Mathematics for Computer Science

revised Wednesday 8th September, 2010, 00:40

Eric Lehman
Google Inc.

F Thomson Leighton
Department of Mathematics and CSAIL, MIT
Akamai Technologies

Albert R Meyer
Massachusets Institute of Technology

Contents

I Proofs

1 Propositions ***5***

1.1 Compound Propositions 6
1.2 Propositional Logic in Computer Programs 10
1.3 Predicates and Quantifiers 11
1.4 Validity 19
1.5 Satisfiability 21

2 Patterns of Proof ***23***

2.1 The Axiomatic Method 23
2.2 Proof by Cases 26
2.3 Proving an Implication 27
2.4 Proving an "If and Only If" 30
2.5 Proof by Contradiction 32
2.6 Proofs about Sets 33
2.7 *Good* Proofs in Practice 40

3 Induction ***43***

3.1 The Well Ordering Principle 43
3.2 Ordinary Induction 46
3.3 Invariants 56
3.4 Strong Induction 64
3.5 Structural Induction 69

4 Number Theory ***81***

4.1 Divisibility 81
4.2 The Greatest Common Divisor 87
4.3 The Fundamental Theorem of Arithmetic 94
4.4 Alan Turing 96
4.5 Modular Arithmetic 100
4.6 Arithmetic with a Prime Modulus 103
4.7 Arithmetic with an Arbitrary Modulus 108
4.8 The RSA Algorithm 113

II Structures

5 Graph Theory *121*

5.1 Definitions 121
5.2 Matching Problems 128
5.3 Coloring 143
5.4 Getting from A to B in a Graph 147
5.5 Connectivity 151
5.6 Around and Around We Go 156
5.7 Trees 162
5.8 Planar Graphs 170

6 Directed Graphs *189*

6.1 Definitions 189
6.2 Tournament Graphs 192
6.3 Communication Networks 196

7 Relations and Partial Orders *213*

7.1 Binary Relations 213
7.2 Relations and Cardinality 217
7.3 Relations on One Set 220
7.4 Equivalence Relations 222
7.5 Partial Orders 225
7.6 Posets and DAGs 226
7.7 Topological Sort 229
7.8 Parallel Task Scheduling 232
7.9 Dilworth's Lemma 235

8 State Machines *237*

III Counting

9 Sums and Asymptotics *243*

9.1 The Value of an Annuity 244
9.2 Power Sums 250
9.3 Approximating Sums 252
9.4 Hanging Out Over the Edge 257
9.5 Double Trouble 269
9.6 Products 272

9.7 Asymptotic Notation 275

10 Recurrences *283*

10.1 The Towers of Hanoi 284
10.2 Merge Sort 291
10.3 Linear Recurrences 294
10.4 Divide-and-Conquer Recurrences 302
10.5 A Feel for Recurrences 309

11 Cardinality Rules *313*

11.1 Counting One Thing by Counting Another 313
11.2 Counting Sequences 314
11.3 The Generalized Product Rule 317
11.4 The Division Rule 321
11.5 Counting Subsets 324
11.6 Sequences with Repetitions 326
11.7 Counting Practice: Poker Hands 329
11.8 Inclusion-Exclusion 334
11.9 Combinatorial Proofs 339
11.10 The Pigeonhole Principle 342
11.11 A Magic Trick 346

12 Generating Functions *355*

12.1 Definitions and Examples 355
12.2 Operations on Generating Functions 356
12.3 Evaluating Sums 361
12.4 Extracting Coefficients 363
12.5 Solving Linear Recurrences 370
12.6 Counting with Generating Functions 374

13 Infinite Sets *379*

13.1 Injections, Surjections, and Bijections 379
13.2 Countable Sets 381
13.3 Power Sets Are Strictly Bigger 384
13.4 Infinities in Computer Science 386

IV Probability

14 Events and Probability Spaces *391*

14.1 Let's Make a Deal 391
14.2 The Four Step Method 392

14.3 Strange Dice 402
14.4 Set Theory and Probability 411
14.5 Infinite Probability Spaces 413

15 Conditional Probability *417*

15.1 Definition 417
15.2 Using the Four-Step Method to Determine Conditional Probability 418
15.3 *A Posteriori* Probabilities 424
15.4 Conditional Identities 427

16 Independence *431*

16.1 Definitions 431
16.2 Independence Is an Assumption 432
16.3 Mutual Independence 433
16.4 Pairwise Independence 435
16.5 The Birthday Paradox 438

17 Random Variables and Distributions *445*

17.1 Definitions and Examples 445
17.2 Distribution Functions 450
17.3 Bernoulli Distributions 452
17.4 Uniform Distributions 453
17.5 Binomial Distributions 456

18 Expectation *467*

18.1 Definitions and Examples 467
18.2 Expected Returns in Gambling Games 477
18.3 Expectations of Sums 483
18.4 Expectations of Products 490
18.5 Expectations of Quotients 492

19 Deviations *497*

19.1 Variance 497
19.2 Markov's Theorem 507
19.3 Chebyshev's Theorem 513
19.4 Bounds for Sums of Random Variables 516
19.5 Mutually Independent Events 523

20 Random Walks *533*

20.1 Unbiased Random Walks 533
20.2 Gambler's Ruin 542
20.3 Walking in Circles 549

I Proofs

Introduction

This text explains how to use mathematical models and methods to analyze problems that arise in computer science. The notion of a proof plays a central role in this work.

Simply put, a proof is a method of establishing truth. Like beauty, "truth" sometimes depends on the eye of the beholder, 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 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

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

even in masterful hands, this approach is not reliable.

Mathematics has its own specific notion of "proof."

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

The three key ideas in this definition are highlighted: proposition, logical deduction, and axiom. These three ideas are explained in 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" (that is, arguments that look like a proof but that contain missteps, or deductions that aren't so logical when examined closely). False proofs are often even more important as examples than correct proofs, because they are uniquely helpful with honing your skills at making sure each step of a proof follows logically from prior steps.

Creating a good proof is a lot like creating a beautiful work of art. In fact, mathematicians often refer to really good proofs as being "elegant" or "beautiful." As with any endeavor, it will probably take a little practice before your fellow students use such praise when referring to your proofs, but to get you started in the right direction, we will provide templates for the most useful proof techniques in Chapters 2 and 3. We then apply these techniques in Chapter 4 to establish some important facts about numbers; facts that form the underpinning of one of the world's most widely-used cryptosystems.

1 Propositions

Definition. A *proposition* is a 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, consider the following statements:

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, 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 —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 IMPLIES 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	**Symbolic 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	$P \longleftrightarrow Q$

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 symbols $\vee$ and $\longrightarrow$, and their meaning is easy to remember. We will use the prior notation for the most part in this text, but 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. Let P be the proposition "I am hungry", and let Q be "I am grumpy". The first sentence says "P IMPLIES Q" and the second says "NOT(Q) IMPLIES NOT(P)". We can compare these two statements in a truth table:

P	Q	P IMPLIES Q	NOT(Q) IMPLIES NOT(P)
T	**T**	**T**	**T**
T	**F**	**F**	**F**
F	**T**	**T**	**T**
F	**F**	**T**	**T**

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

is called the *contrapositive* of 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. 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)	P IFF Q
T	**T**	**T**	**T**	**T**	**T**
T	**F**	**F**	**T**	**F**	**F**
F	**T**	**T**	**F**	**F**	**F**
F	**F**	**T**	**T**	**T**	**T**

1.2 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.

