program recursive data type.

Definition 10.6.5. base cases:

- $[\![x := e]\!]$ is the function from Env to Env defined by the rule:

$$[\![x := e]\!]\rho ::= \rho[x \leftarrow [\![e]\!]\rho].$$

-

$$[\![\textbf{Done}]\!] ::= \mathrm{Id}_{\mathrm{Env}}$$

where $\mathrm{Id}_{\mathrm{Env}}$ is the identity function on Env. In other words, $[\![\mathbf{Done}]\!]\rho ::= \rho$.

constructor cases: If C and D are **while** programs, and T is a test, then:

- $$[\![C;D]\!] ::= [\![D]\!] \circ [\![C]\!]$$

  That is,

  $$[\![C;D]\!]\rho ::= [\![D]\!]([\![C]\!]\rho).$$

- $$[\![\mathbf{if}\ T\ \mathbf{then}\ C\ \mathbf{else}\ D]\!]\rho ::= \begin{cases} [\![C]\!]\rho & \text{if } [\![T]\!]\rho = \mathbf{T} \\ [\![D]\!]\rho & \text{if } [\![T]\!]\rho = \mathbf{F}. \end{cases}$$

- $$[\![\mathbf{while}\ T\ \mathbf{do}\ C\ \mathbf{od}]\!]\rho ::= [\![C]\!]^{(n)}\rho$$

  where n is the least nonnegative integer such that $[\![T]\!]([\![C]\!]^{(n)}\rho) = \mathbf{F}$. (If there is no such n, then $[\![\mathbf{while}\ T\ \mathbf{do}\ C\ \mathbf{od}]\!]\rho$ is undefined.)

We can use the denotational semantics of **while** programs to reason about **while** programs using structural induction on programs, and this is often much simpler than reasoning about them using induction on the number of steps in an execution. This is OK as long as the denotational semantics accurately captures the state machine behavior. In particular, using the notation $\longrightarrow^*$ for the transitive closure of the transition relation:

Theorem 10.6.6.

$$\langle C, \rho\rangle \longrightarrow^* \langle \textbf{\textit{Done}}, \rho'\rangle \quad \textit{iff} \quad [\![C]\!]\rho = \rho'$$

Theorem 10.6.6 can be proved easily by induction; it appear in Problem**??**.

10.6.4 Problems

Homework Problems

10.6.5 Logic of Programs

A typical program specification describes the kind of inputs and environments the program should handle, and then describes what should result from an execution. The specification of the inputs or initial environment is called the *precondition* for program execution, and the prescription of what the result of execution should be is called the *postcondition*. So if P is a logical formula expressing the precondition for a program, C, and likewise Q expresses the postcondition, the specification requires that

If P holds when C is started, then Q will hold if and when C halts.

We'll express this requirement as a formula

$$P \ \{C\} \ Q$$

called a *partial correctness assertion*.

For example, if E is **while** program above for the Euclidean algorithm, then the partial correctness of E can be expressed as

$$(a, b \in \mathbb{N} \text{ AND } \mathtt{x} \neq 0) \ \{C\} \ (\mathtt{x} = \gcd(a, b))\,. \tag{10.9}$$

More precisely, notice that just as the value of an expression in an environment is an integer, the value of a logical formula in an environment is a truth value. For example, if $\rho(\mathtt{x}) = 4$, and $\rho(\mathtt{y}) = -2$, then by (10.8), $[\![\mathtt{x}^2 + \mathtt{y} - 3]\!]\rho = 11$, so

$$[\![\exists \mathtt{z}.\, \mathtt{z} > 4 \text{ AND } \mathtt{x}^2 + \mathtt{y} - 3 = \mathtt{z}]\!]\rho = \mathbf{T},$$

$$[\![\exists \mathtt{z}.\, \mathtt{z} > 13 \text{ AND } \mathtt{x}^2 + \mathtt{y} - 3 = \mathtt{z}]\!]\rho = \mathbf{F}.$$

Definition 10.6.7. For logical formulas P and Q, and **while** program, C, the partial correctness assertion

$$P\ \{C\}\ Q$$

is true proving that for all environments, rho, if $[\![P]\!]\rho$ is true, and $\langle C, \rho\rangle \longrightarrow^* \langle \mathbf{Done}, \rho'\rangle$ for some ρ', then $[\![Q]\!]\rho'$ is true.

In the 1970', Univ. Dublin formulated a set of inference rules for proving partial correctness formulas. These rules are known as *Hoare Logic*.

The first rule captures the fact that strengthening the preconditions and weakening the postconditions makes a partial correctness specification easier to satisfy:

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

The rest of the logical rules follow the recursive definition of **while** programs.

There are axioms for the base case commands:

$$P(x)\ \{x := e\}\ P(e)$$
$$P\ \{\textbf{Done}\}\ P,$$

and proof rules for the constructor cases:

- $$\frac{P\ \{C\}\ Q \text{ AND } Q\ \{D\}\ R}{P\ \{C;D\}\ R}$$

- $$\frac{P \text{ AND } T\ \{C\}\ Q}{P \text{ AND } T\ \{\textbf{if } T \textbf{ then } C \textbf{ else } D\}\ Q \text{ AND } T}$$

- $$\frac{P \text{ AND } T\ \{C\}\ P}{P\ \{\textbf{while } T \textbf{ do } C \textbf{ od}\}\ P \text{ AND } \text{NOT}(T)}$$

Example 10.6.8. **Proof of partial correctness (10.9) for the Euclidean algorithm.**

EDITING NOTE: **TBA - Brief discussion of "relative completeness".** ■

Part III

Counting

Chapter 11

Sums & Asymptotics

EDITING NOTE:

Closed Forms and Approximations

Sums and products arise regularly in the analysis of algorithms and in other technical areas such as finance and probabilistic systems. For example, we used Well Ordering to prove in Theorem 3.2.1 that

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}. \tag{11.1}$$

The simple *closed form* expression $n(n+1)/2$ makes the sum a lot easier to understand and evaluate. But the proof of equation (11.1) by Well Ordering does not explain where it came from in the first place. In Section 11.3, we'll discuss ways to find such closed forms. Even when there are no closed forms exactly equal to a sum, we may still be able to find a closed form that *approximates* a sum with useful accuracy.

The product we focus on in this chapter is the familiar factorial:

$$n! ::= 1 \cdot 2 \cdots (n-1) \cdot n = \prod_{i=1}^{n} i.$$

We'll describe a closed form approximation for it called *Stirling's Formula*.

Finally, when there isn't a good closed form approximation for some expression, there may still be a closed form that characterizes its growth rate. We'll introduce *asymptotic notation*, such as "big Oh", to describe growth rates. ■

11.1 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 thirty years.

In Japan, the standard interest rate is near zero%, and on a few occasions in the past few years has even been slightly negative. It's a mystery why the Japanese populace keeps any money in their banks.

11.1.1 The Future Value of Money

So for an n-year, m-payment annuity, the first payment of m dollars is truly worth m dollars. But the second payment a year later is worth only $m/(1+p)$ dollars. Similarly, the third payment is worth $m/(1+p)^2$, and the n-th payment is worth only $m/(1+p)^{n-1}$. The total value, V, of the annuity is equal to the sum of the payment values. This gives:

$$\begin{aligned}
V &= \sum_{i=1}^{n} \frac{m}{(1+p)^{i-1}} \\
&= m \cdot \sum_{j=0}^{n-1} \left(\frac{1}{1+p}\right)^j && \text{(substitute } j ::= i-1\text{)} \\
&= m \cdot \sum_{j=0}^{n-1} x^j && \text{(substitute } x = \frac{1}{1+p}\text{).} \qquad (11.2)
\end{aligned}$$

The summation in (11.2) is a geometric sum that has a *closed form*, making the evaluation a lot easier, namely[2],

$$\sum_{i=0}^{n-1} x^i = \frac{1-x^n}{1-x}. \qquad (11.3)$$

(The phrase "closed form" refers to a mathematical expression without any summation or product notation.)

[2]To make this equality hold for $x = 0$, we adopt the convention that $0^0 ::= 1$.

Equation (11.3) was proved by induction in Problem **??**, but, as is often the case, the proof by induction gave no hint about how the formula was found in the first place. So we'll take this opportunity to explain where it comes from. The trick is to let S be the value of the sum and then observe what $-xS$ is:

$$\begin{array}{rcllllll} S &=& 1 & +x & +x^2 & +x^3 & + & \cdots & +x^{n-1} \\ -xS &=& & -x & -x^2 & -x^3 & - & \cdots & -x^{n-1} - x^n. \end{array}$$

Adding these two equations gives:

$$S - xS = 1 - x^n,$$

so

$$S = \frac{1 - x^n}{1 - x}.$$

We'll look further into this method of proof in a few weeks when we introduce *generating functions* in Chapter 16.

11.1.2 Closed Form for the Annuity Value

So now we have a simple formula for V, the value of an annuity that pays m dollars at the start of each year for n years.

$$V = m\frac{1-x^n}{1-x} \qquad \text{(by (11.2) and (11.3))} \tag{11.4}$$

$$= m\frac{1+p-(1/(1+p))^{n-1}}{p} \qquad (x = 1/(1+p)). \tag{11.5}$$

The formula (11.5) is much easier to use than a summation with dozens of terms. For example, what is the real value of a winning lottery ticket that pays $\$50,000$ per year for 20 years? Plugging in $m = \$50,000$, $n = 20$, and $p = 0.08$ gives $V \approx \$530,180$. So because payments are deferred, the million dollar lottery is really only worth about a half million dollars! This is a good trick for the lottery advertisers!

11.1.3 Infinite Geometric Series

The question we began with was whether you would prefer a million dollars today or $\$50,000$ a year for the rest of your life. Of course, this depends on how long you live, so optimistically assume that the second option is to receive $\$50,000$ a year *forever*. This sounds like infinite money! But we can compute the value of an annuity with an infinite number of payments by taking the limit of our geometric sum in (11.3) as n tends to infinity.

Theorem 11.1.1. *If $|x| < 1$, then*

$$\sum_{i=0}^{\infty} x^i = \frac{1}{1-x}.$$

Proof.

$$\begin{aligned}
\sum_{i=0}^{\infty} x^i &::= \lim_{n\to\infty} \sum_{i=0}^{n-1} x^i \\
&= \lim_{n\to\infty} \frac{1-x^n}{1-x} && \text{(by (11.3))} \\
&= \frac{1}{1-x}.
\end{aligned}$$

The final line follows from that fact that $\lim_{n\to\infty} x^n = 0$ when $|x| < 1$. ■

In our annuity problem, $x = 1/(1+p) < 1$, so Theorem 11.1.1 applies, and we get

$$\begin{aligned}
V &= m \cdot \sum_{j=0}^{\infty} x^j && \text{(by (11.2))} \\
&= m \cdot \frac{1}{1-x} && \text{(by Theorem 11.1.1)} \\
&= m \cdot \frac{1+p}{p} && (x = 1/(1+p)).
\end{aligned}$$

Plugging in $m = \$50,000$ and $p = 0.08$, the value, V, is only $\$675,000$. Amazingly, a million dollars today is worth much more than $\$50,000$ paid every year forever! Then again, if we had a million dollars today in the bank earning 8% interest, we could take out and spend $\$80,000$ a year forever. So on second thought, this answer really isn't so amazing.

EDITING NOTE:

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 equation (11.3) (for finite sums) or Theorem 11.1.1 (for infinite series).

$$1 + 1/2 + 1/4 + 1/8 + \cdots = \sum_{i=0}^{\infty} (1/2)^i = \frac{1}{1-(1/2)} = 2 \tag{11.6}$$

$$0.999999999\ldots = 0.9 \sum_{i=0}^{\infty} (1/10)^i = 0.9 \frac{1}{1-1/10} = 0.9 \frac{10}{9} = 1 \tag{11.7}$$

$$1 - 1/2 + 1/4 - 1/8 + \cdots = \sum_{i=0}^{\infty} (-1/2)^i = \frac{1}{1-(-1/2)} = 2/3 \tag{11.8}$$

$$1 + 2 + 4 + 8 + \cdots + 2^{n-1} = \sum_{i=0}^{n-1} 2^i = \frac{1-2^n}{1-2} = 2^n - 1 \tag{11.9}$$

$$1 + 3 + 9 + 27 + \cdots + 3^{n-1} = \sum_{i=0}^{n-1} 3^i = \frac{1-3^n}{1-3} = \frac{3^n-1}{2} \tag{11.10}$$

If the terms in a geometric sum or series grow smaller, as in equation (11.6), then the sum is said to be *geometrically decreasing*. If the terms in a geometric sum grow progressively larger, as in (11.9) and (11.10), 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 (11.6) and (11.8), the largest term is equal to 1 and the sums are 2 and 2/3, both relatively close to 1. In equation (11.9), the sum is about twice the largest term. In the final equation (11.10), 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.

Related Sums

We now know all about geometric sums. But in practice one often encounters sums that cannot be transformed by simple variable substitutions to the form $\sum x^i$.

A non-obvious, but useful way to obtain new summation formulas from old is by differentiating or integrating with respect to x. As an example, consider the following sum:

$$\sum_{i=1}^{n} ix^i = x + 2x^2 + 3x^3 + \cdots + nx^n$$

This is not a geometric sum, since the ratio between successive terms is not constant. Our formula for the sum of a geometric sum cannot be directly applied. But suppose that we differentiate that formula:

$$\begin{aligned}
\frac{d}{dx}\sum_{i=0}^{n} x^i &= \frac{d}{dx}\,\frac{1-x^{n+1}}{1-x} \\
\sum_{i=1}^{n} ix^{i-1} &= \frac{-(n+1)x^n(1-x) - (-1)(1-x^{n+1})}{(1-x)^2} \\
&= \frac{-(n+1)x^n + (n+1)x^{n+1} + 1 - x^{n+1}}{(1-x)^2} \\
&= \frac{1-(n+1)x^n + nx^{n+1}}{(1-x)^2}.
\end{aligned}$$

Often differentiating or integrating messes up the exponent of x in every term. In this case, we now have a formula for a sum of the form $\sum ix^{i-1}$, but we want a formula for the series $\sum ix^i$. The solution is simple: multiply by x. This gives:

$$\sum_{i=1}^{n} ix^i = \frac{x-(n+1)x^{n+1}+nx^{n+2}}{(1-x)^2}$$

Since we could easily have made a mistake, it is a good idea to go back and validate a formula obtained this way with a proof by induction.

Notice that if $|x| < 1$, then this series converges to a finite value even if there are infinitely many terms. Taking the limit as n tends infinity gives the following theorem:

Theorem 11.1.2. *If* $|x| < 1$, *then*

$$\sum_{i=1}^{\infty} ix^i = \frac{x}{(1-x)^2}.$$

As a consequence, suppose there is an annuity that pays im dollars at the *end* of each year i forever. For example, if $m = \$50,000$, then the payouts are $\$50,000$ and then $\$100,000$ and then $\$150,000$ and so on. It is hard to believe that the value of this annuity is finite! But we can use the preceding theorem to compute the value:

$$\begin{aligned} V &= \sum_{i=1}^{\infty} \frac{im}{(1+p)^i} \\ &= m \cdot \frac{\frac{1}{1+p}}{(1-\frac{1}{1+p})^2} \\ &= m \cdot \frac{1+p}{p^2}. \end{aligned}$$

The second line follows by an application of Theorem 11.1.2. The third line is obtained by multiplying the numerator and denominator by $(1+p)^2$.

For example, if $m = \$50,000$, and $p = 0.08$ as usual, then the value of the annuity is $V = \$8,437,500$. Even though payments increase every year, the increase is only additive with time; by contrast, dollars paid out in the future decrease in value exponentially with time. The geometric decrease swamps out the additive increase. Payments in the distant future are almost worthless, so the value of the annuity is finite.

The important thing to remember is the trick of taking the derivative (or integral) of a summation formula. Of course, this technique requires one to compute nasty derivatives correctly, but this is at least theoretically possible!

■

11.1.4 Problems

Class Problems

Homework Problems

11.2 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!

11.2.1 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 11.1.

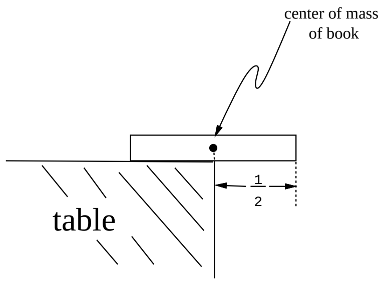


Figure 11.1: One book can overhang half a book length.

Now suppose we have a stack of books that will stick out past the table edge without tipping over—call that a *stable* stack. Let's define the *overhang* of a stable stack to be the largest horizontal distance from the center of mass of the stack to the furthest edge of a book. If we place the center of mass of the stable stack at the edge of the table as in Figure 11.2, that's how far we can get a book in the stack to stick out past the edge.

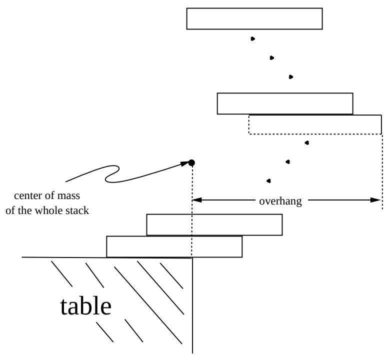


Figure 11.2: Overhanging the edge of the table.

So we want a formula for the maximum possible overhang, B_n, achievable with a stack of n books.

We've already observed that the overhang of one book is 1/2 a book length. That is,

$$B_1 = \frac{1}{2}.$$

Now suppose we have a stable stack of $n + 1$ books with maximum overhang. If the overhang of the n books on top of the bottom book was not maximum, we could get a book to stick out further by replacing the top stack with a stack of n books with larger overhang. So the maximum overhang, B_{n+1}, of a stack of $n + 1$ books is obtained by placing a maximum overhang stable stack of n books on top of the bottom book. And we get the biggest overhang for the stack of $n+1$ books by placing the center of mass of the n books right over the edge of the bottom book as in Figure 11.3.

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,

$$B_{n+1} = B_n + \frac{1}{2(n+1)}, \tag{11.11}$$

as shown in Figure 11.3.

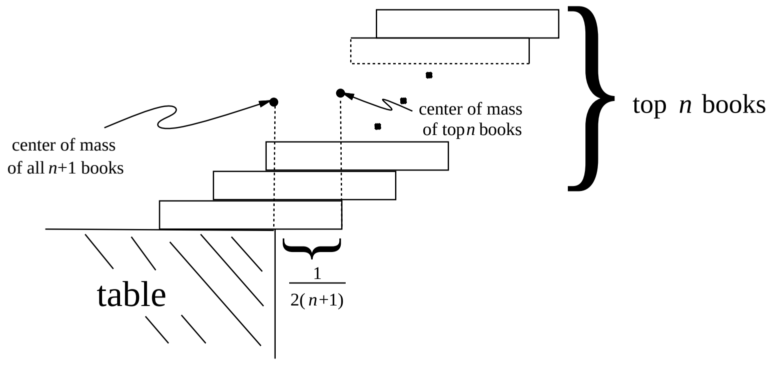

top n books
center of mass of top n books
center of mass of all n+1 books
table
$\frac{1}{2(n+1)}$

Figure 11.3: Additional overhang with $n+1$ books.

Expanding equation (11.11), we have

$$\begin{aligned} B_{n+1} &= B_{n-1} + \frac{1}{2n} + \frac{1}{2(n+1)} \\ &= B_1 + \frac{1}{2\cdot 2} + \cdots + \frac{1}{2n} + \frac{1}{2(n+1)} \\ &= \frac{1}{2}\sum_{i=1}^{n+1}\frac{1}{i}. \end{aligned} \tag{11.12}$$

The nth *Harmonic number*, H_n, is defined to be

Definition 11.2.1.

$$H_n ::= \sum_{i=1}^{n}\frac{1}{i}.$$

So (11.12) means that

$$B_n = \frac{H_n}{2}.$$

The first few Harmonic numbers are easy to compute. For example, $H_4 = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} = \frac{25}{12}$. The fact that H_4 is greater than 2 has special significance; it implies that the total extension of a 4-book stack is greater than one full book! This is the situation shown in Figure 11.4.

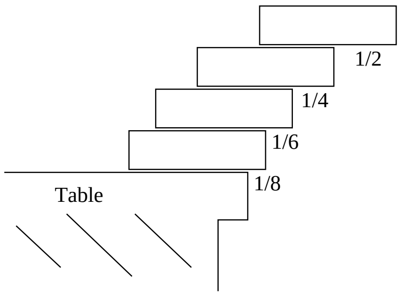


Figure 11.4: Stack of four books with maximum overhang.

11.2.2 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 11.5. The integrals of these functions then bound the value of the sum above and below.

The Integral Method gives the following upper and lower bounds on the harmonic number H_n:

$$H_n \leq 1 + \int_1^n \frac{1}{x}\,dx = 1 + \ln n \tag{11.13}$$

$$H_n \geq \int_0^n \frac{1}{x+1}\,dx = \int_1^{n+1} \frac{1}{x}\,dx = \ln(n+1). \tag{11.14}$$

These bounds imply that the harmonic number H_n is around $\ln n$.

But $\ln n$ grows —slowly —but without bound. That means we can get books to overhang *any distance* past the edge of the table by piling them high enough! For example, to build a stack extending three book lengths beyond the table, we need

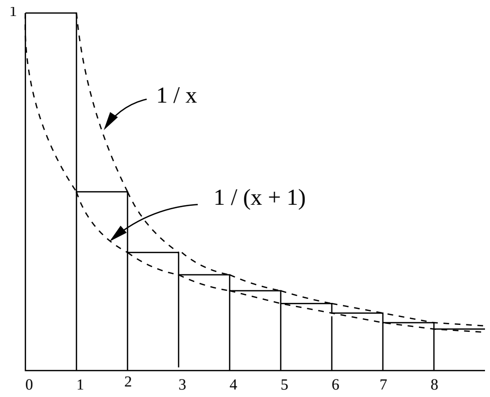


Figure 11.5: *This figure illustrates the Integral Method for bounding a sum. The area under the "stairstep" curve over the interval* $[0, n]$ *is equal to* $H_n = \sum_{i=1}^{n} 1/i$. *The function* $1/x$ *is everywhere greater than or equal to the stairstep and so the integral of* $1/x$ *over this 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.*

a number of books n so that $H_n \geq 6$. By inequality (11.14), this means we want

$$H_n \geq \ln(n+1) \geq 6,$$

so $n \geq e^6 - 1$ books will work, that is, 403 books will be enough to get a three book overhang. Actual calculation of H_6 shows that 227 books is the smallest number that will work.

11.2.3 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 11.2.2. For functions $f, g : \mathbb{R} \to \mathbb{R}$, we say f is *asymptotically equal* to g, in symbols,

$$f(x) \sim g(x)$$

iff

$$\lim_{x \to \infty} f(x)/g(x) = 1.$$

It's tempting to might write $H_n \sim \ln n + \gamma$ to indicate the two leading terms, but it is not really right. According to Definition 11.2.2, $H_n \sim \ln n + c$ where c is *any constant*. The correct way to indicate that γ is the second-largest term is $H_n - \ln n \sim \gamma$.

The reason that the $\sim$ notation is useful is that often we do not care about lower order terms. For example, if $n = 100$, then we can compute $H(n)$ to great precision using only the two leading terms:

$$|H_n - \ln n - \gamma| \leq \left| \frac{1}{200} - \frac{1}{120000} + \frac{1}{120 \cdot 100^4} \right| < \frac{1}{200}.$$

11.2.4 Problems

Class Problems

Homework Problems

11.3 Finding Summation Formulas

The Integral Method offers a way to derive formulas like

EDITING NOTE: equation (11.1) ■

for the sum of consecutive integers,

$$\sum_{i=1}^{n} i = n(n+1)/2,$$

or for the sum of squares,

$$\sum_{i=1}^{n} i^2 = \frac{(2n+1)(n+1)n}{6}$$
$$= \frac{n^3}{3} + \frac{n^2}{2} + \frac{n}{6}. \tag{11.15}$$

These equations appeared in Chapter 3.1 as equations (3.1) and (**??**) where they were proved using the Well-ordering Principle. But those proofs did not explain how someone figured out in the first place that these were the formulas to prove.

Here's how the Integral Method leads to the sum-of-squares formula, for example. First, get a quick estimate of the sum:

$$\int_0^n x^2 \, dx \leq \sum_{i=1}^{n} i^2 \leq \int_0^n (x+1)^2 \, dx,$$

so

$$n^3/3 \leq \sum_{i=1}^{n} i^2 \leq (n+1)^3/3 - 1/3. \tag{11.16}$$

and the upper and lower bounds (11.16) imply that

$$\sum_{i=1}^{n} i^2 \sim n^3/3.$$

To get an exact formula, we then guess the general form of the solution. Where we are uncertain, we can add parameters $a, b, c, \ldots$. For example, we might make the guess:

$$\sum_{i=1}^{n} i^2 = an^3 + bn^2 + cn + d.$$

If the guess is correct, then we can determine the parameters a, b, c, and d by plugging in a few values for n. Each such value gives a linear equation in a, b, c, and d. If we plug in enough values, we may get a linear system with a unique solution. Applying this method to our example gives:

$$\begin{array}{lll} n = 0 & \text{implies} & 0 = d \\ n = 1 & \text{implies} & 1 = a + b + c + d \\ n = 2 & \text{implies} & 5 = 8a + 4b + 2c + d \\ n = 3 & \text{implies} & 14 = 27a + 9b + 3c + d. \end{array}$$

Solving this system gives the solution $a = 1/3$, $b = 1/2$, $c = 1/6$, $d = 0$. Therefore, *if* our initial guess at the form of the solution was correct, then the summation is equal to $n^3/3 + n^2/2 + n/6$, which matches equation (11.15).

The point is that if the desired formula turns out to be a polynomial, then once you get an estimate of the *degree* of the polynomial —by the Integral Method or any other way —all the coefficients of the polynomial can be found automatically.

Be careful! This method let's you discover formulas, but it doesn't guarantee they are right! After obtaining a formula by this method, it's important to go back and *prove* it using induction or some other method, because if the initial guess at the solution was not of the right form, then the resulting formula will be completely wrong!

11.3.1 Double Sums

Sometimes we have to evaluate sums of sums, otherwise known as *double summations*. This can be easy: evaluate the inner sum, replace it with a closed form, and then evaluate the outer sum which no longer has a summation inside it. For example,

$$\begin{aligned}
&\sum_{n=0}^{\infty}\left(y^n\sum_{i=0}^{n}x^i\right)\\
&=\sum_{n=0}^{\infty}\left(y^n\frac{1-x^{n+1}}{1-x}\right) && \text{(geometric sum formula (11.3))}\\
&=\frac{\sum_{n=0}^{\infty}y^n}{1-x}-\frac{\sum_{n=0}^{\infty}y^n x^{n+1}}{1-x}\\
&=\frac{1}{(1-y)(1-x)}-\frac{x\sum_{n=0}^{\infty}(xy)^n}{1-x} && \text{(infinite geometric sum, Theorem 11.1.1)}\\
&=\frac{1}{(1-y)(1-x)}-\frac{x}{(1-xy)(1-x)} && \text{(infinite geometric sum, Theorem 11.1.1)}\\
&=\frac{(1-xy)-x(1-y)}{(1-xy)(1-y)(1-x)}\\
&=\frac{1-x}{(1-xy)(1-y)(1-x)}\\
&=\frac{1}{(1-xy)(1-y)}.
\end{aligned}$$

When there's no obvious closed form for the inner sum, a special trick that is often useful is to try *exchanging the order of summation.* For example, suppose we want to compute the sum of the harmonic numbers

$$\sum_{k=1}^{n}H_k=\sum_{k=1}^{n}\sum_{j=1}^{k}1/j$$

For intuition about this sum, we can try the integral method:

$$\sum_{k=1}^{n} H_k \approx \int_1^n \ln x \, dx \approx n \ln n - n.$$

Now let's look for an exact answer. If we think about the pairs (k, j) over which we are summing, they form a triangle:

		j						
		1	2	3	4	5	$\ldots$	n
k	1	1						
	2	1	$1/2$					
	3	1	$1/2$	$1/3$				
	4	1	$1/2$	$1/3$	$1/4$			
		$\ldots$						
	n	1	$1/2$		$\ldots$			$1/n$

The summation above is summing each row and then adding the row sums. Instead, we can sum the columns and then add the column sums. Inspecting the table we see that this double sum can be written as

$$\begin{aligned}
\sum_{k=1}^{n} H_k &= \sum_{k=1}^{n}\sum_{j=1}^{k} 1/j \\
&= \sum_{j=1}^{n}\sum_{k=j}^{n} 1/j \\
&= \sum_{j=1}^{n} 1/j \sum_{k=j}^{n} 1 \\
&= \sum_{j=1}^{n} \frac{1}{j}(n-j+1) \\
&= \sum_{j=1}^{n} \frac{n-j+1}{j} \\
&= \sum_{j=1}^{n} \frac{n+1}{j} - \sum_{j=1}^{n} \frac{j}{j} \\
&= (n+1)\sum_{j=1}^{n} \frac{1}{j} - \sum_{j=1}^{n} 1 \\
&= (n+1)H_n - n. \qquad (11.17)
\end{aligned}$$

11.4 Stirling's Approximation

The familiar factorial notation, $n!$, is an abbreviation for the product

$$\prod_{i=1}^{n} i.$$

This is by far the most common product in discrete mathematics. In this section we describe a good closed-form estimate of $n!$ called *Stirling's Approximation*. Unfortunately, all we can do is estimate: there is no closed form for $n!$ —though proving so would take us beyond the scope of this text.

11.4.1 Products to Sums

A good way to handle a product is often to convert it into a sum by taking the logarithm. In the case of factorial, this gives

$$\begin{aligned}
\ln(n!) &= \ln(1 \cdot 2 \cdot 3 \cdots (n-1) \cdot n) \\
&= \ln 1 + \ln 2 + \ln 3 + \cdots + \ln(n-1) + \ln n \\
&= \sum_{i=1}^{n} \ln i.
\end{aligned}$$

We've not seen a summation containing a logarithm before! Fortunately, one tool that we used in evaluating sums is still applicable: the Integral Method. We can bound the terms of this sum with $\ln x$ and $\ln(x+1)$ as shown in Figure 11.6. This gives bounds on $\ln(n!)$ as follows:

$$\begin{aligned}
\int_1^n \ln x\,dx \le\ & \textstyle\sum_{i=1}^n \ln i & \le \int_0^n \ln(x+1)\,dx \\
n\ln(\frac{n}{e})+1 \le\ & \textstyle\sum_{i=1}^n \ln i & \le (n+1)\ln\left(\frac{n+1}{e}\right)+1 \\
\left(\frac{n}{e}\right)^n e \le\ & n! & \le \left(\frac{n+1}{e}\right)^{n+1} e.
\end{aligned}$$

The second line follows from the first by completing the integrations. The third line is obtained by exponentiating.

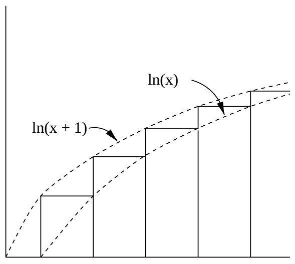


Figure 11.6: *This figure illustrates the Integral Method for bounding the sum $\sum_{i=1}^n$ ln i.*

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)**.**

$$n! \sim \left(\frac{n}{e}\right)^n \sqrt{2\pi n}, \tag{11.18}$$

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)**.**

$$\sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{1/(12n+1)} \le n! \le \sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{1/12n}.$$

Stirling's Approximation implies the asymptotic formula (11.18), since $e^{1/(12n+1)}$ and $e^{1/12n}$ both approach 1 as n grows large. These inequalities can be verified by induction, but the details are nasty.

The bounds in Stirling's formula are very tight. For example, if $n = 100$, then Stirling's bounds are:

$$100! \ge \sqrt{200\pi}\left(\frac{100}{e}\right)^{100} e^{1/1201}$$

$$100! \le \sqrt{200\pi}\left(\frac{100}{e}\right)^{100} e^{1/1200}$$

The only difference between the upper bound and the lower bound is in the final term. In particular $e^{1/1201} \approx 1.00083299$ and $e^{1/1200} \approx 1.00083368$. As a result, the upper bound is no more than $1 + 10^{-6}$ times the lower bound. This is amazingly tight! Remember Stirling's formula; we will use it often.

EDITING NOTE:

Bounds by Double Summing

Another way to derive Stirling's approximation is to remember that $\ln n$ is roughly the same as H_n. This lets us use the result we derived before for $\sum H_k$ via double summation. Our approximation for H_k told us that $\ln(k+1) \leq H_k \leq 1 + \ln k$. Rewriting, we find that $H_k - 1 \leq \ln k \leq H_{k-1}$. It follows that (leaving out the $i = 1$

term in the sum, which contributes 0),

$$\begin{aligned}
\sum_{i=2}^{n} \ln i &\le \sum_{i=2}^{n} H_{i-1} \\
&= \sum_{i=1}^{n-1} H_i \\
&= nH_{n-1} - (n-1) && \text{by (11.17)} \\
&\le n(1 + \ln(n-1)) - (n-1) && \text{by (11.13)} \\
&= n\ln(n-1) + 1,
\end{aligned}$$

roughly the same bound as we proved before via the integral method. We can derive a similar lower bound.

■

11.5 Asymptotic Notation

Asymptotic notation is a shorthand used to give a quick measure of the behavior of a function $f(n)$ as n grows large.

11.5.1 Little Oh

The asymptotic notation, $\sim$, of Definition 11.2.2 is a binary relation indicating that two functions grow at the *same* rate. There is also a binary relation indicating that one function grows at a significantly *slower* rate than another. Namely,

Definition 11.5.1. For functions $f, g : \mathbb{R} \to \mathbb{R}$, with g nonnegative, we say f is *asymptotically smaller* than g, in symbols,

$$f(x) = o(g(x)),$$

iff

$$\lim_{x\to\infty} f(x)/g(x) = 0.$$

For example, $1000x^{1.9} = o(x^2)$, because $1000x^{1.9}/x^2 = 1000/x^{0.1}$ and since $x^{0.1}$ goes to infinity with x and 1000 is constant, we have $\lim_{x\to\infty} 1000x^{1.9}/x^2 = 0$. This argument generalizes directly to yield

Lemma 11.5.2. *$x^a = o(x^b)$ for all nonnegative constants $a < b$.*

Using the familiar fact that $\log x < x$ for all $x > 1$, we can prove

Lemma 11.5.3. *$\log x = o(x^{\epsilon})$ for all $\epsilon > 0$ and $x > 1$.*

Proof. Choose $\epsilon > \delta > 0$ and let $x = z^{\delta}$ in the inequality $\log x < x$. This implies

$$\log z < z^{\delta}/\delta = o(z^{\epsilon}) \qquad \text{by Lemma 11.5.2.} \tag{11.19}$$

■

Corollary 11.5.4. *$x^b = o(a^x)$ for any $a, b \in \mathbb{R}$ with $a > 1$.*

Proof. From (11.19),

$$\log z < z^{\delta}/\delta$$

for all $z > 1$, $\delta > 0$. Hence

$$(e^b)^{\log z} < (e^b)^{z^{\delta}/\delta}$$

$$z^b < \left(e^{\log a (b/\log a)}\right)^{z^{\delta}/\delta}$$

$$= a^{(b/\delta \log a) z^{\delta}}$$

$$< a^z$$

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 11.5.3 and Corollary 11.5.4 can also be proved easily in several other ways, for example, using L'Hopital's Rule or the McLaurin Series for $\log x$ and e^x. Proofs can be found in most calculus texts.

11.5.2 Big Oh

Big Oh is the most frequently used asymptotic notation. It is used to give an upper bound on the growth of a function, such as the running time of an algorithm.

Definition 11.5.5. Given nonnegative functions $f, g : \mathbb{R} \to \mathbb{R}$, we say that

$$f = O(g)$$

iff

$$\limsup_{x \to \infty} f(x)/g(x) < \infty.$$

This definition[3] makes it clear that

[3] We can't simply use the limit as $x \to \infty$ in the definition of $O()$, because if $f(x)/g(x)$ oscil-

Lemma 11.5.6. *If $f = o(g)$ or $f \sim g$, then $f = O(g)$.*

Proof. $\lim f/g = 0$ or $\lim f/g = 1$ implies $\lim f/g < \infty$. ■

It is easy to see that the converse of Lemma 11.5.6 is not true. For example, $2x = O(x)$, but $2x \not\sim x$ and $2x \neq o(x)$.

The usual formulation of Big Oh spells out the definition of $\limsup$ without mentioning it. Namely, here is an equivalent definition:

Definition 11.5.7. Given functions $f, g : \mathbb{R} \to \mathbb{R}$, we say that

$$f = O(g)$$

iff there exists a constant $c \geq 0$ and an x_0 such that for all $x \geq x_0$, $|f(x)| \leq cg(x)$.

lates between, say, 3 and 5 as x grows, then $f = O(g)$ because $f \leq 5g$, but $\lim_{x\to\infty} f(x)/g(x)$ does not exist. So instead of limit, we use the technical notion of $\limsup$. In this oscillating case, $\limsup_{x\to\infty} f(x)/g(x) = 5$.

The precise definition of $\limsup$ is

$$\limsup_{x\to\infty} h(x) ::= \lim_{x\to\infty} \text{lub}_{y \geq x} h(y),$$

where "lub" abbreviates "least upper bound."

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 11.5.8. *If $f = o(g)$, then it is* not *true that $g = O(f)$.*

Proof.

$$\lim_{x\to\infty} \frac{g(x)}{f(x)} = \frac{1}{\lim_{x\to\infty} f(x)/g(x)} = \frac{1}{0} = \infty,$$

so $g \neq O(f)$.

■

Proposition 11.5.9. $100x^2 = O(x^2)$.

Proof. Choose $c = 100$ and $x_0 = 1$. Then the proposition holds, since for all $x \geq 1$, $\left|100x^2\right| \leq 100x^2$. ■

Proposition 11.5.10. $x^2 + 100x + 10 = O(x^2)$.

Proof. $(x^2 + 100x + 10)/x^2 = 1 + 100/x + 10/x^2$ and so its limit as x approaches infinity is $1+0+0 = 1$. So in fact, $x^2+100x+10 \sim x^2$, and therefore $x^2+100x+10 = O(x^2)$. Indeed, it's conversely true that $x^2 = O(x^2+100x+10)$. ■

Proposition 11.5.10 generalizes to an arbitrary polynomial:

Proposition 11.5.11. *For $a_k \neq 0$, $a_k x^k + a_{k-1}x^{k-1} + \cdots + a_1 x + a_0 = O(x^k)$.*

We'll omit the routine proof.

Big Oh notation is especially useful when describing the running time of an algorithm. For example, the usual algorithm for multiplying $n \times n$ matrices requires proportional to n^3 operations in the worst case. This fact can be expressed concisely by saying that the running time is $O(n^3)$. So this asymptotic notation allows the speed of the algorithm to be discussed without reference to constant factors or lower-order terms that might be machine specific. In this case there is another, ingenious matrix multiplication procedure that requires $O(n^{2.55})$ operations. This procedure will therefore be much more efficient on large enough matrices. Unfortunately, the $O(n^{2.55})$-operation multiplication procedure is almost never used because it happens to be less efficient than the usual $O(n^3)$ procedure on matrices of practical size.

EDITING NOTE: It is even conceivable that there is an $O(n^2)$ matrix multiplication procedure, but none is known. ■

11.5.3 Theta

Definition 11.5.12.

$$f = \Theta(g) \quad \text{iff} \quad f = O(g) \text{ and } g = O(f).$$

The statement $f = \Theta(g)$ can be paraphrased intuitively as "f and g are equal to within a constant factor."

The value of these notations is that they highlight growth rates and allow suppression of distracting factors and low-order terms. For example, if the running time of an algorithm is

$$T(n) = 10n^3 - 20n^2 + 1,$$

then

$$T(n) = \Theta(n^3).$$

In this case, we would say that T *is of order* n^3 or that $T(n)$ *grows cubically*.

Another such example is

$$\pi^2 3^{x-7} + \frac{(2.7x^{113} + x^9 - 86)^4}{\sqrt{x}} - 1.08^{3x} = \Theta(3^x).$$

Just knowing that the running time of an algorithm is $\Theta(n^3)$, for example, is useful, because if n doubles we can predict that the running time will *by and large*[4] increase by a factor of at most 8 for large n. In this way, Theta notation preserves information about the scalability of an algorithm or system. Scalability is, of course, a big issue in the design of algorithms and systems.

EDITING NOTE:

Figure 11.7 illustrates the relationships among the asymptotic growth notations we have considered.

■

[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 $T(n) \sim n^3$.

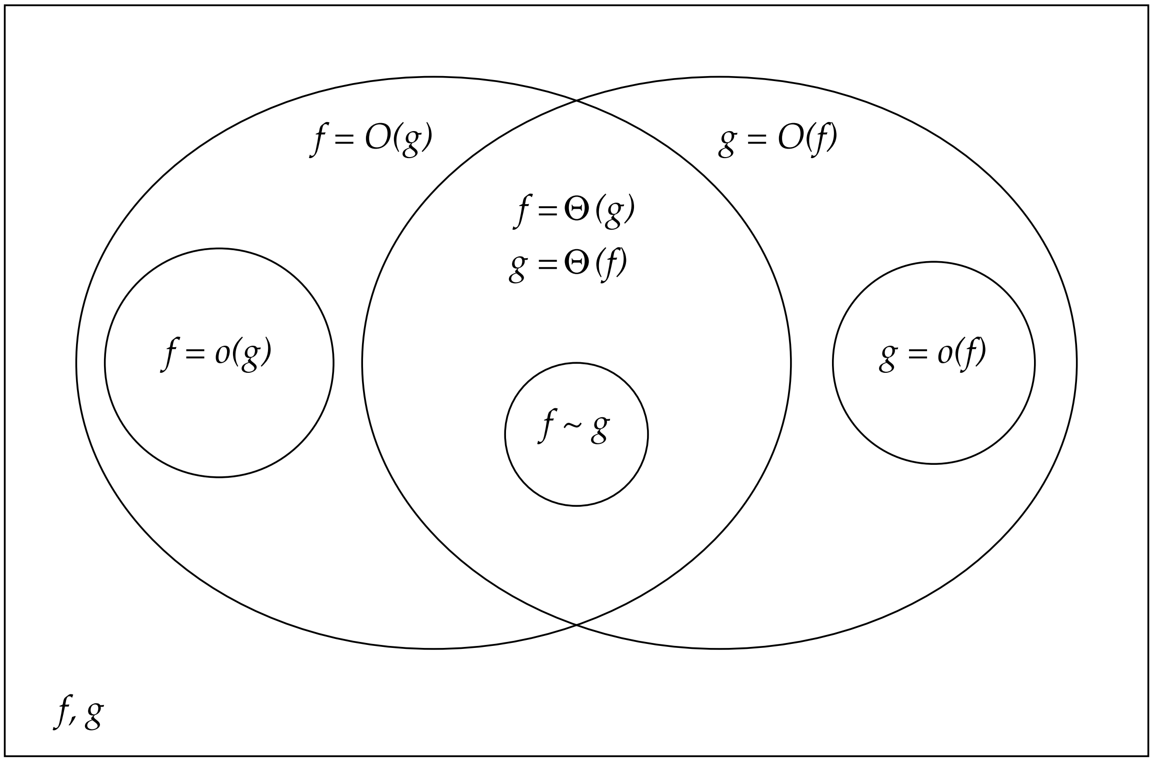


Figure 11.7: Venn Diagram describing Asymptotic Relations

11.5.4 Pitfalls with Big Oh

There is a long list of ways to make mistakes with Big Oh notation. This section presents some of the ways that Big Oh notation can lead to ruin and despair.

The Exponential Fiasco

Sometimes relationships involving Big Oh are not so obvious. For example, one might guess that $4^x = O(2^x)$ since 4 is only a constant factor larger than 2. This reasoning is incorrect, however; actually 4^x grows much faster than 2^x.

Proposition 11.5.13. $4^x \neq O(2^x)$

Proof. $2^x/4^x = 2^x/(2^x 2^x) = 1/2^x$. Hence, $\lim_{x\to\infty} 2^x/4^x = 0$, so in fact $2^x = o(4^x)$. We observed earlier that this implies that $4^x \neq O(2^x)$. ■

Constant Confusion

Every constant is $O(1)$. For example, $17 = O(1)$. This is true because if we let $f(x) = 17$ and $g(x) = 1$, then there exists a $c > 0$ and an x_0 such that $|f(x)| \leq cg(x)$. In particular, we could choose c = 17 and $x_0 = 1$, since $|17| \leq 17 \cdot 1$ for all $x \geq 1$. We can construct a false theorem that exploits this fact.

False Theorem 11.5.14.

$$\sum_{i=1}^{n} i = O(n)$$

False proof. Define $f(n) = \sum_{i=1}^{n} i = 1 + 2 + 3 + \cdots + n$. Since we have shown that every constant i is $O(1)$, $f(n) = O(1) + O(1) + \cdots + O(1) = O(n)$. ■

Of course in reality $\sum_{i=1}^{n} i = n(n+1)/2 \neq O(n)$.

The error stems from confusion over what is meant in the statement $i = O(1)$. For any *constant* $i \in \mathbb{N}$ it is true that $i = O(1)$. More precisely, if f is any constant function, then $f = O(1)$. But in this False Theorem, i is not constant but ranges over a set of values 0,1,. . . ,n that depends on n.

And anyway, we should not be adding $O(1)$'s as though they were numbers. We never even defined what $O(g)$ means by itself; it should only be used in the context "$f = O(g)$" to describe a relation between functions f and g.

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 "$O(f) = g$."

11.5.5 Problems

Practice Problems

Homework Problems

Class Problems

Chapter 12

Counting

12.1 Why Count?

Are there two different subsets of the ninety 25-digit numbers shown below that have the same sum —for example, maybe the sum of the numbers in the first column is equal to the sum of the numbers in the second column?

0020480135385502964448038	3171004832173501394113017
5763257331083479647409398	8247331000042995311646021
0489445991866915676240992	3208234421597368647019265
5800949123548989122628663	8496243997123475922766310
1082662032430379651370981	3437254656355157864869113
6042900801199280218026001	8518399140676002660747477
1178480894769706178994993	3574883393058653923711365
6116171789137737896701405	8543691283470191452333763
1253127351683239693851327	3644909946040480189969149
6144868973001582369723512	8675309258374137092461352
1301505129234077811069011	3790044132737084094417246
6247314593851169234746152	8694321112363996867296665
1311567111143866433882194	3870332127437971355322815
6814428944266874963488274	8772321203608477245851154
1470029452721203587686214	4080505804577801451363100
6870852945543886849147881	8791422161722582546341091
1578271047286257499433886	4167283461025702348124920
6914955508120950093732397	9062628024592126283973285
1638243921852176243192354	4235996831123777788211249
6949632451365987152423541	9137845566925526349897794
1763580219131985963102365	4670939445749439042111220
7128211143613619828415650	9153762966803189291934419
1826227795601842231029694	4815379351865384279613427
7173920083651862307925394	9270880194077636406984249

7858918664240262356610010	9631217114906129219461111
8149436716871371161932035	3157693105325111284321993
3111474985252793452860017	5439211712248901995423441
7898156786763212963178679	9908189853102753335981319
3145621587936120118438701	5610379826092838192760458
8147591017037573337848616	9913237476341764299813987
3148901255628881103198549	5632317555465228677676044
5692168374637019617423712	8176063831682536571306791

Finding two subsets with the same sum may seem like an silly puzzle, but solving problems like this turns out to be useful, for example in finding good ways to fit packages into shipping containers and in decoding secret messages.

The answer to the question turns out to be "yes." Of course this would be easy to confirm just by showing two subsets with the same sum, but that turns out to be kind of hard to do. So before we put a lot of effort into finding such a pair, it would be nice to be sure there were some. Fortunately, it *is* very easy to see why there is such a pair —or at least it will be easy once we have developed a few simple rules for counting things.