1.3 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 straightforward 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[1]

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

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 the checking by evaluating $p(0) = 41$, which is prime. $p(1) = 43$ is also prime. So is $p(2) = 47$, $p(3) = 53$, ..., and $p(20) = 461$, all of which are prime. Hmmm... It is starting to look like $p(n)$ is a prime for every nonnegative integer n. In fact, continued checking reveals that $p(n)$ is prime for all $n \leq 39$. 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!

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 (or even many), 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.

[1] 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.

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 this proposition to be true in 1769. It was checked by humans and then by computers for many values of a, b, c, and d over the next two centuries. 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.

Of course, not all propositions 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[2] regions have different colors.*

The proof of this proposition is difficult and took over a century to perfect. Along the way, many incorrect proofs were proposed, including one that stood for 10 years

[2] 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.

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 intelligible proof of the Four-Color Theorem was found, though a computer is still needed to check the colorability of several hundred special maps.[3]

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

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

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

[3] 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.

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 More 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

$$[P \text{ AND } (Q \text{ OR } R)] \text{ IFF } [(P \text{ AND } Q) \text{ OR } (P \text{ AND } R)] \tag{1.6}$$

is valid. This can be verified by checking the truth table for P AND $(Q$ OR $R)$ and $(P$ AND $Q)$ OR $(P$ AND $R)$:

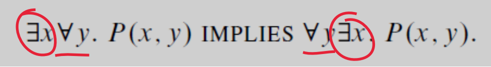
P	Q	R	P AND $(Q$ OR $R)$	$(P$ AND $Q)$ OR $(P$ AND $R)$
T	**T**	**T**	**T**	**T**
T	**T**	**F**	**T**	**T**
T	**F**	**T**	**T**	**T**
T	**F**	**F**	**F**	**F**
F	**T**	**T**	**F**	**F**
F	**T**	**F**	**F**	**F**
F	**F**	**T**	**F**	**F**
F	**F**	**F**	**F**	**F**

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

$$\exists x \forall y.\ P(x,y) \text{ IMPLIES } \forall y \exists x.\ P(x,y). \tag{1.7}$$

Here's an explanation why this is valid:



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

Let D be the domain for the variables and P_0 be some binary predicate[4] on D. We need to show that if

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

holds under this interpretation, then so does

$$\forall y \in D\ \exists x \in D.\ P_0(x,y). \tag{1.9}$$

So suppose (1.8) is true. Then by definition of $\exists$, this means that some element $d_0 \in D$ has the property that

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

By definition of $\forall$, this means that

$$P_0(d_0,d)$$

is true for all $d \in D$. So given any $d \in D$, there is an element in D, namely, d_0, such that $P_0(d_0,d)$ is true. But that's exactly what (1.9) means, so we've proved that (1.9) 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 (1.7), 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 (1.7) into English and then appeal to the meaning, in English, of “for all” and “there exists” as justification.

In contrast to (1.7), the formula

$$\forall y \exists x.\ P(x,y) \text{ IMPLIES } \exists x \forall y.\ P(x,y). \quad ✗ \tag{1.10}$$

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.

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

1.5 Satisfiability

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

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

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

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.

2.1.1 Our Axioms

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

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

2.1.2 Logical Deductions

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

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

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

Rule 2.1.1.

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

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 2.1.2.

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

Rule 2.1.3.

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

Rule 2.1.4.

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

On the other hand,

Non-Rule.

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

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.

2.1.3 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 go through several of these standard patterns, pointing out the basic idea and common pitfalls and giving some examples. Many of these templates fit together; one may give you a top-level outline while others help you at the next level of detail. And we'll show you other, more sophisticated proof techniques 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 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 IMPLIES Q is equivalent to its contrapositive NOT(Q) IMPLIES NOT(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 IMPLIES Q is true if and only if NOT(Q) IMPLIES NOT(P) is true. For example, if $P = \mathbf{T}$ and $Q = \mathbf{F}$, then both P IMPLIES Q and NOT(Q) IMPLIES NOT(P) are false, thereby establishing that $(P$ IMPLIES $Q)$IFF(NOT(Q) IMPLIES NOT$(P))$ is true for this case. If a proposition is true in every possible case, then it is true.

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 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:

[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.

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. ■

2.3 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.

2.3.1 Method #1: Assume P is true

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, **F** 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 2.3.1. *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.

Potential 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 2.3.1 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 2.3.1.

2.3.2 Method #2: Prove the Contrapositive

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

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

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 2.3.2. *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. ■

2.4 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".

2.4.1 Method #1: Prove Each Statement Implies the Other

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

1. Write, "We prove P implies Q and vice-versa."
2. Write, "First, we show P implies Q." Do this by one of the methods in Section 2.3.
3. Write, "Now, we show Q implies P." Again, do this by one of the methods in Section 2.3.

2.4.2 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 2.4.1. *The standard deviation of a sequence of values $x_1, \ldots, x_n$ is zero iff all the values are equal to the mean.*

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 2.4.1 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.

■

2.5 Proof by Contradiction

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

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

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

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

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 2.5.1. *$\sqrt{2}$ is irrational.*

Proof. We use proof by contradiction. Suppose the claim is false; that is, $\sqrt{2}$ is rational. Then we can write $\sqrt{2}$ as a fraction n/d where n and d are positive integers. Furthermore, let's take n and d so that n/d is in *lowest terms* (that is, so that 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. ■

Potential Pitfall

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 2.3.2, 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.

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 intermediate results after a proof by contradiction is completed (for example, that n is even after the proof of Theorem 2.5.1). There was not much risk of that happening in the proof of Theorem 2.5.1, 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 propositions only follow from a (false) assumption in a proof by contradiction.

2.6 Proofs about Sets

Sets are simple, flexible, and everywhere. You will find some set mentioned in nearly every section of this text. In fact, we have already talked about a lot of sets: the set of integers, the set of real numbers, and the set of positive even numbers, to name a few.

In this section, we'll see how to prove basic facts about sets. We'll start with some definitions just to make sure that you know the terminology and that you are comfortable working with sets.

2.6.1 Definitions

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:

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

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

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

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.

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:

$$
\begin{aligned}
X &::= \{1, 2, 3\} \\
Y &::= \{2, 3, 4\}
\end{aligned}
$$

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

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

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

The 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, $A \subseteq \overline{B}$.

Cardinality

The *cardinality* of a set A is the number of elements in A and is denoted by $|A|$. For example,

$$\begin{aligned}
&|\emptyset| = 0, \\
&|\{1, 2, 4\}| = 3, \text{ and} \\
&|\mathbb{N}| \text{ is infinite.}
\end{aligned}$$

The 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 $\mathcal{P}(A)$. In other words, if A is finite, then $|\mathcal{P}(A)| = 2^{|A|}$. For this reason, some authors use the notation 2^A instead of $\mathcal{P}(A)$ to denote the power set of A.

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 and $\emptyset$ for the empty set.

Cross Products

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:

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

2.6.2 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:

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

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

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

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

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

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

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

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

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

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

2.6.3 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 often be formulated and proved as "iff" theorems. For example:

Theorem 2.6.1 (*Distributive Law* for Sets)**.** *Let A, B, and C be sets. Then:*

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

Proof. The equality (2.5) is equivalent to the assertion that

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

for all z. This assertion looks very similar to the Distributive Law for AND and OR that we proved in Section 1.4 (equation 1.6). Namely, if P, Q, and R are propositions, then

$$[P \text{ AND } (Q \text{ OR } R)] \text{ IFF } [(P \text{ AND } Q) \text{ OR } (P \text{ AND } R)] \tag{2.7}$$

Using this fact, we can now prove (2.6) by a chain of iff's:

$$\begin{aligned}
&z \in A \cap (B \cup C) \\
&\quad \text{iff} \quad (z \in A) \text{ AND } (z \in B \cup C) && \text{(def of } \cap) \\
&\quad \text{iff} \quad (z \in A) \text{ AND } (z \in B \text{ OR } z \in C) && \text{(def of } \cup) \\
&\quad \text{iff} \quad (z \in A \text{ AND } z \in B) \text{ OR } (z \in A \text{ AND } z \in C) && \text{(equation 2.7)} \\
&\quad \text{iff} \quad (z \in A \cap B) \text{ OR } (z \in A \cap C) && \text{(def of } \cap) \\
&\quad \text{iff} \quad z \in (A \cap B) \cup (A \cap C) && \text{(def of } \cup)
\end{aligned}$$

■

Many other set equalities can be derived from other valid propositions and proved in an analogous manner. In particular, propositions such as P, Q and R are replaced with sets such as A, B, and C, AND ($\wedge$) is replaced with intersection ($\cap$), OR ($\vee$) is replaced with union ($\cup$), NOT is replaced with complement (for example, $\overline{P}$ would become $\overline{A}$), and IFF becomes set equality ($=$). Of course, you should always check that any alleged set equality derived in this manner is indeed true.

2.6.4 Russell's Paradox and the Logic of Sets

Reasoning naively about sets can sometimes be tricky. In fact, one of the earliest attempts to come up with precise axioms for sets by a late nineteenth century logician named Gotlob *Frege* was shot down by a three line argument known as *Russell's Paradox*:[4] This was an astonishing blow to efforts to provide an axiomatic foundation for mathematics.

Russell's Paradox

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

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

So by definition, for any set S,

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

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*

[4] 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.

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*.

Over time, more efficient axiom systems were developed and today, it is generally agreed that, using some simple logical deduction rules, essentially all of mathematics can be derived from the Axioms of Zermelo-Frankel Set Theory with Choice (ZFC). We are *not* going to be working with these axioms in this course, but just in case you are interested, we have included them as a sidebar below.

The ZFC axioms avoid Russell's Paradox because they imply that no set is ever a member of itself. Unfortunately, this does not necessarily mean that there are not other paradoxes lurking around out there, just waiting to be uncovered by future mathematicians.

ZFC Axioms

***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.

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

$$\forall x.\, x \neq \emptyset \text{ IMPLIES } \exists m.\, \text{member-minimal}(m, x).$$

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.

$$\begin{aligned}
\exists y \forall z \forall w \quad & ((z \in w \text{ AND } w \in x) \text{ IMPLIES} \\
& \quad \exists v \exists u (\exists t ((u \in w \text{AND} \qquad w \in t) \quad \text{AND}(u \in t \text{ AND } t \in y)) \\
& \qquad\qquad\qquad\qquad\qquad\qquad \text{IFF} u = v))
\end{aligned}$$

2.7 *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!

3 Induction

Now that you understand the basics of how to prove that a proposition is true, it is time to equip you with the most powerful methods we have for establishing truth: the Well Ordering Principle, the Induction Rule, and Strong Induction. These methods are especially useful when you need to prove that a predicate is true for all natural numbers.

Although the three methods look and feel different, it turns out that they are equivalent in the sense that a proof using any one of the methods can be automatically reformatted so that it becomes a proof using any of the other methods. The choice of which method to use is up to you and typically depends on whichever seems to be the easiest or most natural for the problem at hand.

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[1] 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, to prove that “$P(n)$ is true for all $n \in \mathbb{N}$” using the Well Ordering Principle, you can take the following steps:

- Define the set, C, of *counterexamples* to P being true. Namely, define[2]

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

- Use a proof by contradiction and assume 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

[1] This means that you are about to see an informal proof by contradiction.

[2] As we learned in Section 2.6.2, the notation $\{n \mid P(n) \text{ is false}\}$ means “the set of all elements n, for which $P(n)$ is false.

3.1.3 Examples

Let's use this this template to prove

Theorem 3.1.1.

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

for all nonnegative integers, n.

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

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

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

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

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

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

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

OK, back to the proof:

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

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

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. ■

Here is another result that can be proved using Well Ordering. It will be useful in Chapter 4 when we study number theory and cryptography.

Theorem 3.1.2. *Every natural number can be factored as a product of primes.*

Proof. By contradiction and Well Ordering. Assume that the theorem is false and let C be the set of all integers greater than one that cannot be factored as a product of primes. We assume that 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 is not empty must therefore be false. ■

3.2 Ordinary 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 A Rule for Ordinary Induction

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

The Principle of Induction.

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

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

then

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

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

Rule. ***Induction Rule***

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

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

3.2.2 A Familiar Example

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 integers 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 we establish two simpler facts:

[3] But see section 3.2.7.

- $P(0)$ 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 from Section 2.3: *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.

3.2.3 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 proposition that you are trying to prove, as in the example above. The predicate $P(n)$ is called the *induction hypothesis*. Sometimes the induction hypothesis will involve several variables, in which case you should indicate which variable serves as n.

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

4. **Prove that $P(n)$ implies $P(n+1)$ for every nonnegative integer n.** This is called the *inductive step*. The basic plan is always the same: assume that $P(n)$ is true and then use this assumption to prove that $P(n+1)$ is true. These two statements should be fairly similar, but bridging the gap may require some ingenuity. Whatever argument you give must be valid for every nonnegative integer n, since the goal is to prove the implications $P(0) \rightarrow P(1)$, $P(1) \rightarrow P(2)$, $P(2) \rightarrow P(3)$, etc. all at once.

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

Always be sure to explicitly label the *base case* and the *inductive step*. It will make your proofs clearer, and it will decrease the chance that you forget a key step (such as checking the base case).

3.2.4 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 of Theorem 3.2.1. 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. ■

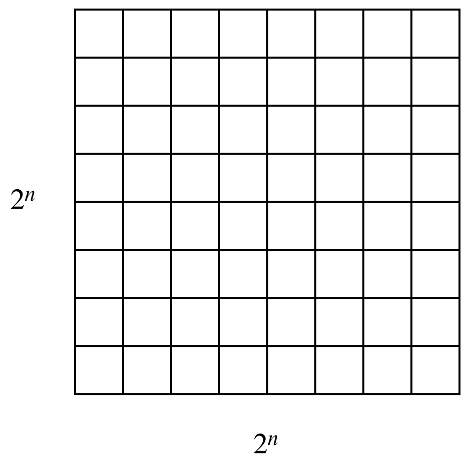


Figure 3.1 A $2^n \times 2^n$ courtyard for $n = 3$.