The Contest to Find Two Sets with the Same Sum

One term one of us authors offered a $100 prize to the first student to actually *find* two different subsets of the above ninety 25-digit numbers that have the same sum. We didn't expect to have to pay off this bet, but we underestimated the ingenuity and initiative of the students. One computer science major wrote a program that cleverly searched only among a reasonably small set of "plausible" sets, sorted them by their sums, and actually found a couple with the same sum. He won the prize. A few days later, a math major figured out how to reformulate the sum problem as a "lattice basis reduction" problem; then he found a software package implementing an efficient basis reduction procedure, and using it, he very quickly found lots of pairs of subsets with the same sum. He didn't win the prize, but he got a standing ovation from the class —staff included.

Counting seems easy enough: 1, 2, 3, 4, etc. This direct approach works well for counting simple things —like your toes —and may be the only approach for extremely complicated things with no identifiable structure. However, subtler methods can help you count many things in the vast middle ground, such as:

- The number of different ways to select a dozen doughnuts when there are five varieties available.
- The number of 16-bit numbers with exactly 4 ones.

Counting is useful in computer science for several reasons:

- Determining the time and storage required to solve a computational problem —a central objective in computer science —often comes down to solving a counting problem.
- Counting is the basis of probability theory, which plays a central role in all sciences, including computer science.
- Two remarkable proof techniques, the "pigeonhole principle" and "combi-

natorial proof," rely on counting. These lead to a variety of interesting and useful insights.

We're going to present a lot of rules for counting. These rules are actually theorems, but most of them are pretty obvious anyway, so we're not going to focus on proving them. Our objective is to teach you simple counting as a practical skill, like integration.

12.2 Counting One Thing by Counting Another

How do you count the number of people in a crowded room? You could count heads, since for each person there is exactly one head. Alternatively, you could count ears and divide by two. Of course, you might have to adjust the calculation if someone lost an ear in a pirate raid or someone was born with three ears. The point here is that you can often *count one thing by counting another*, though some fudge factors may be required.

EDITING NOTE: This is a central theme of counting, from the easiest problems to the hardest. ■

In more formal terms, every counting problem comes down to determining the size of some set. The *size* or *cardinality* of a finite set, S, is the number of elements in it and is denoted $|S|$. In these terms, we're claiming that we can often find the size of one set by finding the size of a related set. We've already seen a general statement of this idea in the Mapping Rule of Lemma 5.6.2.

12.2.1 The Bijection Rule

We've already implicitly used the Bijection Rule of Lemma 3 a lot. For example, when we studied Stable Marriage and Bipartite Matching, we assumed the obvious fact that if we can pair up all the girls at a dance with all the boys, then there must be an equal number of each. If we needed to be explicit about using the Bijection Rule, we could say that A was the set of boys, B was the set of girls, and the bijection between them was how they were paired.

The Bijection Rule acts as a magnifier of counting ability; if you figure out the size of one set, then you can immediately determine the sizes of many other sets via bijections. For example, let's return to two sets mentioned earlier:

$A =$ all ways to select a dozen doughnuts when five varieties are available

$B =$ all 16-bit sequences with exactly 4 ones

Let's consider a particular element of set A:

$$\underbrace{0\,0}_{\text{chocolate}} \qquad \underbrace{\quad}_{\text{lemon-filled}} \qquad \underbrace{0\,0\,0\,0\,0\,0}_{\text{sugar}} \quad \underbrace{0\,0}_{\text{glazed}} \quad \underbrace{0\,0}_{\text{plain}}$$

We've depicted each doughnut with a 0 and left a gap between the different varieties. Thus, the selection above contains two chocolate doughnuts, no lemon-filled, six sugar, two glazed, and two plain. Now let's put a 1 into each of the four gaps:

$$\underbrace{0\,0}_{\text{chocolate}} \quad 1 \quad \underbrace{\quad}_{\text{lemon-filled}} \quad 1 \quad \underbrace{0\,0\,0\,0\,0\,0}_{\text{sugar}} \quad 1 \quad \underbrace{0\,0}_{\text{glazed}} \quad 1 \quad \underbrace{0\,0}_{\text{plain}}$$

We've just formed a 16-bit number with exactly 4 ones— an element of B!

This example suggests a bijection from set A to set B: map a dozen doughnuts consisting of:

c chocolate, l lemon-filled, s sugar, g glazed, and p plain to the sequence:

$$\underbrace{0\ldots0}_{c} \quad 1 \quad \underbrace{0\ldots0}_{l} \quad 1 \quad \underbrace{0\ldots0}_{s} \quad 1 \quad \underbrace{0\ldots0}_{g} \quad 1 \quad \underbrace{0\ldots0}_{p}$$

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 routine.

12.2.2 Counting Sequences

The Bijection Rule lets us count one thing by counting another. This suggests a general strategy: get really good at counting just a *few* things and then use bijections to count *everything else*. This is the strategy we'll follow. In particular, we'll get really good at counting *sequences*. When we want to determine the size of some other set T, we'll find a bijection from T to a set of sequences S. Then we'll use our super-ninja sequence-counting skills to determine $|S|$, which immediately gives us $|T|$. We'll need to hone this idea somewhat as we go along, but that's pretty much the plan!

12.2.3 The Sum Rule

Linus allocates his big sister Lucy a quota of 20 crabby days, 40 irritable days, and 60 generally surly days. On how many days can Lucy be out-of-sorts one way or another? Let set C be her crabby days, I be her irritable days, and S be the generally surly. In these terms, the answer to the question is $|C \cup I \cup S|$. Now assuming that she is permitted at most one bad quality each day, the size of this union of sets is given by the *Sum Rule*:

Rule 1 (Sum Rule)**.** *If $A_1, A_2, \ldots, A_n$ are* disjoint *sets, then:*

$$|A_1 \cup A_2 \cup \ldots \cup A_n| = |A_1| + |A_2| + \ldots + |A_n|$$

Thus, according to Linus' budget, Lucy can be out-of-sorts for:

$$\begin{aligned} |C \cup I \cup S| &= |C| + |I| + |S| \\ &= 20 + 40 + 60 \\ &= 120 \text{ days} \end{aligned}$$

Notice that the Sum Rule holds only for a union of *disjoint* sets. Finding the size of a union of intersecting sets is a more complicated problem that we'll take up later.

12.2.4 The Product Rule

The *Product Rule* gives the size of a product of sets. Recall that if $P_1, P_2, \ldots, P_n$ are sets, then

$$P_1 \times P_2 \times \ldots \times P_n$$

is the set of all sequences whose first term is drawn from P_1, second term is drawn from P_2 and so forth.

Rule 2 (Product Rule). *If $P_1, P_2, \ldots P_n$ are sets, then:*

$$|P_1 \times P_2 \times \ldots \times P_n| = |P_1| \cdot |P_2| \cdots |P_n|$$

Unlike the sum rule, the product rule does not require the sets $P_1, \ldots, P_n$ to be disjoint. For example, suppose a *daily diet* consists of a breakfast selected from set B, a lunch from set L, and a dinner from set D:

$$B = \{\text{pancakes, bacon and eggs, bagel, Doritos}\}$$

$$L = \{\text{burger and fries, garden salad, Doritos}\}$$

$$D = \{\text{macaroni, pizza, frozen burrito, pasta, Doritos}\}$$

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:

$$\begin{aligned} |B \times L \times D| &= |B| \cdot |L| \cdot |D| \\ &= 4 \cdot 3 \cdot 5 \\ &= 60 \end{aligned}$$

12.2.5 Putting Rules Together

Few counting problems can be solved with a single rule. More often, a solution is a flurry of sums, products, bijections, and other methods. Let's look at some examples that bring more than one rule into play.

Counting Passwords

The sum and product rules together are useful for solving problems involving passwords, telephone numbers, and license plates. For example, on a certain computer system, a valid password is a sequence of between six and eight symbols. The first symbol must be a letter (which can be lowercase or uppercase), and the remaining symbols must be either letters or digits. How many different passwords are possible?

Let's define two sets, corresponding to valid symbols in the first and subsequent positions in the password.

$$F = \{a, b, \ldots, z, A, B, \ldots, Z\}$$

$$S = \{a, b, \ldots, z, A, B, \ldots, Z, 0, 1, \ldots, 9\}$$

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:

$$\begin{array}{cccc}
\emptyset & \{x_1\} & \{x_2\} & \{x_1, x_2\} \\
\{x_3\} & \{x_1, x_3\} & \{x_2, x_3\} & \{x_1, x_2, x_3\}
\end{array}$$

There is a natural bijection from subsets of X to n-bit sequences. Let $x_1, x_2, \ldots, x_n$ be the elements of X. Then a particular subset of X maps to the sequence $(b_1, \ldots, b_n)$ where $b_i = 1$ if and only if x_i is in that subset. For example, if $n = 10$, then the subset $\{x_2, x_3, x_5, x_7, x_{10}\}$ maps to a 10-bit sequence as follows:

$$\begin{array}{rcccccccccccc} \text{subset:} & \{ & & x_2, & x_3, & & x_5, & & x_7, & & & x_{10} & \} \\ \text{sequence:} & (& 0, & 1, & 1, & 0, & 1, & 0, & 1, & 0, & 0, & 1 &) \end{array}$$

We just used a bijection to transform the original problem into a question about sequences —*exactly according to plan!* Now if we answer the sequence question, then we've solved our original problem as well.

But how many different n-bit sequences are there? For example, there are 8 different 3-bit sequences:

$$\begin{array}{cccc} (0,0,0) & (0,0,1) & (0,1,0) & (0,1,1) \\ (1,0,0) & (1,0,1) & (1,1,0) & (1,1,1) \end{array}$$

Well, we can write the set of all n-bit sequences as a product of sets:

$$\underbrace{\{0,1\} \times \{0,1\} \times \ldots \times \{0,1\}}_{n \text{ terms}} = \{0,1\}^n$$

Then Product Rule gives the answer:

$$|\{0,1\}^n| = |\{0,1\}|^n$$

$$= 2^n$$

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.

12.2.6 Problems

Practice Problems

Class Problems

Homework Problems

12.3 The Pigeonhole Principle

Here is an old puzzle:

> A drawer in a dark room contains red socks, green socks, and blue socks. How many socks must you withdraw to be sure that you have a matching pair?

For example, picking out three socks is not enough; you might end up with one red, one green, and one blue. The solution relies on the *Pigeonhole Principle*, which is a friendly name for the contrapositive of the injective case 2 of the Mapping Rule of Lemma 5.6.2. Let's write it down:

If $|X| > |Y|$, then no total function[1] $f : X \to Y$ is injective.

And now rewrite it again to eliminate the word "injective."

Rule 3 (Pigeonhole Principle)**.** *If $|X| > |Y|$, then for every total function $f : X \to Y$, there exist two different elements of X that are mapped to the same element of Y.*

What this abstract mathematical statement has to do with selecting footwear under poor lighting conditions is maybe not obvious. However, let A be the set of socks you pick out, let B be the set of colors available, and let f map each sock to its color. The Pigeonhole Principle says that if $|A| > |B| = 3$, then at least two elements of A (that is, at least two socks) must be mapped to the same element of B (that is, the same color). For example, one possible mapping of four socks to three colors is shown below.

[1]This Mapping Rule actually applies even if f is a total injective *relation*.

$$\begin{tikzcd}
A & f & B \\
\text{1st sock} \arrow[rr] & & \text{red} \\
\text{2nd sock} \arrow[rr] & & \text{green} \\
\text{3rd sock} \arrow[rr] & & \text{blue} \\
\text{4th sock} \arrow[rruu] & &
\end{tikzcd}$$

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).

12.3.1 Hairs on Heads

There are a number of generalizations of the pigeonhole principle. For example:

Rule 4 (Generalized Pigeonhole Principle)**.** *If $|X| > k \cdot |Y|$, then every total function $f : X \to Y$ maps at least $k + 1$ different elements of X to the same element of Y.*

For example, if you pick two people at random, surely they are extremely unlikely to have *exactly* the same number of hairs on their heads. However, in the remarkable city of Boston, Massachusetts there are actually *three* people who have exactly the same number of hairs! Of course, there are many bald people in Boston, and they all have zero hairs. But we're talking about non-bald people; say a person is non-bald if they have at least ten thousand hairs on their head.

Boston has about 500,000 non-bald people, and the number of hairs on a person's head is at most 200,000. Let A be the set of non-bald people in Boston, let $B = \{10,000, 10,001, \ldots, 200,000\}$, 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!

12.3.2 Subsets with the Same Sum

We asserted that two different subsets of the ninety 25-digit numbers listed on the first page have the same sum. This actually follows from the Pigeonhole Principle. Let A be the collection of all subsets of the 90 numbers in the list. Now the sum of any subset of numbers is at most $90 \cdot 10^{25}$, since there are only 90 numbers and every 25-digit number is less than 10^{25}. So let B be the set of integers $\{0, 1, \ldots, 90 \cdot 10^{25}\}$, and let f map each subset of numbers (in A) to its sum (in B).

We proved that an n-element set has 2^n different subsets. Therefore:

$$\begin{aligned} |A| &= 2^{90} \\ &\geq 1.237 \times 10^{27} \end{aligned}$$

On the other hand:

$$\begin{aligned} |B| &= 90 \cdot 10^{25} + 1 \\ &\leq 0.901 \times 10^{27} \end{aligned}$$

Both quantities are enormous, but $|A|$ is a bit greater than $|B|$. This means that f maps at least two elements of A to the same element of B. In other words, by the Pigeonhole Principle, two different subsets must have the same sum!

Notice that this proof gives no indication *which* two sets of numbers have the same sum. This frustrating variety of argument is called a *nonconstructive proof*.

Sets with Distinct Subset Sums

How can we construct a set of n positive integers such that all its subsets have *distinct* sums? One way is to use powers of two:

$$\{1, 2, 4, 8, 16\}$$

This approach is so natural that one suspects all other such sets must involve larger numbers. (For example, we could safely replace 16 by 17, but not by 15.) Remarkably, there are examples involving *smaller* numbers. Here is one:

$$\{6, 9, 11, 12, 13\}$$

One of the top mathematicans of the Twentieth Century, Paul Erdős, conjectured in 1931 that there are no such sets involving *significantly* smaller numbers. More precisely, he conjectured that the largest number must be $> c2^n$ for some constant $c > 0$. He offered $500 to anyone who could prove or disprove his conjecture, but the problem remains unsolved.

12.3.3 Problems

Class Problems

Homework Problems

12.4 The Generalized Product Rule

We realize everyone has been working pretty hard this term, and we're considering awarding some prizes for *truly exceptional* coursework. Here are some possible categories:

Best Administrative Critique We asserted that the quiz was closed-book. On the cover page, one strong candidate for this award wrote, "There is no book."

Awkward Question Award "Okay, the left sock, right sock, and pants are in an antichain, but how— even with assistance— could I put on all three at once?"

Best Collaboration Statement Inspired by a student who wrote "I worked alone" on Quiz 1.

In how many ways can, say, three different prizes be awarded to n people? This is easy to answer using our strategy of translating the problem about awards into a problem about sequences. Let P be the set of n people taking the course. Then there is a bijection from ways of awarding the three prizes to the set $P^3 ::= P \times P \times P$. In particular, the assignment:

"person x wins prize #1, y wins prize #2, and z wins prize #3"

maps to the sequence (x, y, z). By the Product Rule, we have $\left|P^3\right| = |P|^3 = n^3$, so there are n^3 ways to award the prizes to a class of n people.

But what if the three prizes must be awarded to *different* students? As before, we could map the assignment

"person x wins prize #1, y wins prize #2, and z wins prize #3"

to the triple $(x, y, z) \in P^3$. But this function is *no longer a bijection*. For example, no valid assignment maps to the triple (Dave, Dave, Becky) because Dave is not allowed to receive two awards. However, there *is* a bijection from prize assignments to the set:

$$S = \big\{(x, y, z) \in P^3 \mid x, y, \text{ and } z \text{ are different people}\big\}$$

This reduces the original problem to a problem of counting sequences. Unfortunately, the Product Rule is of no help in counting sequences of this type because the entries depend on one another; in particular, they must all be different. However, a slightly sharper tool does the trick.

Rule 5 (Generalized Product Rule). *Let S be a set of length-k sequences. If there are:*

- *n_1 possible first entries,*
- *n_2 possible second entries for each first entry,*
- *n_3 possible third entries for each combination of first and second entries, etc.*

then:

$$|S| = n_1 \cdot n_2 \cdot n_3 \cdots n_k$$

In the awards example, S consists of sequences (x, y, z). There are n ways to choose x, the recipient of prize #1. For each of these, there are $n - 1$ ways to choose

y, the recipient of prize #2, since everyone except for person x is eligible. For each combination of x and y, there are $n-2$ ways to choose z, the recipient of prize #3, because everyone except for x and y is eligible. Thus, according to the Generalized Product Rule, there are

$$|S| = n \cdot (n-1) \cdot (n-2)$$

ways to award the 3 prizes to different people.

12.4.1 Defective Dollars

A dollar is *defective* if some digit appears more than once in the 8-digit serial number. If you check your wallet, you'll be sad to discover that defective dollars are all-too-common. In fact, how common are *nondefective* dollars? Assuming that the digit portions of serial numbers all occur equally often, we could answer this question by computing:

$$\text{fraction dollars that are nondefective} = \frac{\text{\# of serial \#'s with all digits different}}{\text{total \# of serial \#'s}}$$

Let's first consider the denominator. Here there are no restrictions; there are are 10 possible first digits, 10 possible second digits, 10 third digits, and so on. Thus, the total number of 8-digit serial numbers is 10^8 by the Product Rule.

Next, let's turn to the numerator. Now we're not permitted to use any digit twice. So there are still 10 possible first digits, but only 9 possible second digits, 8 possible third digits, and so forth. Thus, by the Generalized Product Rule, there are

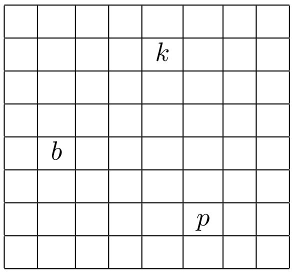
$$\begin{aligned} 10 \cdot 9 \cdot 8 \cdot 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 &= \frac{10!}{2} \\ &= 1,814,400 \end{aligned}$$

serial numbers with all digits different. Plugging these results into the equation above, we find:

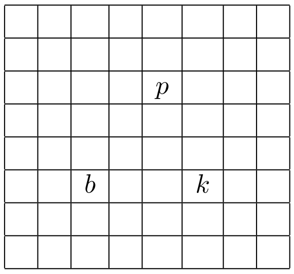
$$\begin{aligned} \text{fraction dollars that are nondefective} &= \frac{1,814,400}{100,000,000} \\ &= 1.8144\% \end{aligned}$$

12.4.2 A Chess Problem

In how many different ways can we place a pawn (p), a knight (k), and a bishop (b) on a chessboard so that no two pieces share a row or a column? A valid configuration is shown below on the left, and an invalid configuration is shown on the right.



valid



invalid

First, we map this problem about chess pieces to a question about sequences. There is a bijection from configurations to sequences

$$(r_p, c_p, r_k, c_k, r_b, c_b)$$

where r_p, r_k, and r_b are distinct rows and c_p, c_k, and c_b are distinct columns. In particular, r_p is the pawn's row, c_p is the pawn's column, r_k is the knight's row, etc. Now we can count the number of such sequences using the Generalized Product Rule:

- r_p is one of 8 rows
- c_p is one of 8 columns
- r_k is one of 7 rows (any one but r_p)
- c_k is one of 7 columns (any one but c_p)
- r_b is one of 6 rows (any one but r_p or r_k)
- c_b is one of 6 columns (any one but c_p or c_k)

Thus, the total number of configurations is $(8 \cdot 7 \cdot 6)^2$.

12.4.3 Permutations

A *permutation* of a set S is a sequence that contains every element of S exactly once.

For example, here are all the permutations of the set $\{a, b, c\}$:

$$\begin{array}{ccc} (a, b, c) & (a, c, b) & (b, a, c) \\ (b, c, a) & (c, a, b) & (c, b, a) \end{array}$$

How many permutations of an n-element set are there? Well, there are n choices for the first element. For each of these, there are $n - 1$ remaining choices for the second element. For every combination of the first two elements, there are $n - 2$ ways to choose the third element, and so forth. Thus, there are a total of

$$n \cdot (n-1) \cdot (n-2) \cdots 3 \cdot 2 \cdot 1 = n!$$

permutations of an n-element set. In particular, this formula says that there are $3! = 6$ permuations of the 3-element set $\{a, b, c\}$, which is the number we found above.

Permutations will come up again in this course approximately 1.6 bazillion times. In fact, permutations are the reason why factorial comes up so often and why we taught you Stirling's approximation:

$$n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

12.5 The Division Rule

Counting ears and dividing by two is a silly way to count the number of people in a room, but this approach is representative of a powerful counting principle.

A *k-to-1 function* maps exactly k elements of the domain to every element of the codomain. For example, the function mapping each ear to its owner is 2-to-1:

$$\begin{tikzcd}
\text{ear 1} \arrow[r] & \text{person A} \\
\text{ear 2} \arrow[dr] & \\
\text{ear 3} \arrow[uur] & \text{person B} \\
\text{ear 4} \arrow[dr] & \\
\text{ear 5} \arrow[r] & \text{person C} \\
\text{ear 6} \arrow[uuur] &
\end{tikzcd}$$

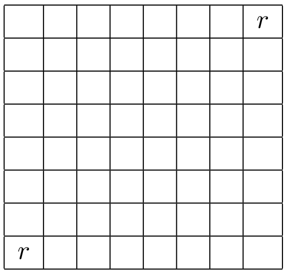
Similarly, the function mapping each finger to its owner is 10-to-1, and the function mapping each finger and toe to its owner is 20-to-1. The general rule is:

Rule 6 (Division Rule)**.** *If $f : A \to B$ is k-to-1, then $|A| = k \cdot |B|$.*

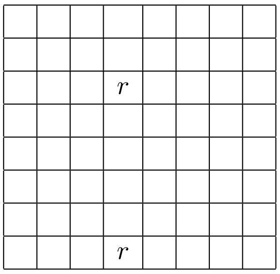
For example, suppose A is the set of ears in the room and B is the set of people. There is a 2-to-1 mapping from ears to people, so by the Division Rule $|A| = 2 \cdot |B|$ or, equivalently, $|B| = |A|/2$, expressing what we knew all along: the number of people is half the number of ears. Unlikely as it may seem, many counting problems are made much easier by initially counting every item multiple times and then correcting the answer using the Division Rule. Let's look at some examples.

12.5.1 Another Chess Problem

In how many different ways can you place two identical rooks on a chessboard so that they do not share a row or column? A valid configuration is shown below on the left, and an invalid configuration is shown on the right.



valid



invalid

Let A be the set of all sequences

$$(r_1, c_1, r_2, c_2)$$

where r_1 and r_2 are distinct rows and c_1 and c_2 are distinct columns. Let B be the set of all valid rook configurations. There is a natural function f from set A to set B; in particular, f maps the sequence (r_1, c_1, r_2, c_2) to a configuration with one rook in row r_1, column c_1 and the other rook in row r_2, column c_2.

But now there's a snag. Consider the sequences:

$$(1, 1, 8, 8) \qquad \text{and} \qquad (8, 8, 1, 1)$$

The first sequence maps to a configuration with a rook in the lower-left corner and a rook in the upper-right corner. The second sequence maps to a configuration with a rook in the upper-right corner and a rook in the lower-left corner. The problem is that those are two different ways of describing the *same* configuration! In fact, this arrangement is shown on the left side in the diagram above.

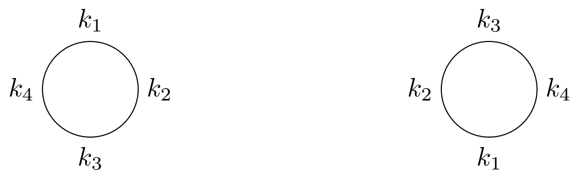
More generally, the function f maps exactly two sequences to *every* board configuration; that is f is a 2-to-1 function. Thus, by the quotient rule, $|A| = 2 \cdot |B|$. Rearranging terms gives:

$$\begin{aligned} |B| &= \frac{|A|}{2} \\ &= \frac{(8 \cdot 7)^2}{2} \end{aligned}$$

On the second line, we've computed the size of A using the General Product Rule just as in the earlier chess problem.

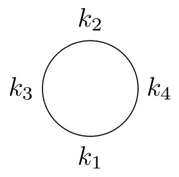
12.5.2 Knights of the Round Table

In how many ways can King Arthur seat n different knights at his round table? Two seatings are considered equivalent if one can be obtained from the other by rotation. For example, the following two arrangements are equivalent:



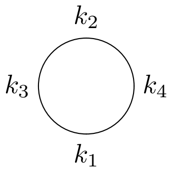
Let A be all the permutations of the knights, and let B be the set of all possible seating arrangements at the round table. We can map each permutation in set A to a circular seating arrangement in set B by seating the first knight in the permutation anywhere, putting the second knight to his left, the third knight to the left of the second, and so forth all the way around the table. For example:

$(k_2, k_4, k_1, k_3) \longrightarrow$



This mapping is actually an n-to-1 function from A to B, since all n cyclic shifts of the original sequence map to the same seating arrangement. In the example, $n = 4$ different sequences map to the same seating arrangement:

$$\begin{array}{c}(k_2, k_4, k_1, k_3)\\(k_4, k_1, k_3, k_2)\\(k_1, k_3, k_2, k_4)\\(k_3, k_2, k_4, k_1)\end{array} \quad \longrightarrow$$



Therefore, by the division rule, the number of circular seating arrangements is:

$$\begin{aligned}|B| &= \frac{|A|}{n}\\ &= \frac{n!}{n}\\ &= (n-1)!\end{aligned}$$

Note that $|A| = n!$ since there are $n!$ permutations of n knights.

12.5.3 Problems

Class Problems

Exam Problems

12.6 Counting Subsets

How many k-element subsets of an n-element set are there? This question arises all the time in various guises:

- In how many ways can I select 5 books from my collection of 100 to bring on vacation?
- How many different 13-card Bridge hands can be dealt from a 52-card deck?
- In how many ways can I select 5 toppings for my pizza if there are 14 available toppings?

This number comes up so often that there is a special notation for it:

$$\binom{n}{k} ::= \text{ the number of } k\text{-element subsets of an } n\text{-element set.}$$

The expression $\binom{n}{k}$ is read "n choose k." Now we can immediately express the answers to all three questions above:

- I can select 5 books from 100 in $\binom{100}{5}$ ways.
- There are $\binom{52}{13}$ different Bridge hands.
- There are $\binom{14}{5}$ different 5-topping pizzas, if 14 toppings are available.

12.6.1 The Subset Rule

We can derive a simple formula for the n-choose-k number using the Division Rule. We do this by mapping any permutation of an n-element set $\{a_1, \ldots, a_n\}$ into a k-element subset simply by taking the first k elements of the permutation. That is, the permutation $a_1 a_2 \ldots a_n$ will map to the set $\{a_1, a_2, \ldots, a_k\}$.

Notice that any other permutation with the same first k elements $a_1, \ldots, a_k$ *in any order* and the same remaining elements $n - k$ elements *in any order* will also map to this set. What's more, a permutation can only map to $\{a_1, a_2, \ldots, a_k\}$ if its first k elements are the elements $a_1, \ldots, a_k$ in some order. Since there are

$k!$ possible permutations of the first k elements and $(n-k)!$ permutations of the remaining elements, we conclude from the Product Rule that exactly $k!(n-k)!$ permutations of the n-element set map to the the particular subset, S. In other words, the mapping from permutations to k-element subsets is $k!(n-k)!$-to-1.

But we know there are $n!$ permutations of an n-element set, so by the Division Rule, we conclude that

$$n! = k!(n-k)!\binom{n}{k}$$

which proves:

Rule 7 (Subset Rule)**.** *The number,*

$$\binom{n}{k},$$

of k-element subsets of an n-element set is

$$\frac{n!}{k!\,(n-k)!}.$$

Notice that this works even for 0-element subsets: $n!/0!n! = 1$. Here we use the fact that $0!$ is a *product* of 0 terms, which by convention equals 1. (A *sum* of zero terms equals 0.)

12.6.2 Bit Sequences

How many n-bit sequences contain exactly k ones? We've already seen the straightforward bijection between subsets of an n-element set and n-bit sequences. For example, here is a 3-element subset of $\{x_1, x_2, \ldots, x_8\}$ and the associated 8-bit sequence:

$$\begin{array}{cccccccccc} \{ & x_1, & & & x_4, & x_5 & & & & \} \\ (& 1, & 0, & 0, & 1, & 1, & 0, & 0, & 0 &) \end{array}$$

Notice that this sequence has exactly 3 ones, each corresponding to an element of the 3-element subset. More generally, the n-bit sequences corresponding to a k-element subset will have exactly k ones. So by the Bijection Rule,

> The number of n-bit sequences with exactly k ones is $\binom{n}{k}$.

12.7 Sequences with Repetitions

12.7.1 Sequences of Subsets

Choosing a k-element subset of an n-element set is the same as splitting the set into a pair of subsets: the first subset of size k and the second subset consisting of the remaining $n - k$ elements. So the Subset Rule can be understood as a rule for counting the number of such splits into pairs of subsets.

We can generalize this to splits into more than two subsets. Namely, let A be an n-element set and $k_1, k_2, \ldots, k_m$ be nonnegative integers whose sum is n. A $(k_1, k_2, \ldots, k_m)$*-split of* A is a sequence

$$(A_1, A_2, \ldots, A_m)$$

where the A_i are pairwise disjoint[2] subsets of A and $|A_i| = k_i$ for $i = 1, \ldots, m$.

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.

[2]That is $A_i \cap A_j = \emptyset$ whenever $i \neq j$. Another way to say this is that no element appears in more than one of the A_i's.

Rule 8 (Subset Split Rule)**.** *The number of* $(k_1, k_2, \dots, k_m)$*-splits of an* n*-element set is*

$$\binom{n}{k_1, \dots, k_m} ::= \frac{n!}{k_1!\, k_2!\, \cdots k_m!}$$

The proof of this Rule is essentially the same as for the Subset Rule. Namely, we map any permutation $a_1 a_2 \dots a_n$ of an n-element set, A, into a $(k_1, k_2, \dots, k_m)$-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 $(k_1, k_2, \dots, k_m)$-splits of A, and the Subset Split Rule now follows from the Division Rule.

12.7.2 The Bookkeeper Rule

We can also generalize our count of n-bit sequences with k-ones to counting length n sequences of letters over an alphabet with more than two letters. For example, how many sequences can be formed by permuting the letters in the 10-letter word BOOKKEEPER?

Notice that there are 1 B, 2 O's, 2 K's, 3 E's, 1 P, and 1 R in BOOKKEEPER. This leads to a straightforward bijection between permutations of BOOKKEEPER and (1,2,2,3,1,1)-splits of $\{1, \ldots, n\}$. Namely, map a permutation to the sequence of sets of positions where each of the different letters occur.

For example, in the permutation BOOKKEEPER itself, the B is in the 1st position, the O's occur in the 2nd and 3rd positions, K's in 4th and 5th, the E's in the 6th, 7th and 9th, P in the 8th, and R is in the 10th position, so BOOKKEEPER maps to

$$(\{1\}, \{2,3\}, \{4,5\}, \{6,7,9\}, \{8\}, \{10\}).$$

From this bijection and the Subset Split Rule, we conclude that the number of ways to rearrange the letters in the word BOOKKEEPER is:

$$\frac{\overbrace{10!}^{\text{total letters}}}{\underbrace{1!}_{\text{B's}}\;\underbrace{2!}_{\text{O's}}\;\underbrace{2!}_{\text{K's}}\;\underbrace{3!}_{\text{E's}}\;\underbrace{1!}_{\text{P's}}\;\underbrace{1!}_{\text{R's}}}$$

This example generalizes directly to an exceptionally useful counting principle which we will call the

Rule 9 (Bookkeeper Rule). *Let $l_1, \ldots, l_m$ be distinct elements. The number of sequences* *with k_1 occurrences of l_1, and k_2 occurrences of $l_2, \ldots$, 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:

$$\frac{20!}{5!^4}$$

12.7.3 A Word about Words

Someday you might refer to the Subset Split Rule or the Bookkeeper Rule in front of a roomful of colleagues and discover that they're all staring back at you blankly. This is not because they're dumb, but rather because we made up the name "Bookkeeper Rule". However, the rule is excellent and the name is apt, so we suggest that you play through: “You know? The Bookkeeper Rule? Don’t you guys know *anything???*”

The Bookkeeper Rule is sometimes called the “formula for permutations with indistinguishable objects.” The size k subsets of an n-element set are sometimes called *k-combinations*. Other similar-sounding descriptions are “combinations with repetition, permutations with repetition, r-permutations, permutations with indistinguishable objects,” and so on. However, the counting rules we’ve taught you are sufficient to solve all these sorts of problems without knowing this jargon, so we won’t burden you with it.

12.7.4 Problems

Class Problems

12.8 Magic Trick

There is a Magician and an Assistant. The Assistant goes into the audience with a deck of 52 cards while the Magician looks away.[3]

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 As-

[3] There are 52 cards in a standard deck. Each card has a *suit* and a *rank*. There are four suits:

$$\spadesuit(\text{ spades}) \qquad \heartsuit(\text{ hearts}) \qquad \clubsuit(\text{ clubs}) \qquad \diamondsuit(\text{ diamonds})$$

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.

sistant 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.

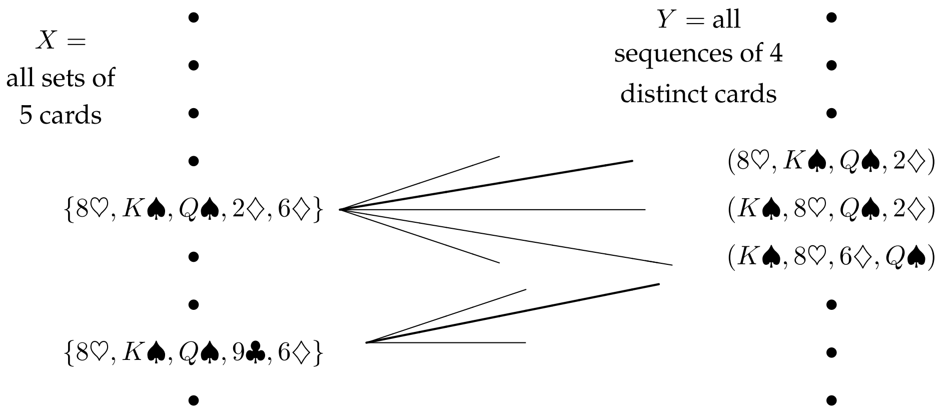
Of course, without cheating, there is still an obvious way the Assistant can communicate to the Magician: he can choose any of the $4! = 24$ permutations of the 4 cards as the order in which to hold up the cards. However, this alone won't quite work: there are 48 cards remaining in the deck, so the Assistant doesn't have enough choices of orders to indicate exactly what the secret card is (though he could narrow it down to two cards).

12.8.1 The Secret

The method the Assistant can use to communicate the fifth card exactly is a nice application of what we know about counting and matching.

The Assistant really has another legitimate way to communicate: he can choose *which of the five cards to keep hidden*. Of course, it's not clear how the Magician could determine which of these five possibilities the Assistant selected by looking at the four visible cards, but there is a way, as we'll now explain.

The problem facing the Magician and Assistant is actually a bipartite matching problem. Put all the *sets* of 5 cards in a collection, X, on the left. And put all the sequences of 4 distinct cards in a collection, Y, on the right. These are the two sets of vertices in the bipartite graph. There is an edge between a set of 5 cards and a sequence of 4 if every card in the sequence is also in the set. In other words, if the audience selects a set of cards, then the Assistant must reveal a sequence of cards that is adjacent in the bipartite graph. Some edges are shown in the diagram below.


$X =$ all sets of 5 cards
$\{8\heartsuit, K\spadesuit, Q\spadesuit, 2\diamondsuit, 6\diamondsuit\}$
$\{8\heartsuit, K\spadesuit, Q\spadesuit, 9\clubsuit, 6\diamondsuit\}$
$Y =$ all sequences of 4 distinct cards
$(8\heartsuit, K\spadesuit, Q\spadesuit, 2\diamondsuit)$
$(K\spadesuit, 8\heartsuit, Q\spadesuit, 2\diamondsuit)$
$(K\spadesuit, 8\heartsuit, 6\diamondsuit, Q\spadesuit)$

For example,

$$\{8\heartsuit, K\spadesuit, Q\spadesuit, 2\diamondsuit, 6\diamondsuit\} \tag{12.1}$$

is an element of X on the left. If the audience selects this set of 5 cards, then there are many different 4-card sequences on the right in set Y that the Assistant could choose to reveal, including $(8\heartsuit, K\spadesuit, Q\spadesuit, 2\diamondsuit)$, $(K\spadesuit, 8\heartsuit, Q\spadesuit, 2\diamondsuit)$, and $(K\spadesuit, 8\heartsuit, 6\diamondsuit, Q\spadesuit)$.

What the Magician and his Assistant need to perform the trick is a *matching* for the X vertices. If they agree in advance on some matching, then when the audience selects a set of 5 cards, the Assistant reveals the matching sequence of 4 cards. The Magician uses the reverse of the matching to find the audience's chosen set of 5 cards, and so he can name the one not already revealed.

For example, suppose the Assistant and Magician agree on a matching containing the two bold edges in the diagram above. If the audience selects the set

$$\{8\heartsuit, K\spadesuit, Q\spadesuit, 9\clubsuit, 6\diamondsuit\}, \tag{12.2}$$

then the Assistant reveals the corresponding sequence

$$(K\spadesuit, 8\heartsuit, 6\diamondsuit, Q\spadesuit). \tag{12.3}$$

Using the matching, the Magician sees that the hand (12.2) is matched to the sequence (12.3), so he can name the one card in the corresponding set not already revealed, namely, the $9\clubsuit$. Notice that the fact that the sets are *matched*, that is, that different sets are paired with *distinct* sequences, is essential. For example, if the audience picked the previous hand (12.1), it would be possible for the Assistant to reveal the same sequence (12.3), but he better not do that: if he did, then the Magician would have no way to tell if the remaining card was the $9\clubsuit$ or the $2\diamondsuit$.

So how can we be sure the needed matching can be found? The reason is that each vertex on the left has degree $5 \cdot 4! = 120$, since there are five ways to select the card kept secret and there are $4!$ permutations of the remaining 4 cards. In addition, each vertex on the right has degree 48, since there are 48 possibilities for the fifth card. So this graph is *degree-constrained* according to Definition 7.7.5, and therefore satisfies Hall's matching condition.

In fact, this reasoning show that the Magician could still pull off the trick if 120 cards were left instead of 48, that is, the trick would work with a deck as large as 124 different cards —without any magic!

12.8.2 The Real Secret

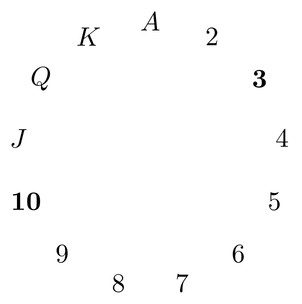
But wait a minute! It's all very well in principle to have the Magician and his Assistant agree on a matching, but how are they supposed to remember a matching with $\binom{52}{5} = 2,598,960$ edges? For the trick to work in practice, there has to be a way to match hands and card sequences mentally and on the fly.

We'll describe one approach. As a running example, suppose that the audience selects:

$$10\heartsuit \quad 9\diamondsuit \quad 3\heartsuit \quad Q\spadesuit \quad J\diamondsuit$$

- The Assistant picks out two cards of the same suit. In the example, the assistant might choose the $3\heartsuit$ and $10\heartsuit$.

- The Assistant locates the ranks of these two cards on the cycle shown below:



For any two distinct ranks on this cycle, one is always between 1 and 6 hops clockwise from the other. For example, the $3\heartsuit$ is 6 hops clockwise from the $10\heartsuit$.

- The more counterclockwise of these two cards is revealed first, and the other becomes the secret card. Thus, in our example, the $10\heartsuit$ would be revealed, and the $3\heartsuit$ would be the secret card. Therefore:

  – The suit of the secret card is the same as the suit of the first card revealed.

  - The rank of the secret card is between 1 and 6 hops clockwise from the rank of the first card revealed.

- All that remains is to communicate a number between 1 and 6. The Magician and Assistant agree beforehand on an ordering of all the cards in the deck from smallest to largest such as:

$$A\clubsuit\ A\diamondsuit\ A\heartsuit\ A\spadesuit\ 2\clubsuit\ 2\diamondsuit\ 2\heartsuit\ 2\spadesuit\ \ldots\ K\heartsuit\ K\spadesuit$$

  The order in which the last three cards are revealed communicates the number according to the following scheme:

$$\begin{array}{ccccc} (& \text{small}, & \text{medium}, & \text{large} &) = 1 \\ (& \text{small}, & \text{large}, & \text{medium} &) = 2 \\ (& \text{medium}, & \text{small}, & \text{large} &) = 3 \\ (& \text{medium}, & \text{large}, & \text{small} &) = 4 \\ (& \text{large}, & \text{small}, & \text{medium} &) = 5 \\ (& \text{large}, & \text{medium}, & \text{small} &) = 6 \end{array}$$

  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).

12.8.3 Same Trick with Four Cards?

Suppose that the audience selects only *four* cards and the Assistant reveals a sequence of *three* to the Magician. Can the Magician determine the fourth card?

Let X be all the sets of four cards that the audience might select, and let Y be all the sequences of three cards that the Assistant might reveal. Now, on one hand, we have

$$|X| = \binom{52}{4} = 270,725$$

by the Subset Rule. On the other hand, we have

$$|Y| = 52 \cdot 51 \cdot 50 = 132,600$$

by the Generalized Product Rule. Thus, by the Pigeonhole Principle, the Assistant must reveal the *same* sequence of three cards for at least

$$\left\lceil \frac{270,725}{132,600} \right\rceil = 3$$

different four-card hands. This is bad news for the Magician: if he sees that sequence of three, then there are at least three possibilities for the fourth card which he cannot distinguish. So there is no legitimate way for the Assistant to communicate exactly what the fourth card is!

12.8.4 Problems

Class Problems

Homework Problems

12.9 Counting Practice: Poker Hands

Five-Card Draw is a card game in which each player is initially dealt a *hand*, a subset of 5 cards. (Then the game gets complicated, but let's not worry about that.) The number of different hands in Five-Card Draw is the number of 5-element subsets of a 52-element set, which is 52 choose 5:

$$\text{total \# of hands} = \binom{52}{5} = 2,598,960$$

Let's get some counting practice by working out the number of hands with various special properties.

12.9.1 Hands with a Four-of-a-Kind

A *Four-of-a-Kind* is a set of four cards with the same rank. How many different hands contain a Four-of-a-Kind? Here are a couple examples:

$$\begin{array}{llllll} \{ & 8\spadesuit, & 8\diamondsuit, & Q\heartsuit, & 8\heartsuit, & 8\clubsuit & \} \\ \{ & A\clubsuit, & 2\clubsuit, & 2\heartsuit, & 2\diamondsuit, & 2\spadesuit & \} \end{array}$$

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:

$$\begin{array}{llllllll} (8, Q, \heartsuit) & \leftrightarrow & \{ & 8\spadesuit, & 8\diamondsuit, & 8\heartsuit, & 8\clubsuit, & Q\heartsuit & \} \\ (2, A, \clubsuit) & \leftrightarrow & \{ & 2\clubsuit, & 2\heartsuit, & 2\diamondsuit, & 2\spadesuit, & A\clubsuit & \} \end{array}$$

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!

12.9.2 Hands with a Full House

A *Full House* is a hand with three cards of one rank and two cards of another rank. Here are some examples:

$$\begin{array}{lccccccr} \{ & 2\spadesuit, & 2\clubsuit, & 2\diamondsuit, & J\clubsuit, & J\diamondsuit & \} \\ \{ & 5\diamondsuit, & 5\clubsuit, & 5\heartsuit, & 7\heartsuit, & 7\clubsuit & \} \end{array}$$

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:

$$\begin{array}{ccccccccc}
(2, \{\spadesuit, \clubsuit, \diamondsuit\}, J, \{\clubsuit, \diamondsuit\}) & \leftrightarrow & \{ & 2\spadesuit, & 2\clubsuit, & 2\diamondsuit, & J\clubsuit, & J\diamondsuit & \} \\
(5, \{\diamondsuit, \clubsuit, \heartsuit\}, 7, \{\heartsuit, \clubsuit\}) & \leftrightarrow & \{ & 5\diamondsuit, & 5\clubsuit, & 5\heartsuit, & 7\heartsuit, & 7\clubsuit & \}
\end{array}$$

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.

12.9.3 Hands with Two Pairs

How many hands have *Two Pairs*; that is, two cards of one rank, two cards of another rank, and one card of a third rank? Here are examples:

$$\begin{array}{ccccccc}
\{ & 3\diamondsuit, & 3\spadesuit, & Q\diamondsuit, & Q\heartsuit, & A\clubsuit & \} \\
\{ & 9\heartsuit, & 9\diamondsuit, & 5\heartsuit, & 5\clubsuit, & K\spadesuit & \}
\end{array}$$

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:

$$\begin{array}{ll} (3, \{\diamondsuit, \spadesuit\}, Q, \{\diamondsuit, \heartsuit\}, A, \clubsuit) & \searrow \\ & \quad \{\ 3\diamondsuit,\ 3\spadesuit,\ Q\diamondsuit,\ Q\heartsuit,\ A\clubsuit\ \} \\ (Q, \{\diamondsuit, \heartsuit\}, 3, \{\diamondsuit, \spadesuit\}, A, \clubsuit) & \nearrow \\ (9, \{\heartsuit, \diamondsuit\}, 5, \{\heartsuit, \clubsuit\}, K, \spadesuit) & \searrow \\ & \quad \{\ 9\heartsuit,\ 9\diamondsuit,\ 5\heartsuit,\ 5\clubsuit,\ K\spadesuit\ \} \\ (5, \{\heartsuit, \clubsuit\}, 9, \{\heartsuit, \diamondsuit\}, K, \spadesuit) & \nearrow \end{array}$$

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:

$$\begin{array}{ccl}
(\{3, Q\}, \{\diamondsuit, \spadesuit\}, \{\diamondsuit, \heartsuit\}, A, \clubsuit) & \leftrightarrow & \{\ 3\diamondsuit,\ 3\spadesuit,\ Q\diamondsuit,\ Q\heartsuit,\ A\clubsuit\ \} \\
(\{9, 5\}, \{\heartsuit, \clubsuit\}, \{\heartsuit, \diamondsuit\}, K, \spadesuit) & \leftrightarrow & \{\ 9\heartsuit,\ 9\diamondsuit,\ 5\heartsuit,\ 5\clubsuit,\ K\spadesuit\ \}
\end{array}$$

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.

12.9.4 Hands with Every Suit

How many hands contain at least one card from every suit? Here is an example of such a hand:

$$\{ \quad 7\diamondsuit, \quad K\clubsuit, \quad 3\diamondsuit, \quad A\heartsuit, \quad 2\spadesuit \quad \}$$

Each such hand is described by a sequence that specifies:

1. The ranks of the diamond, the club, the heart, and the spade, which can be selected in $13 \cdot 13 \cdot 13 \cdot 13 = 13^4$ ways.