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 be covered in Part III of the text.

3.2.5 A More Challenging Example

During the development of MIT's famous Stata Center, as costs rose further and further beyond budget, there were some radical fundraising ideas. One rumored plan was to install a big courtyard with dimensions $2^n \times 2^n$ (as shown in Figure 3.1 for the case where $n = 3$) and to have one of the central squares[4] be occupied by a statue of a wealthy potential donor (who we will refer to as "Bill", for the purposes of preserving anonymity). A complication was that the building's unconventional architect, Frank Gehry, was alleged to require that only special L-shaped tiles (show in Figure 3.2) be used for the courtyard. It was quickly determined that a courtyard meeting these constraints exists, at least for $n = 2$. (See Figure 3.3.) But what about for larger values of n? Is there a way to tile a $2^n \times 2^n$ courtyard with L-shaped tiles around 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.

[4] In the special case $n = 0$, the whole courtyard consists of a single central square; otherwise, there are four central squares.

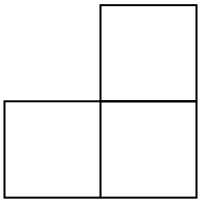


Figure 3.2 The special L-shaped tile.

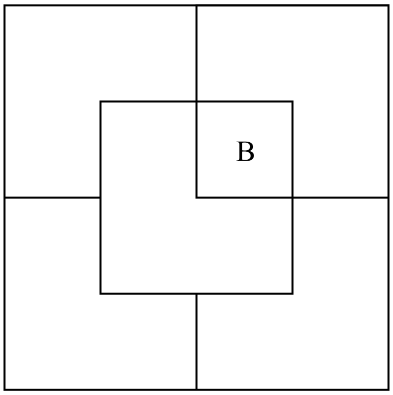


Figure 3.3 A tiling using L-shaped tiles for $n = 2$ with Bill in a center square.

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 as shown in Figure 3.4.

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 $n \geq 0$. Thus $P(m)$ is true for all $n \in \mathbb{N}$, and the theorem follows as a special case where we put Bill in a central square. ■

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

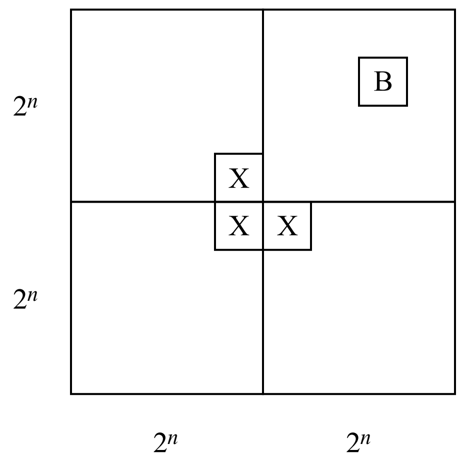


Figure 3.4 Using a stronger inductive hypothesis to prove Theorem 3.2.2.

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"[5]. 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 was easy!

3.2.6 A Faulty Induction Proof

If we have done a good job in writing this text, right about now you should be thinking, "Hey, this induction stuff isn't so hard after all—just show $P(0)$ is true and that $P(n)$ implies $P(n + 1)$ for any number n." And, you would be right, although sometimes when you start doing induction proofs on your own, you can run into trouble. For example, we will now attempt to ruin your day by using induction to "prove" that all horses are the same color. And just when you thought it was safe to skip class and work on your robot program instead. Bummer!

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

[5] 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 Part II of the text.

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.

Bogus 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}—that is, $h_2, \ldots, h_n$)—and likewise h_{n+1} is the same color as the remaining horses besides h_1–$h_2, \ldots, h_n$. Since h_1 and h_{n+1} are the same color as $h_2, \ldots, h_n$, 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 first error in this argument is in the sentence that begins "So h_1 is the same color as the remaining horses besides h_{n+1}—$h_2, \ldots, h_n$)..."

The "..." notation in the expression "$h_1, h_2, \ldots, h_n, h_{n+1}$" creates the impression that there are some remaining horses (namely $h_2, \ldots, h_n$) besides h_1 and h_{n+1}. However, this is not true when $n = 1$. In that case, $h_1, h_2, \ldots, h_n, h_{n+1} =$

h_1, h_2 and there are no remaining horses besides h_1 and h_{n+1}. 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 sets of n non-uniformly-colored horses for all $n \geq 2$.

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.

3.2.7 Induction versus Well Ordering

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

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

3.3 Invariants

One of the most important uses of induction in computer science involves proving that a program or process preserves one or more desirable properties as it proceeds. A property that is preserved through a series of operations or steps is known as an *invariant*. Examples of desirable invariants include properties such as a variable never exceeding a certain value, the altitude of a plane never dropping below 1,000 feet without the wingflaps and landing gear being deployed, and the temperature of a nuclear reactor never exceeding the threshold for a meltdown.

We typically use induction to prove that a proposition is an invariant. In particular, we show that the proposition is true at the beginning (this is the base case) and that if it is true after t steps have been taken, it will also be true after step $t + 1$ (this is the inductive step). We can then use the induction principle to conclude that the proposition is indeed an invariant, namely, that it will always hold.

3.3.1 A Simple Example: The Diagonally-Moving Robot

Invariants are useful in systems that have a *start state* (or *starting configuration*) and a well-defined series of *steps* during which the system can change state.[6] For example, suppose that you have a robot that can walk across diagonals on an infinite 2-dimensional grid. The robot starts at position $(0,0)$ and at each step it moves up or down by 1 unit vertically and left or right by 1 unit horizontally. To be clear, the robot must move by exactly 1 unit in each dimension during each step, since it can only traverse diagonals.

In this example, the *state* of the robot at any time can be specified by a coordinate pair (x, y) that denotes the robot's position. The *start state* is $(0,0)$ since it is given that the robot starts at that position. After the first step, the robot could be in states $(1,1)$, $(1,-1)$, $(-1,1)$, or $(-1,-1)$. After two steps, there are 9 possible states for the robot, including $(0,0)$.

Can the robot ever reach position $(1,0)$?

After playing around with the robot for a bit, it will become apparent that the robot will never be able to reach position $(1,0)$. This is because the robot can only reach positions (x, y) for which $x + y$ is even. This crucial observation quickly leads to the formulation of a predicate

$$P(t) :: \text{if the robot is in state } (x, y) \text{ after } t \text{ steps, then } x + y \text{ is even}$$

which we can prove to be an invariant by induction.

Theorem 3.3.1. *The sum of robot's coordinates is always even.*

Proof. We will prove that P is an invariant by induction.

$P(0)$ is true since the robot starts at $(0,0)$ and $0 + 0$ is even.

Assume that $P(t)$ is true for the inductive step. Let (x, y) be the position of the robot after t steps. Since $P(t)$ is assumed to be true, we know that $x + y$ is even. There are four cases to consider for step $t + 1$, depending on which direction the robot moves.

Case 1 The robot moves to $(x + 1, y + 1)$. Then the sum of the coordinates is $x + y + 2$, which is even, and so $P(t + 1)$ is true.