2. The suit of the extra card, which can be selected in 4 ways.

3. The rank of the extra card, which can be selected in 12 ways.

For example, the hand above is described by the sequence:

$$(7, K, A, 2, \diamondsuit, 3) \quad \leftrightarrow \quad \{ \quad 7\diamondsuit, \quad K\clubsuit, \quad A\heartsuit, \quad 2\spadesuit, \quad 3\diamondsuit \quad \}$$

Are there other sequences that correspond to the same hand? There is one more! We could equally well regard either the $3\diamondsuit$ or the $7\diamondsuit$ as the extra card, so this is actually a 2-to-1 mapping. Here are the two sequences corresponding to the example hand:

$$\begin{array}{ll} (7, K, A, 2, \diamondsuit, 3) & \searrow \\ & \qquad \{ \quad 7\diamondsuit, \quad K\clubsuit, \quad A\heartsuit, \quad 2\spadesuit, \quad 3\diamondsuit \quad \} \\ (3, K, A, 2, \diamondsuit, 7) & \nearrow \end{array}$$

Therefore, the number of hands with every suit is:

$$\frac{13^4 \cdot 4 \cdot 12}{2}$$

12.9.5 Problems

Class Problems

Exam Problems

12.10 Inclusion-Exclusion

How big is a union of sets? For example, suppose there are 60 math majors, 200 EECS majors, and 40 physics majors. How many students are there in these three departments? Let M be the set of math majors, E be the set of EECS majors, and P be the set of physics majors. In these terms, we're asking for $|M \cup E \cup P|$.

The Sum Rule says that the size of union of *disjoint* sets is the sum of their sizes:

$$|M \cup E \cup P| = |M| + |E| + |P| \qquad \text{(if } M\text{, } E\text{, and } P \text{ are disjoint)}$$

However, the sets M, E, and P might *not* be disjoint. For example, there might be a student majoring in both math and physics. Such a student would be counted twice on the right side of this equation, once as an element of M and once as an element of P. Worse, there might be a triple-major[4] counted *three* times on the right side!

Our last counting rule determines the size of a union of sets that are not necessarily disjoint. Before we state the rule, let's build some intuition by considering some easier special cases: unions of just two or three sets.

12.10.1 Union of Two Sets

For two sets, S_1 and S_2, the *Inclusion-Exclusion Rule* is that the size of their union is:

$$|S_1 \cup S_2| = |S_1| + |S_2| - |S_1 \cap S_2| \tag{12.4}$$

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

[4]. . . though not at MIT anymore.

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:

$$S_1 \cup S_2 = (S_1 - S_2) \cup (S_2 - S_1) \cup (S_1 \cap S_2). \tag{12.5}$$

Similarly, we can decompose each of S_1 and S_2 into the elements exclusively in each set and the elements in both:

$$S_1 = (S_1 - S_2) \cup (S_1 \cap S_2), \tag{12.6}$$

$$S_2 = (S_2 - S_1) \cup (S_1 \cap S_2). \tag{12.7}$$

Now we have from (12.6) and (12.7)

$$\begin{aligned} |S_1| + |S_2| &= (|S_1 - S_2| + |S_1 \cap S_2|) + (|S_2 - S_1| + |S_1 \cap S_2|) \\ &= |S_1 - S_2| + |S_2 - S_1| + 2\,|S_1 \cap S_2|\,, \end{aligned} \tag{12.8}$$

which shows the double-counting of $S_1 \cap S_2$ in the sum. On the other hand, we have from (12.5)

$$|S_1 \cup S_2| = |S_1 - S_2| + |S_2 - S_1| + |S_1 \cap S_2|\,. \tag{12.9}$$

Subtracting (12.9) from (12.8), we get

$$(|S_1| + |S_2|) - |S_1 \cup S_2| = |S_1 \cap S_2|$$

which proves (12.4).

12.10.2 Union of Three Sets

So how many students are there in the math, EECS, and physics departments? In other words, what is $|M \cup E \cup P|$ if:

$$|M| = 60$$

$$|E| = 200$$

$$|P| = 40$$

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:

$$|P_{42} \cup P_{04} \cup P_{60}| = 9! + 9! + 9! - 8! - 8! - 8! + 7!$$

12.10.3 Union of *n* Sets

The size of a union of n sets is given by the following rule.

Rule 10 (Inclusion-Exclusion)**.**

$$|S_1 \cup S_2 \cup \cdots \cup S_n| =$$

	the sum of the sizes of the individual sets
minus	*the sizes of all two-way intersections*
plus	*the sizes of all three-way intersections*
minus	*the sizes of all four-way intersections*
plus	*the sizes of all five-way intersections, etc.*

The formulas for unions of two and three sets are special cases of this general rule.

This way of expressing Inclusion-Exclusion is easy to understand and nearly as precise as expressing it in mathematical symbols, but we'll need the symbolic version below, so let's work on deciphering it now.

We already have a standard notation for the sum of sizes of the individual sets, namely,

$$\sum_{i=1}^{n} |S_i| .$$

A "two-way intersection" is a set of the form $S_i \cap S_j$ for $i \neq j$. We regard $S_j \cap S_i$ as the same two-way intersection as $S_i \cap S_j$, so we can assume that $i < j$. Now we can express the sum of the sizes of the two-way intersections as

$$\sum_{1 \leq i < j \leq n} |S_i \cap S_j| .$$

Similarly, the sum of the sizes of the three-way intersections is

$$\sum_{1 \leq i < j < k \leq n} |S_i \cap S_j \cap S_k| .$$

These sums have alternating signs in the Inclusion-Exclusion formula, with the sum of the k-way intersections getting the sign $(-1)^{k-1}$. This finally leads to a symbolic version of the rule:

Rule (Inclusion-Exclusion)**.**

$$\begin{aligned}\left|\bigcup_{i=1}^{n} S_i\right| = &\sum_{i=1}^{n} |S_i| \\ &- \sum_{1\le i<j\le n} |S_i \cap S_j| \\ &+ \sum_{1\le i<j<k\le n} |S_i \cap S_j \cap S_k| + \cdots \\ &+ (-1)^{n-1} \left|\bigcap_{i=1}^{n} S_i\right| .\end{aligned}$$

EDITING NOTE:

(covered in Problem **??** PS_inclusion-exclusion_primes)

Counting Primes

How many of the numbers $1, 2, \dots, 100$ are prime?

Counting Primes

How many of the numbers $1, 2, \ldots, 100$ are prime? One way to answer this question is to test each number up to 100 for primality and keep a count. This requires considerable effort. (Is 57 prime? How about 67?)

Another approach is to use the Inclusion-Exclusion Principle. This requires one trick: to determine the number of primes, we will first count the number of *non-primes*. By the Sum Rule, we can then find the number of primes by subtraction from 100. This trick of "counting the complement" is a good one to remember.

Reduction to a Union of Four Sets

The set of non-primes in the range $1, \ldots, 100$ consists of the set, C, of composite numbers in this range: $4, 6, 8, 9, \ldots, 99, 100$ and the number 1, which is neither prime nor composite. The main job is to determine the size of the set C of composite numbers. For this purpose, define A_m to be the set of numbers in the range

$m+1, \ldots, 100$ that are divisible by m:

$$A_m ::= \{x \leq 100 \mid x > m \text{ and } (m \mid x)\}$$

For example, A_2 is all the even numbers from 4 to 100. The following Lemma will now allow us to compute the cardinality of C by using Inclusion-Exclusion for the union of four sets:

Lemma 12.10.1.

$$C = A_2 \cup A_3 \cup A_5 \cup A_7.$$

Proof. We prove the two sets equal by showing that each contains the other.

To show that $A_2 \cup A_3 \cup A_5 \cup A_7 \subseteq C$, let n be an element of $A_2 \cup A_3 \cup A_5 \cup A_7$. Then $n \in A_m$ for $m = 2, 3, 5$ or 7. This implies that n is in the range $1, \ldots, 100$ and is composite because it has m as a factor. That is, $n \in C$.

Conversely, to show that $C \subseteq A_2 \cup A_3 \cup A_5 \cup A_7$, let n be an element of C. Then n is a composite number in the range $1, \ldots, 100$. This means that n has at least two prime factors. Now if both prime factors are > 10, then their product would be a number > 100 which divided n, contradicting the fact that $n < 100$. So n must have a prime factor ≤ 10. But 2, 3, 5, and 7 are the only primes ≤ 10. This means that n is an element of A_2, A_3, A_5, or A_7, and so $n \in A_2 \cup A_3 \cup A_5 \cup A_7$. ■

Computing the Cardinality of the Union

Now it's easy to find the cardinality of each set A_m: every mth integer is divisible by m, so the number of integers in the range $1, \ldots, 100$ that are divisible by m is simply $\lfloor 100/m \rfloor$. So

$$|A_m| = \left\lfloor \frac{100}{m} \right\rfloor - 1,$$

where the -1 arises because we defined A_m to exclude m itself. This formula gives:

$$\begin{aligned}
|A_2| &= \lfloor \tfrac{100}{2} \rfloor - 1 = & 49 \\
|A_3| &= \lfloor \tfrac{100}{3} \rfloor - 1 = & 32 \\
|A_5| &= \lfloor \tfrac{100}{5} \rfloor - 1 = & 19 \\
|A_7| &= \lfloor \tfrac{100}{7} \rfloor - 1 = & 13
\end{aligned}$$

Notice that these sets A_2, A_3, A_5, and A_7 are not disjoint. For example, 6 is in both A_2 and A_3. Since the sets intersect, we must use the Inclusion-Exclusion Principle:

$$\begin{aligned}
|C| &= |A_2 \cup A_3 \cup A_5 \cup A_7| \\
&= |A_2| + |A_3| + |A_5| + |A_7| \\
&\quad - |A_2 \cap A_3| - |A_2 \cap A_5| - |A_2 \cap A_7| - |A_3 \cap A_5| - |A_3 \cap A_7| - |A_5 \cap A_7| \\
&\quad + |A_2 \cap A_3 \cap A_5| + |A_2 \cap A_3 \cap A_7| + |A_2 \cap A_5 \cap A_7| + |A_3 \cap A_5 \cap A_7| \\
&\quad - |A_2 \cap A_3 \cap A_5 \cap A_7|
\end{aligned}$$

There are a lot of terms here! Fortunately, all of them are easy to evaluate. For example, $|A_3 \cap A_7|$ is the number of multiples of $3 \cdot 7 = 21$ in the range 1 to 100, which is $\lfloor 100/21 \rfloor = 4$. Substituting such values for all of the terms above gives:

$$\begin{aligned}|C| =& 49 + 32 + 19 + 13 \\ & - 16 - 10 - 7 - 6 - 4 - 2 \\ & + 3 + 2 + 1 + 0 \\ & - 0 \\ =& 74\end{aligned}$$

This calculation shows that there are 74 composite numbers in the range 1 to 100. Since the number 1 is neither composite nor prime, there are $100 - 74 - 1 = 25$ primes in this range.

At this point it may seem that checking each number from 1 to 100 for primality and keeping a count of primes might have been easier than using Inclusion-Exclusion. However, the Inclusion-Exclusion approach used here is asymptotically faster as the range of numbers grows large.

The naive strategy requires n runs of a primality test if the upper bound is n. The Inclusion-Exclusion approach seems to require summing an immense number of terms, but fewer than n of these are non-zero and the rest can be ignored. ■

12.10.4 Computing Euler's Function

We will now use Inclusion-Exclusion to calculate Euler's function, $\phi(n)$. By definition, $\phi(n)$ is the number of nonnegative integers less than a positive integer n that are relatively prime to n. But the set, S, of nonnegative integers less than n that are *not* relatively prime to n will be easier to count.

Suppose the prime factorization of n is $p_1^{e_1} \cdots p_m^{e_m}$ for distinct primes p_i. This means that the integers in S are precisely the nonnegative integers less than n that are divisible by at least one of the p_i's. So, letting C_i be the set of nonnegative integers less than n that are divisible by p_i, we have

$$S = \bigcup_{i=1}^{m} C_i.$$

We'll be able to find the size of this union using Inclusion-Exclusion because the intersections of the C_i's are easy to count. For example, $C_1 \cap C_2 \cap C_3$ is the set of nonnegative integers less than n that are divisible by each of p_1, p_2 and p_3.

But since the p_i's are distinct primes, being divisible by each of these primes is that same as being divisible by their product. Now observe that if r is a positive divisor of n, then exactly n/r nonnegative integers less than n are divisible by r, namely, $0, r, 2r, \ldots, ((n/r) - 1)r$. So exactly $n/p_1p_2p_3$ nonnegative integers less than n are divisible by all three primes p_1, p_2, p_3. In other words,

$$|C_1 \cap C_2 \cap C_3| = \frac{n}{p_1p_2p_3}.$$

So reasoning this way about all the intersections among the C_i's and applying Inclusion-Exclusion, we get

$$\begin{aligned}
|S| &= \left|\bigcup_{i=1}^{m} C_i\right| \\
&= \sum_{i=1}^{m} |C_i| - \sum_{1 \le i < j \le m} |C_i \cap C_j| + \sum_{1 \le i < j < k \le m} |C_i \cap C_j \cap C_k| - \cdots + (-1)^{m-1} \left|\bigcap_{i=1}^{m} C_i\right| \\
&= \sum_{i=1}^{m} \frac{n}{p_i} - \sum_{1 \le i < j \le m} \frac{n}{p_ip_j} + \sum_{1 \le i < j < k \le m} \frac{n}{p_ip_jp_k} - \cdots + (-1)^{m-1} \frac{n}{p_1p_2 \cdots p_n} \\
&= n \left(\sum_{i=1}^{m} \frac{1}{p_i} - \sum_{1 \le i < j \le m} \frac{1}{p_ip_j} + \sum_{1 \le i < j < k \le m} \frac{1}{p_ip_jp_k} - \cdots + (-1)^{m-1} \frac{1}{p_1p_2 \cdots p_n} \right)
\end{aligned}$$

But $\phi(n) = n - |S|$ by definition, so

$$\begin{aligned}\phi(n) &= n\left(1 - \sum_{i=1}^{m}\frac{1}{p_i} + \sum_{1\le i<j\le m}\frac{1}{p_ip_j} - \sum_{1\le i<j<k\le m}\frac{1}{p_ip_jp_k} + \cdots + (-1)^m\frac{1}{p_1p_2\cdots p_n}\right)\\ &= n\prod_{i=1}^{m}\left(1-\frac{1}{p_i}\right). \qquad (12.10)\end{aligned}$$

Notice that in case $n = p^k$ for some prime, p, then (12.10) simplifies to

$$\phi(p^k) = p^k\left(1-\frac{1}{p}\right) = p^k - p^{k-1}$$

as claimed in chapter 4.

Quick Question: Why does equation (12.10) imply that

$$\phi(ab) = \phi(a)\phi(b)$$

for relatively prime integers $a, b > 1$, as claimed in Theorem 4.7.4.(a)?

12.10.5 Problems

Practice Problems

Class Problems

Homework Problems

12.11 Binomial Theorem

Counting gives insight into one of the basic theorems of algebra. A *binomial* is a sum of two terms, such as $a + b$. Now consider its 4th power, $(a + b)^4$.

If we multiply out this 4th power expression completely, we get

$$\begin{array}{rcccccccccc} (a+b)^4 & = & & aaaa & + & aaab & + & aaba & + & aabb \\ & & + & abaa & + & abab & + & abba & + & abbb \\ & & + & baaa & + & baab & + & baba & + & babb \\ & & + & bbaa & + & bbab & + & bbba & + & bbbb \end{array}$$

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 12.11.1 (Binomial Theorem)**.** *For all $n \in \mathbb{N}$ and $a, b \in \mathbb{R}$:*

$$(a+b)^n = \sum_{k=0}^{n} \binom{n}{k} a^{n-k}b^k$$

The expression $\binom{n}{k}$ is often called a "binomial coefficient" in honor of its appearance here.

This reasoning about binomials extends nicely to *multinomials*, which are sums of two or more terms. For example, suppose we wanted the coefficient of

$$bo^2k^2e^3pr$$

in the expansion of $(b+o+k+e+p+r)^{10}$. Each term in this expansion is a product of 10 variables where each variable is one of b, o, k, e, p, or r. Now, the coefficient of $bo^2k^2e^3pr$ is the number of those terms with exactly 1 b, 2 o's, 2 k's, 3 e's, 1 p, and 1 r. And the number of such terms is precisely the number of rearrangments of the word BOOKKEEPER:

$$\binom{10}{1,2,2,3,1,1} = \frac{10!}{1!\,2!\,2!\,3!\,1!\,1!}.$$

The expression on the left is called a "multinomial coefficient." This reasoning extends to a general theorem.

Definition 12.11.2. For $n, k_1, \dots, k_m \in naturals$, such that $k_1 + k_2 + \cdots + k_m = n$, define the *multinomial coefficient*

$$\binom{n}{k_1, k_2, \dots, k_m} ::= \frac{n!}{k_1!\,k_2!\,\dots k_m!}.$$

Theorem 12.11.3 (Multinomial Theorem)**.** *For all $n \in \mathbb{N}$ and $z_1, \dots z_m \in \mathbb{R}$:*

$$(z_1 + z_2 + \cdots + z_m)^n = \sum_{\substack{k_1,\dots,k_m \in \mathbb{N} \\ k_1+\cdots+k_m=n}} \binom{n}{k_1, k_2, \dots, k_m} z_1^{k_1} z_2^{k_2} \cdots z_m^{k_m}$$

You'll be better off remembering the reasoning behind the Multinomial Theorem rather than this ugly formal statement.

12.11.1 Problems

Practice Problems

Class Problems

Homework Problems

12.12 Combinatorial Proof

Suppose you have n different T-shirts, but only want to keep k. You could equally well select the k shirts you want to keep or select the complementary set of $n-k$ shirts you want to throw out. Thus, the number of ways to select k shirts from among n must be equal to the number of ways to select $n-k$ shirts from among n. Therefore:

$$\binom{n}{k} = \binom{n}{n-k}$$

This is easy to prove algebraically, since both sides are equal to:

$$\frac{n!}{k!\,(n-k)!}$$

But we didn't really have to resort to algebra; we just used counting principles.

Hmm.

12.12.1 Boxing

Jay, famed Math for Computer Science Teaching Assistant, has decided to try out for the US Olympic boxing team. After all, he's watched all of the *Rocky* movies and spent hours in front of a mirror sneering, "Yo, you wanna piece a' *me*?!" Jay figures that n people (including himself) are competing for spots on the team and only k will be selected. As part of maneuvering for a spot on the team, he needs to work out how many different teams are possible. There are two cases to consider:

- Jay *is* selected for the team, and his $k-1$ teammates are selected from among the other $n-1$ competitors. The number of different teams that can be formed in this way is:

$$\binom{n-1}{k-1}$$

- Jay is *not* selected for the team, and all k team members are selected from among the other $n-1$ competitors. The number of teams that can be formed this way is:

$$\binom{n-1}{k}$$

All teams of the first type contain Jay, and no team of the second type does; therefore, the two sets of teams are disjoint. Thus, by the Sum Rule, the total number of possible Olympic boxing teams is:

$$\binom{n-1}{k-1} + \binom{n-1}{k}$$

Jeremy, equally-famed Teaching Assistant, thinks Jay isn't so tough and so he might as well also try out. He reasons that n people (including himself) are trying out for k spots. Thus, the number of ways to select the team is simply:

$$\binom{n}{k}$$

Jeremy and Jay each correctly counted the number of possible boxing teams; thus, their answers must be equal. So we know:

$$\binom{n-1}{k-1} + \binom{n-1}{k} = \binom{n}{k}$$

This is called *Pascal's Identity*. And we proved it *without any algebra!* Instead, we relied purely on counting techniques.

12.12.2 Finding a Combinatorial Proof

A *combinatorial proof* is an argument that establishes an algebraic fact by relying on counting principles. Many such proofs follow the same basic outline:

1. Define a set S.
2. Show that $|S| = n$ by counting one way.
3. Show that $|S| = m$ by counting another way.
4. Conclude that $n = m$.

In the preceding example, S was the set of all possible Olympic boxing teams. Jay computed

$$|S| = \binom{n-1}{k-1} + \binom{n-1}{k}$$

by counting one way, and Jeremy computed

$$|S| = \binom{n}{k}$$

by counting another. Equating these two expressions gave Pascal's Identity.

More typically, the set S is defined in terms of simple sequences or sets rather than an elaborate story. Here is less colorful example of a combinatorial argument.

Theorem 12.12.1.

$$\sum_{r=0}^{n} \binom{n}{r}\binom{2n}{n-r} = \binom{3n}{n}$$

Proof. We give a combinatorial proof. Let S be all n-card hands that can be dealt from a deck containing n red cards (numbered $1, \ldots, n$) and $2n$ black cards (numbered $1, \ldots, 2n$). First, note that every $3n$-element set has

$$|S| = \binom{3n}{n}$$

n-element subsets.

From another perspective, the number of hands with exactly r red cards is

$$\binom{n}{r}\binom{2n}{n-r}$$

since there are $\binom{n}{r}$ ways to choose the r red cards and $\binom{2n}{n-r}$ ways to choose the $n-r$ black cards. Since the number of red cards can be anywhere from 0 to n, the total number of n-card hands is:

$$|S| = \sum_{r=0}^{n} \binom{n}{r} \binom{2n}{n-r}$$

Equating these two expressions for $|S|$ proves the theorem. ■

Combinatorial proofs are almost magical. Theorem 12.12.1 looks pretty scary, but we proved it without any algebraic manipulations at all. The key to constructing a combinatorial proof is choosing the set S properly, which can be tricky. Generally, the simpler side of the equation should provide some guidance. For example, the right side of Theorem 12.12.1 is $\binom{3n}{n}$, which suggests choosing S to be all n-element subsets of some $3n$-element set.

12.12.3 Problems

Class Problems

Homework Problems

Chapter 13

Generating Functions

Generating Functions are one of the most surprising and useful inventions in Discrete Math. Roughly speaking, generating functions transform problems about *sequences* into problems about *functions*. This is great because we've got piles of mathematical machinery for manipulating functions. Thanks to generating functions, we can apply all that machinery to problems about sequences. In this way, we can use generating functions to solve all sorts of counting problems. There is a huge chunk of mathematics concerning generating functions, so we will only get a taste of the subject.

In this chapter, we'll put sequences in angle brackets to more clearly distinguish them from the many other mathematical expressions floating around.

The *ordinary generating function* for $\langle g_0, g_1, g_2, g_3 \ldots \rangle$ is the power series:

$$G(x) = g_0 + g_1 x + g_2 x^2 + g_3 x^3 + \cdots .$$

There are a few other kinds of generating functions in common use, but ordinary generating functions are enough to illustrate the power of the idea, so we'll stick to them. So from now on *generating function* will mean the ordinary kind.

A generating function is a "formal" power series in the sense that we usually regard x as a placeholder rather than a number. Only in rare cases will we actually evaluate a generating function by letting x take a real number value, so we generally ignore the issue of convergence.

Throughout this chapter, we'll indicate the correspondence between a sequence and its generating function with a double-sided arrow as follows:

$$\langle g_0, g_1, g_2, g_3, \ldots \rangle \longleftrightarrow g_0 + g_1 x + g_2 x^2 + g_3 x^3 + \cdots$$

For example, here are some sequences and their generating functions:

$$\langle 0,0,0,0,\dots\rangle \longleftrightarrow 0+0x+0x^2+0x^3+\cdots = 0$$

$$\langle 1,0,0,0,\dots\rangle \longleftrightarrow 1+0x+0x^2+0x^3+\cdots = 1$$

$$\langle 3,2,1,0,\dots\rangle \longleftrightarrow 3+2x+1x^2+0x^3+\cdots = 3+2x+x^2$$

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:

$$\begin{aligned}
\langle 1,1,1,1,\dots\rangle &\longleftrightarrow 1+x+x^2+x^3+\cdots &&= \frac{1}{1-x} \\
\langle 1,-1,1,-1,\dots\rangle &\longleftrightarrow 1-x+x^2-x^3+x^4-\cdots &&= \frac{1}{1+x} \\
\left\langle 1,a,a^2,a^3,\dots\right\rangle &\longleftrightarrow 1+ax+a^2x^2+a^3x^3+\cdots &&= \frac{1}{1-ax} \\
\langle 1,0,1,0,1,0,\dots\rangle &\longleftrightarrow 1+x^2+x^4+x^6+\cdots &&= \frac{1}{1-x^2}
\end{aligned}$$

13.1 Operations on Generating Functions

The magic of generating functions is that we can carry out all sorts of manipulations on sequences by performing mathematical operations on their associated generating functions. Let's experiment with various operations and characterize their effects in terms of sequences.

13.1.1 Scaling

Multiplying a generating function by a constant scales every term in the associated sequence by the same constant. For example, we noted above that:

$$\langle 1,0,1,0,1,0,\dots \rangle \;\longleftrightarrow\; 1+x^2+x^4+x^6+\cdots = \frac{1}{1-x^2}$$

Multiplying the generating function by 2 gives

$$\frac{2}{1-x^2} = 2+2x^2+2x^4+2x^6+\cdots$$

which generates the sequence:

$$\langle 2,0,2,0,2,0,\dots \rangle$$

Rule 11 (Scaling Rule)**.** *If*

$$\langle f_0, f_1, f_2, \ldots \rangle \longleftrightarrow F(x),$$

then

$$\langle cf_0,\ cf_1,\ cf_2,\ \ldots \rangle \longleftrightarrow c \cdot F(x).$$

The idea behind this rule is that:

$$\begin{aligned}\langle cf_0, cf_1, cf_2, \ldots \rangle \quad &\longleftrightarrow \quad cf_0 + cf_1 x + cf_2 x^2 + \cdots \\ &= \quad c \cdot (f_0 + f_1 x + f_2 x^2 + \cdots) \\ &= \quad cF(x)\end{aligned}$$

13.1.2 Addition

Adding generating functions corresponds to adding the two sequences term by term. For example, adding two of our earlier examples gives:

$$\begin{array}{rcccccccccc} & \langle & 1, & 1, & 1, & 1, & 1, & 1, & \ldots & \rangle & \longleftrightarrow & \dfrac{1}{1-x} \\ + & \langle & 1, & -1, & 1, & -1, & 1, & -1, & \ldots & \rangle & \longleftrightarrow & \dfrac{1}{1+x} \\ \hline & \langle & 2, & 0, & 2, & 0, & 2, & 0, & \ldots & \rangle & \longleftrightarrow & \dfrac{1}{1-x} + \dfrac{1}{1+x} \end{array}$$

We've now derived two different expressions that both generate the sequence $\langle 2, 0, 2, 0, \dots \rangle$.

They are, of course, equal:

$$\frac{1}{1-x} + \frac{1}{1+x} = \frac{(1+x)+(1-x)}{(1-x)(1+x)} = \frac{2}{1-x^2}$$

Rule 12 (Addition Rule). *If*

$$\langle f_0, f_1, f_2, \dots \rangle \longleftrightarrow F(x), \qquad \textit{and}$$

$$\langle g_0, g_1, g_2, \dots \rangle \longleftrightarrow G(x),$$

then

$$\langle f_0 + g_0,\ f_1 + g_1,\ f_2 + g_2,\ \dots \rangle \longleftrightarrow F(x) + G(x).$$

The idea behind this rule is that:

$$\begin{aligned}
\langle f_0 + g_0,\ f_1 + g_1,\ f_2 + g_2,\ \dots \rangle \quad &\longleftrightarrow \quad \sum_{n=0}^{\infty} (f_n + g_n) x^n \\
&= \quad \left(\sum_{n=0}^{\infty} f_n x^n \right) + \left(\sum_{n=0}^{\infty} g_n x^n \right) \\
&= \quad F(x) + G(x)
\end{aligned}$$

13.1.3 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 13 (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:

$$\begin{aligned}
\langle \overbrace{0, 0, \ldots, 0}^{k \text{ zeroes}}, f_0, f_1, f_2, \ldots \rangle \quad &\longleftrightarrow \quad f_0 x^k + f_1 x^{k+1} + f_2 x^{k+2} + \cdots \\
&= \quad x^k \cdot (f_0 + f_1 x + f_2 x^2 + f_3 x^3 + \cdots) \\
&= \quad x^k \cdot F(x)
\end{aligned}$$

13.1.4 Differentiation

What happens if we take the *derivative* of a generating function? As an example, let's differentiate the now-familiar generating function for an infinite sequence of 1's.

$$\begin{aligned}
\frac{d}{dx}\left(1 + x + x^2 + x^3 + x^4 + \cdots\right) \quad &= \quad \frac{d}{dx}\left(\frac{1}{1-x}\right) \\
1 + 2x + 3x^2 + 4x^3 + \cdots \quad &= \quad \frac{1}{(1-x)^2} \qquad (13.1) \\
\langle 1, 2, 3, 4, \ldots \rangle \quad &\longleftrightarrow \quad \frac{1}{(1-x)^2}
\end{aligned}$$

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 14 (Derivative Rule). *If*

$$\langle f_0, f_1, f_2, f_3, \dots \rangle \longleftrightarrow F(x),$$

then

$$\langle f_1, 2f_2, 3f_3, \dots \rangle \longleftrightarrow F'(x).$$

The idea behind this rule is that:

$$\begin{aligned}\langle f_1, 2f_2, 3f_3, \dots \rangle \longleftrightarrow\ & f_1 + 2f_2x + 3f_3x^2 + \cdots \\ &= \frac{d}{dx}\left(f_0 + f_1x + f_2x^2 + f_3x^3 + \cdots\right) \\ &= \frac{d}{dx}\,F(x)\end{aligned}$$

The Derivative Rule is very useful. In fact, there is frequent, independent need for each of differentiation's two effects, multiplying terms by their index and left-shifting one place. Typically, we want just one effect and must somehow cancel out the other. For example, let's try to find the generating function for the sequence of squares, $\langle 0, 1, 4, 9, 16, \dots \rangle$. If we could start with the sequence $\langle 1, 1, 1, 1, \dots \rangle$ and multiply each term by its index two times, then we'd have the desired result:

$$\langle 0\cdot 0,\ 1\cdot 1,\ 2\cdot 2,\ 3\cdot 3,\ \dots\rangle = \langle 0,1,4,9,\dots\rangle$$

A challenge is that differentiation not only multiplies each term by its index, but also shifts the whole sequence left one place. However, the Right-Shift Rule 13 tells how to cancel out this unwanted left-shift: multiply the generating function by x.

Our procedure, therefore, is to begin with the generating function for $\langle 1,1,1,1,\dots\rangle$, differentiate, multiply by x, and then differentiate and multiply by x once more.

$$\begin{aligned}
\langle 1,1,1,1,\dots\rangle &\longleftrightarrow \frac{1}{1-x}\\
\langle 1,2,3,4,\dots\rangle &\longleftrightarrow \frac{d}{dx}\frac{1}{1-x} = \frac{1}{(1-x)^2}\\
\langle 0,1,2,3,\dots\rangle &\longleftrightarrow x\cdot\frac{1}{(1-x)^2} = \frac{x}{(1-x)^2}\\
\langle 1,4,9,16,\dots\rangle &\longleftrightarrow \frac{d}{dx}\frac{x}{(1-x)^2} = \frac{1+x}{(1-x)^3}\\
\langle 0,1,4,9,\dots\rangle &\longleftrightarrow x\cdot\frac{1+x}{(1-x)^3} = \frac{x(1+x)}{(1-x)^3}
\end{aligned}$$

Thus, the generating function for squares is:

$$\frac{x(1+x)}{(1-x)^3} \tag{13.2}$$

13.1.5 Products

Rule 15 (Product Rule)**.** *If*

$$\langle a_0, a_1, a_2, \dots \rangle \longleftrightarrow A(x), \quad \textit{and} \quad \langle b_0, b_1, b_2, \dots \rangle \longleftrightarrow B(x),$$

then

$$\langle c_0,\ c_1,\ c_2,\ \dots \rangle \longleftrightarrow A(x) \cdot B(x),$$

where

$$c_n ::= a_0 b_n + a_1 b_{n-1} + a_2 b_{n-2} + \cdots + a_n b_0.$$

To understand this rule, let

$$C(x) ::= A(x) \cdot B(x) = \sum_{n=0}^{\infty} c_n x^n.$$

We can evaluate the product $A(x) \cdot B(x)$ by using a table to identify all the cross-terms from the product of the sums:

$$\begin{array}{c|ccccc}
 & b_0x^0 & b_1x^1 & b_2x^2 & b_3x^3 & \ldots \\
\hline
a_0x^0 & a_0b_0x^0 & a_0b_1x^1 & a_0b_2x^2 & a_0b_3x^3 & \ldots \\
a_1x^1 & a_1b_0x^1 & a_1b_1x^2 & a_1b_2x^3 & \ldots & \\
a_2x^2 & a_2b_0x^2 & a_2b_1x^3 & \ldots & & \\
a_3x^3 & a_3b_0x^3 & \ldots & & & \\
\vdots & \ldots & & & &
\end{array}$$

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,

$$a_0b_n + a_1b_{n-1} + a_2b_{n-2} + \cdots + a_nb_0. \tag{13.3}$$

This expression (13.3) 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 $\langle b_0, b_1, b_2, \ldots \rangle$.

13.2 The Fibonacci Sequence

Sometimes we can find nice generating functions for more complicated sequences. For example, here is a generating function for the Fibonacci numbers:

$$\langle 0, 1, 1, 2, 3, 5, 8, 13, 21, \ldots \rangle \longleftrightarrow \frac{x}{1 - x - x^2}$$

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.

13.2.1 Finding a Generating Function

Let's begin by recalling the definition of the Fibonacci numbers:

$$
\begin{aligned}
&f_0 = 0 \\
&f_1 = 1 \\
&f_n = f_{n-1} + f_{n-2} \qquad \text{(for } n \geq 2\text{)}
\end{aligned}
$$

We can expand the final clause into an infinite sequence of equations. Thus, the Fibonacci numbers are defined by:

$$
\begin{aligned}
f_0 &= 0 \\
f_1 &= 1 \\
f_2 &= f_1 + f_0 \\
f_3 &= f_2 + f_1 \\
f_4 &= f_3 + f_2 \\
&\vdots
\end{aligned}
$$

Now the overall plan is to *define* a function $F(x)$ that generates the sequence on the left side of the equality symbols, which are the Fibonacci numbers. Then we *derive* a function that generates the sequence on the right side. Finally, we equate the two and solve for $F(x)$. Let's try this. First, we define:

$$F(x) = f_0 + f_1 x + f_2 x^2 + f_3 x^3 + f_4 x^4 + \cdots$$

Now we need to derive a generating function for the sequence:

$$\langle 0,\ 1,\ f_1 + f_0,\ f_2 + f_1,\ f_3 + f_2,\ \ldots \rangle$$

One approach is to break this into a sum of three sequences for which we know generating functions and then apply the Addition Rule:

$$\begin{array}{rcccccccl}
 & \langle\ 0, & 1, & 0, & 0, & 0, & \ldots\ \rangle & \longleftrightarrow & x \\
 & \langle\ 0, & f_0, & f_1, & f_2, & f_3, & \ldots\ \rangle & \longleftrightarrow & xF(x) \\
+ & \langle\ 0, & 0, & f_0, & f_1, & f_2, & \ldots\ \rangle & \longleftrightarrow & x^2F(x) \\
\hline
 & \langle\ 0, & 1 + f_0, & f_1 + f_0, & f_2 + f_1, & f_3 + f_2, & \ldots\ \rangle & \longleftrightarrow & x + xF(x) + x^2F(x)
\end{array}$$

This sequence is almost identical to the right sides of the Fibonacci equations. The one blemish is that the second term is $1 + f_0$ instead of simply 1. However, this amounts to nothing, since $f_0 = 0$ anyway.

Now if we equate $F(x)$ with the new function $x + xF(x) + x^2F(x)$, then we're implicitly writing down *all* of the equations that define the Fibonacci numbers in one fell swoop:

$$\begin{array}{ccccccccc}
F(x) & = & f_0 + & f_1 & x + & f_2 & x^2 + & f_3 & x^3 + \cdots \\
\shortparallel & & \shortparallel & \shortparallel & & \shortparallel & & \shortparallel & \\
x + xF(x) + x^2F(x) & = & 0 & + (1 + f_0) & x + & (f_1 + f_0) & x^2 + & (f_2 + f_1) & x^3 + \cdots
\end{array}$$

Solving for $F(x)$ gives the generating function for the Fibonacci sequence:

$$F(x) = x + xF(x) + x^2F(x)$$

so

$$F(x) = \frac{x}{1 - x - x^2}.$$

Sure enough, this is the simple generating function we claimed at the outset.

13.2.2 Finding a Closed Form

Why should one care about the generating function for a sequence? There are several answers, but here is one: if we can find a generating function for a sequence, then we can often find a closed form for the nth coefficient— which can be pretty useful! For example, a closed form for the coefficient of x^n in the power series for $x/(1 - x - x^2)$ would be an explicit formula for the nth Fibonacci number.

So our next task is to extract coefficients from a generating function. There are several approaches. For a generating function that is a ratio of polynomials, we can use the method of *partial fractions*, which you learned in calculus. Just as the terms in a partial fraction expansion are easier to integrate, the coefficients of those terms are easy to compute.

Let's try this approach with the generating function for Fibonacci numbers. First, we factor the denominator:

$$1 - x - x^2 = (1 - \alpha_1 x)(1 - \alpha_2 x)$$

where $\alpha_1 = \frac{1}{2}(1 + \sqrt{5})$ and $\alpha_2 = \frac{1}{2}(1 - \sqrt{5})$. Next, we find A_1 and A_2 which satisfy:

$$\frac{x}{1 - x - x^2} = \frac{A_1}{1 - \alpha_1 x} + \frac{A_2}{1 - \alpha_2 x}$$

We do this by plugging in various values of x to generate linear equations in A_1 and A_2. We can then find A_1 and A_2 by solving a linear system. This gives:

$$A_1 = \frac{1}{\alpha_1 - \alpha_2} = \frac{1}{\sqrt{5}}$$

$$A_2 = \frac{-1}{\alpha_1 - \alpha_2} = -\frac{1}{\sqrt{5}}$$

Substituting into the equation above gives the partial fractions expansion of

$F(x)$:

$$\frac{x}{1-x-x^2} = \frac{1}{\sqrt{5}}\left(\frac{1}{1-\alpha_1 x} - \frac{1}{1-\alpha_2 x}\right)$$

Each term in the partial fractions expansion has a simple power series given by the geometric sum formula:

$$\frac{1}{1-\alpha_1 x} = 1 + \alpha_1 x + \alpha_1^2 x^2 + \cdots$$

$$\frac{1}{1-\alpha_2 x} = 1 + \alpha_2 x + \alpha_2^2 x^2 + \cdots$$

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.

13.2.3 Problems

Class Problems

Homework Problems

Exam Problems

13.3 Counting with Generating Functions

Generating functions are particularly useful for solving counting problems. In particular, problems involving choosing items from a set often lead to nice generating functions by letting the coefficient of x^n be the number of ways to choose n items.

13.3.1 Choosing Distinct Items from a Set

The generating function for binomial coefficients follows directly from the Binomial Theorem:

$$\begin{aligned}\left\langle \binom{k}{0}, \binom{k}{1}, \binom{k}{2}, \dots, \binom{k}{k}, 0, 0, 0, \dots \right\rangle \quad &\longleftrightarrow \quad \binom{k}{0} + \binom{k}{1}x + \binom{k}{2}x^2 + \cdots + \binom{k}{k}x^k \\ &= \quad (1+x)^k\end{aligned}$$

Thus, the coefficient of x^n in $(1+x)^k$ is $\binom{k}{n}$, the number of ways to choose n distinct items from a set of size k. For example, the coefficient of x^2 is $\binom{k}{2}$, the number of ways to choose 2 items from a set with k elements. Similarly, the coefficient of x^{k+1} is the number of ways to choose $k+1$ items from a size k set, which is zero. (Watch out for this reversal of the roles that k and n played in earlier examples; we're led to this reversal because we've been using n to refer to the power of x in a power series.)

13.3.2 Building Generating Functions that Count

Often we can translate the description of a counting problem directly into a generating function for the solution. For example, we could figure out that $(1+x)^k$ generates the number of ways to select n distinct items from a k-element set without resorting to the Binomial Theorem or even fussing with binomial coefficients!

Here is how. First, consider a single-element set $\{a_1\}$. The generating function for the number of ways to select n elements from this set is simply $1+x$: we have 1 way to select zero elements, 1 way to select one element, and 0 ways to select more than one element. Similarly, the number of ways to select n elements from the set $\{a_2\}$ is also given by the generating function $1+x$. The fact that the elements differ in the two cases is irrelevant.

Now here is the main trick: *the generating function for choosing elements from a union of disjoint sets is the product of the generating functions for choosing from each set.* We'll justify this in a moment, but let's first look at an example. According to this principle, the generating function for the number of ways to select n elements from the $\{a_1, a_2\}$ is:

$$\underbrace{(1+x)}_{\substack{\text{gen func for}\\ \text{selecting an } a_1}} \cdot \underbrace{(1+x)}_{\substack{\text{gen func for}\\ \text{selecting an } a_2}} = \underbrace{(1+x)^2}_{\substack{\text{gen func for}\\ \text{selecting from}\\ \{a_1,a_2\}}} = 1 + 2x + x^2$$

Sure enough, for the set $\{a_1, a_2\}$, we have 1 way to select zero elements, 2 ways to select one element, 1 way to select two elements, and 0 ways to select more than two elements.

Repeated application of this rule gives the generating function for selecting n items from a k-element set $\{a_1, a_2, \ldots, a_k\}$:

$$\underbrace{(1+x)}_{\substack{\text{gen func for}\\ \text{selecting an } a_1}} \cdot \underbrace{(1+x)}_{\substack{\text{gen func for}\\ \text{selecting an } a_2}} \cdots \underbrace{(1+x)}_{\substack{\text{gen func for}\\ \text{selecting an } a_k}} = \underbrace{(1+x)^k}_{\substack{\text{gen func for}\\ \text{selecting from}\\ \{a_1,a_2,\ldots,a_k\}}}$$

This is the same generating function that we obtained by using the Binomial Theorem. But this time around we translated directly from the counting problem to the generating function.

We can extend these ideas to a general principle:

Rule 16 (Convolution Rule). *Let $A(x)$ be the generating function for selecting items* *from set $\mathcal{A}$, and let $B(x)$ be the generating function for selecting items from set $\mathcal{B}$. If $\mathcal{A}$ and $\mathcal{B}$ are disjoint, then the generating function for selecting items from the union $\mathcal{A} \cup \mathcal{B}$ is the product $A(x) \cdot B(x)$.*

This rule is rather ambiguous: what exactly are the rules governing the selection of items from a set? Remarkably, the Convolution Rule remains valid under *many* interpretations of selection. For example, we could insist that distinct items be selected or we might allow the same item to be picked a limited number of times or any number of times. Informally, the only restrictions are that (1) the order in which items are selected is disregarded and (2) restrictions on the selection of items from sets $\mathcal{A}$ and $\mathcal{B}$ also apply in selecting items from $\mathcal{A} \cup \mathcal{B}$. (Formally, there must be a bijection between n-element selections from $\mathcal{A} \cup \mathcal{B}$ and ordered pairs of selections from $\mathcal{A}$ and $\mathcal{B}$ containing a total of n elements.)

To count the number of ways to select n items from $\mathcal{A} \cup \mathcal{B}$, we observe that we can select n items by choosing j items from $\mathcal{A}$ and $n - j$ items from $\mathcal{B}$, where j is any number from 0 to n. This can be done in $a_j b_{n-j}$ ways. Summing over all the possible values of j gives a total of

$$a_0b_n + a_1b_{n-1} + a_2b_{n-2} + \cdots + a_nb_0$$

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 $A(x)B(x)$.

13.3.3 Choosing Items with Repetition

The first counting problem we considered was the number of ways to select a dozen doughnuts when five flavors were available. We can generalize this question as follows: in how many ways can we select n items from a k-element set if we're allowed to pick the same item multiple times? In these terms, the doughnut problem asks in how many ways we can select $n = 12$ doughnuts from the set of $k = 5$ flavors

$$\{\text{chocolate}, \text{lemon-filled}, \text{sugar}, \text{glazed}, \text{plain}\}$$

where, of course, we're allowed to pick several doughnuts of the same flavor. Let's approach this question from a generating functions perspective.

Suppose we make n choices (with repetition allowed) of items from a set containing a single item. Then there is one way to choose zero items, one way to choose one item, one way to choose two items, etc. Thus, the generating function for choosing n elements with repetition from a 1-element set is:

$$\begin{aligned}\langle 1, 1, 1, 1, \ldots \rangle \quad &\longleftrightarrow \quad 1 + x + x^2 + x^3 + \cdots \\ &= \quad \frac{1}{1-x}\end{aligned}$$

The Convolution Rule says that the generating function for selecting items from a union of disjoint sets is the product of the generating functions for selecting items from each set:

$$\underbrace{\frac{1}{1-x}}_{\substack{\text{gen func for}\\ \text{choosing } a_1\text{'s}}} \cdot \underbrace{\frac{1}{1-x}}_{\substack{\text{gen func for}\\ \text{choosing } a_2\text{'s}}} \cdots \underbrace{\frac{1}{1-x}}_{\substack{\text{gen func for}\\ \text{choosing } a_k\text{'s}}} = \underbrace{\frac{1}{(1-x)^k}}_{\substack{\text{gen func for}\\ \text{repeated choice from}\\ \{a_1, a_2, \ldots, a_k\}}}$$

Therefore, the generating function for choosing items from a k-element set with repetition allowed is $1/(1-x)^k$.

Now the Bookkeeper Rule tells us that the number of ways to choose n items with repetition from an k element set is

$$\binom{n+k-1}{n},$$

so this is the coefficient of x^n in the series expansion of $1/(1-x)^k$.

On the other hand, it's instructive to derive this coefficient algebraically, which we can do using Taylor's Theorem:

Theorem 13.3.1 (Taylor's Theorem)**.**

$$f(x) = f(0) + f'(0)x + \frac{f''(0)}{2!}x^2 + \frac{f'''(0)}{3!}x^3 + \cdots + \frac{f^{(n)}(0)}{n!}x^n + \cdots .$$

This theorem says that the nth coefficient of $1/(1-x)^k$ is equal to its nth derivative evaluated at 0 and divided by $n!$. Computing the nth derivative turns out not to be very difficult (Problem **??**).

EDITING NOTE:

Let

$$G(x) ::= \frac{1}{(1-x)^k} = (1-x)^{-k}.$$

Then we have:

$$G'(x) = k(1-x)^{-(k+1)}$$

$$G''(x) = k(k+1)(1-x)^{-(k+2)}$$

$$G'''(x) = k(k+1)(k+2)(1-x)^{-(k+3)}$$

$$G^{(n)}(x) = k(k+1)\cdots(k+n-1)(1-x)^{-(k+n)}$$

Thus, the coefficient of x^n in the generating function is:

$$\begin{aligned} G^{(n)}(0)/n! &= \frac{k(k+1)\cdots(k+n-1)}{n!} \\ &= \frac{(k+n-1)!}{(k-1)!\,n!} \\ &= \binom{n+k-1}{n}. \end{aligned}$$

■

13.3.4 Problems

Practice Problems

Class Problems

Homework Problems

Exam Problems

13.3.5 An "Impossible" Counting Problem

EDITING NOTE: Not so impossible. From Rebecca Freund, F09:

Note that the fruits can be divided into two groups, the apples-and-pears and the bananas-and-oranges. Once you know how many are apples-and-pears, there's only one way to distribute them: Use a pear if the number is odd, otherwise don't. Make the rest apples. Similarly, once you've decided on the number of bananas-and-oranges, you have to throw in the-greatest-multiple-of-five-less-than-or-equal-to-that bananas and add oranges as needed. So the number of apples-and-pears exactly determines the arrangement. You can have 0-n apples-and-pears, so there are n+1 possibilities. ■

So far everything we've done with generating functions we could have done another way. But here is an absurd counting problem —really over the top! In how many ways can we fill a bag with n fruits subject to the following constraints?

- The number of apples must be even.
- The number of bananas must be a multiple of 5.
- There can be at most four oranges.
- There can be at most one pear.

For example, there are 7 ways to form a bag with 6 fruits:

Apples	6	4	4	2	2	0	0
Bananas	0	0	0	0	0	5	5
Oranges	0	2	1	4	3	1	0
Pears	0	0	1	0	1	0	1

These constraints are so complicated that the problem seems hopeless! But let's see what generating functions reveal.

Let's first construct a generating function for choosing apples. We can choose a set of 0 apples in one way, a set of 1 apple in zero ways (since the number of apples must be even), a set of 2 apples in one way, a set of 3 apples in zero ways, and so forth. So we have:

$$A(x) = 1 + x^2 + x^4 + x^6 + \cdots = \frac{1}{1 - x^2}$$

Similarly, the generating function for choosing bananas is:

$$B(x) = 1 + x^5 + x^{10} + x^{15} + \cdots = \frac{1}{1 - x^5}$$

Now, we can choose a set of 0 oranges in one way, a set of 1 orange in one way, and so on. However, we can not choose more than four oranges, so we have the generating function:

$$O(x) = 1 + x + x^2 + x^3 + x^4 = \frac{1 - x^5}{1 - x}$$

Here we're using the geometric sum formula. Finally, we can choose only zero or one pear, so we have:

$$P(x) = 1 + x$$

The Convolution Rule says that the generating function for choosing from among all four kinds of fruit is:

$$\begin{aligned} A(x)B(x)O(x)P(x) &= \frac{1}{1-x^2}\frac{1}{1-x^5}\frac{1-x^5}{1-x}(1+x) \\ &= \frac{1}{(1-x)^2} \\ &= 1+2x+3x^2+4x^3+\cdots \end{aligned}$$

Almost everything cancels! We're left with $1/(1-x)^2$, which we found a power series for earlier: the coefficient of x^n is simply $n+1$. Thus, the number of ways to form a bag of n fruits is just $n+1$. This is consistent with the example we worked out, since there were 7 different fruit bags containing 6 fruits. *Amazing!*

13.3.6 Problems

Practice Problems

Homework Problems

Exam Problems

Part IV

Probability

Chapter 14

Introduction to Probability

Probability plays a key role in the sciences —"hard" and social —including computer science. Many algorithms rely on randomization. Investigating their correctness and performance requires probability theory. Moreover, computer systems designs, such as memory management, branch prediction, packet routing, and load balancing are based on probabilistic assumptions and analyses. Probability is central as well in related subjects such as information theory, cryptography, artificial intelligence, and game theory. But we'll start with a more down-to-earth application: getting a prize in a game show.

14.1 Monty Hall

In the September 9, 1990 issue of *Parade* magazine, the columnist Marilyn vos Savant responded to this letter:

> *Suppose you're on a game show, and you're given the choice of three doors. Behind one door is a car, behind the others, goats. You pick a door, say number 1, and the host, who knows what's behind the doors, opens another door, say number 3, which has a goat. He says to you, "Do you want to pick door number 2?" Is it to your advantage to switch your choice of doors?*
>
> Craig. F. Whitaker
>
> Columbia, MD

The letter describes a situation like one faced by contestants on the 1970's game show *Let's Make a Deal*, hosted by Monty Hall and Carol Merrill. Marilyn replied that the contestant should indeed switch. She explained that if the car was behind either of the two unpicked doors —which is twice as likely as the the car being behind the picked door —the contestant wins by switching. But she soon received a torrent of letters, many from mathematicians, telling her that she was wrong. The problem generated thousands of hours of heated debate.

This incident highlights a fact about probability: the subject uncovers lots of examples where ordinary intuition leads to completely wrong conclusions. So until you've studied probabilities enough to have refined your intuition, a way to avoid errors is to fall back on a rigorous, systematic approach such as the Four Step Method.

14.1.1 The Four Step Method

Every probability problem involves some sort of randomized experiment, process, or game. And each such problem involves two distinct challenges:

1. How do we model the situation mathematically?

2. How do we solve the resulting mathematical problem?

In this section, we introduce a four step approach to questions of the form, "What is the probability that —— ?" In this approach, we build a probabilistic model step-by-step, formalizing the original question in terms of that model. Remarkably, the structured thinking that this approach imposes provides simple solutions to many famously-confusing problems. For example, as you'll see, the four step method cuts through the confusion surrounding the Monty Hall problem like a Ginsu knife. However, more complex probability questions may spin off challenging counting, summing, and approximation problems— which, fortunately, you've already spent weeks learning how to solve.

14.1.2 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?"

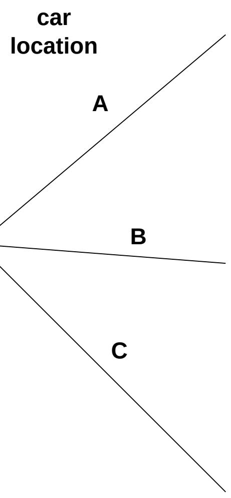
14.1.3 Step 1: Find the Sample Space

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.

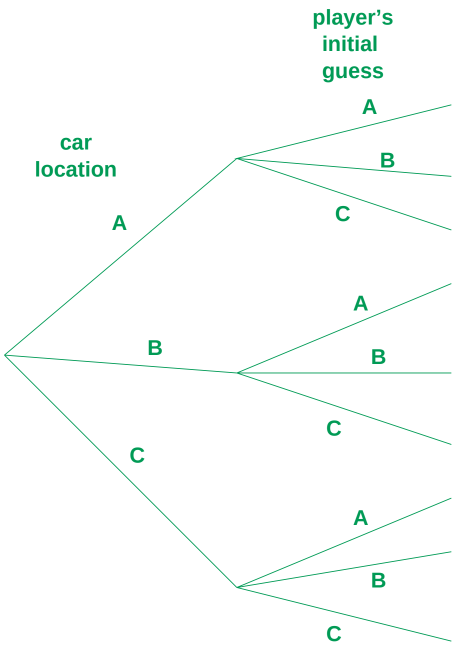
A *tree diagram* is a graphical tool that can help us work through the four step approach when the number of outcomes is not too large or the problem is nicely structured. In particular, we can use a tree diagram to help understand the sample space of an experiment. The first randomly-determined quantity in our experiment is the door concealing the prize. We represent this as a tree with three branches:



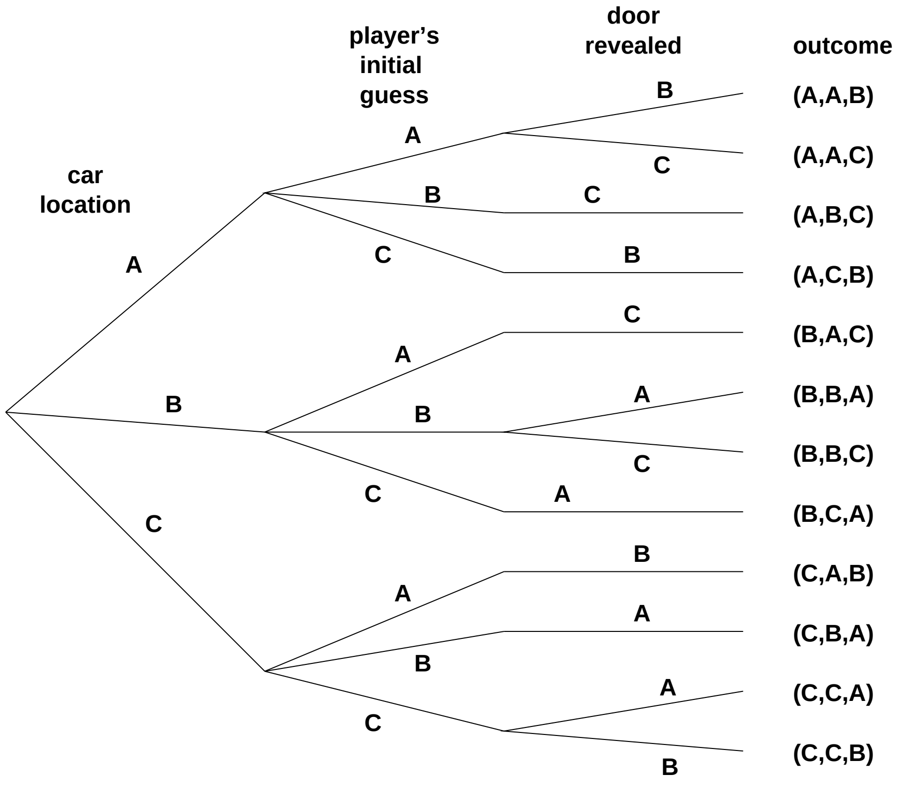
In this diagram, the doors are called A, B, and C instead of 1, 2, and 3 because we'll be adding a lot of other numbers to the picture later.

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:

EDITING NOTE:


player's initial guess
car location
A
B
C
A
B
C
A
B
C
A
B
C

■



Notice that the third layer reflects the fact that the host has either one choice or two, depending on the position of the car and the door initially selected by the player. For example, if the prize is behind door A and the player picks door B, then the host must open door C. However, if the prize is behind door A and the player picks door A, then the host could open either door B or door C.

Now let's relate this picture to the terms we introduced earlier: the leaves of the tree represent *outcomes* of the experiment, and the set of all leaves represents the *sample space*. Thus, for this experiment, the sample space consists of 12 outcomes. For reference, we've labeled each outcome with a triple of doors indicating:

$$(\text{door concealing prize},\ \text{door initially chosen},\ \text{door opened to reveal a goat})$$

In these terms, the sample space is the set:

$$\left\{ \begin{array}{cccccc} (A,A,B), & (A,A,C), & (A,B,C), & (A,C,B), & (B,A,C), & (B,B,A), \\ (B,B,C), & (B,C,A), & (C,A,B), & (C,B,A), & (C,C,A), & (C,C,B) \end{array} \right\}$$

The tree diagram has a broader interpretation as well: we can regard the whole experiment as following a path from the root to a leaf, where the branch taken at each stage is "randomly" determined. Keep this interpretation in mind; we'll use it again later.

14.1.4 Step 2: Define Events of Interest

Our objective is to answer questions of the form "What is the probability that ...?", where the missing phrase might be "the player wins by switching", "the player initially picked the door concealing the prize", or "the prize is behind door C", for example. Each of these phrases characterizes a set of outcomes: the outcomes specified by "the prize is behind door C" is:

$$\{(C, A, B), (C, B, A), (C, C, A), (C, C, B)\}$$

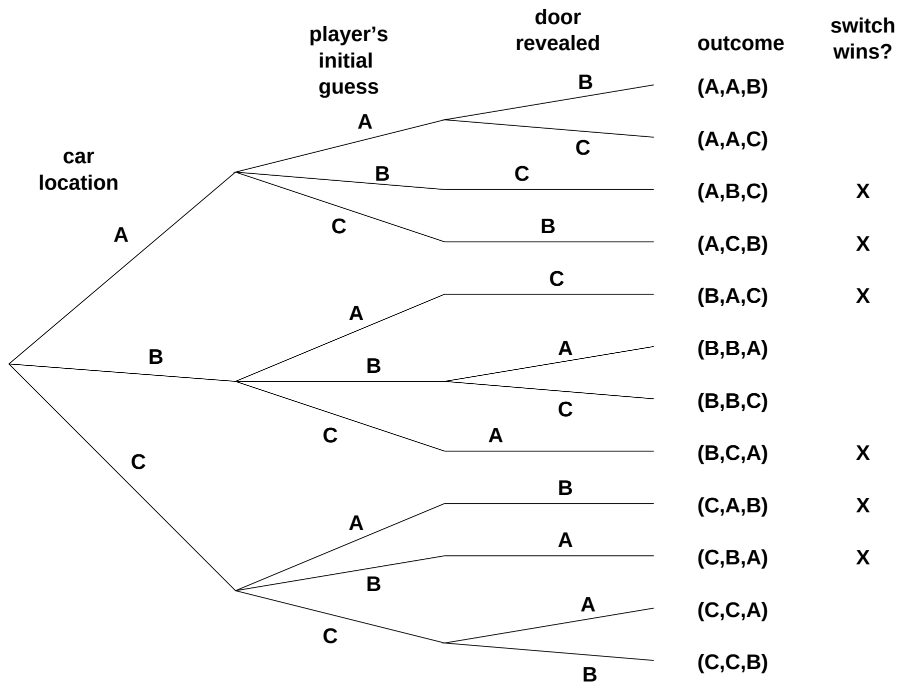
A set of outcomes is called an *event*. So the event that the player initially picked the door concealing the prize is the set:

$$\{(A, A, B), (A, A, C), (B, B, A), (B, B, C), (C, C, A), (C, C, B)\}$$

And what we're really after, the event that the player wins by switching, is the set of outcomes:

$$\{(A, B, C), (A, C, B), (B, A, C), (B, C, A), (C, A, B), (C, B, A)\}$$

Let's annotate our tree diagram to indicate the outcomes in this event.



Notice that exactly half of the outcomes are marked, meaning that the player wins by switching in half of all outcomes. You might be tempted to conclude that a player who switches wins with probability $1/2$. *This is wrong.* The reason is that these outcomes are not all equally likely, as we'll see shortly.

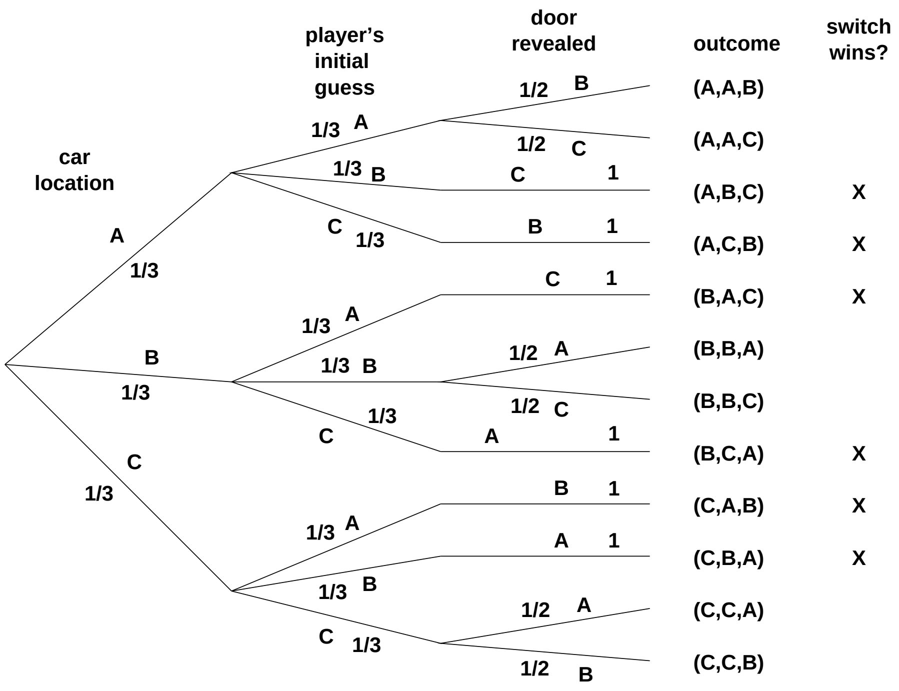
14.1.5 Step 3: Determine Outcome Probabilities

So far we've enumerated all the possible outcomes of the experiment. Now we must start assessing the likelihood of those outcomes. In particular, the goal of this step is to assign each outcome a probability, indicating the fraction of the time this outcome is expected to occur. The sum of all outcome probabilities must be one, reflecting the fact that there always is an outcome.

Ultimately, outcome probabilities are determined by the phenomenon we're modeling and thus are not quantities that we can derive mathematically. However, mathematics can help us compute the probability of every outcome *based on fewer and more elementary modeling decisions.* In particular, we'll break the task of determining outcome probabilities into two stages.

Step 3a: Assign Edge Probabilities

First, we record a probability on each *edge* of the tree diagram. These edge-probabilities are determined by the assumptions we made at the outset: that the prize is equally likely to be behind each door, that the player is equally likely to pick each door, and that the host is equally likely to reveal each goat, if he has a choice. Notice that when the host has no choice regarding which door to open, the single branch is assigned probability 1.



Step 3b: Compute Outcome Probabilities

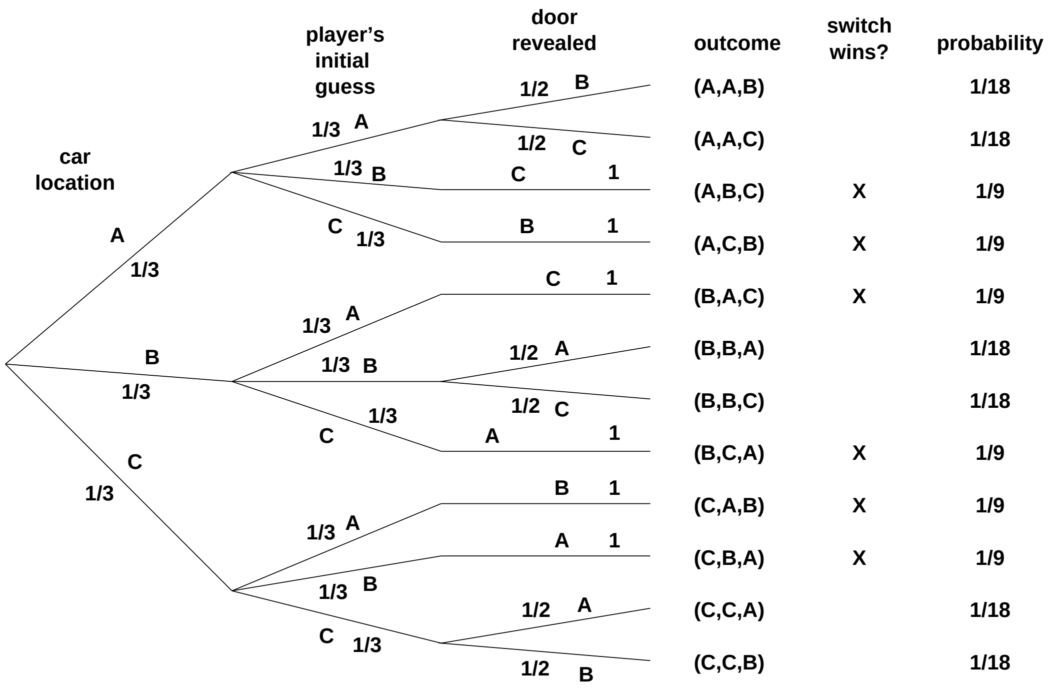
Our next job is to convert edge probabilities into outcome probabilities. This is a purely mechanical process: *the probability of an outcome is equal to the product of the edge-probabilities on the path from the root to that outcome*. For example, the probability of the topmost outcome, (A, A, B) is

$$\frac{1}{3} \cdot \frac{1}{3} \cdot \frac{1}{2} = \frac{1}{18}.$$

There's an easy, intuitive justification for this rule. As the steps in an experiment progress randomly along a path from the root of the tree to a leaf, the probabilities on the edges indicate how likely the walk is to proceed along each branch. For example, a path starting at the root in our example is equally likely to go down each of the three top-level branches.

Now, how likely is such a walk to arrive at the topmost outcome, (A, A, B)? Well, there is a 1-in-3 chance that a walk would follow the A-branch at the top level, a 1-in-3 chance it would continue along the A-branch at the second level, and 1-in-2 chance it would follow the B-branch at the third level. Thus, it seems that about 1 walk in 18 should arrive at the (A, A, B) leaf, which is precisely the probability we assign it.

Anyway, let's record all the outcome probabilities in our tree diagram.



Specifying the probability of each outcome amounts to defining a function that maps each outcome to a probability. This function is usually called **Pr**. In these terms, we've just determined that:

$$\Pr\{(A,A,B)\} = \frac{1}{18}$$

$$\Pr\{(A,A,C)\} = \frac{1}{18}$$

$$\Pr\{(A,B,C)\} = \frac{1}{9}$$

etc.

14.1.6 Step 4: Compute Event Probabilities

We now have a probability for each *outcome*, but we want to determine the probability of an *event* which will be the sum of the probabilities of the outcomes in it. The probability of an event, E, is written $\Pr\{E\}$. For example, the probability of the event that the player wins by switching is:

$$\begin{aligned}\Pr\{\text{switching wins}\} &= \Pr\{(A,B,C)\} + \Pr\{(A,C,B)\} + \Pr\{(B,A,C)\} + \\ &\quad \Pr\{(B,C,A)\} + \Pr\{(C,A,B)\} + \Pr\{(C,B,A)\} \\ &= \frac{1}{9} + \frac{1}{9} + \frac{1}{9} + \frac{1}{9} + \frac{1}{9} + \frac{1}{9} \\ &= \frac{2}{3}\end{aligned}$$

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.

14.1.7 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!

14.1.8 Problems

Class Problems

Homework Problems

14.2 Set Theory and Probability

Let's abstract what we've just done in this Monty Hall example into a general mathematical definition of probability. In the Monty Hall example, there were only finitely many possible outcomes. Other examples in this course will have a countably infinite number of outcomes.

General probability theory deals with uncountable sets like the set of real numbers, but we won't need these, and sticking to countable sets lets us define the probability of events using sums instead of integrals. It also lets us avoid some distracting technical problems in set theory like the Banach-Tarski "paradox" mentioned in Chapter 5.2.4.

14.2.1 Probability Spaces

Definition 14.2.1. A countable *sample space*, $\mathcal{S}$, is a nonempty countable set. An element $w \in \mathcal{S}$ is called an *outcome*. A subset of $\mathcal{S}$ is called an *event*.

Definition 14.2.2. A *probability function* on a sample space, $\mathcal{S}$, is a total function $\Pr\{\} : \mathcal{S} \to \mathbb{R}$ such that

- $\Pr\{w\} \geq 0$ for all $w \in \mathcal{S}$, and
- $\sum_{w \in \mathcal{S}} \Pr\{w\} = 1$.

The sample space together with a probability function is called a *probability space*.

For any event, $E \subseteq \mathcal{S}$, the *probability of E* is defined to be the sum of the probabilities of the outcomes in E:

$$\Pr\{E\} ::= \sum_{w \in E} \Pr\{w\}.$$

An immediate consequence of the definition of event probability is that for *disjoint* events, E, F,

$$\Pr\{E \cup F\} = \Pr\{E\} + \Pr\{F\}.$$

This generalizes to a countable number of events. Namely, a collection of sets is *pairwise disjoint* when no element is in more than one of them —formally, $A \cap B = \emptyset$ for all sets $A \neq B$ in the collection.

Rule (Sum Rule). *If $\{E_0, E_1, \dots\}$ is collection of pairwise disjoint events, then*

$$\Pr\left\{\bigcup_{n\in\mathbb{N}} E_n\right\} = \sum_{n\in\mathbb{N}} \Pr\{E_n\}.$$

The Sum Rule[1] lets us analyze a complicated event by breaking it down into simpler cases. For example, if the probability that a randomly chosen MIT student

[1]If you think like a mathematician, you should be wondering if the infinite sum is really necessary. Namely, suppose we had only used finite sums in Definition 14.2.2 instead of sums over all natural numbers. Would this imply the result for infinite sums? It's hard to find counterexamples, but there are some: it is possible to find a pathological "probability" measure on a sample space satisfying the Sum Rule for finite unions, in which the outcomes $w_0, w_1, \dots$ each have probability zero, and the probability assigned to any event is either zero or one! So the infinite Sum Rule fails dramatically, since the whole space is of measure one, but it is a union of the outcomes of measure zero.

is native to the United States is 60%, to Canada is 5%, and to Mexico is 5%, then the probability that a random MIT student is native to North America is 70%.

Another consequence of the Sum Rule is that $\Pr\{A\} + \Pr\{\overline{A}\} = 1$, which follows because $\Pr\{\mathcal{S}\} = 1$ and $\mathcal{S}$ is the union of the disjoint sets A and $\overline{A}$. This equation often comes up in the form

Rule (Complement Rule)**.**

$$\Pr\{\overline{A}\} = 1 - \Pr\{A\}.$$

Sometimes the easiest way to compute the probability of an event is to compute the probability of its complement and then apply this formula.

Some further basic facts about probability parallel facts about cardinalities of

The construction of such weird examples is beyond the scope of this text. You can learn more about this by taking a course in Set Theory and Logic that covers the topic of "ultrafilters."

finite sets. In particular:

$$\Pr\{B - A\} = \Pr\{B\} - \Pr\{A \cap B\}, \qquad \text{(Difference Rule)}$$

$$\Pr\{A \cup B\} = \Pr\{A\} + \Pr\{B\} - \Pr\{A \cap B\}, \qquad \text{(Inclusion-Exclusion)}$$

$$\Pr\{A \cup B\} \leq \Pr\{A\} + \Pr\{B\}. \qquad \text{(Boole's Inequality)}$$

The Difference Rule follows from the Sum Rule because B is the union of the disjoint sets $B - A$ and $A \cap B$. Inclusion-Exclusion then follows from the Sum and Difference Rules, because $A \cup B$ is the union of the disjoint sets A and $B - A$. Boole's inequality is an immediate consequence of Inclusion-Exclusion since probabilities are nonnegative.

The two event Inclusion-Exclusion equation above generalizes to n events in the same way as the corresponding Inclusion-Exclusion rule for n sets. Boole's inequality also generalizes to

$$\Pr\{E_1 \cup \cdots \cup E_n\} \leq \Pr\{E_1\} + \cdots + \Pr\{E_n\}. \qquad \text{(Union Bound)}$$

This simple Union Bound is actually useful in many calculations. For example, suppose that E_i is the event that the i-th critical component in a spacecraft fails.

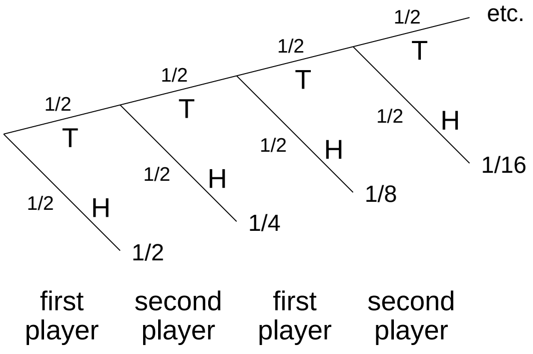
Then $E_1 \cup \cdots \cup E_n$ is the event that *some* critical component fails. The Union Bound can give an adequate upper bound on this vital probability.

Similarly, the Difference Rule implies that

$$\text{If } A \subseteq B, \text{ then } \Pr\{A\} \leq \Pr\{B\}. \qquad \text{(Monotonicity)}$$

14.2.2 An Infinite Sample Space

Suppose two players take turns flipping a fair coin. Whoever flips heads first is declared the winner. What is the probability that the first player wins? A tree diagram for this problem is shown below:



The event that the first player wins contains an infinite number of outcomes, but we can still sum their probabilities:

$$\begin{aligned}\Pr\{\text{first player wins}\} &= \frac{1}{2} + \frac{1}{8} + \frac{1}{32} + \frac{1}{128} + \cdots \\ &= \frac{1}{2}\sum_{n=0}^{\infty}\left(\frac{1}{4}\right)^n \\ &= \frac{1}{2}\left(\frac{1}{1-1/4}\right) = \frac{2}{3}.\end{aligned}$$

Similarly, we can compute the probability that the second player wins:

$$\begin{aligned}\Pr\{\text{second player wins}\} &= \frac{1}{4} + \frac{1}{16} + \frac{1}{64} + \frac{1}{256} + \cdots \\ &= \frac{1}{3}.\end{aligned}$$

To be formal about this, sample space is the infinite set

$$\mathcal{S} ::= \{\mathtt{T}^n\mathtt{H} \mid n \in \mathbb{N}\}$$

where $\mathtt{T}^n$ stands for a length n string of $\mathtt{T}$'s. The probability function is

$$\Pr\{\mathtt{T}^n\mathtt{H}\} ::= \frac{1}{2^{n+1}}.$$

Since this function is obviously nonnegative, To verify that this is a probability space, we just have to check that all the probabilities sum to 1. But this follows directly from the formula for the sum of a geometric series:

$$\sum_{\mathtt{T}^n\mathtt{H}\in\mathcal{S}} \Pr\{\mathtt{T}^n\mathtt{H}\} = \sum_{n\in\mathbb{N}} \frac{1}{2^{n+1}} = \frac{1}{2}\sum_{n\in\mathbb{N}} \frac{1}{2^n} = 1.$$

Notice that this model does not have an outcome corresponding to the possibility that both players keep flipping tails forever —in the diagram, flipping forever corresponds to following the infinite path in the tree without ever reaching a leaf/outcome. If leaving this possibility out of the model bothers you, you're welcome to fix it by adding another outcome, w_{forever}, to indicate that that's what happened. Of course since the probabililities of the other outcomes already sum to 1, you have to define the probability of w_{forever} to be 0. Now outcomes with probability zero will have no impact on our calculations, so there's no harm in adding it in if it makes you happier. On the other hand, there's also no harm in simply leaving it out as we did, since it has no impact.

The mathematical machinery we've developed is adequate to model and analyze many interesting probability problems with infinite sample spaces. However, some intricate infinite processes require uncountable sample spaces along with more powerful (and more complex) measure-theoretic notions of probability. For example, if we generate an infinite sequence of random bits $b_1, b_2, b_3, \ldots$, then what is the probability that

$$\frac{b_1}{2^1} + \frac{b_2}{2^2} + \frac{b_3}{2^3} + \cdots$$

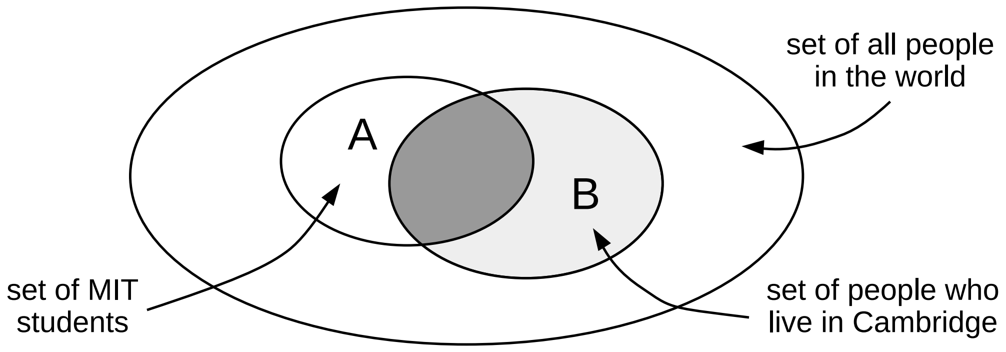
is a rational number? Fortunately, we won't have any need to worry about such things.

14.2.3 Problems

Class Problems

14.3 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:


set of all people in the world
A
B
set of MIT students
set of people who live in Cambridge

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 14.3.1.

$$\Pr\{A \mid B\} ::= \frac{\Pr\{A \cap B\}}{\Pr\{B\}}$$

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.

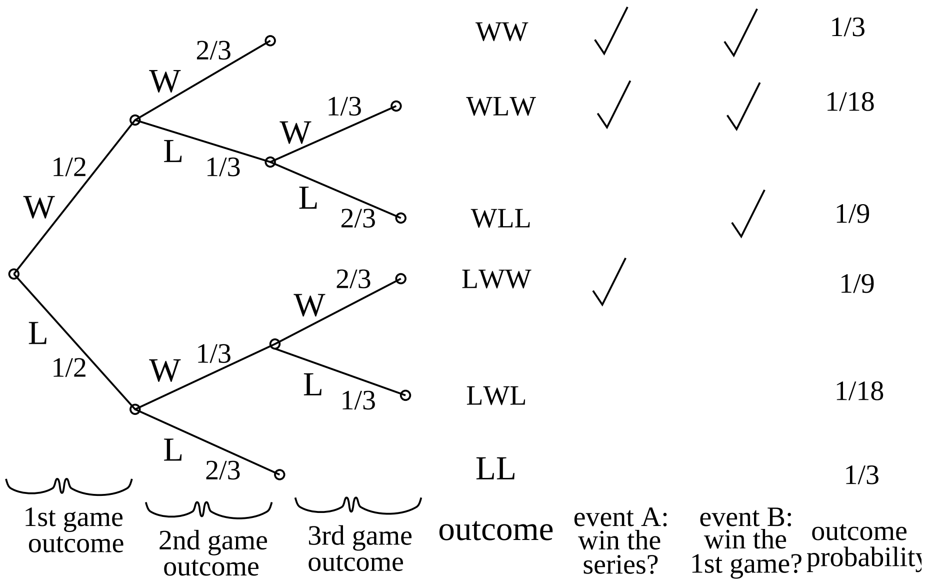
14.3.1 The "Halting Problem"

The *Halting Problem* was the first example of a property that could not be tested by any program. It was introduced by Alan Turing in his seminal 1936 paper. The problem is to determine whether a Turing machine halts on a given …yadda yadda yadda …what's much *more important*, it was the name of the MIT EECS department's famed C-league hockey team.

In a best-of-three tournament, the Halting Problem wins the first game with probability $1/2$. In subsequent games, their probability of winning is determined by the outcome of the previous game. If the Halting Problem won the previous game, then they are invigorated by victory and win the current game with probability $2/3$. If they lost the previous game, then they are demoralized by defeat and win the current game with probablity only $1/3$. What is the probability that the Halting Problem wins the tournament, given that they win the first game?

This is a question about a conditional probability. Let A be the event that the Halting Problem wins the tournament, and let B be the event that they win the first game. Our goal is then to determine the conditional probability $\Pr\{A \mid B\}$.

We can tackle conditional probability questions just like ordinary probability problems: using a tree diagram and the four step method. A complete tree diagram is shown below, followed by an explanation of its construction and use.



Step 1: Find the Sample Space

Each internal vertex in the tree diagram has two children, one corresponding to a win for the Halting Problem (labeled W) and one corresponding to a loss (labeled L). The complete sample space is:

$$\mathcal{S} = \{WW,\ WLW,\ WLL,\ LWW,\ LWL,\ LL\}$$

Step 2: Define Events of Interest

The event that the Halting Problem wins the whole tournament is:

$$T = \{WW,\ WLW,\ LWW\}$$

And the event that the Halting Problem wins the first game is:

$$F = \{WW, WLW, WLL\}$$

The outcomes in these events are indicated with checkmarks in the tree diagram.

Step 3: Determine Outcome Probabilities

Next, we must assign a probability to each outcome. We begin by labeling edges as specified in the problem statement. Specifically, The Halting Problem has a $1/2$ chance of winning the first game, so the two edges leaving the root are each assigned probability $1/2$. Other edges are labeled $1/3$ or $2/3$ based on the outcome of the preceding game. We then find the probability of each outcome by multiplying all probabilities along the corresponding root-to-leaf path. For example, the probability of outcome WLL is:

$$\frac{1}{2} \cdot \frac{1}{3} \cdot \frac{2}{3} = \frac{1}{9}$$

Step 4: Compute Event Probabilities

We can now compute the probability that The Halting Problem wins the tournament, given that they win the first game:

$$\begin{aligned}
\Pr\{A \mid B\} &= \frac{\Pr\{A \cap B\}}{\Pr\{B\}} \\
&= \frac{\Pr\{\{WW, WLW\}\}}{\Pr\{\{WW, WLW, WLL\}\}} \\
&= \frac{1/3 + 1/18}{1/3 + 1/18 + 1/9} \\
&= \frac{7}{9}
\end{aligned}$$

We're done! If the Halting Problem wins the first game, then they win the whole tournament with probability $7/9$.

14.3.2 Why Tree Diagrams Work

We've now settled into a routine of solving probability problems using tree diagrams. But we've left a big question unaddressed: what is the mathematical justification behind those funny little pictures? Why do they work?

The answer involves conditional probabilities. In fact, the probabilities that we've been recording on the edges of tree diagrams *are* conditional probabilities. For example, consider the uppermost path in the tree diagram for the Halting Problem, which corresponds to the outcome WW. The first edge is labeled $1/2$, which is the probability that the Halting Problem wins the first game. The second edge is labeled $2/3$, which is the probability that the Halting Problem wins the second game, *given* that they won the first— that's a conditional probability! More generally, on each edge of a tree diagram, we record the probability that the experiment proceeds along that path, given that it reaches the parent vertex.

So we've been using conditional probabilities all along. But why can we multiply edge probabilities to get outcome probabilities? For example, we concluded that:

$$\begin{aligned}\Pr\{WW\} &= \frac{1}{2}\cdot\frac{2}{3}\\ &= \frac{1}{3}\end{aligned}$$

Why is this correct?

The answer goes back to Definition 14.3.1 of conditional probability which could be written in a form called the *Product Rule* for probabilities:

Rule (Product Rule for 2 Events)**.** *If* $\Pr\{E_1\} \neq 0$*, then:*

$$\Pr\{E_1 \cap E_2\} = \Pr\{E_1\} \cdot \Pr\{E_2 \mid E_1\}$$

Multiplying edge probabilities in a tree diagram amounts to evaluating the right side of this equation. For example:

$$\begin{aligned}&\Pr\{\text{win first game} \cap \text{win second game}\}\\ &\qquad = \Pr\{\text{win first game}\} \cdot \Pr\{\text{win second game} \mid \text{win first game}\}\\ &\qquad = \frac{1}{2}\cdot\frac{2}{3}\end{aligned}$$

So the Product Rule is the formal justification for multiplying edge probabilities to get outcome probabilities! Of course to justify multiplying edge probabilities along longer paths, we need a Product Rule for n events. The pattern of the n event rule should be apparent from

Rule (Product Rule for 3 Events)**.**

$$\Pr\{E_1 \cap E_2 \cap E_3\} = \Pr\{E_1\} \cdot \Pr\{E_2 \mid E_1\} \cdot \Pr\{E_3 \mid E_2 \cap E_1\}$$

providing $\Pr\{E_1 \cap E_2\} \neq 0$.

This rule follows from the definition of conditional probability and the trivial identity

$$\Pr\{E_1 \cap E_2 \cap E_3\} = \Pr\{E_1\} \cdot \frac{\Pr\{E_2 \cap E_1\}}{\Pr\{E_1\}} \cdot \frac{\Pr\{E_3 \cap E_2 \cap E_1\}}{\Pr\{E_2 \cap E_1\}}$$

14.3.3 The Law of Total Probability

Breaking a probability calculation into cases simplifies many problems. The idea is to calculate the probability of an event A by splitting into two cases based on whether or not another event E occurs. That is, calculate the probability of $A \cap E$ and $A \cap \overline{E}$. By the Sum Rule, the sum of these probabilities equals $\Pr\{A\}$. Expressing the intersection probabilities as conditional probabilities yields

Rule (Total Probability)**.**

$$\Pr\{A\} = \Pr\{A \mid E\} \cdot \Pr\{E\} + \Pr\{A \mid \overline{E}\} \cdot \Pr\{\overline{E}\}.$$

For example, suppose we conduct the following experiment. First, we flip a coin. If heads comes up, then we roll one die and take the result. If tails comes up, then we roll two dice and take the sum of the two results. What is the probability that this process yields a 2? Let E be the event that the coin comes up heads, and let A be the event that we get a 2 overall. Assuming that the coin is fair, $\Pr\{E\} = \Pr\{\overline{E}\} = 1/2$. There are now two cases. If we flip heads, then we roll a 2 on a single die with probabilty $\Pr\{A \mid E\} = 1/6$. On the other hand, if we flip tails, then we get a sum of 2 on two dice with probability $\Pr\{A \mid \overline{E}\} = 1/36$. Therefore, the probability that the whole process yields a 2 is

$$\Pr\{A\} = \frac{1}{2} \cdot \frac{1}{6} + \frac{1}{2} \cdot \frac{1}{36} = \frac{7}{72}.$$

There is also a form of the rule to handle more than two cases.

Rule (Multicase Total Probability)**.** *If $E_1, \ldots, E_n$ are pairwise disjoint events whose union is the whole sample space, then:*

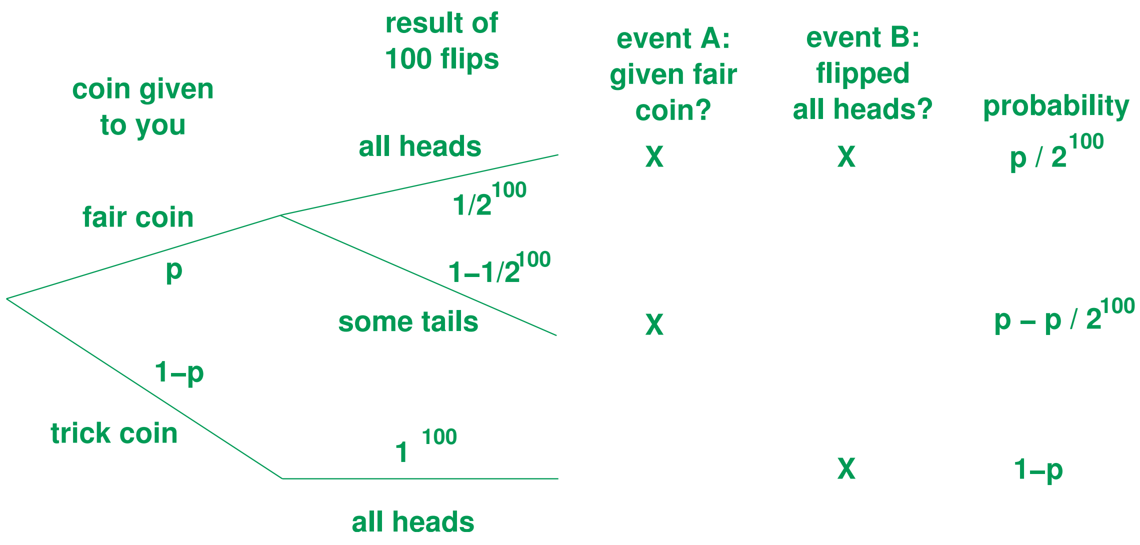
$$\Pr\{A\} = \sum_{i=1}^{n} \Pr\{A \mid E_i\} \cdot \Pr\{E_i\}.$$

EDITING NOTE:

A Coin Problem

Someone hands you either a fair coin or a trick coin with heads on both sides. You flip the coin 100 times and see heads every time. What can you say about the probability that you flipped the fair coin? Remarkably— nothing!

In order to make sense out of this outrageous claim, let's formalize the problem. The sample space is worked out in the tree diagram below. We do not know the probability that you were handed the fair coin initially— you were just given one coin or the other— so let's call that p.



Let A be the event that you were handed the fair coin, and let B be the event that you flipped 100 heads. Now, we're looking for $\Pr\{A \mid B\}$, the probability that you were handed the fair coin, given that you flipped 100 heads. The outcome probabilities are worked out in the tree diagram. Plugging the results into the definition of conditional probability gives:

$$\begin{aligned}\Pr\{A \mid B\} &= \frac{\Pr\{A \cap B\}}{\Pr\{B\}} \\ &= \frac{p/2^{100}}{1-p+p/2^{100}} \\ &= \frac{p}{2^{100}(1-p)+p}\end{aligned}$$

This expression is very small for moderate values of p because of the 2^{100} term in the denominator. For example, if $p = 1/2$, then the probability that you were given the fair coin is essentially zero.

But we *do not know* the probability p that you were given the fair coin. And perhaps the value of p is *not* moderate; in fact, maybe $p = 1 - 2^{-100}$. Then there is nearly an even chance that you have the fair coin, given that you flipped 100 heads. In fact, maybe you were handed the fair coin with probability $p = 1$. Then the probability that you were given the fair coin is, well, 1!

A similar problem arises in polling before an election. A pollster picks a random American and asks his or her party affiliation. If this process is repeated many times, what can be said about the population as a whole? To clarify the analogy, suppose that the country contains only two people. There is either one Republican and one Democrat (like the fair coin), or there are two Republicans (like the trick coin). The pollster picks a random citizen 100 times, which is analogous to flipping the coin 100 times. Suppose that he picks a Republican every single time. However, even given this polling data, the probability that there is one citizen in each party could still be anywhere between 0 and 1!

What the pollster *can* say is that either:

1. Something earth-shatteringly unlikely happened during the poll.

2. There are two Republicans.

This is as far as probability theory can take us; from here, you must draw your own conclusions. Based on life experience, many people would consider the second possibility more plausible. However, if you are just *convinced* that the country isn't entirely Republican (say, because you're a citizen and a Democrat), then you might believe that the first possibility is actually more likely.

■

14.3.4 Medical Testing

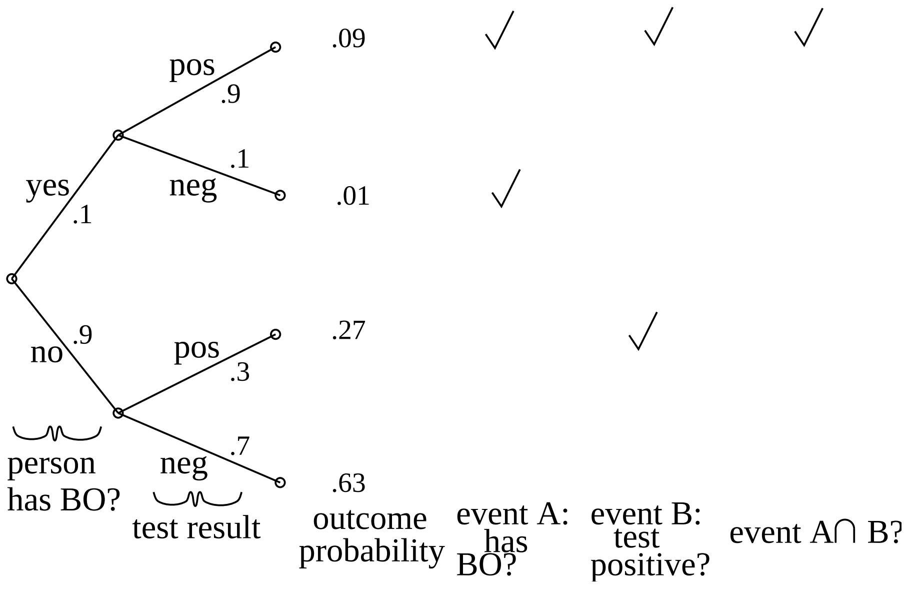
There is an unpleasant condition called *BO* suffered by 10% of the population. There are no prior symptoms; victims just suddenly start to stink. Fortunately, there is a test for latent *BO* before things start to smell. The test is not perfect, however:

- If you have the condition, there is a 10% chance that the test will say you do not. (These are called "false negatives".)
- If you do not have the condition, there is a 30% chance that the test will say you do. (These are "false positives".)

Suppose a random person is tested for latent *BO*. If the test is positive, then what is the probability that the person has the condition?

Step 1: Find the Sample Space

The sample space is found with the tree diagram below.



Step 2: Define Events of Interest

Let A be the event that the person has BO. Let B be the event that the test was positive. The outcomes in each event are marked in the tree diagram. We want to find $\Pr\{A \mid B\}$, the probability that a person has BO, given that the test was positive.

Step 3: Find Outcome Probabilities

First, we assign probabilities to edges. These probabilities are drawn directly from the problem statement. By the Product Rule, the probability of an outcome is the product of the probabilities on the corresponding root-to-leaf path. All probabilities are shown in the figure.

Step 4: Compute Event Probabilities

p

$$\begin{aligned}
\Pr\{A \mid B\} &= \frac{\Pr\{A \cap B\}}{\Pr\{B\}} \\
&= \frac{0.09}{0.09 + 0.27} \\
&= \frac{1}{4}
\end{aligned}$$

If you test positive, then there is only a 25% chance that you have the condition!

This answer is initially surprising, but makes sense on reflection. There are two ways you could test positive. First, it could be that you are sick and the test is correct. Second, it could be that you are healthy and the test is incorrect. The problem is that almost everyone is healthy; therefore, most of the positive results arise from incorrect tests of healthy people!

We can also compute the probability that the test is correct for a random person. This event consists of two outcomes. The person could be sick and the test positive (probability 0.09), or the person could be healthy and the test negative (probability 0.63). Therefore, the test is correct with probability $0.09 + 0.63 = 0.72$. This is a relief; the test is correct almost three-quarters of the time.