Case 2 The robot moves to $(x+1, y-1)$. The the sum of the coordinates is $x+y$, which is even, and so $P(t + 1)$ is true.

[6] Such systems are known as state machines and we will study them in greater detail in Chapter 8.

Case 3 The robot moves to $(x-1, y+1)$. The the sum of the coordinates is $x+y$, as with Case 2, and so $P(t+1)$ is true.

Case 4 The robot moves to $(x-1, y-1)$. The the sum of the coordinates is $x+y-2$, which is even, and so $P(t+1)$ is true.

In every case, $P(t+1)$ is true and so we have proved $P(t)$ IMPLIES $P(t+1)$ and so, by induction, we know that $P(t)$ is true for all $t \geq 0$. ■

Corollary 3.3.2. *The robot can never reach position* $(1, 0)$.

Proof. By Theorem 3.3.1, we know the robot can only reach positions with coordinates that sum to an even number, and thus it cannot reach position $(1, 0)$. ■

Since this was the first time we proved that a predicate was an invariant, we were careful to go through all four cases in gory detail. As you become more experienced with such proofs, you will likely become more brief as well. Indeed, if we were going through the proof again at a later point in the text, we might simply note that the sum of the coordinates after step $t+1$ can be only $x+y$, $x+y+2$ or $x+y-2$ and therefore that the sum is even.

3.3.2 The Invariant Method

In summary, if you would like to prove that some property NICE holds for every step of a process, then it is often helpful to use the following method:

- Define $P(t)$ to be the predicate that NICE holds immediately after step t.
- Show that $P(0)$ is true, namely that NICE holds for the start state.
- Show that

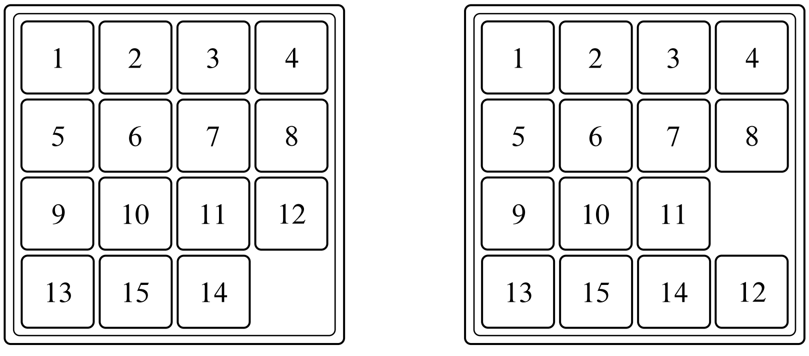
  $$\forall t \in \mathbb{N}.\ P(t) \text{ IMPLIES } P(t+1),$$

  namely, that for any $t \geq 0$, if NICE holds immediately after step t, it must also hold after the following step.

3.3.3 A More Challenging Example: The 15-Puzzle

In the late 19th century, Noyes Chapman, a postmaster in Canastota, New York, invented the 15-puzzle[7], which consisted of a 4×4 grid containing 15 numbered blocks in which the 14-block and the 15-block were out of order. The objective was to move the blocks one at a time into an adjacent hole in the grid so as to eventually

[7] Actually, there is a dispute about who really invented the 15-puzzle. Sam Lloyd, a well-known puzzle designer, claimed to be the inventor, but this claim has since been discounted.



(a) (b)

Figure 3.5 The 15-puzzle in its starting configuration (a) and after the 12-block is moved into the hole below (b).

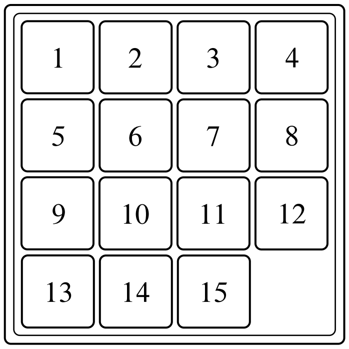
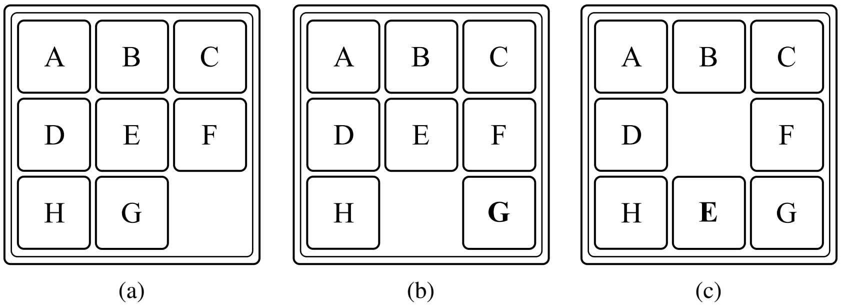


Figure 3.6 The desired final configuration for the 15-puzzle. Can it be achieved by only moving one block at a time into an adjacent hole?

get all 15 blocks into their natural order. A picture of the 15-puzzle is shown in Figure 3.5 along with the configuration after the 12-block is moved into the hole below. The desired final configuration is shown in Figure 3.6.

The 15-puzzle became very popular in North America and Europe and is still sold in game and puzzle shops today. Prizes were offered for its solution, but it is doubtful that they were ever awarded, since it is impossible to get from the configuration in Figure 3.5(a) to the configuration in Figure 3.6 by only moving one block at a time into an adjacent hole. The proof of this fact is a little tricky so we have left it for you to figure out on your own! Instead, we will prove that the analogous task for the much easier 8-puzzle cannot be performed. Both proofs, of course, make use of the Invariant Method.



(a) (b) (c)

Figure 3.7 The 8-Puzzle in its initial configuration (a) and after one (b) and two (c) possible moves.

3.3.4 The 8-Puzzle

In the 8-Puzzle, there are 8 lettered tiles (A–H) and a blank square arranged in a 3 × 3 grid. Any lettered tile adjacent to the blank square can be slid into the blank. For example, a sequence of two moves is illustrated in Figure 3.7.

In the initial configuration shown in Figure 3.7(a), the G and H tiles are out of order. We can find a way of swapping G and H so that they are in the right order, but then other letters may be out of order. Can you find a sequence of moves that puts these two letters in correct order, but returns every other tile to its original position? Some experimentation suggests that the answer is probably “no,” and we will prove that is so by finding an invariant, namely, a property of the puzzle that is always maintained, no matter how you move the tiles around. If we can then show that putting all the tiles in the correct order would violate the invariant, then we can conclude that the puzzle cannot be solved.

Theorem 3.3.3. *No sequence of legal moves transforms the configuration in Figure 3.7(a) into the configuration in Figure 3.8.*

We’ll build up a sequence of observations, stated as lemmas. Once we achieve a critical mass, we’ll assemble these observations into a complete proof of Theorem 3.3.3.

Define a *row move* as a move in which a tile slides horizontally and a *column move* as one in which the tile slides vertically. Assume that tiles are read top-to-bottom and left-to-right like English text, that is, the *natural order*, defined as follows: So when we say that two tiles are “out of order”, we mean that the larger letter precedes the smaller letter in this natural order.