But wait! There is a simple way to make the test correct 90% of the time: always return a negative result! This "test" gives the right answer for all healthy people and the wrong answer only for the 10% that actually have the condition. The best strategy is to completely ignore the test result!

There is a similar paradox in weather forecasting. During winter, almost all days in Boston are wet and overcast. Predicting miserable weather every day may be more accurate than really trying to get it right!

14.3.5 Conditional Identities

The probability rules above extend to probabilities conditioned on the same event. For example, the Inclusion-Exclusion formula for two sets holds when all probabilities are conditioned on an event C:

$$\Pr\{A \cup B \mid C\} = \Pr\{A \mid C\} + \Pr\{B \mid C\} - \Pr\{A \cap B \mid C\}.$$

This follows from the fact that if $\Pr\{C\} \neq 0$ and we define

$$\Pr_C\{A\} ::= \Pr\{A \mid C\}$$

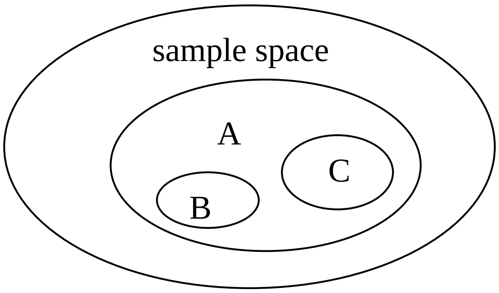
then $\Pr_C\{\}$ satisfies the definition of being probability function.

It is important not to mix up events before and after the conditioning bar. For example, the following is *not* a valid identity:

False Claim.

$$\Pr\{A \mid B \cup C\} = \Pr\{A \mid B\} + \Pr\{A \mid C\} - \Pr\{A \mid B \cap C\}. \tag{14.1}$$

A counterexample is shown below. In this case, $\Pr\{A \mid B\} = 1$, $\Pr\{A \mid C\} = 1$, and $\Pr\{A \mid B \cup C\} = 1$. However, since $1 \neq 1 + 1$, the equation above does not hold.



So you're convinced that this equation is false in general, right? Let's see if you *really* believe that.

14.3.6 Discrimination Lawsuit

Several years ago there was a sex discrimination lawsuit against Berkeley. A female professor was denied tenure, allegedly because she was a woman. She argued that in every one of Berkeley's 22 departments, the percentage of male applicants accepted was greater than the percentage of female applicants accepted. This sounds very suspicious!

However, Berkeley's lawyers argued that across the whole university the percentage of male tenure applicants accepted was actually *lower* than the percentage of female applicants accepted. This suggests that if there was any sex discrimination, then it was against men! Surely, at least one party in the dispute must be lying.

Let's simplify the problem and express both arguments in terms of conditional probabilities. Suppose that there are only two departments, EE and CS, and consider the experiment where we pick a random applicant. Define the following events:

- Let A be the event that the applicant is accepted.
- Let F_{EE} the event that the applicant is a female applying to EE.
- Let F_{CS} the event that the applicant is a female applying to CS.
- Let M_{EE} the event that the applicant is a male applying to EE.
- Let M_{CS} the event that the applicant is a male applying to CS.

Assume that all applicants are either male or female, and that no applicant applied to both departments. That is, the events F_{EE}, F_{CS}, M_{EE}, and M_{CS} are all disjoint.

In these terms, the plaintiff is make the following argument:

$$\Pr\{A \mid F_{EE}\} < \Pr\{A \mid M_{EE}\}$$

$$\Pr\{A \mid F_{CS}\} < \Pr\{A \mid M_{CS}\}$$

That is, in both departments, the probability that a woman is accepted for tenure is less than the probability that a man is accepted. The university retorts that overall a woman applicant is *more* likely to be accepted than a man:

$$\Pr\{A \mid F_{EE} \cup F_{CS}\} > \Pr\{A \mid M_{EE} \cup M_{CS}\}$$

It is easy to believe that these two positions are contradictory. In fact, we might even try to prove this by adding the plaintiff's two inequalities and then arguing as follows:

$$\begin{aligned} & \Pr\{A \mid F_{EE}\} + \Pr\{A \mid F_{CS}\} < \Pr\{A \mid M_{EE}\} + \Pr\{A \mid M_{CS}\} \\ \Rightarrow \quad & \Pr\{A \mid F_{EE} \cup F_{CS}\} < \Pr\{A \mid M_{EE} \cup M_{CS}\} \end{aligned}$$

The second line exactly contradicts the university's position! But there is a big problem with this argument; the second inequality follows from the first only if we accept the false identity (14.1). This argument is bogus! Maybe the two parties do not hold contradictory positions after all!

In fact, the table below shows a set of application statistics for which the assertions of both the plaintiff and the university hold:

CS	0 females accepted, 1 applied	0%
	50 males accepted, 100 applied	50%
EE	70 females accepted, 100 applied	70%
	1 male accepted, 1 applied	100%
Overall	70 females accepted, 101 applied	$\approx 70\%$
	51 males accepted, 101 applied	$\approx 51\%$

In this case, a higher percentage of males were accepted in both departments, but overall a higher percentage of females were accepted! Bizarre!

14.3.7 *A Posteriori* Probabilities

Suppose that we turn the hockey question around: what is the probability that the Halting Problem won their first game, given that they won the series?

This seems like an absurd question! After all, if the Halting Problem won the series, then the winner of the first game has already been determined. Therefore, who won the first game is a question of fact, not a question of probability. However, our mathematical theory of probability contains no notion of one event preceding another— there is no notion of time at all. Therefore, from a mathematical perspective, this is a perfectly valid question. And this is also a meaningful question from a practical perspective. Suppose that you're told that the Halting Problem won the series, but not told the results of individual games. Then, from your perspective, it makes perfect sense to wonder how likely it is that The Halting Problem won the first game.

A conditional probability $\Pr\{B \mid A\}$ is called *a posteriori* if event B precedes event A in time. Here are some other examples of a posteriori probabilities:

- The probability it was cloudy this morning, given that it rained in the afternoon.
- The probability that I was initially dealt two queens in Texas No Limit Hold 'Em poker, given that I eventually got four-of-a-kind.

Mathematically, a posteriori probabilities are *no different* from ordinary probabilities; the distinction is only at a higher, philosophical level. Our only reason for drawing attention to them is to say, "Don't let them rattle you."

Let's return to the original problem. The probability that the Halting Problem won their first game, given that they won the series is $\Pr\{B \mid A\}$. We can compute this using the definition of conditional probability and our earlier tree diagram:

$$\begin{aligned}\Pr\{B \mid A\} &= \frac{\Pr\{B \cap A\}}{\Pr\{A\}} \\ &= \frac{1/3 + 1/18}{1/3 + 1/18 + 1/9} \\ &= \frac{7}{9}\end{aligned}$$

This answer is suspicious! In the preceding section, we showed that $\Pr\{A \mid B\}$ was also $7/9$. Could it be true that $\Pr\{A \mid B\} = \Pr\{B \mid A\}$ in general? Some reflection suggests this is unlikely. For example, the probability that I feel uneasy, given that I was abducted by aliens, is pretty large. But the probability that I was abducted by aliens, given that I feel uneasy, is rather small.

Let's work out the general conditions under which $\Pr\{A \mid B\} = \Pr\{B \mid A\}$. By the definition of conditional probability, this equation holds if an only if:

$$\frac{\Pr\{A \cap B\}}{\Pr\{B\}} = \frac{\Pr\{A \cap B\}}{\Pr\{A\}}$$

This equation, in turn, holds only if the denominators are equal or the numerator is 0:

$$\Pr\{B\} = \Pr\{A\} \qquad \text{or} \qquad \Pr\{A \cap B\} = 0$$

The former condition holds in the hockey example; the probability that the Halting Problem wins the series (event A) is equal to the probability that it wins the first game (event B). In fact, both probabilities are $1/2$.

Such pairs of probabilities are related by Bayes' Rule:

Theorem 14.3.2 (Bayes' Rule)**.** *If* $\Pr\{A\}$ *and* $\Pr\{B\}$ *are nonzero, then:*

$$\frac{\Pr\{A \mid B\} \cdot \Pr\{B\}}{\Pr\{A\}} = \Pr\{B \mid A\} \tag{14.2}$$

Proof. When $\Pr\{A\}$ and $\Pr\{B\}$ are nonzero, we have

$$\Pr\{A \mid B\} \cdot \Pr\{B\} = \Pr\{A \cap B\} = \Pr\{B \mid A\} \cdot \Pr\{A\}$$

by definition of conditional probability. Dividing by $\Pr\{A\}$ gives (14.2).

■

In the hockey problem, the probability that the Halting Problem wins the first game is $1/2$ and so is the probability that the Halting Problem wins the series.

Therefore, $\Pr\{A\} = \Pr\{B\} = 1/2$. This, together with Bayes' Rule, explains why $\Pr\{A \mid B\}$ and $\Pr\{B \mid A\}$ turned out to be equal in the hockey example.

14.3.8 Problems

Practice Problems

Class Problems

Homework Problems

14.4 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.

14.4.1 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 $1/10$.

14.4.2 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*

$$\Pr\{A \mid B\} = \Pr\{A\}. \tag{14.3}$$

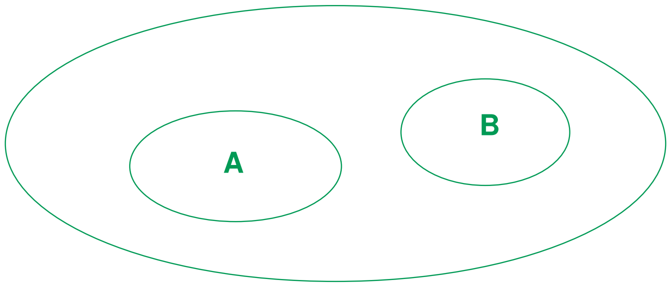
Equation (14.3) says that events A and B are independent if the probability of A is unaffected by the fact that B happens. In these terms, the two coin tosses of the previous section were independent, because the probability that one coin comes up heads is unaffected by the fact that the other came up heads. Turning to our other example, the probability of clouds in the sky is strongly affected by the fact that it is raining. So, as we noted before, these events are not independent.

Warning: Students sometimes get the idea that disjoint events are independent. The *opposite* is true: if $A \cap B = \emptyset$, then knowing that A happens means you know that B does not happen. So disjoint events are *never* independent —unless one of them has probability zero.

EDITING NOTE:

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

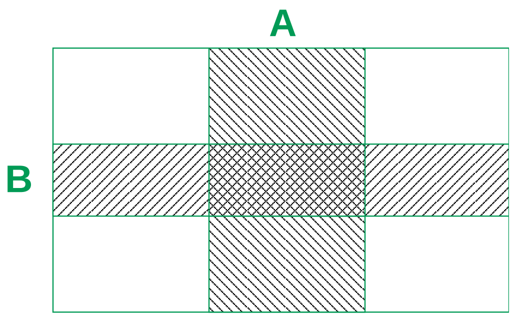
$$\Pr\{A \cap B\} = 0$$

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:

$$\Pr\{A \cap B\} = \Pr\{A\} \cdot \Pr\{B\}$$

■

14.4.3 Mutual Independence

We have defined what it means for two events to be independent. But how can we talk about independence when there are more than two events? For example, how can we say that the orientations of n coins are all independent of one another?

Events $E_1, \ldots, E_n$ are *mutually independent* if and only if *for every subset* of the events, the probability of the intersection is the product of the probabilities. In other words, all of the following equations must hold:

$$\Pr\{E_i \cap E_j\} = \Pr\{E_i\} \cdot \Pr\{E_j\} \qquad \text{for all distinct } i, j$$

$$\Pr\{E_i \cap E_j \cap E_k\} = \Pr\{E_i\} \cdot \Pr\{E_j\} \cdot \Pr\{E_k\} \qquad \text{for all distinct } i, j, k$$

$$\Pr\{E_i \cap E_j \cap E_k \cap E_l\} = \Pr\{E_i\} \cdot \Pr\{E_j\} \cdot \Pr\{E_k\} \cdot \Pr\{E_l\} \qquad \text{for all distinct } i, j, k, l$$

$$\dots$$

$$\Pr\{E_1 \cap \dots \cap E_n\} = \Pr\{E_1\} \cdots \Pr\{E_n\}$$

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 $E_1, \dots, E_{100}$ are mutually independent.

EDITING NOTE:

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?

■

14.4.4 Pairwise Independence

The definition of mutual independence seems awfully complicated— there are so many conditions! Here's an example that illustrates the subtlety of independence when more than two events are involved and the need for all those conditions. Suppose that we flip three fair, mutually-independent coins. Define the following events:

- A_1 is the event that coin 1 matches coin 2.
- A_2 is the event that coin 2 matches coin 3.
- A_3 is the event that coin 3 matches coin 1.

Are A_1, A_2, A_3 mutually independent?

The sample space for this experiment is:

$$\{HHH,\ HHT,\ HTH,\ HTT,\ THH,\ THT,\ TTH,\ TTT\}$$

Every outcome has probability $(1/2)^3 = 1/8$ by our assumption that the coins are mutually independent.

To see if events A_1, A_2, and A_3 are mutually independent, we must check a sequence of equalities. It will be helpful first to compute the probability of each event A_i:

$$\begin{aligned}
\Pr\{A_1\} &= \Pr\{HHH\} + \Pr\{HHT\} + \Pr\{TTH\} + \Pr\{TTT\} \\
&= \frac{1}{8} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8} \\
&= \frac{1}{2}
\end{aligned}$$

By symmetry, $\Pr\{A_2\} = \Pr\{A_3\} = 1/2$ as well. Now we can begin checking all the equalities required for mutual independence.

$$\begin{aligned}\Pr\{A_1 \cap A_2\} &= \Pr\{HHH\} + \Pr\{TTT\}\\ &= \frac{1}{8} + \frac{1}{8}\\ &= \frac{1}{4}\\ &= \frac{1}{2} \cdot \frac{1}{2}\\ &= \Pr\{A_1\}\Pr\{A_2\}\end{aligned}$$

By symmetry, $\Pr\{A_1 \cap A_3\} = \Pr\{A_1\} \cdot \Pr\{A_3\}$ and $\Pr\{A_2 \cap A_3\} = \Pr\{A_2\} \cdot \Pr\{A_3\}$ must hold also. Finally, we must check one last condition:

$$\begin{aligned}\Pr\{A_1 \cap A_2 \cap A_3\} &= \Pr\{HHH\} + \Pr\{TTT\}\\ &= \frac{1}{8} + \frac{1}{8}\\ &= \frac{1}{4}\\ &\neq \Pr\{A_1\}\Pr\{A_2\}\Pr\{A_3\} = \frac{1}{8}\end{aligned}$$

The three events A_1, A_2, and A_3 are not mutually independent even though any two of them are independent! This not-quite mutual independence seems weird at first, but it happens. It even generalizes:

Definition 14.4.1. A set $A_0, A_1, \ldots$ of events is *k-way independent* iff every set of k of these events is mutually independent. The set is *pairwise independent* iff it is 2-way independent.

So the sets A_1, A_2, A_3 above are pairwise independent, but not mutually independent. Pairwise independence is a much weaker property than mutual independence, but it's all that's needed to justify a standard approach to making probabilistic estimates that will come up later.

EDITING NOTE:

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.

■

14.4.5 Problems

Class Problems

14.5 The Birthday Principle

There are 85 students in a class. What is the probability that some birthday is shared by two people? Comparing 85 students to the 365 possible birthdays, you might guess the probability lies somewhere around $1/4$ —but you'd be wrong: the probability that there will be two people in the class with matching birthdays is actually more than 0.9999.

To work this out, we'll assume that the probability that a randomly chosen student has a given birthday is $1/d$, where $d = 365$ in this case. We'll also assume that a class is composed of n randomly and independently selected students, with $n = 85$ in this case. These randomness assumptions are not really true, since more babies are born at certain times of year, and students' class selections are typically not independent of each other, but simplifying in this way gives us a start on analyzing the problem. More importantly, these assumptions are justifiable in important computer science applications of birthday matching. For example, the birthday matching is a good model for collisions between items randomly inserted into a hash table. So we won't worry about things like Spring procreation preferences that make January birthdays more common, or about twins' preferences to take classes together (or not).

EDITING NOTE: or that fact that a student can't be selected twice in making up a class list. ■

Selecting a sequence of n students for a class yields a sequence of n birthdays. Under the assumptions above, the d^n possible birthday sequences are equally likely outcomes. Let's examine the consequences of this probability model by focussing on the ith and jth elements in a birthday sequence, where $1 \leq i \neq j \leq n$. It makes for a better story if we refer to the ith birthday as "Alice's" and the jth as "Bob's."

Now since Bob's birthday is assumed to be independent of Alice's, it follows that whichever of the d birthdays Alice's happens to be, the probability that Bob has the same birthday $1/d$. Next, If we look at two other birthdays —call them "Carol's" and "Don's" —then whether Alice and Bob have matching birthdays has nothing to do with whether Carol and Don have matching birthdays. That is, the event that Alice and Bob have matching birthdays is independent of the event that Carol and Don have matching birthdays. In fact, for any set of non-overlapping couples, the events that a couple has matching birthdays are mutually independent.

In fact, it's pretty clear that the probability that Alice and Bob have matching birthdays remains $1/d$ whether or not Carol and *Alice* have matching birthdays. That is, the event that Alice and Bob match is also independent of Alice and Carol matching. In short, the set of all events in which a couple has macthing birthdays is *pairwise* independent, despite the overlapping couples. This will be important in Chapter 17 because pairwise independence will be enough to justify some conclusions about the expected number of matches. However, it's obvious that these matching birthday events are *not* mutually independent, not even 3-way independent: if Alice and Bob match and also Alice and Carol match, then Bob and Carol will match.

We could justify all these assertions of independence routinely using the four step method, but it's pretty boring, and we'll skip it.

It turns out that as long as the number of students is noticeably smaller than the number of possible birthdays, we can get a pretty good estimate of the birthday matching probabilities by *pretending* that the matching events are mutually independent. (An intuitive justification for this is that with only a small number of matching pairs, it's likely that none of the pairs overlap.) Then the probability of *no* matching birthdays would be the same as rth power of the probability that a couple does *not* have matching birthdays, where $r ::= \binom{n}{2}$ is the number of couples. That is, the probability of no matching birthdays would be

$$(1 - 1/d)^{\binom{n}{2}}. \tag{14.4}$$

Using the fact that $e^x > 1 + x$ for all x,[2] we would conclude that the probability of

[2]This approximation is obtained by truncating the Taylor series $e^{-x} = 1 - x + x^2/2! - x^3/3! + \cdots$.

no matching birthdays is at most

$$e^{-\frac{\binom{n}{2}}{d}}. \tag{14.5}$$

The matching birthday problem fits in here so far as a nice example illustrating pairwise and mutual independence. But it's actually not hard to justify the bound (14.5) without any pretence or any explicit consideration of independence. Namely, there are $d(d-1)(d-2)\cdots(d-(n-1))$ length n sequences of distinct birthdays. So the probability that everyone has a different birthday is:

$$\begin{aligned}
&\frac{d(d-1)(d-2)\cdots(d-(n-1))}{d^n}\\
&= \frac{d}{d}\cdot\frac{d-1}{d}\cdot\frac{d-2}{d}\cdots\frac{d-(n-1)}{d}\\
&= \left(1-\frac{0}{d}\right)\left(1-\frac{1}{d}\right)\left(1-\frac{2}{d}\right)\cdots\left(1-\frac{n-1}{d}\right)\\
&< e^0\cdot e^{-1/d}\cdot e^{-2/d}\cdots e^{-(n-1)/d} && \text{(since } 1+x<e^x\text{)}\\
&= e^{-\left(\sum_{i=1}^{n-1} i/d\right)}\\
&= e^{-(n(n-1)/2d)}\\
&= \text{the bound (14.5)}.
\end{aligned}$$

The approximation $e^{-x}\approx 1-x$ is pretty accurate when x is small.

For $n = 85$ and $d = 365$, (14.5) is less than $1/17,000$, which means the probability of having some pair of matching birthdays actually is more than $1-1/17,000 > 0.9999$. So it would be pretty astonishing if there were no pair of students in the class with matching birthdays.

For $d \leq n^2/2$, the probability of no match turns out to be asymptotically equal to the upper bound (14.5). For $d = n^2/2$ in particular, the probability of no match is asymptotically equal to $1/e$. This leads to a rule of thumb which is useful in many contexts in computer science:

The Birthday Principle

If there are d days in a year and $\sqrt{2d}$ people in a room, then the probability that two share a birthday is about $1 - 1/e \approx 0.632$.

For example, the Birthday Principle says that if you have $\sqrt{2 \cdot 365} \approx 27$ people in a room, then the probability that two share a birthday is about 0.632. The actual probability is about 0.626, so the approximation is quite good.

Among other applications, the Birthday Principle famously comes into play as the basis of “birthday attacks” that crack certain cryptographic systems.

Chapter 15

Random Processes

Random Walks are used to model situations in which an object moves in a sequence of steps in randomly chosen directions. For example in Physics, three-dimensional random walks are used to model Brownian motion and gas diffusion. In this chapter we'll examine two examples of random walks. First, we'll model gambling as a simple 1-dimensional random walk —a walk along a straight line. Then we'll explain how the Google search engine used random walks through the graph of world-wide web links to determine the relative importance of websites.

15.1 Gamblers' Ruin

EDITING NOTE: In the Mathematical literature, random walks are for some reason traditionally discussed in the context of some social vice. A one-dimensional random walk is often described as the path of a drunkard who randomly staggers left or right at each step. We'll examine one-dimensional random walks using the language of gambling. ■

a Suppose a gambler starts with an initial stake of n dollars and makes a sequence of \$1 bets. If he wins an individual bet, he gets his money back plus another \$1. If he loses, he loses the \$1.

We can model this scenario as a random walk between integer points on the reall line. The position on the line at any time corresponds to the gambler's cash-on-hand or *capital*. Walking one step to the right (left) corresponds to winning (losing) a \$1 bet and thereby increasing (decreasing) his capital by \$1. The gambler plays until either he is bankrupt or increases his capital to a target amount of T dollars. If he reaches his target, then he is called an overall *winner*, and his *profit*,

m, will be $T - n$ dollars. If his capital reaches zero dollars before reaching his target, then we say that he is "ruined" or *goes broke*. We'll assume that the gambler has the same probability, p, of winning each individual \$1 bet and that the bets are mutually independent. We'd like to find the probability that the gambler wins.

The gambler's situation as he proceeds with his \$1 bets is illustrated in Figure 15.1. The random walk has boundaries at 0 and T. If the random walk ever reaches either of these boundary values, then it terminates.

In a *fair game*, the gambler is equally likely to win or lose each bet, that is $p = 1/2$. The corresponding random walk is called *unbiased*. The gambler is more likely to win if $p > 1/2$ and less likely to win if $p < 1/2$; these random walks are called *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.

EDITING NOTE:

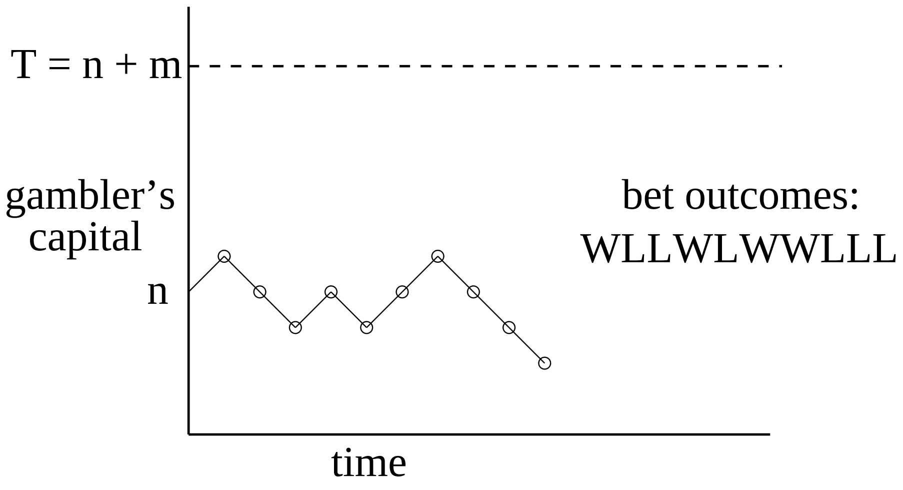


Figure 15.1: *This is a graph of the gambler's capital versus time for one possible sequence of bet outcomes. At each time step, the graph goes up with probability* p *and down with probability* $1 - p$*. The gambler continues betting until the graph reaches either 0 or* T*.*

Make this a pset problem

The Probability Space

Each random-walk game corresponds to a path like the one in Figure 15.1 that starts at the point $(n, 0)$. A winning path never touches the x axis and ends when it first touches the line $y = T$. Likewise, a losing path never touches the line $y = T$ and ends when it first touches the x axis.

figure causing errors omitted here

Any length k path can be characterized by the history of wins and losses on individual $1 bets, so we use a length k string of W's and L's to model a path, and assign probability $p^r q^{k-r}$ to a string that contains r W's. The *outcomes* in our sample space will be precisely those string corresponding to winning or losing walks, that is, when $r = 2k$.

What about the infinite walks in which the gambler plays forever, neither reaching his target nor going bankrupt? A recitation problem will show the probability of playing forever is zero, so we don't need to include any such outcomes in our sample space.

As a sanity check on this definition of the probability space, we should verify that the sum of the outcome probabilities is one, but we omit this calculation.

■

Let's begin by supposing the coin is fair, the gambler starts with 100 dollars, and he wants to double his money. That is, he plays until he goes broke or reaches a target of 200 dollars. Since he starts equidistant from his target and bankruptcy, it's clear by symmetry that his probability of winning in this case is 1/2.

We'll show below that starting with n dollars and aiming for a target of $T \geq n$ dollars, the probability the gambler reaches his target before going broke is n/T. For example, suppose he want to win the same $100, but instead starts out with $500. Now his chances are pretty good: the probability of his making the 100 dollars is $5/6$. And if he started with one million dollars still aiming to win $100 dollars he almost certain to win: the probability is $1M/(1M + 100) > .9999$.

So in the fair game, the larger the initial stake relative to the target, the higher the probability the gambler will win, which makes some intuitive sense. But note that although the gambler now wins nearly all the time, the game is still fair. When he wins, he only wins \$100; when he loses, he loses big: \$1M. So the gambler's average win is actually zero dollars.

EDITING NOTE:

Example 15.1.1. Suppose Albert starts with \$100, and Eric starts with \$10. They flip a fair coin, and every time a Head appears, Albert wins \$1 from Eric, and vice versa for Tails. They play this game until one person goes bankrupt. What is the probability of Albert winning?

This problem is identical to the Gambler's Ruin problem with $n = 100$ and $T = 100 + 10 = 110$. The probability of Albert winning is $100/110 = 10/11$, namely, the ratio of his wealth to the combined wealth. Eric's chances of winnning are $1/11$.

$\blacksquare$

Now suppose instead that the gambler chooses to play roulette in an American casino, always betting \$1 on red. A roulette wheel has 18 black numbers, 18 red numbers, and 2 green numbers, designed so that each number is equally likely to appear. So this game is slightly biased against the gambler: the probability of winning a single bet is $p = 18/38 \approx 0.47$. It's the two green numbers that slightly bias the bets and give the casino an edge. Still, the bets are almost fair, and you might expect that starting with \$500, the gambler has a reasonable chance of winning \$100 —the 5/6 probability of winning in the unbiased game surely gets reduced, but perhaps not too drastically.

Not so! The gambler's odds of winning \$100 making one dollar bets against the "slightly" unfair roulette wheel are less than 1 in 37,000. If that seems surprising, listen to this: *no matter how much money* the gambler has to start —\$5000, \$50,000, $\$5 \cdot 10^{12}$ —his odds are still less than 1 in 37,000 of winning a mere 100 dollars!

Moral: Don't play!

The theory of random walks is filled with such fascinating and counter-intuitive conclusions.

15.1.1 A Recurrence for the Probability of Winning

The probability the gambler wins is a function of his initial capital, n, his target, $T \geq n$, and the probability, p, that he wins an individual one dollar bet. Let's let p and T be fixed, and let w_n be the gambler's probabiliity of winning when his initial capital is n dollars. For example, w_0 is the probability that the gambler will win given that he starts off broke and w_T is the probability he will win if he starts off with his target amount, so clearly

$$w_0 = 0, \tag{15.1}$$

$$w_T = 1. \tag{15.2}$$

Otherwise, the gambler starts with n dollars, where $0 < n < T$. Consider the outcome of his first bet. The gambler wins the first bet with probability p. In this case, he is left with $n + 1$ dollars and becomes a winner with probability w_{n+1}. On the other hand, he loses the first bet with probability $q ::= 1-p$. Now he is left with $n-1$ dollars and becomes a winner with probability w_{n-1}. By the Total Probability Rule, he wins with probability $w_n = pw_{n+1} + qw_{n-1}$. Solving for w_{n+1} we have

$$w_{n+1} = \frac{w_n}{p} - rw_{n-1} \tag{15.3}$$

where

$$r ::= \frac{q}{p}.$$

This recurrence holds only for $n+1 \leq T$, but there's no harm in using (15.3) to define w_{n+1} for all $n+1 > 1$. Now, letting

$$W(x) ::= w_0 + w_1 x + w_2 x^2 + \cdots$$

be the generating function for the w_n, we derive from (15.3) and (15.1) using our generating function methods that

$$xW(x) = \frac{w_1 x}{(1-x)(1-rx)}, \tag{15.4}$$

so if $p \neq q$, then using partial fractions we can calculate that

EDITING NOTE:

$$W(x) = \frac{A}{1-x} + \frac{B}{1-(q/p)x}. \tag{15.5}$$

where $A = w_1/(1-(q/p))$

From (15.4) and (15.5), 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

■

$$W(x) = \frac{w_1}{r-1}\left(\frac{1}{1-rx} - \frac{1}{1-x}\right),$$

which implies

$$w_n = w_1 \frac{r^n - 1}{r-1}. \tag{15.6}$$

Now we can use (15.6) to solve for w_1 by letting $n = T$ to get

EDITING NOTE:

$$1 = w_T = \frac{w_1}{(q/p) - 1}\left(\left(\frac{q}{p}\right)^T - 1\right)$$

so ■

$$w_1 = \frac{r-1}{r^T - 1}.$$

Plugging this value of w_1 into (15.6), we finally arrive at the solution:

$$w_n = \frac{r^n - 1}{r^T - 1}. \tag{15.7}$$

The expression (15.7) for the probability that the Gambler wins in the biased game is a little hard to interpret. There is a simpler upper bound which is nearly tight when the gambler's starting capital is large and the game is biased *against* the gambler. Then both the numerator and denominator in the quotient in (15.7) are positive, and the quotient is less than one. This implies that

$$w_n < \frac{r^n}{r^T} = r^{T-n},$$

which proves:

Corollary 15.1.2. *In the Gambler's Ruin game with probability $p < 1/2$ of winning each individual bet, with initial capital, n, and target, T,*

$$\Pr\{\textit{the gambler is a winner}\} < \left(\frac{p}{q}\right)^{T-n} \tag{15.8}$$

The amount $T - n$ is called the Gambler's *intended profit*. So the gambler gains his intended profit before going broke with probability at most p/q raised to the intended-profit power. Notice that this upper bound does not depend on the gambler's starting capital, but only on his intended profit. This has the amazing consequence we announced above: *no matter how much money he starts with*, if he makes \$1 bets on red in roulette aiming to win \$100, the probability that he wins is less than

$$\left(\frac{18/38}{20/38}\right)^{100} = \left(\frac{9}{10}\right)^{100} < \frac{1}{37,648}.$$

The bound (15.8) 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.

EDITING NOTE:

The odds of winning a little money are not so bad. Applying the exact formula (15.7), 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 (15.7) only applies to biased walks, but the method above works just as well in getting a formula for the unbiased case (except that the partial fractions involve a repeated root). But it's simpler settle the fair case simply by taking the limit as r approaches 1 of (15.7). By L'Hopital's Rule this limit is n/T, as we claimed above.

15.1.2 Intuition

Why is the gambler so unlikely to make money when the game is slightly biased against him? Intuitively, there are two forces at work. First, the gambler's capital has random upward and downward *swings* due to runs of good and bad luck. Second, the gambler's capital will have a steady, downward *drift*, because the negative bias means an average loss of a few cents on each $1 bet. The situation is shown in Figure 15.2.

EDITING NOTE:

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, say, a billion dollars, then he is sure to play for a very long time, so at some point there should be a lucky, upward swing that puts him \$100 ahead. The problem is that his capital is steadily drifting downward. If the gambler does not have a lucky, upward swing early on, then he is doomed. After his capital drifts downward a few hundred dollars, he needs a huge upward swing to save himself. And such a huge swing is extremely improbable. As a rule of thumb, *drift dominates swings* in the long term.

EDITING NOTE:

We can quantify these drifts and swings. After k rounds for $k \leq \min(m, n)$, the number of wins by our player has a binomial distribution with parameters $p < 1/2$ and k. His expected win on any single bet is $p - q = 2p - 1$ dollars, so his expected capital is $n - k(1 - 2p)$. Now to be a winner, his actual number of wins must exceed the expected number by $m + k(1 - 2p)$. But we saw before that the binomial distribution has a standard deviation of only $\sqrt{kp(1-p)}$. So for the gambler to

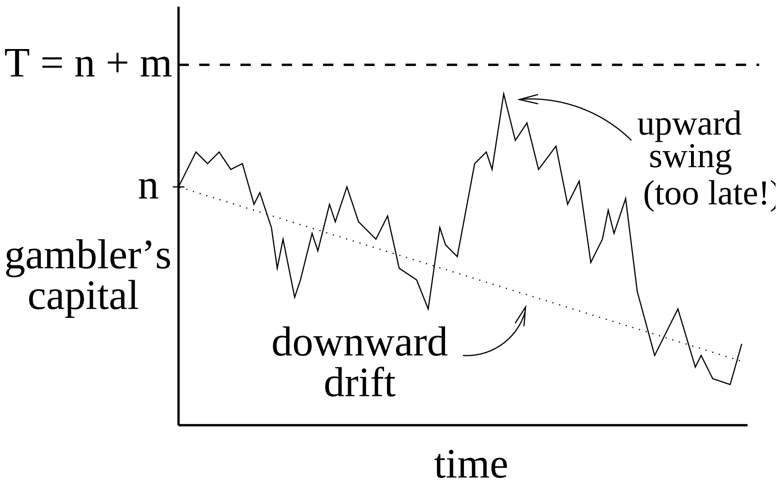


Figure 15.2: *In an unfair game, the gambler's capital swings randomly up and down, but steadily drifts downward. If the gambler does not have a winning swing early on, then his capital drifts downward, and later upward swings are insufficient to make him a winner.*

win, he needs his number of wins to deviate by

$$\frac{m + k(1-2p)}{\sqrt{kp(1-2p)}} = \Theta(\sqrt{k})$$

times its standard deviation. In our study of binomial tails we saw that this was extremely unlikely.

In a fair game, there is no drift; swings are the only effect. In the absence of downward drift, our earlier intuition is correct. If the gambler starts with a trillion dollars then almost certainly there will eventually be a lucky swing that puts him \$100 ahead.

If we start with \$10 and play to win only \$10 more, then the difference between the fair and unfair games is relatively small. We saw that the probability of winning is $1/2$ versus about $1/4$. Since swings of \$10 are relatively common, the game usually ends before the gambler's capital can drift very far. That is, the game does not last long enough for drift to dominate the swings.

How Long a Walk?

Now that we know the probability, w_n, that the gambler is a winner in both fair and unfair games, we consider how many bets he needs on average to either win or go broke.

Duration of a Biased Walk

Let Q be the number of bets the gambler makes until the game ends. Since the gambler's expected win on any bet is $2p-1$, Wald's Theorem should tell us that his game winnings, G, will have expectation $\mathrm{E}[Q](2p-1)$. That is,

$$\mathrm{E}[G] = (2p-1)\,\mathrm{E}[Q]\,, \tag{15.9}$$

In an unbiased game (15.9) is trivially true because both $2p-1$ and the expected overall winnings, $\mathrm{E}[G]$, are zero. On the other hand, in the unfair case, $2p-1 \neq 0$. Also, we know that

$$\mathrm{E}[G] = w_n(T-n) - (1-w_n)n = w_nT - n.$$

So assuming (15.9), we conclude

Theorem 15.1.3. *In the biased Gambler's Ruin game with initial capital, n, target, T, and probability, $p \neq 1/2$, of winning each bet,*

$$\mathrm{E}\left[\textit{number of bets till game ends}\right] = \frac{\Pr\{\textit{gambler is a winner}\}\, T - n}{2p-1}. \tag{15.10}$$

The only problem is that (15.9) is not a special case of Wald's Theorem because $G = \sum_{i=1}^{Q} G_i$ is not a sum of *nonnegative* variables: when the gambler loses the ith bet, the random variable G_i equals -1. However, this is easily dealt with.[1]

Example 15.1.4. If the gambler aims to profit \$100 playing roulette with n dollars

[1]The random variable $G_i + 1$ is nonnegative, and $\mathrm{E}\left[G_i + 1 \mid Q \geq i\right] = \mathrm{E}\left[G_i \mid Q \geq i\right] + 1 = 2p$, so by Wald's Theorem

$$\mathrm{E}\left[\sum_{i=1}^{Q}(G_i+1)\right] = 2p\,\mathrm{E}\left[Q\right]. \tag{15.11}$$

But

$$\begin{aligned} q1\,\mathrm{E}\left[\sum_{i=1}^{Q}(G_i+1)\right] &= \mathrm{E}\left[\sum_{i=1}^{Q}G_i + \sum_{i=1}^{Q}1\right] \\ &= \mathrm{E}\left[(\sum_{i=1}^{Q}G_i) + Q\right] \\ &= \mathrm{E}\left[\sum_{i=1}^{Q}G_i\right] + \mathrm{E}\left[Q\right] \\ &= \mathrm{E}\left[G\right] + \mathrm{E}\left[Q\right]. \end{aligned} \tag{15.12}$$

Now combining (15.11) and (15.12) confirms the truth of our assumption (15.9).

to start, he can expect to make $((n + 100)/37,648 - n)/(2(18/38) - 1) \approx 19n$ bets before the game ends. So he can enjoy playing for a good while before almost surely going broke.

Duration of an Unbiased Walk

This time, we need the more general approach of recurrences to handle the unbiased case. We consider the expected number of bets as a function of the gambler's initial capital. That is, for fixed p and T, let e_n be the expected number of bets until the game ends when the gambler's initial capital is n dollars. Since the game is over in no steps if $n = 0$ or T, the boundary conditions this time are $e_0 = e_T = 0$.

Otherwise, the gambler starts with n dollars, where $0 < n < T$. Now by the conditional expectation rule, the expected number of steps can be broken down into the expected number of steps given the outcome of the first bet weighted by the probability of that outcome. That is,

$$e_n = p\,\mathrm{E}[Q \mid \text{gambler wins first bet}] + q\,\mathrm{E}[Q \mid \text{gambler loses first bet}].$$

But after the gambler wins the first bet, his capital is $n+1$, so he can expect to make another e_{n+1} bets. That is,

$$\mathrm{E}\left[Q \mid \text{gambler wins first bet}\right] = 1 + e_{n+1},$$

and similarly,

$$\mathrm{E}\left[Q \mid \text{gambler loses first bet}\right] = 1 + e_{n-1}.$$

So we have

$$e_n = p(1 + e_{n+1}) + q(1 + e_{n-1}) = pe_{n+1} + qe_{n-1} + 1,$$

which yields the linear recurrence

$$e_{n+1} = \frac{e_n}{p} - \frac{q}{p}e_{n-1} - \frac{1}{p}.$$

For $p = q = 1/2$, this equation simplifies to

$$e_{n+1} = 2e_n - e_{n-1} - 2. \tag{15.13}$$

There is a general theory for solving linear recurrences like (15.13) in which the value at $n + 1$ is a linear combination of values at some arguments $k < n + 1$ plus another simple term—in this case plus the constant -2. This theory implies that

$$e_n = (T - n)n. \tag{15.14}$$

Fortunately, we don't need the general theory to *verify* this solution. Equation (15.14) can be verified routinely from the boundary conditions and (15.13) using strong induction on n.

So we have shown

Theorem 15.1.5. *In the unbiased Gambler's Ruin game with initial capital, n, and target, T, and probability, $p = 1/2$, of winning each bet,*

$$\mathrm{E}\left[\textit{number of bets till game ends}\right] = n(T - n). \tag{15.15}$$

Another way to phrase Theorem 15.1.5 is

$$\mathrm{E}\left[\text{number of bets till game ends}\right] = \text{initial capital} \cdot \text{intended profit}. \tag{15.16}$$

Now for example, we can conclude that if the gambler starts with \$10 dollars and plays until he is broke or ahead \$10, then $10 \cdot 10 = 100$ bets are required on average. If he starts with \$500 and plays until he is broke or ahead \$100, then the expected number of bets until the game is over is $500 \times 100 = 50,000$.

Notice that (15.16) is a very simple answer that cries out for an intuitive proof, but we have not found one.

■

EDITING NOTE:

Quit While You Are Ahead

Suppose that the gambler never quits while he is ahead. That is, he starts with $n > 0$ dollars, ignores any target T, but plays until he is flat broke. Then it turns out that if the game is not favorable, *i.e.*, $p \leq 1/2$, the gambler is sure to go broke. In particular, he is even sure to go broke in a "fair" game with $p = 1/2$.

If the game is favorable to the gambler, *i.e.*, $p > 1/2$, then there is a positive probability that the gambler will play forever. We'll leave a proof of this to Problem**??**.

Lemma 15.1.6. *If the gambler starts with one or more dollars and plays a fair game until* *he is broke, then he will go broke with probability 1.*

Proof. If the gambler has initial capital n and goes broke in a game without reaching a target T, then he would also go broke if he were playing and ignored the target. So the probability that he will lose if he keeps playing without stopping at any target T must be at least as large as the probability that he loses when he has a target $T > n$.

But we know that in a fair game, the probability that he loses is $1 - n/T$. This number can be made arbitrarily close to 1 by choosing a sufficiently large value of T. Hence, the probability of his losing while playing without any target has a lower bound arbitrarily close to 1, which means it must in fact be 1. ■

So even if the gambler starts with a million dollars and plays a perfectly fair game, he will eventually lose it all with probability 1. In fact, if the game is unfavorable, then Theorem 15.1.3 and Corollary 15.1.2 imply that his expected time to go broke is essentially proportional to his initial capital, *i.e.*, $\Theta(n)$.

But there is good news: if the game is fair, he can "expect" to play for a very long time before going broke; in fact, he can expect to play forever!

Lemma 15.1.7. *If the gambler starts with one or more dollars and plays a fair game until he goes broke, then his expected number of plays is infinite.*

Proof. Consider the gambler's ruin game where the gambler starts with initial capital n, and let u_n be the expected number of bets for the *unbounded* game to end. Also, choose any $T \geq n$, and as above, let e_n be the expected number of bets for the game to end when the gambler's target is T.

The unbounded game will have a larger expected number of bets compared to the bounded game because, in addition to the possibility that the gambler goes broke, in the bounded game there is also the possibility that the game will end when the gambler reaches his target, T. That is,

$$u_n \geq e_n.$$

So by (15.14),

$$u_n \geq n(T - n).$$

But $n \geq 1$, and T can be any number greater than or equal to n, so this lower bound on u_n can be arbitrarily large. This implies that u_n must be infinite.

Now by Lemma 15.1.6, with probability 1, the unbounded game ends when the gambler goes broke. So the expected time for the unbounded game to *end* is the *same* as the expected time for the gambler to *go broke*. Therefore, the expected time to go broke is infinite. ■

In particular, even if the gambler starts with just one dollar, his expected number of plays before going broke is infinite! Of course, this does not mean that it is likely he will play for long. For example, there is a 50% chance he will lose the very first bet and go broke right away.

Lemma 15.1.7 says that the gambler can "expect" to play forever, while Lemma 15.1.6 says that with probability 1 he will go broke. These Lemmas sound contradictory, but our analysis showed that they are not. A moral is that naive intuition about "expectation" is misleading when we consider limiting behavior according to the technical mathematical definition of expectation.

■

15.1.3 Problems

Class Problems

Homework Problems

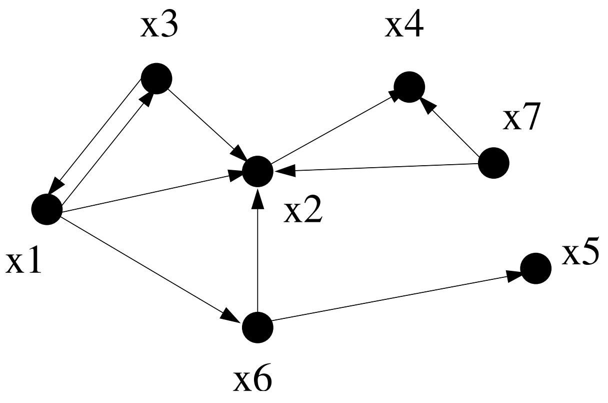
15.2 Random Walks on Graphs

EDITING NOTE:

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 vertices are the web pages with a directed edge from vertex x to vertex y if x has a link to y. For example, in the following graph the vertices $x_1, \ldots, x_n$ correspond to web pages and $x_i \to x_j$ is a directed edge when page x_i contains a hyperlink to page x_j.



The web graph is an enormous graph with many billions and probably even trillions of vertices. At first glance, this graph wouldn't seem to be very interesting. But in 1995, two students at Stanford, Larry Page and indexBrin, Sergey Sergey Brin realized that the structure of this graph could be very useful in building a search engine. Traditional document searching programs had been around for a long time and they worked in a fairly straightforward way. Basically, you would enter some search terms and the searching program would return all documents containing those terms. A relevance score might also be returned for each document based on the frequency or position that the search terms appeared in the document. For example, if the search term appeared in the title or appeared

100 times in a document, that document would get a higher score. So if an author wanted a document to get a higher score for certain keywords, he would put the keywords in the title and make it appear in lots of places. You can even see this today with some bogus web sites.

This approach works fine if you only have a few documents that match a search term. But on the web, there are billions of documents and millions of matches to a typical search.

For example, a few years ago a search on Google for "Math for Computer Science notes" gave 378,000 hits! How does Google decide which 10 or 20 to show first? It wouldn't be smart to pick a page that gets a high keyword score because it has "Math Math . . . Math" across the front of the document.

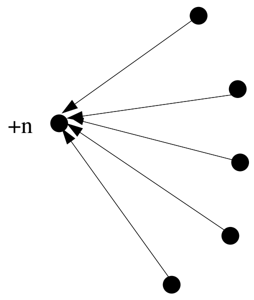
One way to get placed high on the list is to pay Google an advertising fees —and Google gets an enormous revenue stream from these fees. Of course an early listing is worth a fee only if an advertiser's target audience is attracted to the listing. But an audience does get attracted to Google listings because its ranking method is really good at determining the most relevant web pages. For example, Google demonstrated its accuracy in our case by giving first rank to our Fall 2002 Math for Computer Science notes on the MIT Open Courseware webpage `:-)` . So how did Google know to pick our class webage to be first out of $378,000$?

Well back in 1995, Larry and Sergey got the idea to allow the digraph structure of the web to determine which pages are likely to be the most important.

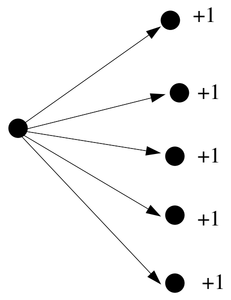
15.2.1 A First Crack at Page Rank

Looking at the web graph, any idea which vertex/page might be the best to rank 1st? Assume that all the pages match the search terms for now. Well, intuitively, we should choose x_2, since lots of other pages point to it. This leads us to their first idea: try defining the *page rank* of x to be the number of links pointing to x, that is, indegree(x). The idea is to think of web pages as voting for the most important page —the more votes, the better rank.

Of course, there are some problems with this idea. Suppose you wanted to have your page get a high ranking. One thing you could do is to create lots of dummy pages with links to your page.



There is another problem —a page could become unfairly influential by having lots of links to other pages it wanted to hype.



So this strategy for high ranking would amount to, "vote early, vote often," which is no good if you want to build a search engine that's worth paying fees for. So, admittedly, their original idea was not so great. It was better than nothing, but certainly not worth billions of dollars.

15.2.2 Random Walk on the Web Graph

But then Sergey and Larry thought some more and came up with a couple of improvements. Instead of just counting the indegree of a vertex, they considered the probability of being at each page after a long random walk on the web graph. In particular, they decided to model a user's web experience as following each link on a page with uniform probability. That is, they assigned each edge $x \to y$ of the web graph with a probability conditioned on being on page x:

$$\Pr\{\text{follow link } x \to y \mid \text{ at page } x\} ::= \frac{1}{\text{outdegree}(x)}.$$

The user experience is then just a random walk on the web graph.

For example, if the user is at page x, and there are three links from page x, then each link is followed with probability $1/3$.

We can also compute the probability of arriving at a particular page, y, by summing over all edges pointing to y. We thus have

$$\begin{aligned}\Pr\{\text{go to } y\} &= \sum_{\text{edges } x \to y} \Pr\{\text{follow link } x \to y \mid \text{at page } x\} \cdot \Pr\{\text{at page } x\} \\ &= \sum_{\text{edges } x \to y} \frac{\Pr\{\text{at } x\}}{\text{outdegree}(x)} \qquad (15.17)\end{aligned}$$

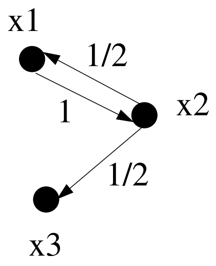
For example, in our web graph, we have

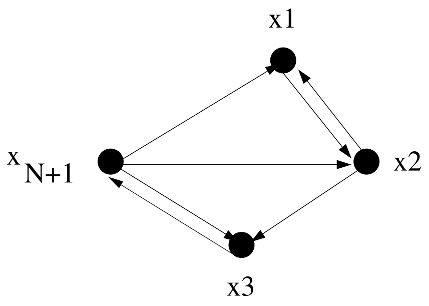
$$\Pr\{\text{go to } x_4\} = \frac{\Pr\{\text{at } x_7\}}{2} + \frac{\Pr\{\text{at } x_2\}}{1}.$$

One can think of this equation as x_7 sending half its probability to x_2 and the other half to x_4. The page x_2 sends all of its probability to x_4.

There's one aspect of the web graph described thus far that doesn't mesh with the user experience —some pages have no hyperlinks out. Under the current model, the user cannot escape these pages. In reality, however, the user doesn't fall off the end of the web into a void of nothingness. Instead, he restarts his web journey.

To model this aspect of the web, Sergey and Larry added a supervertex to the web graph and had every page with no hyperlinks point to it. Moreover, the supervertex points to every other vertex in the graph, allowing you to restart the walk from a random place. For example, below left is a graph and below right is the same graph after adding the supervertex x_{N+1}.





The addition of the supervertex also removes the possibility that the value $1/\text{outdegree}(x)$ might involve a division by zero.

15.2.3 Stationary Distribution & Page Rank

The basic idea of page rank is just a stationary distribution over the web graph,

EDITING NOTE: (there are some more details, but this is the main idea) ■

so let's define a stationary distribution.

Suppose each vertex is assigned a probability that corresponds, intuitively, to the likelihood that a random walker is at that vertex at a randomly chosen time.

We assume that the walk never leaves the vertices in the graph, so we require that

$$\sum_{\text{vertices } x} \Pr\{\text{at } x\} = 1. \tag{15.18}$$

Definition 15.2.1. An assignment of probabililties to vertices in a digraph is a *stationary distribution* if for all vertices x

$$\Pr\{\text{at } x\} = \Pr\{\text{go to } x \text{ at next step}\}$$

Sergey and Larry defined their page ranks to be a stationary distribution. They did this by solving the following system of linear equations: find a nonnegative number, $\text{PR}(x)$, for each vertex, x, such that

$$\text{PR}(x) = \sum_{\text{edges } y \to x} \frac{\text{PR}(y)}{\text{outdegree}(y)}, \tag{15.19}$$

corresponding to the intuitive equations given in (15.17). These numbers must also satisfy the additional constraint corresponding to (15.18):

$$\sum_{\text{vertices } x} \text{PR}(x) = 1. \tag{15.20}$$

So if there are n vertices, then equations (15.19) and (15.20) provide a system of $n + 1$ linear equations in the n variables, $\text{PR}(x)$. Note that constraint (15.20) is needed because the remaining constraints (15.19) could be satisfied by letting $\text{PR}(x) ::= 0$ for all x, which is useless.

Sergey and Larry were smart fellows, and they set up their page rank algorithm so it would always have a meaningful solution. Their addition of a supervertex ensures there is always a *unique* stationary distribution. Moreover, starting from *any* vertex and taking a sufficiently long random walk on the graph, the probability of being at each page will get closer and closer to the stationary distribution. Note that general digraphs without supervertices may have neither of these properties: there may not be a unique stationary distribution, and even when there is, there may be starting points from which the probabilities of positions during a random walk do not converge to the stationary distribution.

EDITING NOTE: 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 $x_i \rightarrow x_j$ exists, and $w_{ij} = 0$ otherwise. For example, in our last example with the 4-vertex graph (including the supervertex), 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

$$\sum_{1 \leq i \leq n} w_{ij} \cdot \text{PR}(x_i) = \sum_{i \mid x_i \rightarrow x_j} \frac{\text{PR}(x_i)}{\text{outdegree}(x_i)},$$

which is exactly the constraint corresponding to vertex x_j in (15.19).

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 vertices are billions of web pages is a daunting task. That's why Google is building power plants. Indeed, Larry and Sergey named their system Google after the number 10^{100} —which called a "googol" —to reflect the fact that the web graph is so enormous.

Anyway, now you can see how our Math for Computer Science notes ranked first out of 378,000 matches. Lots of other universities used our notes and presumably have links to the notes on the Open Courseware site, and the university sites themselves are legitimate, which ultimately leads to our notes getting a high page rank in the web graph.

15.2.4 Problems

Class Problems

Homework Problems

Exam Problems

Chapter 16

Random Variables

So far we focused on probabilities of events —that you win the Monty Hall game; that you have a rare medical condition, given that you tested positive; .... Now we focus on quantitative questions: *How many* contestants must play the Monty Hall game until one of them finally wins? ...*How long* will this condition last? *How much* will I lose playing silly Math games all day? Random variables are the mathematical tool for addressing such questions, and in this chapter we work out their basic properties, especially properties of their *mean* or *expected value*.

16.1 Random Variable Examples

Definition 16.1.1. A *random variable*, R, on a probability space is a total function whose domain is the sample space.

The codomain of R can be anything, but will usually be a subset of the real numbers. Notice that the name "random variable" is a misnomer; random variables are actually functions!

For example, suppose we toss three independent, unbiased coins. Let C be the number of heads that appear. Let $M = 1$ if the three coins come up all heads or all tails, and let $M = 0$ otherwise. Now every outcome of the three coin flips uniquely determines the values of C and M. For example, if we flip heads, tails, heads, then $C = 2$ and $M = 0$. If we flip tails, tails, tails, then $C = 0$ and $M = 1$. In effect, C counts the number of heads, and M indicates whether all the coins match.

Since each outcome uniquely determines C and M, we can regard them as functions mapping outcomes to numbers. For this experiment, the sample space is:

$$\mathcal{S} = \{HHH, HHT, HTH, HTT, THH, THT, TTH, TTT\}.$$

Now C is a function that maps each outcome in the sample space to a number as follows:

$$\begin{array}{rclcrcl} C(HHH) & = & 3 & \qquad & C(THH) & = & 2 \\ C(HHT) & = & 2 & & C(THT) & = & 1 \\ C(HTH) & = & 2 & & C(TTH) & = & 1 \\ C(HTT) & = & 1 & & C(TTT) & = & 0. \end{array}$$

Similarly, M is a function mapping each outcome another way:

$$\begin{array}{rclcrcl} M(HHH) & = & 1 & \qquad & M(THH) & = & 0 \\ M(HHT) & = & 0 & & M(THT) & = & 0 \\ M(HTH) & = & 0 & & M(TTH) & = & 0 \\ M(HTT) & = & 0 & & M(TTT) & = & 1. \end{array}$$

So C and M are random variables.

16.1.1 Indicator Random Variables

An *indicator random variable* is a random variable that maps every outcome to either 0 or 1. These are also called *Bernoulli variables*. The random variable M is an example. If all three coins match, then $M = 1$; otherwise, $M = 0$.

Indicator random variables are closely related to events. In particular, an indicator partitions the sample space into those outcomes mapped to 1 and those outcomes mapped to 0. For example, the indicator M partitions the sample space into two blocks as follows:

$$\underbrace{HHH \quad TTT}_{M=1} \quad \underbrace{HHT \quad HTH \quad HTT \quad THH \quad THT \quad TTH}_{M=0}.$$

In the same way, an event, E, partitions the sample space into those outcomes in E and those not in E. So E is naturally associated with an indicator random variable, I_E, where $I_E(p) = 1$ for outcomes $p \in E$ and $I_E(p) = 0$ for outcomes $p \notin E$. Thus, $M = I_F$ where F is the event that all three coins match.

16.1.2 Random Variables and Events

There is a strong relationship between events and more general random variables as well. A random variable that takes on several values partitions the sample space into several blocks. For example, C partitions the sample space as follows:

$$\underbrace{TTT}_{C=0} \quad \underbrace{TTH \quad THT \quad HTT}_{C=1} \quad \underbrace{THH \quad HTH \quad HHT}_{C=2} \quad \underbrace{HHH}_{C=3}.$$

Each block is a subset of the sample space and is therefore an event. Thus, we can regard an equation or inequality involving a random variable as an event. For example, the event that $C = 2$ consists of the outcomes THH, HTH, and HHT. The event $C \leq 1$ consists of the outcomes TTT, TTH, THT, and HTT.

Naturally enough, we can talk about the probability of events defined by properties of random variables. For example,

$$\begin{aligned}
\Pr\{C = 2\} &= \Pr\{THH\} + \Pr\{HTH\} + \Pr\{HHT\} \\
&= \frac{1}{8} + \frac{1}{8} + \frac{1}{8} = \frac{3}{8}.
\end{aligned}$$

EDITING NOTE:

As another example:

$$\begin{aligned}
\Pr\{M = 1\} &= \Pr\{TTT\} + \Pr\{HHH\} \\
&= \frac{1}{8} + \frac{1}{8} = \frac{1}{4}.
\end{aligned}$$

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.

■

16.1.3 Independence

The notion of independence carries over from events to random variables as well. Random variables R_1 and R_2 are *independent* iff for all x_1 in the codomain of R_1, and x_2 in the codomain of R_2, we have:

$$\Pr\{R_1 = x_1 \text{ AND } R_2 = x_2\} = \Pr\{R_1 = x_1\} \cdot \Pr\{R_2 = x_2\}.$$

As with events, we can formulate independence for random variables in an equivalent and perhaps more intuitive way: random variables R_1 and R_2 are independent if for all x_1 and x_2

$$\Pr\{R_1 = x_1 \mid R_2 = x_2\} = \Pr\{R_1 = x_1\}.$$

whenever the lefthand conditional probability is defined, that is, whenever $\Pr\{R_2 = x_2\} > 0$.

As an example, are C and M independent? Intuitively, the answer should be "no". The number of heads, C, completely determines whether all three coins match; that is, whether $M = 1$. But, to verify this intuition, we must find some $x_1, x_2 \in \mathbb{R}$ such that:

$$\Pr\{C = x_1 \text{ AND } M = x_2\} \neq \Pr\{C = x_1\} \cdot \Pr\{M = x_2\}.$$

One appropriate choice of values is $x_1 = 2$ and $x_2 = 1$. In this case, we have:

$$\Pr\{C = 2 \text{ AND } M = 1\} = 0 \neq \frac{1}{4} \cdot \frac{3}{8} = \Pr\{M = 1\} \cdot \Pr\{C = 2\}.$$

The first probability is zero because we never have exactly two heads ($C = 2$) when all three coins match ($M = 1$). The other two probabilities were computed earlier.

On the other hand, let H_1 be the indicator variable for event that the first flip is a Head, so

$$[H_1 = 1] = \{HHH, HTH, HHT, HTT\}.$$

Then H_1 is independent of M, since

$$\Pr\{M = 1\} = 1/4 = \Pr\{M = 1 \mid H_1 = 1\} = \Pr\{M = 1 \mid H_1 = 0\}$$

$$\Pr\{M = 0\} = 3/4 = \Pr\{M = 0 \mid H_1 = 1\} = \Pr\{M = 0 \mid H_1 = 0\}$$

This example is an instance of a simple lemma:

Lemma 16.1.2. *Two events are independent iff their indicator variables are independent.*

As with events, the notion of independence generalizes to more than two random variables.

Definition 16.1.3. Random variables $R_1, R_2, \ldots, R_n$ are *mutually independent* iff

$$\begin{aligned}&\Pr\{R_1 = x_1 \text{ AND } R_2 = x_2 \text{ AND } \cdots \text{ AND } R_n = x_n\}\\ = \quad &\Pr\{R_1 = x_1\} \cdot \Pr\{R_2 = x_2\} \cdots \Pr\{R_n = x_n\}.\end{aligned}$$

for all $x_1, x_2, \ldots, x_n$.

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 probabilities 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:

$$\Pr\{R_1 = 7 \text{ AND } R_7 = 9.1 \text{ AND } R_{23} = \pi\} = \Pr\{R_1 = 7\} \cdot \Pr\{R_7 = 9.1\} \cdot \Pr\{R_{23} = \pi\}.$$

16.2 Probability Distributions

A random variable maps outcomes to values, but random variables that show up for different spaces of outcomes wind up behaving in much the same way because they have the same probability of taking any given value. Namely, random variables on different probability spaces may wind up having the same probability density function.

Definition 16.2.1. Let R be a random variable with codomain V. The *probability* *density function (pdf)* of R is a function $\text{PDF}_R : V \to [0,1]$ defined by:

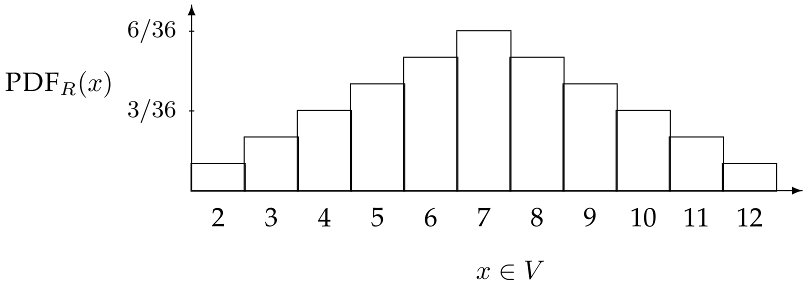
$$\text{PDF}_R(x) ::= \begin{cases} \Pr\{R = x\} & \text{if } x \in \text{range}(R), \\ 0 & \text{if } x \notin \text{range}(R). \end{cases}$$

A consequence of this definition is that

$$\sum_{x \in \text{range}(R)} \text{PDF}_R(x) = 1.$$

This follows because R has a value for each outcome, so summing the probabilities over all outcomes is the same as summing over the probabilities of each value in the range of R.

As an example, let's return to the experiment of rolling two fair, independent dice. As before, let T be the total of the two rolls. This random variable takes on values in the set $V = \{2, 3, \ldots, 12\}$. A plot of the probability density function is shown below:

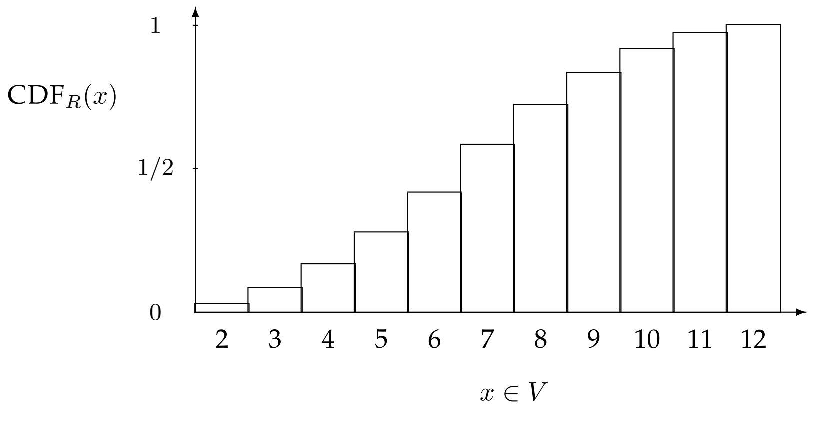


The lump in the middle indicates that sums close to 7 are the most likely. The total area of all the rectangles is 1 since the dice must take on exactly one of the sums in $V = \{2, 3, \ldots, 12\}$.

A closely-related idea is the *cumulative distribution function (cdf)* for a random variable R whose codomain is real numbers. This is a function $\text{CDF}_R : \mathbb{R} \to [0, 1]$ defined by:

$$\text{CDF}_R(x) = \Pr\{R \le x\}$$

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:

$$\begin{aligned}
\text{CDF}_R(x) &= \Pr\{R \le x\} \\
&= \sum_{y \le x} \Pr\{R = y\} \\
&= \sum_{y \le x} \text{PDF}_R(y)
\end{aligned}$$

In summary, $\text{PDF}_R(x)$ measures the probability that $R = x$ and $\text{CDF}_R(x)$ measures the probability that $R \le x$. 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.

EDITING NOTE: 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.

16.2.1 Bernoulli Distribution

Indicator random variables are perhaps the most common type because of their close association with events. The probability density function of an indicator random variable, B, is always

$$\mathrm{PDF}_B(0) = p$$

$$\mathrm{PDF}_B(1) = 1 - p$$

where $0 \leq p \leq 1$. The corresponding cumulative distribution function is:

$$\mathrm{CDF}_B(0) = p$$

$$\mathrm{CDF}_B(1) = 1$$

16.2.2 Uniform Distribution

A random variable that takes on each possible value with the same probability is called *uniform*. For example, the probability density function of a random variable

U that is uniform on the set $\{1, 2, \ldots, N\}$ is:

$$\text{PDF}_U(k) = \frac{1}{N}$$

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 $\{1, 2, \ldots, 6\}$.

16.2.3 The Numbers Game

Let's play a game! I have two envelopes. Each contains an integer in the range $0, 1, \ldots, 100$, and the numbers are distinct. To win the game, you must determine which envelope contains the larger number. To give you a fighting chance, I'll let you peek at the number in one envelope selected at random. Can you devise a strategy that gives you a better than 50% chance of winning?

For example, you could just pick an envelope at random and guess that it contains the larger number. But this strategy wins only 50% of the time. Your challenge is to do better.

So you might try to be more clever. Suppose you peek in the left envelope and see the number 12. Since 12 is a small number, you might guess that that other number is larger. But perhaps I'm sort of tricky and put small numbers in *both* envelopes. Then your guess might not be so good!

An important point here is that the numbers in the envelopes may *not* be random. I'm picking the numbers and I'm choosing them in a way that I think will defeat your guessing strategy. I'll only use randomization to choose the numbers if that serves *my* end: making you lose!

Intuition Behind the Winning Strategy

Amazingly, there is a strategy that wins more than 50% of the time, regardless of what numbers I put in the envelopes!

Suppose that you somehow knew a number x *between* my lower number and higher numbers. Now you peek in an envelope and see one or the other. If it is bigger than x, then you know you're peeking at the higher number. If it is smaller than x, then you're peeking at the lower number. In other words, if you know a number x between my lower and higher numbers, then you are certain to win the game.

The only flaw with this brilliant strategy is that you do *not* know x. Oh well.

But what if you try to *guess* x? There is some probability that you guess correctly. In this case, you win 100% of the time. On the other hand, if you guess incorrectly, then you're no worse off than before; your chance of winning is still 50%. Combining these two cases, your overall chance of winning is better than 50%!

Informal arguments about probability, like this one, often sound plausible, but do not hold up under close scrutiny. In contrast, this argument sounds completely implausible— but is actually correct!

Analysis of the Winning Strategy

For generality, suppose that I can choose numbers from the set $\{0, 1, \ldots, n\}$. Call the lower number L and the higher number H.

Your goal is to guess a number x between L and H. To avoid confusing equality cases, you select x at random from among the half-integers:

$$\left\{\frac{1}{2},\ 1\frac{1}{2},\ 2\frac{1}{2},\ \ldots,\ n-\frac{1}{2}\right\}$$

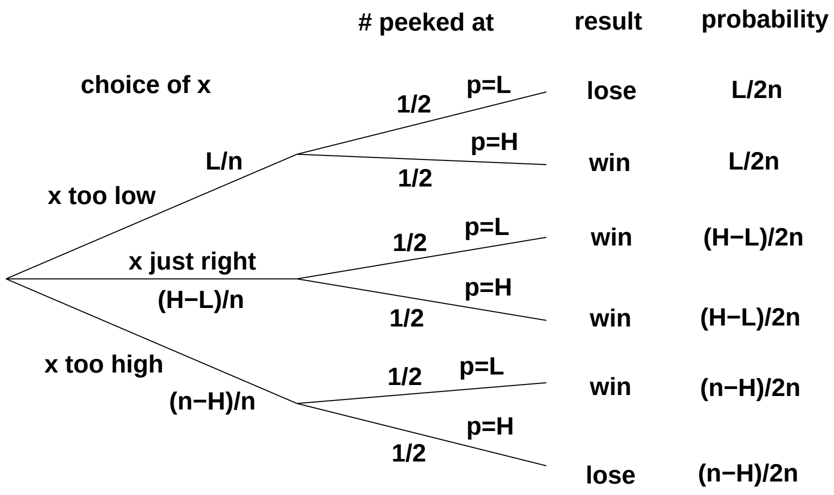
But what probability distribution should you use?

The uniform distribution turns out to be your best bet. An informal justification is that if I figured out that you were unlikely to pick some number— say $50\frac{1}{2}$— then I'd always put 50 and 51 in the evelopes. Then you'd be unlikely to pick an x between L and H and would have less chance of winning.

After you've selected the number x, you peek into an envelope and see some number p. If $p > x$, then you guess that you're looking at the larger number. If $p < x$, then you guess that the other number is larger.

All that remains is to determine the probability that this strategy succeeds. We can do this with the usual four step method and a tree diagram.

Step 1: Find the sample space. You either choose x too low ($< L$), too high ($> H$), or just right ($L < x < H$). Then you either peek at the lower number ($p = L$) or the higher number ($p = H$). This gives a total of six possible outcomes.



Step 2: Define events of interest. The four outcomes in the event that you win are marked in the tree diagram.

Step 3: Assign outcome probabilities. First, we assign edge probabilities. Your guess x is too low with probability L/n, too high with probability $(n - H)/n$, and just right with probability $(H - L)/n$. Next, you peek at either the lower or higher number with equal probability. Multiplying along root-to-leaf paths gives the outcome probabilities.

Step 4: Compute event probabilities. The probability of the event that you win is the sum of the probabilities of the four outcomes in that event:

$$\begin{aligned}
\Pr\{\text{win}\} &= \frac{L}{2n} + \frac{H-L}{2n} + \frac{H-L}{2n} + \frac{n-H}{2n} \\
&= \frac{1}{2} + \frac{H-L}{2n} \\
&\geq \frac{1}{2} + \frac{1}{2n}
\end{aligned}$$

The final inequality relies on the fact that the higher number H is at least 1 greater than the lower number L since they are required to be distinct.

Sure enough, you win with this strategy more than half the time, regardless of the numbers in the envelopes! For example, if I choose numbers in the range $0, 1, \ldots, 100$, then you win with probability at least $\frac{1}{2} + \frac{1}{200} = 50.5\%$. Even better, if I'm allowed only numbers in the range $0, \ldots, 10$, then your probability of winning rises to 55%! By Las Vegas standards, those are great odds!

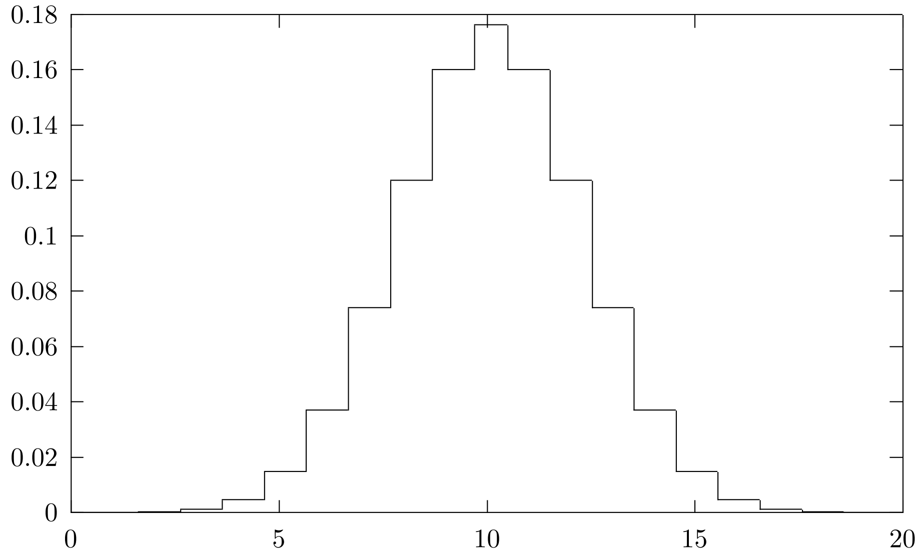
16.2.4 Binomial Distribution

The *binomial distribution* plays an important role in Computer Science as it does in most other sciences. The standard example of a random variable with a binomial distribution is the number of heads that come up in n independent flips of a coin; call this random variable H_n. If the coin is fair, then H_n has an *unbiased binomial density function*:

$$\text{PDF}_{H_n}(k) = \binom{n}{k} 2^{-n}.$$

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*.



In many fields, including Computer Science, probability analyses come down to getting small bounds on the tails of the binomial distribution. In the context of a problem, this typically means that there is very small probability that something *bad* happens, which could be a server or communication link overloading or a randomized algorithm running for an exceptionally long time or producing the wrong result.

As an example, we can calculate the probability of flipping at most 25 heads in 100 tosses of a fair coin and see that it is very small, namely, less than 1 in 3,000,000.

EDITING NOTE: Add calculation that the ratio of the $k-1$st and kth terms for $k \leq 25$ is less than 1/4(?), so the probability of $< k$ heads is less than 1/2 the prob of exactly k heads. ■

In fact, the tail of the distribution falls off so rapidly that the probability of flipping exactly 25 heads is nearly twice the probability of flipping fewer than 25 heads! That is, the probability of flipping exactly 25 heads —small as it is —is still nearly twice as large as the probability of flipping exactly 24 heads *plus* the probability of flipping exactly 23 heads *plus* . . . the probability of flipping no heads.

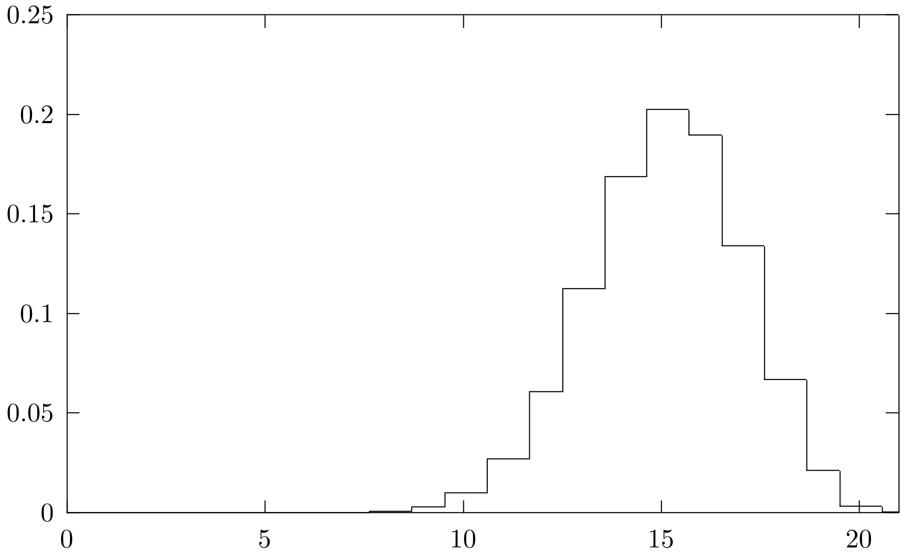
The General Binomial Distribution

Now let J be the number of heads that come up on n independent coins, each of which is heads with probability p. Then J has a *general binomial density function*:

$$\text{PDF}_J(k) = \binom{n}{k} p^k (1-p)^{n-k}.$$

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 $\mathrm{PDF}_J(k)$ corresponding to flipping $n = 20$ independent coins that are heads with probabilty $p = 0.75$. The graph shows that we are most likely to get around $k = 15$ heads, as you might expect. Once again, the probability falls off quickly for larger and smaller values of k.



EDITING NOTE:

Approximating the Binomial Density Function

Computing the general binomial density function is daunting if not impossible when n is up in the thousands. Fortunately, there is an approximate closed-form formula for this function based on an approximation for the binomial coefficient. In the formula, k is replaced by αn where α is a number between 0 and 1.

Lemma 16.2.2.

$$\binom{n}{\alpha n} \leq \frac{2^{nH(\alpha)}}{\sqrt{2\pi\alpha(1-\alpha)n}}$$

where $H(\alpha)$ is the famous entropy function*:*

$$H(\alpha) ::= \alpha \log_2 \frac{1}{\alpha} + (1-\alpha)\log_2 \frac{1}{1-\alpha}$$

The graph of H is shown in Figure 16.1.

The upper bound(16.2.2) on the binomial coefficient is tight enough to serve as an excellent approximation. We'll skip its derivation, which consists of plugging in Stirling's formula for the factorials in the binomial coefficient and then simplifying.

Now let's plug this formula into the general binomial density function. The

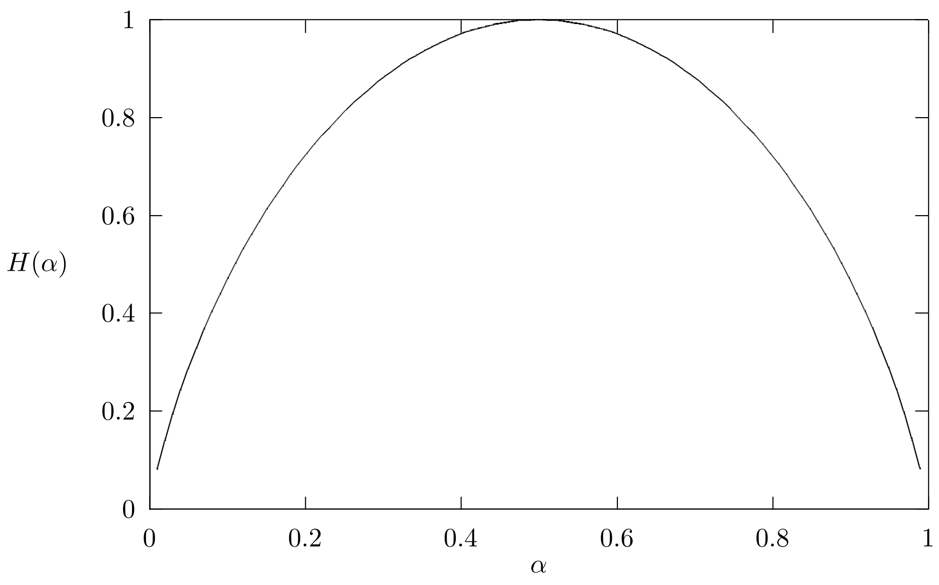


Figure 16.1: The Entropy Function

probability of flipping αn heads in n tosses of a coin that comes up heads with probability p is:

$$\text{PDF}_J(\alpha n) \leq \frac{2^{nH(\alpha)}}{\sqrt{2\pi\alpha(1-\alpha)n}} \cdot p^{\alpha n}(1-p)^{(1-\alpha)n} \tag{16.1}$$

This formula is ugly as a bowling shoe, but is useful because it's easy to evaluate. 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 $p = 1/2$ and $H(1/2) = 1$ into (16.1) gives:

$$\begin{aligned}
\mathrm{PDF}_J(\alpha n) &\le \frac{2^{nH(1/2)}}{\sqrt{2\pi(1/2)(1-(1/2))n}} \cdot 2^{-n} \\
&= \sqrt{\frac{2}{\pi n}}
\end{aligned}$$

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%.

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* αn heads is given by the cumulative binomial distribution function:

$$\mathrm{CDF}_J(\alpha n) = \Pr\{J \le \alpha n\} = \sum_{i=0}^{\alpha n} \mathrm{PDF}_J(i) \tag{16.2}$$

We can bound this sum by bounding the ratio of successive terms. This yields a geometric sum from 0 to $\mathrm{PDF}_J(\alpha n)$ that bounds (16.2). Then applying the formula for a geometric sum gives

$$\mathrm{CDF}_J(\alpha n) \leq \frac{1-\alpha}{1-\alpha/p} \cdot \mathrm{PDF}_J(\alpha n), \tag{16.3}$$

which holds providing $\alpha < p$. This is all we need, since we already have the bound (16.1) for $\mathrm{PDF}_J(\alpha n)$.

It would be awkward to evaluate (16.3) 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 $n = 100$:

$$\mathrm{CDF}_J\left(\frac{n}{4}\right) \leq \frac{1-(1/4)}{1-(1/4)/(1/2)} \cdot \mathrm{PDF}_J\left(\frac{n}{4}\right) \leq \frac{3}{2} \cdot 1.913 \cdot 10^{-7}.$$

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.

■

16.2.5 Problems

Class Problems

Homework Problems

16.3 Average & Expected Value

The *expectation* of a random variable is its average value, where each value is weighted according to the probability that it comes up. The expectation is also called the *expected value* or the *mean* of the random variable.

For example, suppose we select a student uniformly at random from the class, and let R be the student's quiz score. Then $\mathrm{E}[R]$ is just the class average —the first thing everyone wants to know after getting their test back! For similar reasons, the first thing you usually want to know about a random variable is its expected value.

Definition 16.3.1.

$$\mathrm{E}[R] ::= \sum_{x \in \text{range}(R)} x \cdot \Pr\{R = x\} \tag{16.4}$$

$$= \sum_{x \in \text{range}(R)} x \cdot \text{PDF}_R(x).$$

Let's work through an example. Let R be the number that comes up on a fair, six-sided die. Then by (16.4), the expected value of R is:

$$\begin{aligned}\mathrm{E}[R] &= \sum_{k=1}^{6} k \cdot \frac{1}{6} \\ &= 1 \cdot \frac{1}{6} + 2 \cdot \frac{1}{6} + 3 \cdot \frac{1}{6} + 4 \cdot \frac{1}{6} + 5 \cdot \frac{1}{6} + 6 \cdot \frac{1}{6} \\ &= \frac{7}{2}\end{aligned}$$

This calculation shows that the name "expected value" is a little misleading; the random variable might *never* actually take on that value. You don't ever expect to roll a $3\frac{1}{2}$ on an ordinary die!

There is an even simpler formula for expectation:

Theorem 16.3.2. *If R is a random variable defined on a sample space, $\mathcal{S}$, then*

$$\mathrm{E}\left[R\right] = \sum_{\omega \in \mathcal{S}} R(\omega) \Pr\left\{\omega\right\} \tag{16.5}$$

The proof of Theorem 16.3.2, like many of the elementary proofs about expectation in this chapter, follows by judicious regrouping of terms in the defining sum (16.4):

Proof.

$$
\begin{aligned}
\mathrm{E}[R] &::= \sum_{x \in \mathrm{range}(R)} x \cdot \Pr\{R = x\} && \text{(Def 16.3.1 of expectation)} \\
&= \sum_{x \in \mathrm{range}(R)} x \left(\sum_{\omega \in [R=x]} \Pr\{\omega\} \right) && \text{(def of } \Pr\{R = x\}\text{)} \\
&= \sum_{x \in \mathrm{range}(R)} \sum_{\omega \in [R=x]} x \Pr\{\omega\} && \text{(distributing } x \text{ over the inner sum)} \\
&= \sum_{x \in \mathrm{range}(R)} \sum_{\omega \in [R=x]} R(\omega) \Pr\{\omega\} && \text{(def of the event } [R = x]\text{)} \\
&= \sum_{\omega \in \mathcal{S}} R(\omega) \Pr\{\omega\}
\end{aligned}
$$

The last equality follows because the events $[R = x]$ for $x \in \mathrm{range}(R)$ partition the sample space, $\mathcal{S}$, so summing over the outcomes in $[R = x]$ for $x \in \mathrm{range}(R)$ is the same as summing over $\mathcal{S}$. ■

In general, the defining sum (16.4) 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 (16.5)is sometimes easier to use in proofs about expectation.

16.3.1 Expected Value of an Indicator Variable

The expected value of an indicator random variable for an event is just the probability of that event.

Lemma 16.3.3. *If I_A is the indicator random variable for event A, then*

$$\mathrm{E}[I_A] = \Pr\{A\}.$$

Proof.

$$\begin{aligned}
\mathrm{E}[I_A] &= 1 \cdot \Pr\{I_A = 1\} + 0 \cdot \Pr\{I_A = 0\} \\
&= \Pr\{I_A = 1\} \\
&= \Pr\{A\}. && \text{(def of } I_A\text{)}
\end{aligned}$$

■

For example, if A is the event that a coin with bias p comes up heads, $\mathrm{E}[I_A] = \Pr\{I_A = 1\} = p$.

16.3.2 Conditional Expectation

Just like event probabilities, expectations can be conditioned on some event.

Definition 16.3.4. The *conditional expectation*, $\mathrm{E}[R \mid A]$, of a random variable, R, given event, A, is:

$$\mathrm{E}[R \mid A] ::= \sum_{r \in \text{range}(R)} r \cdot \Pr\{R = r \mid A\}. \tag{16.6}$$

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 (16.6),

$$\mathrm{E}[R \mid R \geq 4] = \sum_{i=1}^{6} i \cdot \Pr\{R = i \mid R \geq 4\} = 1 \cdot 0 + 2 \cdot 0 + 3 \cdot 0 + 4 \cdot \tfrac{1}{3} + 5 \cdot \tfrac{1}{3} + 6 \cdot \tfrac{1}{3} = 5.$$

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 16.3.5. *Let $A_1, A_2, \ldots$ be a partition of the sample space. Then*

Rule (Law of Total Expectation)**.**

$$\mathrm{E}[R] = \sum_i \mathrm{E}[R \mid A_i] \Pr\{A_i\}.$$

Proof.

$$\begin{aligned}
\mathrm{E}[R] &::= \sum_{r \in \mathrm{range}(R)} r \cdot \Pr\{R = r\} && \text{(Def 16.3.1 of expectation)} \\
&= \sum_r r \cdot \sum_i \Pr\{R = r \mid A_i\} \Pr\{A_i\} && \text{(Law of Total Probability)} \\
&= \sum_r \sum_i r \cdot \Pr\{R = r \mid A_i\} \Pr\{A_i\} && \text{(distribute constant } r\text{)} \\
&= \sum_i \sum_r r \cdot \Pr\{R = r \mid A_i\} \Pr\{A_i\} && \text{(exchange order of summation)} \\
&= \sum_i \Pr\{A_i\} \sum_r r \cdot \Pr\{R = r \mid A_i\} && \text{(factor constant } \Pr\{A_i\}\text{)} \\
&= \sum_i \Pr\{A_i\} \mathrm{E}[R \mid A_i]. && \text{(Def 16.3.4 of cond. expectation)}
\end{aligned}$$

■

16.3.3 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 $\text{E}[C]$. 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 (16.4) for expectation, we have

$$\begin{aligned}
\text{E}[C] &= \sum_{i \in \mathbb{N}} i \cdot \Pr\{R = i\} \\
&= \sum_{i \in \mathbb{N}^+} i(1-p)^{i-1}p \\
&= p \sum_{i \in \mathbb{N}^+} i(1-p)^{i-1} \\
&= p \frac{1}{(1-(1-p))^2} && \text{(by (13.1))} \\
&= \frac{1}{p}
\end{aligned}$$

A simple alternative derivation that does not depend on the formula (13.1) (which you remembered, right?) is based on conditional expectation. Given that the computer crashes in the first hour, the expected number of hours to the first crash is obviously 1! On the other hand, given that the computer does not crash in the first hour, then the expected total number of hours till the first crash is the expectation of one plus the number of additional hours to the first crash. So,

$$\mathrm{E}[C] = p \cdot 1 + (1-p)\,\mathrm{E}[C+1] = p + \mathrm{E}[C] - p\,\mathrm{E}[C] + 1 - p,$$

from which we immediately calculate that $\mathrm{E}[C] = 1/p$.

EDITING NOTE: There is a useful trick for calculating expectations of nonegative integer valued variables:

Lemma 16.3.6. *If R is a nonegative integer-valued random variable, then:*

$$\mathrm{E}[R] = \sum_{i \in \mathbb{N}} \Pr\{R > i\} \tag{16.7}$$

Proof. Consider the sum:

$$\begin{array}{cccccccc}
\Pr\{R=1\} & + & \Pr\{R=2\} & + & \Pr\{R=3\} & + & \cdots \\
 & + & \Pr\{R=2\} & + & \Pr\{R=3\} & + & \cdots \\
 & & & + & \Pr\{R=3\} & + & \cdots \\
 & & & & & + & \cdots
\end{array}$$

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 (16.4). On the other hand, the successive rows sum to $\Pr\{R > 0\}, \Pr\{R > 1\}, \Pr\{R > 2\}, \ldots$. 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 (16.7) gives:

$$\begin{aligned}
\mathrm{E}[C] &= \sum_{i \in \mathbb{N}} (1-p)^i \\
&= \frac{1}{1-(1-p)} && \text{(sum of geometric series)} \\
&= \frac{1}{p}
\end{aligned}$$

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 $1/p$.

■

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.

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?

16.3.4 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 16.3.7. *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 (16.5)

$$\begin{aligned}
\mathrm{E}[T] &= \sum_{\omega\in\mathcal{S}} T(\omega)\cdot\Pr\{\omega\} && \text{(Theorem 16.3.2)}\\
&= \sum_{\omega\in\mathcal{S}} (R_1(\omega)+R_2(\omega))\cdot\Pr\{\omega\} && \text{(def of } T\text{)}\\
&= \sum_{\omega\in\mathcal{S}} R_1(\omega)\Pr\{\omega\} + \sum_{\omega\in\mathcal{S}} R_2(\omega)\Pr\{\omega\} && \text{(rearranging terms)}\\
&= \mathrm{E}[R_1] + \mathrm{E}[R_2]. && \text{(Theorem 16.3.2)}
\end{aligned}$$

■

A small extension of this proof, which we leave to the reader, implies

Theorem 16.3.8 (Linearity of Expectation)**.** *For random variables R_1, R_2 and constants $a_1, a_2 \in \mathbb{R}$,*

$$\mathrm{E}[a_1R_1 + a_2R_2] = a_1\,\mathrm{E}[R_1] + a_2\,\mathrm{E}[R_2].$$

In other words, expectation is a linear function. A routine induction extends the result to more than two variables:

Corollary 16.3.9. *For any random variables $R_1, \ldots, R_k$ and constants $a_1, \ldots, a_k \in \mathbb{R}$,*

$$\mathrm{E}\left[\sum_{i=1}^{k} a_iR_i\right] = \sum_{i=1}^{k} a_i\,\mathrm{E}[R_i].$$

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.

EDITING NOTE: Even when the random variables *are* independent, we know from previous experience that proving independence requires a lot of work. ■

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:

$$\mathrm{E}[R_1 + R_2] = \mathrm{E}[R_1] + \mathrm{E}[R_2] = 3.5 + 3.5 = 7.$$

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 each individual die remainw 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!

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:

$$G = G_1 + G_2 + \cdots + G_n. \tag{16.8}$$

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 16.3.3. Now we can take the expected value of both sides of equation (16.8) 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!

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 16.3.3, 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.

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:

blue green green red blue orange blue orange 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:

$$\begin{aligned}
\mathrm{E}[T] &= \mathrm{E}[X_0 + X_1 + \cdots + X_{n-1}] \\
&= \mathrm{E}[X_0] + \mathrm{E}[X_1] + \cdots + \mathrm{E}[X_{n-1}] \\
&= \frac{n}{n-0} + \frac{n}{n-1} + \cdots + \frac{n}{3} + \frac{n}{2} + \frac{n}{1} \\
&= n\left(\frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{3} + \frac{1}{2} + \frac{1}{1}\right) \\
& n\left(\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n-1} + \frac{1}{n}\right) \\
&= nH_n \sim n \ln n.
\end{aligned}$$

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:

$$365H_{365} = 2364.6\ldots$$

EDITING NOTE: unedited from F02

Let A_i be the event that coin i comes up heads. Since the coin is fair, $\Pr\{A_i\} = 1/2$. Since there are N coins in all, there are N such events. By linearity of expectation (Theorem 16.3.9), the expected number of events that occur —the number of coins that come up heads —is $N(1/2) = N/2$.

Let's try to solve the same problem the hard way. In this case, assume that the coins are fair. Let the random variable R be the number of heads. We want to compute the expected value of R.

$$\begin{aligned} \mathrm{E}[R] &= \sum_{i=0}^{N} i \cdot \Pr\{R = i\} \\ &= \sum_{i=0}^{N} i \binom{N}{i} 2^{-N} \end{aligned}$$

The first equation follows from the definition of expectation. In the second step, we evaluate $\Pr\{R = i\}$. An outcome of tossing the N coins can be represented by a length N sequence of H's and T's. An H in position i indicates that the ith coin is heads, and a T indicates that the ith coin is tails. The sample space consists of all 2^N such sequences. The outcomes are equiprobable, and so each has probability

2^{-N}. The number of outcomes with exactly i heads is the number of length N sequences with i H's, which is $\binom{N}{i}$. Therefore, $\Pr\{R = i\} = \binom{N}{i}2^{-N}$.

The answer from linearity of expectation and from the hard way must be the same, so we can equate the two results to obtain a neat identity.[1]

$$\begin{aligned} \sum_{i=0}^{N} i\binom{N}{i}2^{-N} &= \frac{N}{2} \\ \sum_{i=0}^{N} i\binom{N}{i} &= N2^{N-1} \end{aligned}$$

The expected number of heads is $N/2$, even if some coins are glued together.

We can extend this reasoning to n tosses of a coin with probability p of a head, rather than $1/2$. If we do this, we get the generalized combinatorial identity:

$$\sum_{i=0}^{N} i\binom{N}{i}p^i(1-p)^{N-i} = Np$$

Here, the p^i factor gives the probabilities for the heads and the $(1-p)^{N-i}$ factor gives the probabilities for the tails. The right-hand side is the sum of N terms, each giving the probability of a particular A_i, which is p. The total is Np. For example,

[1]The identity also has a simple combinatorial proof given in Problem **??**.

consider an ordinary die. Let A_1 be the event that the value is odd, A_2 the event that the value is 1, 2, or 3, and A_3 the event that the value is 4, 5, or 6. These events are not mutually independent. However, the expected number of these events that occur is still obtainable by adding $\Pr\{A_1\} + \Pr\{A_2\} + \Pr\{A_3\}$, which yields 3/2.

The Number-Picking Game

Here is a game that you and I could play that reveals a strange property of expectation.