Our difficulty is that one pair of tiles (the G and H) is out of order initially. An immediate observation is that row moves alone are of little value in addressing this

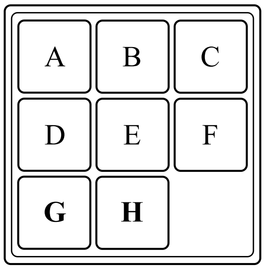
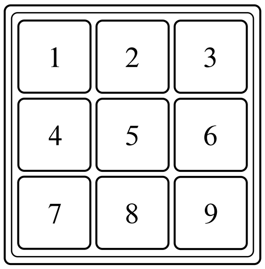


Figure 3.8 The desired final configuration of the 8-puzzle.



problem:

Lemma 3.3.4. *A row move does not change the order of the tiles.*

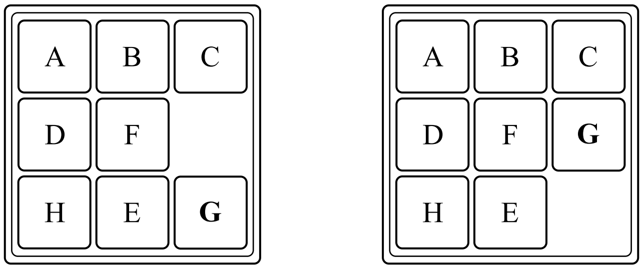
Proof. A row move moves a tile from cell i to cell $i + 1$ or vice versa. This tile does not change its order with respect to any other tile. Since no other tile moves, there is no change in the order of any of the other pairs of tiles. ■

Let’s turn to column moves. This is the more interesting case, since here the order can change. For example, the column move in Figure 3.9 changes the relative order of the pairs (G, H) and (G, E).

Lemma 3.3.5. *A column move changes the relative order of exactly two pairs of tiles.*

Proof. Sliding a tile down moves it after the next two tiles in the order. Sliding a tile up moves it before the previous two tiles in the order. Either way, the relative order changes between the moved tile and each of the two tiles it crosses. The relative order between any other pair of tiles does not change. ■

These observations suggest that there are limitations on how tiles can be swapped. Some such limitation may lead to the invariant we need. In order to reason about swaps more precisely, let’s define a term referring to a pair of items that are out of order:

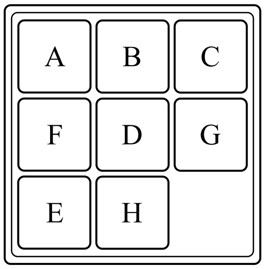


(a) (b)

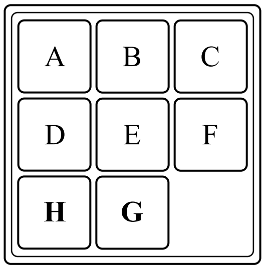
Figure 3.9 An example of a column move in which the G-tile is moved into the adjacent hole above. In this case, G changes order with E and H.

Definition 3.3.6. A pair of letters L_1 and L_2 is an *inversion* if L_1 precedes L_2 in the alphabet, but L_1 appears after L_2 in the puzzle order.

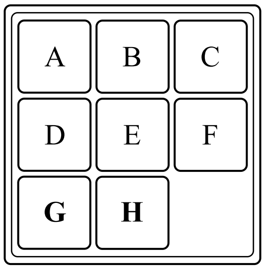
For example, in the puzzle below, there are three inversions: (D, F), (E, F), (E, G).



There is exactly one inversion (G, H) in the start state:



There are no inversions in the end state:



Let's work out the effects of row and column moves in terms of inversions.

Lemma 3.3.7. *During a move, the number of inversions can only increase by 2, decrease by 2, or remain the same.*

Proof. By Lemma 3.3.4, a row move does not change the order of the tiles, and so a row move does not change the number of inversions.

By Lemma 3.3.5, a column move changes the relative order of exactly 2 pairs of tiles. There are three cases: If both pairs were originally in order, then the number of inversions after the move goes up by 2. If both pairs were originally inverted, then the number of inversions after the move goes down by 2. If one pair was originally inverted and the other was originally in order, then the number of inversions stays the same (since changing the former pair makes the number of inversions smaller by 1, and changing the latter pair makes the number of inversions larger by 1). ■

We are almost there. If the number of inversions only changes by 2, then what about the parity of the number of inversions? (The "parity" of a number refers to whether the number is even or odd. For example, 7 and 5 have odd parity, and 18 and 0 have even parity.)

Since adding or subtracting 2 from a number does not change its parity, we have the following corollary to Lemma 3.3.7:

Corollary 3.3.8. *Neither a row move nor a column move ever changes the parity of the number of inversions.*

Now we can bundle up all these observations and state an *invariant*, that is, a property of the puzzle that never changes, no matter how you slide the tiles around.

Lemma 3.3.9. *In every configuration reachable from the configuration shown in Figure 3.7(a), the* parity of the number of inversions is odd.

Proof. We use induction. Let $P(n)$ be the proposition that after n moves from the above configuration, the parity of the number of inversions is odd.

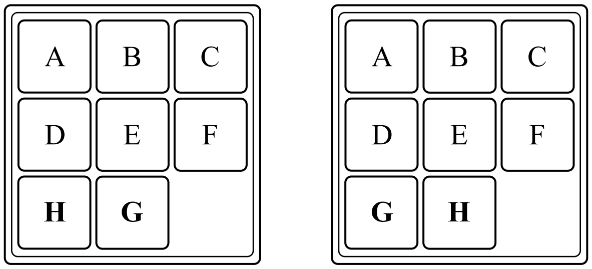
Base case: After zero moves, exactly one pair of tiles is inverted (G and H), which is an odd number. Therefore $P(0)$ is true.

Inductive step: Now we must prove that $P(n)$ implies $P(n+1)$ for all $n \geq 0$. So assume that $P(n)$ is true; that is, after n moves the parity of the number of inversions is odd. Consider any sequence of $n+1$ moves $m_1, \ldots, m_{n+1}$. By the induction hypothesis $P(n)$, we know that the parity after moves $m_1, \ldots, m_n$ is odd. By Corollary 3.3.8, we know that the parity does not change during m_{n+1}. Therefore, the parity of the number of inversions after moves $m_1, \ldots, m_{n+1}$ is odd, so we have that $P(n+1)$ is true.

By the principle of induction, $P(n)$ is true for all $n \geq 0$. ■

The theorem we originally set out to prove is restated below. With our invariant in hand, the proof is simple.

Theorem. *No sequence of legal moves transforms the board below on the left into the board below on the right.*



Proof. In the target configuration on the right, the total number of inversions is zero, which is even. Therefore, by Lemma 3.3.9, the target configuration is unreachable. ■

3.4 Strong Induction

Strong induction is a variation of ordinary induction that is useful when the predicate $P(n+1)$ naturally depends on $P(a)$ for values of $a < n$. As with ordinary induction, strong induction is useful to prove that a predicate $P(n)$ is true for all $n \in \mathbb{N}$.

3.4.1 A Rule for Strong Induction

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

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

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

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

Formulated as a proof rule, strong induction is

Rule. ***Strong Induction Rule***

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

The template for strong induction proofs is identical to the template given in Section 3.2.3 for ordinary induction except for two things:

- you should state that your proof is by strong induction, and
- you can assume that $P(0)$, $P(1)$, …, $P(n)$ are all true instead of only $P(n)$ during the inductive step.