First, you think of a probability density function on the natural numbers. Your distribution can be absolutely anything you like. For example, you might choose a uniform distribution on $1, 2, \ldots, 6$, like the outcome of a fair die roll. Or you might choose a binomial distribution on $0, 1, \ldots, n$. You can even give every natural number a non-zero probability, provided that the sum of all probabilities is 1.

Next, I pick a random number z according to your distribution. Then, you pick a random number y_1 according to the same distribution. If your number is bigger than mine ($y_1 > z$), then the game ends. Otherwise, if our numbers are equal or mine is bigger ($z \geq y_1$), then you pick a new number y_2 with the same distribution, and keep picking values y_3, y_4, etc. until you get a value that is strictly bigger than my number, z. What is the expected number of picks that you must make?

Certainly, you always need at least one pick, so the expected number is greater than one. An answer like 2 or 3 sounds reasonable, though one might suspect that the answer depends on the distribution. Let's find out whether or not this intuition is correct.

The number of picks you must make is a natural-valued random variable, so from formula (16.7) we have:

$$\mathrm{E}\left[\text{\# picks by you}\right] = \sum_{k \in \mathbb{N}} \Pr\left\{(\text{\# picks by you}) > k\right\} \tag{16.9}$$

Suppose that I've picked my number z, and you have picked k numbers $y_1, y_2, \ldots, y_k$. There are two possibilities:

- If there is a unique largest number among our picks, then my number is as likely to be it as any one of yours. So with probability $1/(k+1)$ my number is larger than all of yours, and you must pick again.

- Otherwise, there are several numbers tied for largest. My number is as likely to be one of these as any of your numbers, so with probability greater than $1/(k+1)$ you must pick again.

In both cases, with probability at least $1/(k+1)$, you need more than k picks to beat me. In other words:

$$\Pr\{(\text{\# picks by you}) > k\} \geq \frac{1}{k+1} \tag{16.10}$$

This suggests that in order to minimize your rolls, you should choose a distribution such that ties are very rare. For example, you might choose the uniform distribution on $\{1, 2, \ldots, 10^{100}\}$. In this case, the probability that you need more than k picks to beat me is very close to $1/(k+1)$ for moderate values of k. For example, the probability that you need more than 99 picks is almost exactly 1%. This sounds very promising for you; intuitively, you might expect to win within a reasonable number of picks on average!

Unfortunately for intuition, there is a simple proof that the expected number of picks that you need in order to beat me is *infinite*, regardless of the distribution!

Let's plug (16.10) into (16.9):

$$\begin{aligned}\text{E}\left[\#\text{ picks by you}\right] &= \sum_{k\in\mathbb{N}} \frac{1}{k+1}\\ &= \infty\end{aligned}$$

This phenomenon can cause all sorts of confusion! For example, suppose you have a communication network where each packet of data has a $1/k$ chance of being delayed by k or more steps. This sounds good; there is only a 1% chance of being delayed by 100 or more steps. But the *expected* delay for the packet is actually infinite!

There is a larger point here as well: not every random variable has a well-defined expectation. This idea may be disturbing at first, but remember that an expected value is just a weighted average. And there are many sets of numbers that have no conventional average either, such as:

$$\{1, -2, 3, -4, 5, -6, \dots\}$$

Strictly speaking, we should qualify virtually all theorems involving expectation with phrases such as "...provided all expectations exist." But we're going to leave that assumption implicit.

Random variables with infinite or ill-defined expectations are more the exception than the rule, but they do creep in occasionally.

■

16.4 Expectation of a Quotient

16.4.1 A RISC Paradox

The following data is taken from a paper by some famous professors. They wanted to show that programs on a RISC processor are generally shorter than programs on a CISC processor. For this purpose, they applied a RISC compiler and then a CISC compiler to some benchmark source programs and made a table of compiled program lengths.

Benchmark	RISC	CISC	CISC/RISC
E-string search	150	120	0.8
F-bit test	120	180	1.5
Ackerman	150	300	2.0
Rec 2-sort	2800	1400	0.5
Average			1.2

Each row contains the data for one benchmark. The numbers in the second and third columns are program lengths for each type of compiler. The fourth column contains the ratio of the CISC program length to the RISC program length. Averaging this ratio over all benchmarks gives the value 1.2 in the lower right. The authors conclude that "CISC programs are 20% longer on average".

However, some critics of their paper took the same data and argued this way: redo the final column, taking the other ratio, RISC/CISC instead of CISC/RISC.

Benchmark	RISC	CISC	RISC/CISC
E-string search	150	120	1.25
F-bit test	120	180	0.67
Ackerman	150	300	0.5
Rec 2-sort	2800	1400	2.0
Average			1.1

From this table, we would conclude that RISC programs are 10% longer than CISC programs on average! We are using the same reasoning as in the paper, so this conclusion is equally justifiable— yet the result is opposite! What is going on?

16.4.2 A Probabilistic Interpretation

To resolve these contradictory conclusions, we can model the RISC vs. CISC debate with the machinery of probability theory.

Let the sample space be the set of benchmark programs. Let the random variable R be the length of the compiled RISC program, and let the random variable C be the length of the compiled CISC program. We would like to compare the average length, $\mathrm{E}[R]$, of a RISC program to the average length, $\mathrm{E}[C]$, of a CISC program.

To compare average program lengths, we must assign a probability to each sample point; in effect, this assigns a "weight" to each benchmark. One might like to weigh benchmarks based on how frequently similar programs arise in practice. Lacking such data, however, we will assign all benchmarks equal weight; that is, our sample space is uniform.

In terms of our probability model, the paper computes C/R for each sample point, and then averages to obtain $\mathrm{E}[C/R] = 1.2$. This much is correct. The authors then conclude that "CISC programs are 20% longer on average"; that is, they conclude that $\mathrm{E}[C] = 1.2\ \mathrm{E}[R]$.

Similarly, the critics calculation correctly showed that $\mathrm{E}[R/C] = 1.1$. They then concluded that $\mathrm{E}[R] = 1.1\ \mathrm{E}[C]$, that is, a RISC program is 10% longer than a CISC program on average.

These arguments make a natural assumption, namely, that

False Claim 16.4.1. *If S and T are independent random variables with $T > 0$, then*

$$\mathrm{E}\left[\frac{S}{T}\right] = \frac{\mathrm{E}[S]}{\mathrm{E}[T]}.$$

In other words False Claim 16.4.1 simply generalizes the rule for expectation of a product to a rule for the expectation of a quotient. But the rule for requires independence, and we surely don't expect C and R to be independent: large source programs will lead to large compiled programs, so when the RISC program is large, so the CISC would be too.

However, we can easily compensate for this kind of dependence: we should compare the lengths of the programs *relative to the size of the source code*. While the lengths of C and R are dependent, it's more plausible that their *relative* lengths will be independent. So we really want to divide the second and third entries in each row of the table by a "normalizing factor" equal to the length of the benchmark program in the first entry of the row.

But note that normalizing this way will have no effect on the fourth column! That's because the normalizing factors applied to the second and and third entries of the rows will cancel. So the independence hypothesis of False Claim 16.4.1 may be justified, in which case the authors' conclusions would be justified. But then, so would the contradictory conclusions of the critics. Something must be wrong! Maybe it's False Claim 16.4.1 (duh!), so let's try and prove it.

False proof.

$$\begin{aligned}
\mathrm{E}\left[\frac{S}{T}\right] &= \mathrm{E}\left[S \cdot \frac{1}{T}\right] \\
&= \mathrm{E}[S] \cdot \mathrm{E}\left[\frac{1}{T}\right] && \text{(independence of } S \text{ and } T\text{)} \qquad (16.11) \\
&= \mathrm{E}[S] \cdot \frac{1}{\mathrm{E}[T]}. && (16.12) \\
&= \frac{\mathrm{E}[S]}{\mathrm{E}[T]}.
\end{aligned}$$

Note that line (16.11) uses the fact that if S and T are independent, then so are S and $1/T$. This holds because functions of independent random variables yield independent random variables, as shown in Problem **??**.

■

But this proof is bogus! The bug is in line (16.12), which assumes

False Theorem 16.4.2.

$$\mathrm{E}\left[\frac{1}{T}\right] = \frac{1}{\mathrm{E}[T]}.$$

Here is a counterexample:

Example. Suppose $T = 1$ with probability $1/2$ and $T = 2$ with probability $1/2$.

Then

$$\begin{aligned}
\frac{1}{\mathrm{E}[T]} &= \frac{1}{1 \cdot \frac{1}{2} + 2 \cdot \frac{1}{2}} \\
&= \frac{2}{3} \\
&\neq \frac{3}{4} \\
&= \frac{1}{1} \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2} \\
&= \mathrm{E}\left[\frac{1}{T}\right].
\end{aligned}$$

The two quantities are not equal, so False Claim 16.4.2 really is false.

Unfortunately, the fact that Claim 16.4.1 and 16.4.2 are false does not mean that they are never used!

16.4.3 The Proper Quotient

We can compute $\mathrm{E}[R]$ and $\mathrm{E}[C]$ as follows:

$$\begin{aligned}\mathrm{E}[R] &= \sum_{i \in \text{Range(R)}} i \cdot \Pr\{R = i\} \\ &= \frac{150}{4} + \frac{120}{4} + \frac{150}{4} + \frac{2800}{4} \\ &= 805\end{aligned}$$

$$\begin{aligned}\mathrm{E}[C] &= \sum_{i \in \text{Range(C)}} i \cdot \Pr\{C = i\} \\ &= \frac{120}{4} + \frac{180}{4} + \frac{300}{4} + \frac{1400}{4} \\ &= 500\end{aligned}$$

Now since $\mathrm{E}[R] / \mathrm{E}[C] = 1.61$, we conclude that the average RISC program is 61% longer than the average CISC program. This is a third answer, completely different from the other two! Furthermore, this answer makes RISC look really bad in terms of code length. This one is the correct conclusion, under our assumption that the benchmarks deserve equal weight. Neither of the earlier results were correct—not surprising since both were based on the same false Claim.

16.4.4 A Simpler Example

The source of the problem is clearer in the following, simpler example. Suppose the data were as follows.

Benchmark	Processor A	Processor B	B/A	A/B
Problem 1	2	1	$1/2$	2
Problem 2	1	2	2	$1/2$
Average			1.25	1.25

Now the data for the processors A and B is exactly symmetric; the two processors are equivalent. Yet, from the third column we would conclude that Processor B programs are 25% longer on average, and from the fourth column we would conclude that Processor A programs are 25% longer on average. Both conclusions are obviously wrong.

The moral is that one must be very careful in summarizing data, we must not take an average of ratios blindly!

EDITING NOTE:

Infinite Linearity of Expectation

We know that expectation is linear over finite sums. It's useful to extend this result to infinite summations. This works as long as we avoid sums whose values may depend on the order of summation.

Convergence Conditions for Infinite Linearity

Theorem 16.4.3. *[Linearity of Expectation] Let R_0, R_1, ..., be random variables such that*

$$\sum_{i=0}^{\infty} \mathrm{E}\left[|R_i|\right]$$

converges. Then

$$\mathrm{E}\left[\sum_{i=0}^{\infty} R_i\right] = \sum_{i=0}^{\infty} \mathrm{E}\left[R_i\right].$$

Proof. Let $T ::= \sum_{i=0}^{\infty} R_i$.

We leave it to the reader to verify that, under the given convergence hypothesis, all the sums in the following derivation are absolutely convergent, which justifies rearranging them as follows:

$$\begin{aligned}
\sum_{i=0}^{\infty} \mathrm{E}[R_i] &= \sum_{i=0}^{\infty} \sum_{s \in \mathcal{S}} R_i(s) \cdot \Pr\{s\} && \text{(Def. 16.5)} \\
&= \sum_{s \in \mathcal{S}} \sum_{i=0}^{\infty} R_i(s) \cdot \Pr\{s\} && \text{(exchanging order of summation)} \\
&= \sum_{s \in \mathcal{S}} \left[\sum_{i=0}^{\infty} R_i(s) \right] \cdot \Pr\{s\} && \text{(factoring out } \Pr\{s\}\text{)} \\
&= \sum_{s \in \mathcal{S}} T(s) \cdot \Pr\{s\} && \text{(Def. of } T\text{)} \\
&= \mathrm{E}[T] && \text{(Def. 16.5)} \\
&= \mathrm{E}\left[\sum_{i=0}^{\infty} R_i \right]. && \text{(Def. of } T\text{).}
\end{aligned}$$

■

Note that the finite linearity of expectation we established in Corollary 16.3.9 follows as a special case of Theorem 16.4.3: since $\mathrm{E}[R_i]$ is finite, so is $\mathrm{E}[|R_i|]$, and therefore so is their sum for $0 \leq i \leq n$. Hence the convergence hypothesis of Theorem 16.4.3 is trivially satisfied if there are only finitely many R_i's.

Exercise: Show that linearity of expectation fails for the sum of two variables, one with expectation $+\infty$ and the other with $-\infty$.

A Paradox

One of the simplest casino bets is on “red” or “black” at the roulette table. In each play at roulette, a small ball is set spinning around a roulette wheel until it lands in a red, black, or green colored slot. The payoff for a bet on red or black matches the bet; for example, if you bet $\$10$ on red and the ball lands in a red slot, you get back your original $\$10$ bet plus another matching $\$10$.

In the US, a roulette wheel has 2 green slots among 18 black and 18 red slots, so the probability of red is $p ::= 18/38 \approx 0.473$. In Europe, where roulette wheels have only 1 green slot, the odds for red are a little better —that is, $p = 18/37 \approx 0.486$— but still less than even. To make the game fair, we might agree to ignore green, so that $p = 1/2$.

There is a notorious gambling strategy which seems to guarantee a profit at roulette: bet $\$10$ on red, and keep doubling the bet until a red comes up. This strategy implies that a player will leave the game as a net winner of $\$10$ as soon as the red first appears. Of course the player may need an awfully large bankroll to avoid going bankrupt before red shows up—but we know that the mean time until a red occurs is $1/p$, so it seems possible that a moderate bankroll might actually work out. (In this setting, a "win" on red corresponds to a "failure" in a mean-time-to-failure situation.)

Suppose we have the good fortune to gamble against a fair roulette wheel. In this case, our expected win on any spin is zero, since at the ith spin we are equally likely to win or lose $10 \cdot 2^{i-1}$ dollars. So our expected win after any finite number of spins remains zero, and therefore our expected win using this gambling strategy is zero. This is just what we should have anticipated in a fair game.

But wait a minute. As long as there is a fixed, positive probability of red appearing on each spin of the wheel—even if the wheel is unfair—it's *certain* that red will eventually come up. So with probability one, we leave the casino having won $\$10$, and our expected dollar win is obviously $\$10$, not zero!

Something's wrong here. What?

Solution to the Paradox

The expected amount won is indeed $\$10$.

The argument claiming the expectation is zero is flawed by an invalid use of linearity of expectation for an infinite sum. To pinpoint this flaw, let's first make the sample space explicit: a sample point is a sequence $B^n R$ representing a run of $n \geq 0$ black spins terminated by a red spin. Since the wheel is fair, the probability of $B^n R$ is $2^{-(n+1)}$.

Let C_i be the number of dollars won on the ith spin. So $C_i = 10 \cdot 2^{i-1}$ when red comes up for the first time on the ith spin, that is, at precisely one sample point, namely $B^{i-1}R$. Similarly, $C_i = -10 \cdot 2^{i-1}$ when the first red spin comes up after the ith spin, namely, at the sample points $B^n R$ for $n \geq i$. Finally, we will define C_i by convention to be zero at sample points in which the session ends before the ith spin, that is, at points $B^n R$ for $n < i - 1$.

The dollar amount won in any gambling session is the value of the sum $\sum_{i=1}^{\infty} C_i$.

At any sample point $B^n R$, the value of this sum is

$$10 \cdot -(1 + 2 + 2^2 + \cdots + 2^{n-1}) + 10 \cdot 2^n = 10,$$

which trivially implies that its expectation is 10 as well. That is, the amount we are *certain* to leave the casino with, as well as expectation of the amount we win, is \$10.

Moreover, our reasoning that $\mathrm{E}[C_i] = 0$ is sound, so

$$\sum_{i=1}^{\infty} \mathrm{E}[C_i] = \sum_{i=1}^{\infty} 0 = 0.$$

The flaw in our argument is the claim that, since the expectation at each spin was zero, therefore the final expectation would also be zero. Formally, this corresponds to concluding that

$$\mathrm{E}[\text{amount won}] = \mathrm{E}\left[\sum_{i=1}^{\infty} C_i\right] = \sum_{i=1}^{\infty} \mathrm{E}[C_i] = 0.$$

The flaw lies exactly in the second equality. This is a case where linearity of expectation fails to hold—even though both $\sum_{i=1}^{\infty} \mathrm{E}[C_i]$ and $\mathrm{E}[\sum_{i=1}^{\infty} C_i]$ are finite—

because the convergence hypothesis needed for linearity is false. Namely, the sum

$$\sum_{i=1}^{\infty} \mathrm{E}\left[|C_i|\right]$$

does not converge. In fact, the expected value of $|C_i|$ is 10 because $|C_i| = 10 \cdot 2^i$ with probability 2^{-i} and otherwise is zero, so this sum rapidly approaches infinity.

Probability theory truly leads to this apparently paradoxical conclusion: a game allowing an unbounded—even though always finite—number of "fair" moves may not be fair in the end. In fact, our reasoning leads to an even more startling conclusion: even against an *unfair* wheel, as long as there is some fixed positive probability of red on each spin, we are certain to win $\$10$!

This is clearly a case where naive intuition is unreliable: we don't expect to beat a fair game, and we do expect to lose when the odds are against us. Nevertheless, the "paradox" that in fact we always win by bet-doubling cannot be denied.

But remember that from the start we chose to assume that no one goes bankrupt while executing our bet-doubling strategy. This assumption is crucial, because the expected loss while waiting for the strategy to produce its ten dollar profit is actually infinite! So it's not surprising, after all, that we arrived at an apparently paradoxical conclusion from an unrealistic assumption.

This example also serves a warning that in making use of infinite linearity of expectation, the convergence hypothesis which justifies it had better be checked.

For WALD'S theorem see F02 ln11-12.

■

16.4.5 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:

$$\mathrm{E}\left[R_1 \cdot R_2\right] = \mathrm{E}\left[R_1\right] \cdot \mathrm{E}\left[R_2\right] = 3.5 \cdot 3.5 = 12.25. \tag{16.13}$$

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 16.4.4. *For any two* independent *random variables R_1, R_2,*

$$\mathrm{E}[R_1 \cdot R_2] = \mathrm{E}[R_1] \cdot \mathrm{E}[R_2].$$

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

$$\begin{aligned}
&\mathrm{E}[R_1 \cdot R_2] \\
&::= \sum_{r \in \mathrm{range}(R_1 \cdot R_2)} r \cdot \Pr\{R_1 \cdot R_2 = r\} \\
&= \sum_{r_i \in \mathrm{range}(R_i)} r_1 r_2 \cdot \Pr\{R_1 = r_1 \text{ and } R_2 = r_2\} \\
&= \sum_{r_1 \in \mathrm{range}(R_1)} \sum_{r_2 \in \mathrm{range}(R_2)} r_1 r_2 \cdot \Pr\{R_1 = r_1 \text{ and } R_2 = r_2\} && \text{(ordering terms in the sum)} \\
&= \sum_{r_1 \in \mathrm{range}(R_1)} \sum_{r_2 \in \mathrm{range}(R_2)} r_1 r_2 \cdot \Pr\{R_1 = r_1\} \cdot \Pr\{R_2 = r_2\} && \text{(indep. of } R_1, R_2\text{)} \\
&= \sum_{r_1 \in \mathrm{range}(R_1)} \left(r_1 \Pr\{R_1 = r_1\} \cdot \sum_{r_2 \in \mathrm{range}(R_2)} r_2 \Pr\{R_2 = r_2\} \right) && \text{(factoring out } r_1 \Pr\{R_1 = r_1\}\text{)} \\
&= \sum_{r_1 \in \mathrm{range}(R_1)} r_1 \Pr\{R_1 = r_1\} \cdot \mathrm{E}[R_2] && \text{(def of } \mathrm{E}[R_2]\text{)} \\
&= \mathrm{E}[R_2] \cdot \sum_{r_1 \in \mathrm{range}(R_1)} r_1 \Pr\{R_1 = r_1\} && \text{(factoring out } \mathrm{E}[R_2]\text{)} \\
&= \mathrm{E}[R_2] \cdot \mathrm{E}[R_1]. && \text{(def of } \mathrm{E}[R_1]\text{)}
\end{aligned}$$

■

Theorem 16.4.4 extends routinely to a collection of mutually independent variables.

Corollary 16.4.5. *If random variables $R_1, R_2, \ldots, R_k$ are mutually independent, then*

$$\mathrm{E}\left[\prod_{i=1}^{k} R_i\right] = \prod_{i=1}^{k} \mathrm{E}\left[R_i\right].$$

16.4.6 Problems

Practice Problems

Class Problems

Homework Problems

Chapter 17

Deviation from the Mean

17.1 Why the Mean?

In the previous chapter we took it for granted that expectation is important, and we developed a bunch of techniques for calculating expected (mean) values. But why should we care about the mean? After all, a random variable may never take a value anywhere near its expected value.

The most important reason to care about the mean value comes from its connection to estimation by sampling. For example, suppose we want to estimate the average age, income, family size, or other measure of a population. To do this, we determine a random process for selecting people —say throwing darts at census lists. This process makes the selected person's age, income, and so on into a random variable whose *mean* equals the *actual average* age or income of the population. So we can select a random sample of people and calculate the average of people in the sample to estimate the true average in the whole population. Many fundamental results of probability theory explain exactly how the reliability of such estimates improves as the sample size increases, and in this chapter we'll examine a few such results.

In particular, when we make an estimate by repeated sampling, we need to know how much confidence we should have that our estimate is OK. Technically, this reduces to finding the probability that an estimate *deviates* a lot from its expected value. This topic of *deviation from the mean* is the focus of this final chapter.

The first technical result about deviation will be Markov's Theorem, which gives a simple, but typically coarse, upper bound on the probability that the value of a random variable is more than a certain multiple of its mean. Markov's result holds if we know nothing about a random variable except what its mean is and that its values are nonnegative. Accordingly, Markov's Theorem is very general, but also is much weaker than results which take into account more information about the distribution of the variable.

In many situations, we not only know the mean, but also another numerical quantity called the *variance* of the random variable. The second basic result is Chebyshev's Theorem, which combines Markov's Theorem and information about the variance to give more refined bounds.

The final result we obtain about deviation is Chernoff's bound. Chernoff's bound applies to a random variable that is a sum of bounded independent random variables. Its bound is exponentially tighter than the other two.

EDITING NOTE: A random variable may never take a value anywhere near its expected value, so why is its expected value important? The reason is suggested by a property of gambling games that most people recognize intuitively. Suppose your gamble hinges on the roll of two dice, where you win if the sum of the dice is seven. If the dice are fair, the probabilty you win is $1/6$, which is also your expected number of wins in one roll. Of course there's no such thing as $1/6$ of a win in one roll, since either you win or you don't. But if you play *many times*, you would expect that the *fraction* of times you win would be close to $1/6$. In fact, if you played a lot of times and found that your fraction of wins wasn't pretty close to $1/6$, you would become pretty sure that the dice weren't fair. ■

17.2 Markov's Theorem

Markov's theorem is an easy result that gives a generally rough estimate of the probability that a random variable takes a value *much larger* than its mean.

The idea behind Markov's Theorem can be explained with a simple example of *intelligence quotient*, IQ. This quantity was devised so that the average IQ measurement would be 100. Now from this fact alone we can conclude that at most 1/3 the population can have an IQ of 300 or more, because if more than a third had an IQ of 300, then the average would have to be *more* than $(1/3)300 = 100$, contradicting the fact that the average is 100. So the probability that a randomly chosen person has an IQ of 300 or more is at most 1/3. Of course this is not a very strong conclusion; in fact no IQ of over 300 has ever been recorded. But by the same logic, we can also conclude that at most 2/3 of the population can have an IQ of 150 or more. IQ's of over 150 have certainly been recorded, though again, a much smaller fraction than 2/3 of the population actually has an IQ that high.

But although these conclusions about IQ are weak, they are actually the strongest general conclusions that can be reached about a random variable using *only* the fact that it is nonnegative and its mean is 100. For example, if we choose a random variable equal to 300 with probability 1/3, and 0 with probability 2/3, then its mean is 100, and the probability of a value of 300 or more really is 1/3. So we can't hope to get a better upper bound based solely on this limited amount of information.

EDITING NOTE:

Note that very different distributions can still have the same mean.

Example 17.2.1. Suppose that we roll a fair die. This gives a random variable uniformly distributed on $1, 2, \ldots 6$. The mean, or expected value, is 3.5. Of course, this random variable never takes on exactly the expected value; in fact, the outcome deviates from the mean by at least 0.5 with probability 1. Furthermore, there is a $\frac{2}{3}$ probability that the outcome deviates from the mean by at least 1.5 (roll 1, 2, 5, or 6), a $\frac{1}{3}$ probability that the outcome deviates by at least 2.5 (roll 1 or 6), and zero probability that the outcome deviates by more than 2.5.

Example 17.2.2. A random variable with the binomial distribution is much less likely to deviate far from the mean. For example, suppose we flip 100 fair, mutually independent coins and count the number of heads. The expected number of heads is 50. There is an 8% chance that the outcome is exactly the mean, and the probability of flipping more than 75 heads or fewer than 25 is less than 1 in a billion.

The probability distribution functions for the two preceding examples are graphed

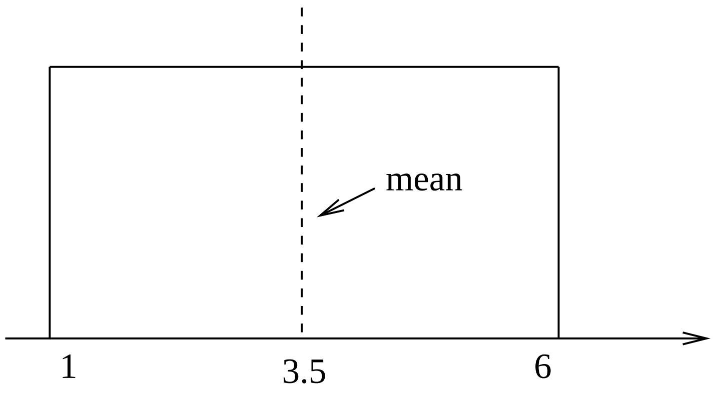


Figure 17.1: This is a graph of the uniform distribution arising from rolling a fair die. Outcomes within the range of the distribution are equally likely, regardless of distance from the mean.

in Figure 17.1 and Figure 17.2. There is a big difference! For the uniform distribution, the graph is flat; that is, outcomes far from the mean are as likely as outcomes close to the mean. However, the binomial distribution has a peak centered on the expected value and the tails fall off rapidly. This shape implies that outcomes close to the expected value are vastly more likely than outcomes far from the expected value. In other words, a random variable with the binomial distribution rarely deviates far from the mean.

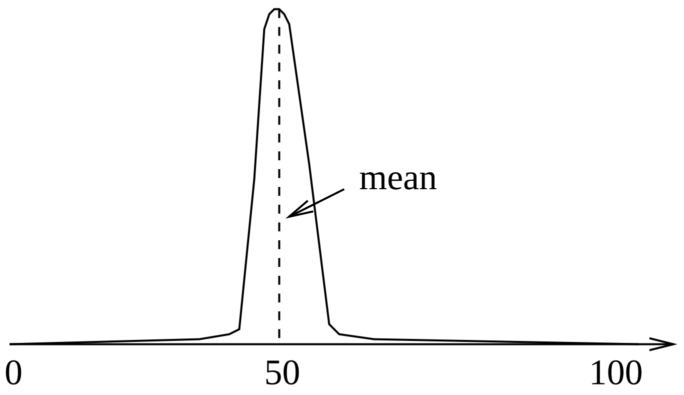


Figure 17.2: This is a rough graph of the binomial distribution given by the number of heads that come up when we flip 100 fair, mutually independent coins. Outcomes close to the mean are much more likely than outcomes far from the mean.

On the other hand, we can define a random variable that always deviates substantially from its expected value. Suppose that we glue 100 coins together, so that with probability 1/2 all are heads and with probability 1/2 all are tails. The graph of the probability distribution function for the number of heads is shown in Figure 17.3. While the expected value of this random variable is 50, the actual value is always 0 or 100.

Even in this last example, however, the random variable is twice the mean with

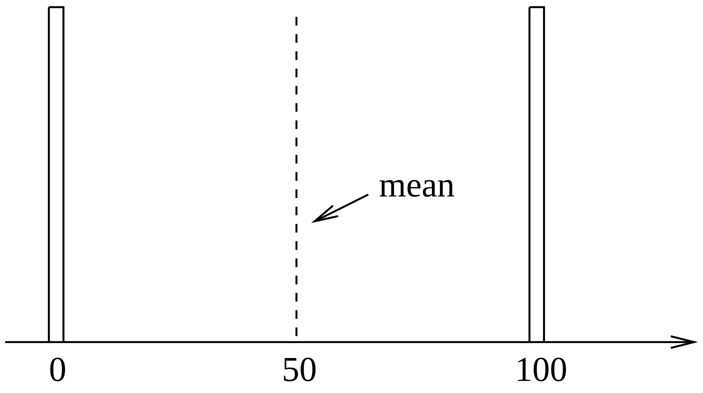


Figure 17.3: This is the nasty distribution corresponding to the number of heads that come up when we flip 100 coins that are all glued together. The outcome always differs from the mean by at least 50.

probability only 1/2. In fact, we will see that this is a worst-case distribution with respect to deviation from the mean.

Theorem Statement and Some Applications

■

Theorem 17.2.3 (Markov's Theorem)**.** *If R is a nonnegative random variable, then for* *all* $x > 0$

$$\Pr\{R \geq x\} \leq \frac{\mathrm{E}[R]}{x}.$$

EDITING NOTE:

Before we prove Markov's Theorem, let's apply it to the three examples in the preceding subsection. First, let the random variable R be the number that comes up when we roll a fair die. By Markov's Theorem, the probability of rolling a 6 is at most:

$$\Pr\{R \geq 6\} \leq \frac{\mathrm{E}[R]}{6} = \frac{3.5}{6} = 0.583\ldots$$

This conclusion is true, but weak. The actual probability of rolling a 6 is $1/6 = 0.166\ldots$.

This is typical of Markov's Theorem. The theorem is easy to apply because it requires so little information about a random variable, only the expected value and nonnegativity. But as a consequence, Markov's Theorem often leads to weak conclusions like the one above.

Suppose that we flip 100 mutually independent, fair coins. Markov's Theorem says that the probability of throwing 75 or more heads is at most:

$$\Pr\{\text{heads} \geq 75\} \leq \frac{\mathrm{E}[\text{heads}]}{75} = \frac{50}{75} = \frac{2}{3}.$$

Markov's Theorem says that the probability of 75 or more heads is at most $2/3$, but the actual probability is less than 1 in a billion!

These two examples show that Markov's Theorem gives weak results for well-behaved random variables; however, the theorem is actually tight for some nasty examples. Suppose we flip 100 fair coins and use Markov's Theorem to compute the probability of getting all heads:

$$\Pr\{\text{heads} \geq 100\} \leq \frac{\mathrm{E}[\text{heads}]}{100} = \frac{50}{100} = \frac{1}{2}.$$

If the coins are mutually independent, then the actual probability of getting all heads is a miniscule 1 in 2^{100}. In this case, Markov's Theorem looks very weak. However, in applying Markov's Theorem, we made no independence assumptions. In fact, if all the coins are glued together, then probability of throwing all heads is exactly $1/2$. In this nasty case, Markov's Theorem is actually tight!

Proof of Markov's Theorem

Let R be the weight of a person selected randomly and uniformly. Suppose that an average person weighs 100 pounds; that is, $\mathrm{E}[R] = 100$. What is the probability that a random person weighs at least 200 pounds?

There is insufficient information for an exact answer. However, we can safely say that the probability that $R \geq 200$ is most $1/2$. If more than half of the people weigh 200 pounds or more, then the average weight would exceed 100 pounds, even if everyone else weighed zero! Markov's Theorem gives the same result:

$$\Pr\{R \geq 200\} \leq \frac{\mathrm{E}[R]}{200} = \frac{100}{200} = \frac{1}{2}.$$

Reasoning similar to that above underlies the proof of Markov's Theorem. Since expectation is a weighted average of all the outcomes of the random variable, that is, a sum over all the variables the random variable can assume, we can give a lower bound on the expectation by removing some of the terms from the sum defining the expectation; this new sum can then be modified into an expression involving the probability of an event in the tail $[R \geq x]$.

■

Proof. For any $x > 0$

$$\begin{aligned}
\mathrm{E}[R] &::= \sum_{y \in \text{range}(R)} y \Pr\{R = y\} \\
&\geq \sum_{\substack{y \geq x, \\ y \in \text{range}(R)}} y \Pr\{R = y\} && \text{(because } R \geq 0\text{)} \\
&\geq \sum_{\substack{y \geq x, \\ y \in \text{range}(R)}} x \Pr\{R = y\} \\
&= x \sum_{\substack{y \geq x, \\ y \in \text{range}(R)}} \Pr\{R = y\} \\
&= x \Pr\{R \geq x\}. && (17.1)
\end{aligned}$$

Dividing the first and last expression (17.1) by x gives the desired result. ■

Our focus is deviation from the mean, so it's useful to rephrase Markov's Theorem this way:

Corollary 17.2.4. *If R is a nonnegative random variable, then for all $c \geq 1$*

$$\Pr\{R \geq c \cdot \mathrm{E}[R]\} \leq \frac{1}{c}. \tag{17.2}$$

This Corollary follows immediately from Markov's Theorem(17.2.3) by letting x be $c \cdot \mathrm{E}[R]$.

17.2.1 Applying Markov's Theorem

Let's consider the Hat-Check problem again. Now we ask what the probability is that x or more men get the right hat, this is, what the value of $\Pr\{G \geq x\}$ is.

We can compute an upper bound with Markov's Theorem. Since we know $\mathrm{E}[G] = 1$, Markov's Theorem implies

$$\Pr\{G \geq x\} \leq \frac{\mathrm{E}[G]}{x} = \frac{1}{x}.$$

For example, there is no better than a 20% chance that 5 men get the right hat, regardless of the number of people at the dinner party.

The Chinese Appetizer problem is similar to the Hat-Check problem. In this case, n people are eating appetizers arranged on a circular, rotating Chinese banquet tray. Someone then spins the tray so that each person receives a random appetizer. What is the probability that everyone gets the same appetizer as before?

There are n equally likely orientations for the tray after it stops spinning. Everyone gets the right appetizer in just one of these n orientations. Therefore, the correct answer is $1/n$.

But what probability do we get from Markov's Theorem? Let the random variable, R, be the number of people that get the right appetizer. Then of course $\mathrm{E}[R] = 1$ (right?), so applying Markov's Theorem, we find:

$$\Pr\{R \geq n\} \leq \frac{\mathrm{E}[R]}{n} = \frac{1}{n}.$$

So for the Chinese appetizer problem, Markov's Theorem is tight!

On the other hand, Markov's Theorem gives the same $1/n$ bound for the probability everyone gets their hat in the Hat-Check problem in the case that all permutations are equally likely. But the probability of this event is $1/(n!)$. So for this case, Markov's Theorem gives a probability bound that is way off.

17.2.2 Markov's Theorem for Bounded Variables

Suppose we learn that the average IQ among MIT students is 150 (which is not true, by the way). What can we say about the probability that an MIT student has an IQ of more than 200? Markov's theorem immediately tells us that no more than $150/200$ or $3/4$ of the students can have such a high IQ. Here we simply applied Markov's Theorem to the random variable, R, equal to the IQ of a random MIT student to conclude:

$$\Pr\{R > 200\} \leq \frac{\mathrm{E}[R]}{200} = \frac{150}{200} = \frac{3}{4}.$$

But let's observe an additional fact (which may be true): no MIT student has an IQ less than 100. This means that if we let $T ::= R - 100$, then T is nonnegative and $\mathrm{E}[T] = 50$, so we can apply Markov's Theorem to T and conclude:

$$\Pr\{R > 200\} = \Pr\{T > 100\} \leq \frac{\mathrm{E}[T]}{100} = \frac{50}{100} = \frac{1}{2}.$$

So only half, not 3/4, of the students can be as amazing as they think they are. A bit of a relief!

More generally, we can get better bounds applying Markov's Theorem to $R - l$

instead of R for any lower bound $l > 0$ on R.

Similarly, if we have any upper bound, u, on a random variable, S, then $u - S$ will be a nonnegative random variable, and applying Markov's Theorem to $u - S$ will allow us to bound the probability that S is much *less* than its expectation.

EDITING NOTE:

Why *R* Must be Nonnegative

Remember that Markov's Theorem applies only to nonnegative random variables! The following example shows that the theorem is false if this restriction is removed. Let R be -10 with probability $1/2$ and 10 with probability $1/2$. Then we have:

$$\mathrm{E}[R] = -10 \cdot \frac{1}{2} + 10 \cdot \frac{1}{2} = 0$$

Suppose that we now tried to compute $\Pr\{R \geq 5\}$ using Markov's Theorem:

$$\Pr\{R \geq 5\} \leq \frac{\mathrm{E}[R]}{5} = \frac{0}{5} = 0.$$

This is the wrong answer! Obviously, R is at least 5 with probability $1/2$.

On the other hand, we can still apply Markov's Theorem indirectly to derive a bound on the probability that an arbitrary variable like R is 5 more. Namely, given any random variable, R with expectation 0 and values ≥ -10, we can conclude that $\Pr\{R \geq 5\} \leq 2/3$.

Proof. Let $T ::= R + 10$. Now T is a nonnegative random variable with expectation $\mathrm{E}[R+10] = \mathrm{E}[R] + 10 = 10$, so Markov's Theorem applies and tells us that $\Pr\{T \geq 15\} \leq 10/15 = 2/3$. But $T \geq 15$ iff $R \geq 5$, so $\Pr\{R \geq 5\} \leq 2/3$, as claimed. ■

Deviation Below the Mean

Markov's Theorem says that a random variable is unlikely to greatly exceed the mean. Correspondingly, there is a theorem that says a random variable is unlikely to be much smaller than its mean.

Theorem 17.2.5. *Let l be a real number and let R be a random variable such that $R \leq l$.*

For all $x < l$, we have:

$$\Pr\{R \le x\} \le \frac{l - \mathrm{E}[R]}{l - x}.$$

Proof. The event that $R \le x$ is the same as the event that $l - R \ge l - x$. Therefore:

$$\begin{aligned}\Pr\{R \le x\} &= \Pr\{l - R \ge l - x\} \\ &\le \frac{\mathrm{E}[l-R]}{l-x}. && \text{(by Markov' Theorem)} \qquad (17.3)\end{aligned}$$

Applying Markov's Theorem in line (17.3) is permissible since $l - R$ is a nonnegative random variable and $l - x > 0$. ■

For example, suppose that the class average on a midterm was 75/100. What fraction of the class scored below 50?

There is not enough information here to answer the question exactly, but Theorem 17.2.5 gives an upper bound. Let R be the score of a random student. Since 100 is the highest possible score, we can set $L = 100$ to meet the condition in the theorem that $R \le L$. Applying Theorem 17.2.5, we find:

$$\Pr\{R \le 50\} \le \frac{100 - 75}{100 - 50} = \frac{1}{2}.$$

That is, at most half of the class scored 50 or worse. This makes sense; if more than half of the class scored 50 or worse, then the class average could not be 75, even if everyone else scored 100. As with Markov's Theorem, Theorem 17.2.5 often gives weak results. In fact, based on the data given, the entire class could have scored above 50.

■

EDITING NOTE:

Using Markov To Analyze Non-Random Events

In the previous examples, we used a theorem about a random variable to conclude facts about non-random data. For example, we concluded that if the average score on a test is 75, then at most $1/2$ the class scored 50 or worse. There is no randomness in this problem, so how can we apply Theorem 17.2.5 to reach this conclusion?

The explanation is not difficult. For any set of scores $S = \{s_1, s_2, \ldots, s_n\}$, we introduce a random variable, R, such that

$$\Pr\{R = s_i\} = \frac{(\text{\# of students with score } s_i)}{n}$$

We then use Theorem 17.2.5 to conclude that $\Pr\{R \leq 50\} \leq 1/2$. To see why this means (with certainty) that at most $1/2$ of the students scored 50 or less, we observe that

$$\begin{aligned}
\Pr\{R \leq 50\} &= \sum_{s_i \leq 50} \Pr\{R = s_i\} \\
&= \sum_{s_i \leq 50} \frac{(\text{\# of students with score } s_i)}{n} \\
&= \frac{1}{n}(\text{\# of students with score 50 or less}).
\end{aligned}$$

So, if $\Pr\{R \leq 50\} \leq 1/2$, then the number of students with score 50 or less is at most $n/2$.

■

17.2.3 Problems

Class Problems

17.3 Chebyshev's Theorem

There's a really good trick for getting more mileage out of Markov's Theorem: instead of applying it to the variable, R, apply it to some function of R. One useful choice of functions to use turns out to be taking a power of $|R|$.

In particular, since $|R|^\alpha$ is nonnegative, Markov's inequality also applies to the event $[|R|^\alpha \geq x^\alpha]$. But this event is equivalent to the event $[|R| \geq x]$, so we have:

Lemma 17.3.1. *For any random variable R, $\alpha \in \mathbb{R}^+$, and $x > 0$,*

$$\Pr\{|R| \geq x\} \leq \frac{\mathrm{E}[|R|^\alpha]}{x^\alpha}.$$

Rephrasing (17.3.1) in terms of the random variable, $|R - \mathrm{E}[R]|$, that measures R's deviation from its mean, we get

$$\Pr\{|R - \mathrm{E}[R]| \geq x\} \leq \frac{\mathrm{E}[(R - \mathrm{E}[R])^\alpha]}{x^\alpha}. \tag{17.4}$$

The case when $\alpha = 2$ is turns out to be so important that numerator of the right hand side of (17.4) has been given a name:

Definition 17.3.2. The *variance*, $\text{Var}[R]$, of a random variable, R, is:

$$\text{Var}[R] ::= \text{E}\left[(R - \text{E}[R])^2\right].$$

The restatement of (17.4) for $\alpha = 2$ is known as *Chebyshev's Theorem*.

Theorem 17.3.3 (Chebyshev). *Let R be a random variable and $x \in \mathbb{R}^+$. Then*

$$\Pr\{|R - \text{E}[R]| \geq x\} \leq \frac{\text{Var}[R]}{x^2}.$$

The expression $\text{E}\left[(R - \text{E}[R])^2\right]$ for variance is a bit cryptic; the best approach is to work through it from the inside out. The innermost expression, $R - \text{E}[R]$, is precisely the deviation of R above its mean. Squaring this, we obtain, $(R - \text{E}[R])^2$. This is a random variable that is near 0 when R is close to the mean and is a large positive number when R deviates far above or below the mean. So if R is always close to the mean, then the variance will be small. If R is often far from the mean, then the variance will be large.

17.3.1 Variance in Two Gambling Games

The relevance of variance is apparent when we compare the following two gambling games.

Game A: We win \$2 with probability $2/3$ and lose \$1 with probability $1/3$.

Game B: We win \$1002 with probability $2/3$ and lose \$2001 with probability $1/3$.

Which game is better financially? We have the same probability, 2/3, of winning each game, but that does not tell the whole story. What about the expected return for each game? Let random variables A and B be the payoffs for the two games. For example, A is 2 with probability 2/3 and -1 with probability 1/3. We can compute the expected payoff for each game as follows:

$$\mathrm{E}[A] = 2 \cdot \frac{2}{3} + (-1) \cdot \frac{1}{3} = 1,$$

$$\mathrm{E}[B] = 1002 \cdot \frac{2}{3} + (-2001) \cdot \frac{1}{3} = 1.$$

The expected payoff is the same for both games, but they are obviously very different! This difference is not apparent in their expected value, but is captured by variance. We can compute the $\operatorname{Var}[A]$ by working "from the inside out" as follows:

$$\begin{aligned}
A - \mathrm{E}[A] &= \begin{cases} 1 & \text{with probability } \frac{2}{3} \\ -2 & \text{with probability } \frac{1}{3} \end{cases} \\
(A - \mathrm{E}[A])^2 &= \begin{cases} 1 & \text{with probability } \frac{2}{3} \\ 4 & \text{with probability } \frac{1}{3} \end{cases} \\
\mathrm{E}\left[(A - \mathrm{E}[A])^2\right] &= 1 \cdot \frac{2}{3} + 4 \cdot \frac{1}{3} \\
\operatorname{Var}[A] &= 2.
\end{aligned}$$

Similarly, we have for $\operatorname{Var}[B]$:

$$\begin{aligned}
B - \mathrm{E}[B] &= \begin{cases} 1001 & \text{with probability } \frac{2}{3} \\ -2002 & \text{with probability } \frac{1}{3} \end{cases} \\
(B - \mathrm{E}[B])^2 &= \begin{cases} 1,002,001 & \text{with probability } \frac{2}{3} \\ 4,008,004 & \text{with probability } \frac{1}{3} \end{cases} \\
\mathrm{E}\left[(B - \mathrm{E}[B])^2\right] &= 1,002,001 \cdot \frac{2}{3} + 4,008,004 \cdot \frac{1}{3} \\
\operatorname{Var}[B] &= 2,004,002.
\end{aligned}$$

The variance of Game A is 2 and the variance of Game B is more than two million! Intuitively, this means that the payoff in Game A is usually close to the expected value of \$1, but the payoff in Game B can deviate very far from this expected value.

High variance is often associated with high risk. For example, in ten rounds of Game A, we expect to make \$10, but could conceivably lose \$10 instead. On the other hand, in ten rounds of game B, we also expect to make \$10, but could actually lose more than \$20,000!

17.3.2 Standard Deviation

Because of its definition in terms of the square of a random variable, the variance of a random variable may be very far from a typical deviation from the mean. For example, in Game B above, the deviation from the mean is 1001 in one outcome and -2002 in the other. But the variance is a whopping 2,004,002. From a dimensional analysis viewpoint, the "units" of variance are wrong: if the random variable is in dollars, then the expectation is also in dollars, but the variance is in square dollars. For this reason, people often describe random variables using standard deviation instead of variance.

Definition 17.3.4. The *standard deviation*, σ_R, of a random variable, R, is the square root of the variance:

$$\sigma_R ::= \sqrt{\text{Var}[R]} = \sqrt{\text{E}[(R - \text{E}[R])^2]}.$$

So the standard deviation is the square root of the mean of the square of the deviation, or the *root mean square* for short. It has the same units —dollars in our example —as the original random variable and as the mean. Intuitively, it measures the average deviation from the mean, since we can think of the square root on the outside as canceling the square on the inside.

Example 17.3.5. The standard deviation of the payoff in Game B is:

$$\sigma_B = \sqrt{\text{Var}[B]} = \sqrt{2,004,002} \approx 1416.$$

The random variable B actually deviates from the mean by either positive 1001 or negative 2002; therefore, the standard deviation of 1416 describes this situation reasonably well.

Intuitively, the standard deviation measures the "width" of the "main part" of the distribution graph, as illustrated in Figure 17.4.

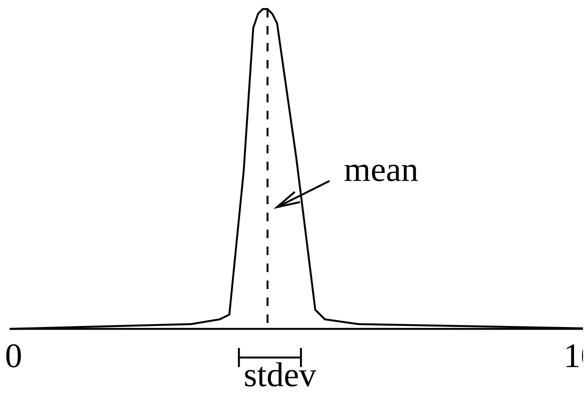


Figure 17.4: The standard deviation of a distribution indicates how wide the "main part" of it is.

It's useful to rephrase Chebyshev's Theorem in terms of standard deviation.

Corollary 17.3.6. *Let R be a random variable, and let c be a positive real number.*

$$\Pr\{|R - \mathrm{E}[R]| \geq c\sigma_R\} \leq \frac{1}{c^2}.$$

Here we see explicitly how the "likely" values of R are clustered in an $O(\sigma_R)$-sized region around $\mathrm{E}[R]$, confirming that the standard deviation measures how spread out the distribution of R is around its mean.

Proof. Substituting $x = c\sigma_R$ in Chebyshev's Theorem gives:

$$\Pr\{|R - \mathrm{E}[R]| \geq c\sigma_R\} \leq \frac{\mathrm{Var}[R]}{(c\sigma_R)^2} = \frac{\sigma_R^2}{(c\sigma_R)^2} = \frac{1}{c^2}.$$

■

The IQ Example

Suppose that, in addition to the national average IQ being 100, we also know the standard deviation of IQ's is 10. How rare is an IQ of 300 or more?

Let the random variable, R, be the IQ of a random person. So we are supposing that $\mathrm{E}[R] = 100$, $\sigma_R = 10$, and R is nonnegative. We want to compute $\Pr\{R \geq 300\}$.

We have already seen that Markov's Theorem 17.2.3 gives a coarse bound, namely,

$$\Pr\{R \geq 300\} \leq \frac{1}{3}.$$

Now we apply Chebyshev's Theorem to the same problem:

$$\Pr\{R \geq 300\} = \Pr\{|R - 100| \geq 200\} \leq \frac{\mathrm{Var}[R]}{200^2} = \frac{10^2}{200^2} = \frac{1}{400}.$$

So Chebyshev's Theorem implies that at most one person in four hundred has an IQ of 300 or more. We have gotten a much tighter bound using the additional information, namely the variance of R, than we could get knowing only the expectation.

17.4 Properties of Variance

The definition of variance of R as $\mathrm{E}\left[(R - \mathrm{E}[R])^2\right]$ may seem rather arbitrary.

EDITING NOTE:

The variance is the average *of the square* of the deviation from the mean. For this reason, variance is sometimes called the "mean squared deviation." But why bother squaring? Why not simply compute the average deviation from the mean? That is, why not define variance to be $\mathrm{E}[R - \mathrm{E}[R]]$?

The problem with this definition is that the positive and negative deviations from the mean exactly cancel. By linearity of expectation, we have:

$$\mathrm{E}[R - \mathrm{E}[R]] = \mathrm{E}[R] - \mathrm{E}[\mathrm{E}[R]].$$

Since $\mathrm{E}[R]$ is a constant, its expected value is itself. Therefore

$$\mathrm{E}[R - \mathrm{E}[R]] = \mathrm{E}[R] - \mathrm{E}[R] = 0.$$

By this definition, every random variable has zero variance. That is not useful! Because of the square in the conventional definition, both positive and negative deviations from the mean increase the variance; positive and negative deviations do not cancel.

Of course, we could also prevent positive and negative deviations from canceling by taking an absolute value. ■

A direct measure of average deviation would be $\mathrm{E}[\,|R - \mathrm{E}[R]|\,]$. But the direct measure doesn't have the many useful properties that variance has, which is what this section is about.

17.4.1 A Formula for Variance

Applying linearity of expectation to the formula for variance yields a convenient alternative formula.

Lemma 17.4.1.

$$\mathrm{Var}[R] = \mathrm{E}\left[R^2\right] - \mathrm{E}^2[R],$$

for any random variable, R.

Here we use the notation $\mathrm{E}^2[R]$ as shorthand for $(\mathrm{E}[R])^2$.

EDITING NOTE: Remember that $\mathrm{E}\left[R^2\right]$ is generally not equal to $\mathrm{E}^2[R]$. We know the expected value of a product is the product of the expected values for independent variables, but not in general. And R is not independent of itself unless it is constant.

■

Proof. Let $\mu = \mathrm{E}[R]$. Then

$$\begin{aligned}
\mathrm{Var}[R] &= \mathrm{E}\left[(R - \mathrm{E}[R])^2\right] && \text{(Def 17.3.2 of variance)} \\
&= \mathrm{E}\left[(R-\mu)^2\right] && \text{(def of } \mu\text{)} \\
&= \mathrm{E}\left[R^2 - 2\mu R + \mu^2\right] \\
&= \mathrm{E}\left[R^2\right] - 2\mu\,\mathrm{E}[R] + \mu^2 && \text{(linearity of expectation)} \\
&= \mathrm{E}\left[R^2\right] - 2\mu^2 + \mu^2 && \text{(def of } \mu\text{)} \\
&= \mathrm{E}\left[R^2\right] - \mu^2 \\
&= \mathrm{E}\left[R^2\right] - \mathrm{E}^2[R]. && \text{(def of } \mu\text{)}
\end{aligned}$$

■

For example, if B is a Bernoulli variable where $p ::= \Pr\{B = 1\}$, then

Lemma 17.4.2.

$$\mathrm{Var}[B] = p - p^2 = p(1-p). \tag{17.5}$$

Proof. By Lemma 16.3.3, $\mathrm{E}[B] = p$. But since B only takes values 0 and 1, $B^2 = B$.

So Lemma 17.4.2 follows immediately from Lemma 17.4.1. ■

17.4.2 Variance of Time to Failure

According to section 16.3.3, the mean time to failure is $1/p$ for a process that fails during any given hour with probability p. What about the variance? That is, let C be the hour of the first failure, so $\Pr\{C = i\} = (1-p)^{i-1}p$. We'd like to find a formula for $\text{Var}[C]$.

By Lemma 17.4.1,

$$\text{Var}[C] = \text{E}\left[C^2\right] - (1/p)^2 \tag{17.6}$$

so all we need is a formula for $\text{E}\left[C^2\right]$:

$$\begin{aligned}
\text{E}\left[C^2\right] &::= \sum_{i \geq 1} i^2 (1-p)^{i-1} p \\
&= p \sum_{i \geq 1} i^2 x^{i-1} \qquad\qquad \text{(where } x = 1-p\text{).}
\end{aligned} \tag{17.7}$$

But (13.2) gives the generating function $x(1+x)/(1-x)^3$ for the nonnegative integer squares, and this implies that the generating function for the sum in (17.7) is $(1 +$

$x)/(1-x)^3$. So,

$$\begin{aligned} \mathrm{E}\left[C^2\right] &= p\,\frac{(1+x)}{(1-x)^3} && \text{(where } x = 1-p\text{)} \\ &= p\,\frac{2+p}{p^3} \\ &= \frac{1-p}{p^2} + \frac{1}{p^2}, && (17.8) \end{aligned}$$

Combining (17.6) and (17.8) gives a simple answer:

$$\mathrm{Var}\left[C\right] = \frac{1-p}{p^2}\,. \tag{17.9}$$

It's great to be able to apply generating function expertise to knock off equation (17.9) mechanically just from the definition of variance, but there's a more elementary, and memorable, alternative. In section 16.3.3 we used conditional expectation to find the mean time to failure, and a similar approach works for the variance. Namely, the expected value of C^2 is the probability, p, of failure in the first hour times 1^2, plus $(1-p)$ times the expected value of $(C+1)^2$. So

$$\begin{aligned} \mathrm{E}\left[C^2\right] &= p \cdot 1^2 + (1-p)\,\mathrm{E}\left[(C+1)^2\right] \\ &= p + (1-p)\left(\mathrm{E}\left[C^2\right] + \frac{2}{p} + 1\right), \end{aligned}$$

which directly simplifies to (17.8).

EDITING NOTE:

Lemma 17.4.1 gives a convenient way to compute the variance of a random variable: find the expected value of the square and subtract the square of the expected value. For example, we can compute the variance of the outcome of a fair die as follows:

$$\text{E}\left[R^2\right] = \frac{1}{6}(1^2 + 2^2 + 3^2 + 4^2 + 5^2 + 6^2) = \frac{91}{6},$$

$$\text{E}^2\left[R\right] = \left(3\frac{1}{2}\right)^2 = \frac{49}{4},$$

$$\text{Var}\left[R\right] = \text{E}\left[R^2\right] - \text{E}^2\left[R\right] = \frac{91}{6} - \frac{49}{4} = \frac{35}{12}.$$

This result is particularly useful when we want to estimate the variance of a random variable from a sequence $x_1, x_2, \ldots, x_n$, of sample values of the variable.

Definition. For any sequence of real numbers $x_1, x_2, \ldots, x_n$, define the *sample* *mean*, μ_n, and the *sample variance*, v_n, of the sequence to be:

$$\begin{aligned}\mu_n &::= \frac{\sum_{i=1}^n x_i}{n},\\ v_n &::= \frac{\sum_{i=1}^n (x_i - \mu_n)^2}{n}.\end{aligned}$$

Notice that if we define a random variable, R, which is equally likely to take each of the values in the sequence, that is $\Pr\{R = x_i\} = 1/n$ for $i = 1, \ldots, n$, then $\mu_n = \mathrm{E}[R]$ and $v_n = \mathrm{Var}[R]$. So Lemma 17.4.1 applies to R and lets us conclude that

$$v_n = \frac{\sum_{i=1}^n x_i^2}{n} - \left(\frac{\sum_{i=1}^n x_i}{n}\right)^2. \tag{17.10}$$

This leads to a simple procedure for computing the sample mean and variance while reading the sequence $x_1, \ldots, x_n$ from left to right. Namely, maintain a sum of all numbers seen and also maintain a sum of the squares of all numbers seen. That is, we store two values, starting with the values x_1 and x_1^2. Then, as we get to the next number, x_i, we add it to the first sum and add its square, x_i^2, to the second sum. After a single pass through the sequence $x_1, \ldots, x_n$, we wind up with the values of the two sums $\sum_{i=1}^n x_i$ and $\sum_{i=1}^n x_i^2$. Then we just plug these two values into (17.10) to find the sample variance.

■

EDITING NOTE:

Expectation Squared

The alternate definition of variance given in Lemma 17.4.1 has a cute implication:

Corollary 17.4.3. *If R is a random variable, then* $\mathrm{E}\left[R^2\right] \geq \mathrm{E}^2\left[R\right]$.

Proof. We first defined $\mathrm{Var}\left[R\right]$ as an average of a squared expression, so $\mathrm{Var}\left[R\right]$ is nonnegative. Then we proved that $\mathrm{Var}\left[R\right] = \mathrm{E}\left[R^2\right] - \mathrm{E}^2\left[R\right]$. This implies that $\mathrm{E}\left[R^2\right] - \mathrm{E}^2\left[R\right]$ is nonnegative. Therefore, $\mathrm{E}\left[R^2\right] \geq \mathrm{E}^2\left[R\right]$. ■

In words, the expectation of a square is at least the square of the expectation. The two are equal exactly when the variance is zero:

$$\mathrm{E}\left[R^2\right] = \mathrm{E}^2\left[R\right] \text{ iff } \mathrm{E}\left[R^2\right] - \mathrm{E}^2\left[R\right] = 0 \text{ iff } \mathrm{Var}\left[R\right] = 0.$$

■

EDITING NOTE:

Zero Variance

When does a random variable, R, have zero variance?...when the random variable *never* deviates from the mean!

Lemma. *The variance of a random variable, R, is zero if and only if* $\Pr\{R = \mathrm{E}[R]\} = 1$.

So saying that $\mathrm{Var}[R] = 0$ is almost the same as saying that R is constant. Namely, it takes the constant value equal to its expectation on all sample points with nonzero probability. (It can take on any finite values on sample points with zero probability without affecting the variance.)

Proof. By the definition of variance,

$$\mathrm{Var}[R] = 0 \quad \text{iff} \quad \mathrm{E}\left[(R - \mathrm{E}[R])^2\right] = 0.$$

The inner expression on the right, $(R - \mathrm{E}[R])^2$, is always nonnegative because of the square. As a result, $\mathrm{E}\left[(R - \mathrm{E}[R])^2\right] = 0$ if and only if $\Pr\left\{(R - \mathrm{E}[R])^2 \neq 0\right\}$ is zero, which is the same as saying that $\Pr\left\{(R - \mathrm{E}[R])^2 = 0\right\}$ is one. That is,

$$\mathrm{Var}[R] = 0 \text{ IFF } \Pr\left\{(R - \mathrm{E}[R])^2 = 0\right\} = 1.$$

But the $(R - \mathrm{E}[R])^2 = 0$ and $R = \mathrm{E}[R]$ are different descriptions of the same event.

Therefore,

$$\mathrm{Var}[R] = 0 \quad \text{iff} \quad \Pr\{R = \mathrm{E}[R]\} = 1.$$

■

■

17.4.3 Dealing with Constants

It helps to know how to calculate the variance of $aR + b$:

Theorem 17.4.4. *Let R be a random variable, and a a constant. Then*

$$\mathrm{Var}[aR] = a^2\,\mathrm{Var}[R]. \tag{17.11}$$

Proof. Beginning with the definition of variance and repeatedly applying linearity of expectation, we have:

$$\begin{aligned}
\text{Var}[aR] &::= \text{E}\left[(aR - \text{E}[aR])^2\right] \\
&= \text{E}\left[(aR)^2 - 2aR\,\text{E}[aR] + \text{E}^2[aR]\right] \\
&= \text{E}\left[(aR)^2\right] - \text{E}[2aR\,\text{E}[aR]] + \text{E}^2[aR] \\
&= a^2\,\text{E}\left[R^2\right] - 2\,\text{E}[aR]\,\text{E}[aR] + \text{E}^2[aR] \\
&= a^2\,\text{E}\left[R^2\right] - a^2\text{E}^2[R] \\
&= a^2\left(\text{E}\left[R^2\right] - \text{E}^2[R]\right) \\
&= a^2\,\text{Var}[R] && \text{(by Lemma 17.4.1)}
\end{aligned}$$

■

It's even simpler to prove that adding a constant does not change the variance, as the reader can verify:

Theorem 17.4.5. *Let R be a random variable, and b a constant. Then*

$$\text{Var}[R + b] = \text{Var}[R]\,. \tag{17.12}$$

Recalling that the standard deviation is the square root of variance, this implies that the standard deviation of $aR + b$ is simply $|a|$ times the standard deviation of R:

Corollary 17.4.6.

$$\sigma_{aR+b} = |a| \, \sigma_R.$$

17.4.4 Variance of a Sum

In general, the variance of a sum is not equal to the sum of the variances, but variances do add for *independent* variables. In fact, *mutual* independence is not necessary: *pairwise* independence will do. This is useful to know because there are some important situations involving variables that are pairwise independent but not mutually independent.

Theorem 17.4.7. *If R_1 and R_2 are independent random variables, then*

$$\text{Var}\,[R_1 + R_2] = \text{Var}\,[R_1] + \text{Var}\,[R_2]\,. \tag{17.13}$$

Proof. We may assume that $\text{E}\,[R_i] = 0$ for $i = 1, 2$, since we could always replace

R_i by $R_i - \mathrm{E}[R_i]$ in equation (17.13). This substitution preserves the independence of the variables, and by Theorem 17.4.5, does not change the variances.

Now by Lemma 17.4.1, $\mathrm{Var}[R_i] = \mathrm{E}\left[R_i^2\right]$ and $\mathrm{Var}[R_1 + R_2] = \mathrm{E}\left[(R_1 + R_2)^2\right]$, so we need only prove

$$\mathrm{E}\left[(R_1 + R_2)^2\right] = \mathrm{E}\left[R_1^2\right] + \mathrm{E}\left[R_2^2\right]. \tag{17.14}$$

But (17.17) follows from linearity of expectation and the fact that

$$\mathrm{E}[R_1 R_2] = \mathrm{E}[R_1]\,\mathrm{E}[R_2] \tag{17.15}$$

since R_1 and R_2 are independent:

$$\begin{aligned}
\mathrm{E}\left[(R_1 + R_2)^2\right] &= \mathrm{E}\left[R_1^2 + 2R_1R_2 + R_2^2\right] \\
&= \mathrm{E}\left[R_1^2\right] + 2\,\mathrm{E}[R_1R_2] + \mathrm{E}\left[R_2^2\right] \\
&= \mathrm{E}\left[R_1^2\right] + 2\,\mathrm{E}[R_1]\,\mathrm{E}[R_2] + \mathrm{E}\left[R_2^2\right] && \text{(by (17.18))} \\
&= \mathrm{E}\left[R_1^2\right] + 2 \cdot 0 \cdot 0 + \mathrm{E}\left[R_2^2\right] \\
&= \mathrm{E}\left[R_1^2\right] + \mathrm{E}\left[R_2^2\right]
\end{aligned}$$

■

An independence condition is necessary. If we ignored independence, then we would conclude that $\text{Var}[R+R] = \text{Var}[R] + \text{Var}[R]$. However, by Theorem 17.4.4, the left side is equal to $4\,\text{Var}[R]$, whereas the right side is $2\,\text{Var}[R]$. This implies that $\text{Var}[R] = 0$, which, by the Lemma above, essentially only holds if R is constant.

The proof of Theorem 17.4.7 carries over straightforwardly to the sum of any finite number of variables. So we have:

Theorem 17.4.8. *[Pairwise Independent Additivity of Variance] If $R_1, R_2, \dots, R_n$ are* pairwise *independent random variables, then*

$$\text{Var}[R_1 + R_2 + \cdots + R_n] = \text{Var}[R_1] + \text{Var}[R_2] + \cdots + \text{Var}[R_n]. \tag{17.16}$$

EDITING NOTE:

Proof. We may assume that $\text{E}[R_i] = 0$ for $i = 1, \dots, n$, since we could always replace R_i by $(R_i - \text{E}[R_i])$ in equation (17.16). This substitution preserves the independence of the variables, and by Theorem 17.4.5, does not change the variances.

Now by Lemma 17.4.1, $\text{Var}[R_i] = \text{E}[R_i^2]$ and

$$\text{Var}[R_1 + R_2 + \cdots + R_n] = \text{E}\left[(R_1 + R_2 + \cdots + R_n)^2\right],$$

so we need only prove

$$\text{E}\left[(R_1 + R_2 + \cdots + R_n)^2\right] = \text{E}\left[R_1^2\right] + \text{E}\left[R_2^2\right] + \cdots + \text{E}\left[R_n^2\right] \tag{17.17}$$

But (17.17) follows from linearity of expectation and the fact that

$$\text{E}[R_i R_j] = \text{E}[R_i]\,\text{E}[R_j] = 0 \cdot 0 = 0 \tag{17.18}$$

for $i \neq j$, since R_i and R_j are independent. Namely,

$$\begin{aligned}
\text{E}\left[(R_1 + R_2 + \cdots + R_n)^2\right] &= \text{E}\left[\sum_{1 \le i,j \le n} R_i R_j\right] \\
&= \sum_{1 \le i,j \le n} \text{E}[R_i R_j] && \text{linearity of E []} \\
&= \sum_{1 \le i \le n} \text{E}\left[R_i^2\right] + \sum_{1 \le i \ne j \le n} \text{E}[R_i R_j] && \text{(rearranging the sum)} \\
&= \sum_{1 \le i \le n} \text{E}\left[R_i^2\right] + \sum_{1 \le i \ne j \le n} 0 && \text{(by (17.18))} \\
&= \text{E}\left[R_1^2\right] + \text{E}\left[R_2^2\right] + \cdots + \text{E}\left[R_n^2\right].
\end{aligned}$$

■

■

Now we have a simple way of computing the variance of a variable, J, that has an (n, p)-binomial distribution. We know that $J = \sum_{k=1}^{n} I_k$ where the I_k are mutually independent indicator variables with $\Pr\{I_k = 1\} = p$. The variance of each I_k is $p(1-p)$ by Lemma 17.4.2, so by linearity of variance, we have

Lemma (Variance of the Binomial Distribution)**.** *If J has the (n, p)-binomial distribution, then*

$$\text{Var}[J] = n\,\text{Var}[I_k] = np(1-p). \tag{17.19}$$

17.4.5 Problems

Practice Problems

Class Problems

Homework Problems

17.5 Estimation by Random Sampling

Polling again

EDITING NOTE:

This paragraph reflects an alternative exposition where polling estimation and confidence were based only on binomial distribution properties, even before expectation was introduced.

In Chapter [none], we used bounds on the binomial distribution to determine confidence levels for a poll of voter preferences of Franken vs. Coleman. Now that we know the variance of the binomial distribution, we can use Chebyshev's

Theorem as an alternative approach to calculate poll size.

The setup is the same as in Chapter [none] 🟥

Suppose we had wanted an advance estimate of the fraction of the Massachusetts voters who favored Scott Brown over everyone else in the recent Democratic primary election to fill Senator Edward Kennedy's seat.

Let p be this unknown fraction, and let's suppose we have some random process —say throwing darts at voter registration lists —which will select each voter with equal probability. We can define a Bernoulli variable, K, by the rule that $K = 1$ if the random voter most prefers Brown, and $K = 0$ otherwise.

Now to estimate p, we take a large number, n, of random choices of voters[1] and count the fraction who favor Brown. That is, we define variables $K_1, K_2, \ldots,$ where K_i is interpreted to be the indicator variable for the event that the ith cho-

[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.

sen voter prefers Brown. Since our choices are made independently, the K_i's are independent. So formally, we model our estimation process by simply assuming we have mutually independent Bernoulli variables $K_1, K_2, \dots,$ each with the same probability, p, of being equal to 1. Now let S_n be their sum, that is,

$$S_n ::= \sum_{i=1}^{n} K_i. \tag{17.20}$$

So S_n has the binomial distribution with parameter n, which we can choose, and unknown parameter p.

The variable S_n/n describes the fraction of voters we will sample who favor Scott Brown. Most people intuitively expect this sample fraction to give a useful approximation to the unknown fraction, p —and they would be right. So we will use the sample value, S_n/n, as our *statistical estimate* of p and use the Pairwise Independent Sampling Theorem 17.5.1 to work out how good an estinate this is.

17.5.1 Sampling

Suppose we want our estimate to be within 0.04 of the Brown favoring fraction, p, at least 95% of the time. This means we want

$$\Pr\left\{ \left| \frac{S_n}{n} - p \right| \leq 0.04 \right\} \geq 0.95 \,. \tag{17.21}$$

So we better determine the number, n, of times we must poll voters so that inequality (17.21) will hold.

EDITING NOTE: 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 Brown.

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 Brown. ■

Now S_n is binomially distributed, so from (17.19) we have

$$\text{Var}\,[S_n] = n(p(1-p)) \leq n \cdot \frac{1}{4} = \frac{n}{4}$$

The bound of $1/4$ follows from the fact that $p(1-p)$ is maximized when $p = 1-p$, that is, when $p = 1/2$ (check this yourself!).

Next, we bound the variance of S_n/n:

$$\begin{aligned} \text{Var}\left[\frac{S_n}{n}\right] &= \left(\frac{1}{n}\right)^2 \text{Var}\,[S_n] && \text{(by (17.11))} \\ &\leq \left(\frac{1}{n}\right)^2 \frac{n}{4} && \text{(by (17.5.1))} \\ &= \frac{1}{4n} && (17.22) \end{aligned}$$

Now from Chebyshev and (17.22) we have:

$$\Pr\left\{\left|\frac{S_n}{n} - p\right| \geq 0.04\right\} \leq \frac{\text{Var}\,[S_n/n]}{(0.04)^2} = \frac{1}{4n(0.04)^2} = \frac{156.25}{n} \qquad (17.23)$$

To make our our estimate with 95% confidence, we want the righthand side of (17.23) to be at most $1/20$. So we choose n so that

$$\frac{156.25}{n} \leq \frac{1}{20},$$

that is,

$$n \geq 3,125.$$

A more exact calculation of the tail of this binomial distribution shows that the above sample size is about four times larger than necessary, but it is still a feasible size to sample. The fact that the sample size derived using Chebyshev's Theorem was unduly pessimistic should not be surprising. After all, in applying the Chebyshev Theorem, we only used the variance of S_n. It makes sense that more detailed information about the distribution leads to better bounds. But working through this example using only the variance has the virtue of illustrating an approach to estimation that is applicable to arbitrary random variables, not just binomial variables.

17.5.2 Matching Birthdays

There are important cases where the relevant distributions are not binomial because the mutual independence properties of the voter preference example do not hold. In these cases, estimation methods based on the Chebyshev bound may be the best approach. Birthday Matching is an example. We already saw in Section 14.5 that in a class of 85 students it is virtually certain that two or more students will have the same birthday. This suggests that quite a few pairs of students are likely to have the same birthday. How many?

So as before, suppose there are n students and d days in the year, and let D be the number of pairs of students with the same birthday. Now it will be easy to calculate the expected number of pairs of students with matching birthdays. Then we can take the same approach as we did in estimating voter preferences to get an estimate of the probability of getting a number of pairs close to the expected number.

Unlike the situation with voter preferences, having matching birthdays for different pairs of students are not mutually independent events, but the matchings are *pairwise independent*, as explained in Section 14.5. as we did for voter preference. Namely, let $B_1, B_2, \ldots, B_n$ be the birthdays of n independently chosen people, and let $E_{i,j}$ be the indicator variable for the event that the ith and jth people chosen have the same birthdays, that is, the event $[B_i = B_j]$. So our probability model, the B_i's are mutually independent variables, the $E_{i,j}$'s are pairwise independent. Also, the expectations of $E_{i,j}$ for $i \neq j$ equals the probability that $B_i = B_j$, namely, $1/d$.

Now, D, the number of matching pairs of birthdays among the n choices is simply the sum of the $E_{i,j}$'s:

$$D ::= \sum_{1 \le i < j \le n} E_{i,j}. \tag{17.24}$$

So by linearity of expectation

$$\mathrm{E}[D] = \mathrm{E}\left[\sum_{1 \le i < j \le n} E_{i,j}\right] = \sum_{1 \le i < j \le n} \mathrm{E}[E_{i,j}] = \binom{n}{2} \cdot \frac{1}{d}.$$

Similarly,

$$\begin{aligned}
\mathrm{Var}[D] &= \mathrm{Var}\left[\sum_{1 \le i < j \le n} E_{i,j}\right] \\
&= \sum_{1 \le i < j \le n} \mathrm{Var}[E_{i,j}] && \text{(by Theorem 17.4.8)} \\
&= \binom{n}{2} \cdot \frac{1}{d}\left(1 - \frac{1}{d}\right). && (by Lemma\ 17.4.2)
\end{aligned}$$

In particular, for a class of $n = 85$ students with $d = 365$ possible birthdays, we have E $[D] \approx 9.7$ and Var $[D] < 9.7(1 - 1/365) < 9.7$. So by Chebyshev's Theorem

$$\Pr\{|D - 9.7| \geq x\} < \frac{9.7}{x^2}.$$

Letting $x = 5$, we conclude that there is a better than 50% chance that in a class of 85 students, the number of pairs of students with the same birthday will be between 5 and 14.

17.5.3 Pairwise Independent Sampling

The reasoning we used above to analyze voter polling and matching birthdays is very similar. We summarize it in slightly more general form with a basic result we call the Pairwise Independent Sampling Theorem. In particular, we do not need to restrict ourselves to sums of zero-one valued variables, or to variables with the same distribution. For simplicity, we state the Theorem for pairwise independent variables with possibly different distributions but with the same mean and variance.

Theorem 17.5.1 (Pairwise Independent Sampling)**.** *Let $G_1, \ldots, G_n$ be pairwise independent variables with the same mean, μ, and deviation, σ. Define*

$$S_n ::= \sum_{i=1}^{n} G_i. \tag{17.25}$$

Then

$$\Pr\left\{\left|\frac{S_n}{n} - \mu\right| \geq x\right\} \leq \frac{1}{n}\left(\frac{\sigma}{x}\right)^2.$$

Proof. We observe first that the expectation of S_n/n is μ:

$$\begin{aligned}
\mathrm{E}\left[\frac{S_n}{n}\right] &= \mathrm{E}\left[\frac{\sum_{i=1}^{n} G_i}{n}\right] && \text{(def of } S_n\text{)} \\
&= \frac{\sum_{i=1}^{n} \mathrm{E}\left[G_i\right]}{n} && \text{(linearity of expectation)} \\
&= \frac{\sum_{i=1}^{n} \mu}{n} \\
&= \frac{n\mu}{n} = \mu.
\end{aligned}$$

The second important property of S_n/n is that its variance is the variance of G_i

divided by n:

$$\begin{aligned}
\text{Var}\left[\frac{S_n}{n}\right] &= \left(\frac{1}{n}\right)^2 \text{Var}\left[S_n\right] && \text{(by (17.11))} \\
&= \frac{1}{n^2}\text{Var}\left[\sum_{i=1}^{n} G_i\right] && \text{(def of } S_n\text{)} \\
&= \frac{1}{n^2}\sum_{i=1}^{n}\text{Var}\left[G_i\right] && \text{(pairwise independent additivity)} \\
&= \frac{1}{n^2}\cdot n\sigma^2 = \frac{\sigma^2}{n}. && (17.26)
\end{aligned}$$

This is enough to apply Chebyshev's Theorem and conclude:

$$\begin{aligned}
\Pr\left\{\left|\frac{S_n}{n} - \mu\right| \geq x\right\} &\leq \frac{\text{Var}\left[S_n/n\right]}{x^2}. && \text{(Chebyshev's bound)} \\
&= \frac{\sigma^2/n}{x^2} && \text{(by (17.26))} \\
&= \frac{1}{n}\left(\frac{\sigma}{x}\right)^2.
\end{aligned}$$

■

The Pairwise Independent Sampling Theorem provides a precise general statement about how the average of independent samples of a random variable approaches the mean. In particular, it proves what is known as the Law of Large

Numbers[2] : by choosing a large enough sample size, we can get arbitrarily accurate estimates of the mean with confidence arbitrarily close to 100%.

Corollary 17.5.2. *[Weak Law of Large Numbers] Let $G_1, \ldots, G_n$ be pairwise independent variables with the same mean, μ, and the same finite deviation, and let*

$$S_n ::= \frac{\sum_{i=1}^{n} G_i}{n}.$$

Then for every $\epsilon > 0$,

$$\lim_{n \to \infty} \Pr\{|S_n - \mu| \leq \epsilon\} = 1.$$

17.6 Confidence versus Probability

So Chebyshev's Bound implies that sampling 3,125 voters will yield a fraction that, 95% of the time, is within 0.04 of the actual fraction of the voting population who prefer Brown.

EDITING NOTE: Estimates of the binomial distribution show that a sample size

[2] This is the *Weak* Law of Large Numbers. As you might suppose, there is also a Strong Law, but it's outside the scope of this text.

around 664 would do. ■

Notice that the actual size of the voting population was never considered because *it did not matter*. People who have not studied probability theory often insist that the population size should matter. But our analysis shows that polling a little over 3000 people people is always sufficient, whether there are ten thousand, or million, or billion . . . voters. You should think about an intuitive explanation that might persuade someone who thinks population size matters.

Now suppose a pollster actually takes a sample of 3,125 random voters to estimate the fraction of voters who prefer Brown, and the pollster finds that 1250 of them prefer Brown. It's tempting, **but sloppy**, to say that this means:

False Claim. *With probability 0.95, the fraction, p, of voters who prefer Brown is* $1250/3125 \pm 0.04$. *Since* $1250/3125 - 0.04 > 1/3$, *there is a 95% chance that more than a third of the voters prefer Brown to all other candidates.*

What's objectionable about this statement is that it talks about the probability or "chance" that a real world fact is true, namely that the actual fraction, p, of voters favoring Brown is more than $1/3$. But p is what it is, and it simply makes no sense to talk about the probability that it is something else. For example, suppose p is actually 0.3; then it's nonsense to ask about the probability that it is within 0.04 of 1250/3125 —it simply isn't.

This example of voter preference is typical: we want to estimate a fixed, unknown real-world quantity. But *being unknown does not make this quantity a random variable*, so it makes no sense to talk about the probability that it has some property.

A more careful summary of what we have accomplished goes this way:

> We have described a probabilistic procedure for estimating the value of the actual fraction, p. The probability that *our estimation procedure* will yield a value within 0.04 of p is 0.95.

This is a bit of a mouthful, so special phrasing closer to the sloppy language is commonly used. The pollster would describe his conclusion by saying that

> At the 95% *confidence level*, the fraction of voters who prefer Brown is $1250/3125 \pm 0.04$.

So confidence levels refer to the results of estimation procedures for real-world quantities. The phrase "confidence level" should be heard as a reminder that some statistical procedure was used to obtain an estimate, and in judging the credibility of the estimate, it may be important to learn just what this procedure was.

EDITING NOTE: Maybe include example from CP_drug_confidence here. ■

17.6.1 Problems

Practice Problems

Class Problems

Exam Problems

17.7 The Chernoff Bound

Fussbook is a new social networking site oriented toward unpleasant people.

Like all major web services, Fussbook has a load balancing problem. Specifically, Fussbook receives 24,000 forum posts every 10 minutes. Each post is assigned to one of m computers for processing, and each computer works sequentially through its assigned tasks. Processing an average post takes a computer $1/4$ second. Some posts, such as pointless grammar critiques and snide witticisms, are easier. But the most protracted harangues require 1 full second.

Balancing the work load across the m computers is vital; if any computer is assigned more than 10 minutes of work in a 10-minute interval, then that computer is overloaded and system performance suffers. That would be bad, because Fussbook users are *not* a tolerant bunch.

An early idea was to assign each computer an alphabetic range of forum topics. ("That oughta work!", one programmer said.) But after the computer handling the "*pr*ivacy" and "*pr*eferred text editor" threads melted, the drawback of an ad hoc approach was clear: there are no guarantees.

If the length of every task were known in advance, then finding a balanced distribution would be a kind of "bin packing" problem. Such problems are hard to solve exactly, though approximation algorithms can come close. But in this case task lengths are not known in advance, which is typical for workload problems.

So the load balancing problem seems sort of hopeless, because there is no data available to guide decisions. Heck, we might as well assign tasks to computers at random!

As it turns out, random assignment not only balances load reasonably well, but also permits provable performance guarantees in place of "That oughta work!" assertions. In general, a randomized approach to a problem is worth considering when a deterministic solution is hard to compute or requires unavailable information.

Some arithmetic shows that Fussbook's traffic is sufficient to keep $m = 10$ computers running at 100% capacity with perfect load balancing. Surely, more than 10 servers are needed to cope with random fluctuations in task length and imperfect load balance. But how many is enough? 11? 15? 20? We'll answer that question with a new mathematical tool.

17.7.1 The Chernoff Bound

The Chernoff bound is a hammer that you can use to nail a great many problems. Roughly, the Chernoff bound says that certain random variables are very unlikely to significantly exceed their expectation. For example, if the expected load on a computer is just a bit below its capacity, then that computer is unlikely to be overloaded, provided the conditions of the Chernoff bound are satisfied.

More precisely, the Chernoff Bound says that *the sum of lots of little, independent random variables is unlikely to significantly exceed the mean*. The Markov and Chebychev bounds lead to the same kind of conclusion but typically provide much weaker conclusions.

EDITING NOTE: In particular, the Markov and Chebychev bounds are polynomial, while the Chernoff bound is exponential. ■

Here is the theorem. The proof is at the end of the chapter.

Theorem 17.7.1 (Chernoff Bound)**.** *Let $T_1, \dots T_n$ be mutually independent random* *variables such that $0 \leq T_i \leq 1$ for all i. Let $T = T_1 + \cdots + T_n$. Then for all $c \geq 1$,*

$$\Pr\{T \geq c \operatorname{E}[T]\} \leq e^{-k \operatorname{E}[T]} \tag{17.27}$$

where $k = c \ln c - c + 1$.

The Chernoff bound applies only to distributions of sums of independent random variables that take on values in the interval $[0, 1]$. The binomial distribution is of course such a distribution, but are lots of other distributions because the Chernoff bound allows the variables in the sum to have differing, arbitrary, and even unknown distributions over the range $[0, 1]$. Furthermore, there is no direct dependence on the number of random variables in the sum or their expectations. In short, the Chernoff bound gives strong results for lots of problems based on little information —no wonder it is widely used!

A Simple Example

The Chernoff bound is pretty easy to apply, though the details can be daunting at first. Let's walk through a simple example to get the hang of it.

What are the odds that the number of heads that come up in 1000 independent tosses of a fair coin exceeds the expectation by 20% or more? Let T_i be an indicator variable for the event that the i-th coin is heads. Then the total number of heads is $T = T_1 + \cdots + T_{1000}$. The Chernoff bound requires that the random variables T_i be mututally independent and take on values in the range $[0, 1]$. Both conditions hold here. In fact, this example is similar to many applications of the Chernoff bound in that every T_i is *either* 0 or 1, since they're indicators.

The goal is to bound the probability that the number of heads exceeds its expectation by 20% or more; that is, to bound $\Pr\{T \geq c\,\mathrm{E}[T]\}$ where c = 1.2. To that end, we compute k as defined in the theorem:

$$k = c \ln c - c + 1 = 0.0187\ldots$$

Plugging this value into the Chernoff bound gives:

$$\begin{aligned}\Pr\{T \geq 1.2\,\mathrm{E}[T]\} &\leq e^{-k\,\mathrm{E}[T]} \\ &= e^{-(0.0187\ldots)\cdot 500} \\ &< 0.0000834\end{aligned}$$

So the probability of getting 20% or more extra heads on 1000 coins is less than 1 in 10,000.

The bound becomes much stronger as the number of coins increases, because the expected number of heads appears in the exponent of the upper bound. For example, the probability of getting at least 20% extra heads on a million coins is at most

$$e^{-(0.0187\dots)\cdot 500000} < e^{-9392}$$

which is pretty darn small.

Alternatively, the bound also becomes stronger for larger deviations. For example, suppose we're interested in the odds of getting 30% or more extra heads in 1000 tosses, rather than 20%. In that case, $c = 1.3$ instead of 1.2. Consequently, the parameter k rises from 0.0187 to about 0.0410, which may seem insignificant. But because k appears in the exponent of the upper bound, the final probability decreases from around 1 in 10,000 to about 1 in a billion!

Pick-4

Pick-4 is a lottery game where you pick a 4-digit number between 0000 and 9999. If your number comes up in a random drawing, then you win. Your chance of winning is 1 in 10,000. And if 10 million people play, then the expected number of winners is 1000. The lottery operator's nightmare is that the number of winners is much greater; say, 2000 or more. What are the odds of that?

Let T_i be an indicator for the event that the i-th player wins. Then $T = T_1 + \cdots + T_n$ is the total number of winners. If we assume that the players' picks and the winning number are independent and uniform, then the indicators T_i are independent, as required by the Chernoff bound.

EDITING NOTE: Add comment about how unrealistic these assumptions are because people frequently play a few favorite numbers.

The assumptions would be plausible for a version where people buy tickets with randomly assigned numbers, so they can't pick their own number. ■

Now, 2000 winners would be twice the expected number. So we choose $c = 2$, compute $k = c \ln c - c + 1 = 0.386\ldots$, and plug these values into the Chernoff bound:

$$\begin{aligned}
\Pr\{T \geq 2000\} &= \Pr\{T \geq 2\,\mathrm{E}[T]\} \\
&\leq e^{-k\,\mathrm{E}[T]} \\
&= e^{-(0.386\ldots)\cdot 1000} \\
&< e^{-386}
\end{aligned}$$

So there is almost no chance that the lottery operator pays out double. In fact, the number of winners won't even be 10% higher than expected very often. To prove that, let $c = 1.1$, compute $k = c \ln c - c + 1 = 0.00484\ldots$, and plug in again:

$$\begin{aligned}
\Pr\{T \geq 1.1\,\mathrm{E}[T]\} &\leq e^{-k\,\mathrm{E}[T]} \\
&= e^{-}0.00484 \cdots * 1000 \\
&< 0.01
\end{aligned}$$

So the Pick-4 lottery may be exciting for the players, but the lottery operator has little doubt about the outcome!

17.7.2 Randomized Load Balancing

Now let's return to Fussbook and its load balancing problem. Specifically, we need to determine how many machines suffice to ensure that no server is overloaded; that is, assigned to do more than 10 minutes of work in a 10-minute interval.

To begin, let's find the probability that the first server is overloaded. Let T_i be the number of seconds that the first server spends on the i-th task. So T_i is zero if the task is assigned to another machine, and otherwise T_i is the length of the task. Then $T = \sum T_i$ is the total length of tasks assigned to the server. We need to upper bound $\Pr\{T \geq 600\}$; that is, the probability that the first server is assigned more than 600 seconds (or, equivalently, 10 minutes) of work.

The Chernoff bound is applicable only if the T_i are mutually independent and take on values in the range $[0, 1]$. The first condition is satisfied if we assume that tasks lengths and assignments are independent. And the second condition is satisfied because processing even the most interminable harangue takes at most 1 second.

In all, there are 24,000 tasks each with an expected length of $1/4$ second. Since tasks are assigned to computers at random, the expected load on the first server is:

$$\begin{aligned} \mathrm{E}[T] &= \frac{24,000 \text{ tasks} \cdot 1/4 \text{ second per task}}{m \text{ machines}} \\ &= 6000/m \text{ seconds} \end{aligned}$$

For example, if there are $m = 10$ machines, then the expected load on the first server is 600 seconds, which is 100% of its capacity.

Now we can use the Chernoff bound to upper bound the probability that the first server is overloaded:

$$\begin{aligned} \Pr\{T \geq 600\} &= \Pr\{T \geq c\,\mathrm{E}[T]\} \\ &\leq e^{-(c \ln c - c + 1) \cdot 6000/m} \end{aligned}$$

Equality holds on the first line when $c = m/10$, since $c\,\mathrm{E}[T] = (m/10) \cdot (6000/m) = 600$. The probability that *some* server is overloaded is at most m times the probability that the first server is overloaded:

$$\Pr\{\text{some server is overloaded}\} \leq m e^{-(c \ln c - c + 1) \cdot 6000/m}$$

Some values of this upper bound are tabulated below:

$$\begin{aligned} m &= 11: \quad 0.784\ldots \\ m &= 12: \quad 0.000999\ldots \\ m &= 13: \quad 0.0000000760\ldots \end{aligned}$$

These values suggest that a system with $m = 11$ machines might suffer immediate overload, $m = 12$ machines could fail in a few days, but $m = 13$ should be fine for a century or two!

17.7.3 Proof of the Chernoff Bound

The proof of the Chernoff bound is somewhat involved. Heck, even *Chernoff* didn't come up with it! His friend, Herman Rubin, showed him the argument. Thinking the bound not very significant, Chernoff did not credit Rubin in print. He felt pretty bad when it became famous!

EDITING NOTE: References: "A Conversation with Herman Chernoff" Statistical Science 1996, Vol 11, No 4, pp 335-350.

Here is the theorem again, for reference:

Theorem 17.7.2 (Chernoff Bound)**.** *Let $T_1, \ldots T_n$ be mutually independent random* *variables such that $0 \leq T_i \leq 1$ for all i. Let $T = T_1 + \cdots + T_n$. Then for all $c \geq 1$,*

$$\Pr\{T \geq c\,\mathrm{E}[T]\} \leq e^{-k\,\mathrm{E}[T]}$$

where $k = c \ln c - c + 1$.

■

For clarity, we'll go through the proof "top down"; that is, we'll use facts that are proved immediately afterward.

Proof. The key step is to exponentiate both sides of the inequality $T > c\,\mathrm{E}[T]$ and then apply the Markov bound.

$$\begin{aligned}
\Pr\{T \geq c\,\mathrm{E}[T]\} &= \Pr\left\{c^T \geq c^{c\,\mathrm{E}[T]}\right\} \\
&\leq \frac{\mathrm{E}\left[c^T\right]}{c^{c\,\mathrm{E}[T]}} && \text{(by Markov)} \\
&\leq \frac{e^{(c-1)\,\mathrm{E}[T]}}{c^{c\,\mathrm{E}[T]}} \\
&= e^{-(c\ln c - c + 1)\,\mathrm{E}[T]}
\end{aligned}$$

In the third step, the numerator is rewritten using the inequality

$$\mathrm{E}\left[c^T\right] \leq e^{(c-1)\,\mathrm{E}[T]}$$

which is proved below in Lemma 17.7.3. The final step is simplification. (Recall that c^c is equal to $e^{c \ln c}$.) ■

Algebra aside, there is a brilliant idea in this proof: in this context, exponentiating somehow supercharges the Markov bound. This is not true in general! One unfortunate side-effect is that we have to bound some nasty expectations involving exponentials in order to complete the proof. This is done in the two lemmas below, where variables take on values as in Theorem 17.7.1.

Lemma 17.7.3.

$$\mathrm{E}\left[c^T\right] \le e^{(c-1)\,\mathrm{E}[T]}$$

Proof.

$$\begin{aligned}
\mathrm{E}\left[c^T\right] &= \mathrm{E}\left[c^{T_1+\cdots+T_n}\right] \\
&= \mathrm{E}\left[c^{T_1}\cdots c^{T_n}\right] \\
&= \mathrm{E}\left[c^{T_1}\right]\cdots \mathrm{E}\left[c^{T_n}\right] \\
&\le e^{(c-1)\,\mathrm{E}[T_1]}\cdots e^{(c-1)\,\mathrm{E}[T_n]} \\
&= e^{(c-1)(\mathrm{E}[T_1]+\cdots+\mathrm{E}[T_n])} \\
&= e^{(c-1)\,\mathrm{E}[T_1+\cdots+T_n]} \\
&= e^{(c-1)\,\mathrm{E}[T]}
\end{aligned}$$

The first step uses the definition of T, and the second is just algegra. The third step uses the fact that the expectation of a product of independent random variables is the product of the expectations. (This is where the requirement that the T_i be independent is used.) Then we bound each term using the inquality

$$\mathrm{E}\left[c^{T_i}\right] \le e^{(c-1)\,\mathrm{E}[T_i]}$$

which is proved in Lemma 17.7.4. The last steps are simplifications using algebra and linearity of expectation. ■

Lemma 17.7.4.

$$\mathrm{E}\left[c^{T_i}\right] \le e^{(c-1)\,\mathrm{E}[T_i]}$$

Proof. All summations below range over values v taken on by the random variable T_i, which are all required to be in the interval $[0, 1]$.

$$\begin{aligned}
\mathrm{E}\left[c^{T_i}\right] &= \sum c^v \Pr\{T_i = v\} \\
&\le \sum (1 + (c-1)v) \Pr\{T_i = v\} \\
&= \sum \Pr\{T_i = v\} + (c-1)v \Pr\{T_i = v\} \\
&= \sum \Pr\{T_i = v\} + \sum (c-1)v \Pr\{T_i = v\} \\
&= 1 + (c-1) \sum v \Pr\{T_i = v\} \\
&= 1 + (c-1)\,\mathrm{E}[T_i] \\
&\le e^{(c-1)\,\mathrm{E}[T_i]}
\end{aligned}$$

The first step uses the definition of expectation. The second step relies on the inequality $c^v \leq 1 + (c-1)v$, which holds for all v in $[0,1]$ and $c \geq 1$. This follows from the general principle that a convex function, namely c^v, is less than the linear function, $1 + (c-1)v$, between their points of intersection, namely $v = 0$ and 1. This inequality is why the variables T_i are restricted to the interval $[0,1]$. We then multiply out inside the summation and split into two sums. The first sum adds the probabilities of all possible outcomes, so it is equal to 1. After pulling the constant $c - 1$ out of the second sum, we're left with the definition of $\text{E}[T_i]$. The final step uses the standard inequality $1 + z \leq e^z$, which holds for all real z. ■

EDITING NOTE: Add problems ■

Index