3.4.2 Some Examples

Products of Primes

As a first example, we'll use strong induction to re-prove Theorem 3.1.2 which we previously proved using Well Ordering.

Lemma 3.4.1. *Every integer greater than 1 is a product of primes.*

Proof. We will prove Lemma 3.4.1 by strong induction, letting the induction hypothesis, $P(n)$, be

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

So Lemma 3.4.1 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, and 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 the 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 $n \geq 2$.

■

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 a 5Sg coin 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.

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

Figure 3.10 An example of the stacking game with $n = 10$ boxes. On each line, the underlined stack is divided in the next step.

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.

Since $n \geq 0$, we know that $n + 1 \geq 1$ and thus that the three cases cover every possibility. Since $P(n + 1)$ is true in every case, we can conclude by strong induction that for all $n \geq 0$, 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 shown in Figure 3.10. Can you find a better strategy?

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

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

- The template for a strong induction proof mirrors the template 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. ■

3.4.3 Strong Induction versus Induction

Is strong induction really "stronger" than ordinary induction? It certainly looks that way. After all, you can assume a lot more when proving the induction step. But actually, any proof using strong induction can be reformatted into a proof using ordinary induction—you just need to use a "stronger" induction hypothesis.

Which method should you use? Whichever you find easier. But whichever method you choose, be sure to state the method up front so that the reader can understand and more easily verify your proof.

3.5 Structural Induction

Up to now, we have focussed on induction over the natural numbers. But the idea of induction is far more general—it can be applied to a much richer class of sets. In particular, it is especially useful in connection with sets or data types that are defined recursively.

3.5.1 Recursive Data Types

Recursive data types play a central role in programming. They are specified by *recursive definitions* that say how to build something from its parts. *Recursive* definitions have two parts:

- **Base case(s)** that don't depend on anything else.
- **Constructor case(s)** that depend on previous cases.

Let's see how this works in a couple of examples: Strings of brackets and expression evaluation.

Example 1: Strings of Brackets

Let brkts be the set of all sequences (or strings) of square brackets. For example, the following two strings are in brkts:

$$\texttt{]]}\,\texttt{[[[[[]]} \quad \text{and} \quad \texttt{[[[]][]][]} \tag{3.5}$$

Definition 3.5.1. The set brkts of strings of brackets can be defined recursively as follows:

- **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 can be "matched up" in the usual way. For example, the left hand string in 3.5 is not matched because its second right bracket does not have a matching left bracket. The string on the right is matched. The set of matched strings can be defined recursively as follows.

Definition 3.5.2. Recursively define the set, RecMatch, of strings as follows:

- **Base case:** $\lambda \in \text{RecMatch}$.
- **Constructor case:** If $s, t \in \text{RecMatch}$, then

$$[\,s\,]\,t \in \text{RecMatch}.$$

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

Using this definition, we can see that $\lambda \in \text{RecMatch}$ by the Base case, so

$$[\lambda]\,\lambda = [\,] \in \text{RecMatch}$$

by the Constructor case. So now,

$$\begin{aligned} [\lambda][\,] &= [\,][\,] \in \text{RecMatch} && (\text{letting } s = \lambda, t = [\,]) \\ [[\,]]\,\lambda &= [[\,]] \in \text{RecMatch} && (\text{letting } s = [\,], t = \lambda) \\ &[[\,]][\,] \in \text{RecMatch} && (\text{letting } s = [\,], t = [\,]) \end{aligned}$$

are also strings in RecMatch by repeated applications of the Constructor case.

In general, RecMatch will contain precisely the strings with matching brackets. This is because the constructor case is, in effect, identifying the bracket that matches the leftmost bracket in any string. Since that matching bracket is unique, this method of constructing RecMatch gives a unique way of constructing any string with matched brackets. This will turn out to be important later when we talk about ambiguity.

Strings with matched brackets arise in the area of expression parsing. A brief history of the advances in this field is provided in the box on the next page.

Example 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 3.5.3. The set Aexp is defined recursively as follows:

- **Base cases:**

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

$$\begin{aligned} &[[x + y] * z^2 \div y] + 7, \text{ or,} \\ &x + [y * z^2 \div [y + 7]], \text{ or,} \\ &[x + [y * z^2]] \div [y + 7], \end{aligned}$$

or ... ?

The Turing award (the “Nobel Prize” of computer science) was ultimately bestowed on Robert Floyd, for, among other things, being discoverer of a simple program that would insert the brackets properly.

In the 70’s and 80’s, this parsing technology was packaged into high-level compiler-compilers that automatically generated parsers from expression grammars. This automation of parsing was so effective that the subject stopped demanding attention and largely disappeared from the computer science curriculum by the 1990’s.

1. The variable, x, is in Aexp.
2. The arabic numeral, $\texttt{k}$, 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

$$((\texttt{3} * (x * x)) + ((\texttt{2} * x) + \texttt{1})). \tag{3.6}$$

These parentheses and $*$'s clutter up examples, so we'll often use simpler expressions like "$3x^2 + 2x + 1$" instead of (3.6). But it's important to recognize that $3x^2 + 2x + 1$ is not an Aexp; it's an *abbreviation* for an Aexp.

3.5.2 Structural Induction on Recursive Data Types

Structural induction is a method for proving that some property, P, holds for 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 (recursively-defined) matched strings always have an equal number of left and right brackets. To do this, define a predicate, P, on strings $s \in \texttt{brkts}$:

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

Theorem 3.5.4. *$P(s)$ holds for all $s \in$ RecMatch.*

Proof. By structural induction on the definition that $s \in$ RecMatch, using $P(s)$ as the induction hypothesis.

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

Constructor case: For $r = [\,s\,]\,t$, we must show that $P(r)$ holds, given that $P(s)$ and $P(t)$ holds. So let n_s, n_t be, respectively, the number of left brackets in s and t. So the number of left brackets in r is $1 + n_s + n_t$.

Now from the respective hypotheses $P(s)$ and $P(t)$, we know that the number of right brackets in s is n_s, and likewise, the number of right brackets in t is n_t. So the number of right brackets in r is $1 + n_s + n_t$, which is the same as the number of left brackets. This proves $P(r)$. We conclude by structural induction that $P(s)$ holds for all $s \in \text{RecMatch}$. ■

3.5.3 Functions on Recursively-defined Data Types

A Quick Review of 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

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

Notice that f_5 does not assign a value to a 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.[8] The set of values that arise from applying f to all possible arguments is called the *range* of f. That is,

$$\text{range}(f) ::= f(\text{domain}(f)).$$

[8] 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.

Recursively-Defined Functions

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, consider the function

$$\text{eval} : \text{Aexp} \times \mathbb{Z} \rightarrow \mathbb{Z},$$

which evaluates any expression in Aexp using the value n for x. It is useful to express this function with a recursive definition as follows:

Definition 3.5.5. The *evaluation function*, eval : Aexp × $\mathbb{Z} \rightarrow \mathbb{Z}$, is defined recursively on expressions, $e \in$ 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 $\texttt{k}$ 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:

$$\begin{aligned}
\text{eval}((\texttt{3} + (x * x)), 2) &= \text{eval}(\texttt{3}, 2) + \text{eval}((x * x), 2) && \text{(by Def 3.5.5.3)} \\
&= 3 + \text{eval}((x * x), 2) && \text{(by Def 3.5.5.2)} \\
&= 3 + (\text{eval}(x, 2) \cdot \text{eval}(x, 2)) && \text{(by Def 3.5.5.4)} \\
&= 3 + (2 \cdot 2) && \text{(by Def 3.5.5.1)} \\
&= 3 + 4 = 7.
\end{aligned}$$

A Second Example

We next consider the function on matched strings that specifies the depth of the matched brackets in any string. This function can be specified recursively as follows:

Definition 3.5.6. The *depth* $d(s)$ of a string $s \in \text{RecMatch}$ is defined recursively by the rules:

- $d(\lambda) ::= 0.$
- $d(\,[\, s \,]\, t) ::= \max\{d(s) + 1, d(t)\}$

Ambiguity

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 another definition of the matched strings.

Definition 3.5.7. Define the set, $M \subseteq \texttt{brkts}$ recursively as follows:

- **Base case:** $\lambda \in M$,
- **Constructor cases:** if $s, t \in M$, then the strings $[\, s \,]$ and st are also in M.

By using structural induction, it is possible to prove that $M = \text{RecMatch}$. Indeed, the definition of M might even seem like a more natural way to define the set of matched strings than the definition of RecMatch. But the definition of M is ambiguous, while the (perhaps less natural) definition of RecMatch is unambiguous. Does this ambiguity matter? Yes, it can. For example, suppose we defined

$$\begin{aligned} f(\lambda) &::= 1, \\ f(\,\textbf{[}\,s\,\textbf{]}\,) &::= 1 + f(s), \\ f(st) &::= (f(s)+1) \cdot (f(t)+1) \qquad\qquad \text{for } st \neq \lambda. \end{aligned}$$

Let a be the string $\textbf{[\,[\,]\,]} \in M$ built by two successive applications of the first M constructor starting with λ. Next let

$$\begin{aligned} b &::= aa \\ &= \textbf{[\,[\,]\,]\,[\,[\,]\,]} \end{aligned}$$

and

$$\begin{aligned} c &::= bb \\ &= \textbf{[\,[\,]\,]\,[\,[\,]\,]\,[\,[\,]\,]\,[\,[\,]\,]} \end{aligned}$$

each be 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 $t = a$.

By applying these rules to the first way of constructing c, $f(a) = 2$, $f(b) = (2+1)(2+1) = 9$, and $f(c) = f(bb) = (9+1)(9+1) = 100$. Using the second way of constructing c, we find that $f(ba) = (9+1)(2+1) = 27$ and $f(c) = f(ba\,a) = (27+1)(2+1) = 84$. The outcome is that $f(c)$ is defined to be both 100 and 84, which shows that the rules defining f are inconsistent.

Note that structural induction remains a sound proof method even for ambiguous recursive definitions, which is why it is easy to prove that $M =$ RecMatch.

3.5.4 Recursive Functions on $\mathbb{N}$—Structural Induction versus Ordinary Induction

The nonnegative integers can be understood as a recursive data type.

Definition 3.5.8. The set, $\mathbb{N}$, is a data type defined recursivly as:

- **Base Case:** $0 \in \mathbb{N}$.
- **Constructor Case:** If $n \in \mathbb{N}$, then the *successor*, $n + 1$, of n is in $\mathbb{N}$.

This means that ordinary induction is a special case of structural induction on the recursive Definition 3.5.8. Conversely, most proofs based on structural induction that you will encounter in computer science can also be reformatted into proofs that use only ordinary induction. The decision as to which technique to use is up to you, but it will often be the case that structural induction provides the easiest approach when you are dealing with recursive data structures or functions.

Definition 3.5.8 also justifies the familiar recursive definitions of functions on the nonnegative integers. Here are some examples.

The Factorial Function

The factorial function is often written "$n!$." You will be seeing it a lot in Parts III and IV of this text. For now, we'll use the notation $\text{fac}(n)$ and define it recursively as follows:

- **Base Case:** $\text{fac}(0) ::= 1$.
- **Constructor Case:** $\text{fac}(n + 1) ::= (n + 1) \cdot \text{fac}(n)$ for $n \geq 0$.

The Fibonacci numbers.

Fibonacci numbers arose out of an effort 800 years ago to model population growth. We will study them at some length in Part III. The nth Fibonacci number, $\text{fib}(n)$, can be defined recursively by:

- **Base Cases:** $\text{fib}(0) ::= 0$ and $\text{fib}(1) ::= 1$
- **Constructor Case:** $\text{fib}(n) ::= \text{fib}(n - 1) + \text{fib}(n - 2)$ for $n \geq 2$.

Here the recursive step starts at $n = 2$ with base cases for $n = 0$ and $n = 1$. 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

- **Base Case:** $S(0) ::= 0$.
- **Constructor Case:** $S(n + 1) ::= f(n + 1) + S(n)$ for $n \geq 0$.

Ill-formed Function Definitions

There are some blunders to watch out for when defining functions recursively. Below are some function specifications that resemble good definitions of functions on the nonnegative integers, but they aren't.

Definition 3.5.9.

$$f_1(n) ::= 2 + f_1(n-1). \tag{3.7}$$

This "definition" has no base case. If some function, f_1, satisfied (3.7), so would a function obtained by adding a constant to the value of f_1. So equation (3.7) does not uniquely define an f_1.

Definition 3.5.10.

$$f_2(n) ::= \begin{cases} 0, & \text{if } n = 0, \\ f_2(n+1) & \text{otherwise.} \end{cases} \tag{3.8}$$

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 (3.8) 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 (3.8) as defining a *partial* function, f_2, that is undefined everywhere but 0.

Definition 3.5.11.

$$f_3(n) ::= \begin{cases} 0, & \text{if } n \text{ is divisible by } 2, \\ 1, & \text{if } n \text{ is divisible by } 3, \\ 2, & \text{otherwise.} \end{cases} \tag{3.9}$$

This "definition" is inconsistent: it requires $f_3(6) = 0$ and $f_3(6) = 1$, so (3.9) doesn't define anything.

A Mysterious Function

Mathematicians have been wondering about the following function specification for many years:

$$f_4(n) ::= \begin{cases} 1, & \text{if } n \leq 1, \\ f_4(n/2) & \text{if } n > 1 \text{ is even,} \\ f_4(3n+1) & \text{if } n > 1 \text{ is odd.} \end{cases} \tag{3.10}$$

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 (3.10), 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 (3.10) equals 1 for all n up to over a billion.

4 Number Theory

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

> [Number theorists] may be justified in rejoicing that there is one science, at any rate, and that their own, whose very remoteness from ordinary human activities should keep it gentle and clean.

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

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.*
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 $a \mid c$. ■

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

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

[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.

[3] 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.

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

Twin Prime Conjecture There are infinitely many primes p such that $p + 2$ is also a prime. In 1966 Chen showed that there are infinitely many primes p such that $p + 2$ is the product of at most two primes. So the conjecture is known to be *almost* true!

Primality Testing There is an efficient way to determine whether a number is prime. A naive search for factors of an integer n takes a number of steps proportional to $\sqrt{n}$, which is exponential in the *size* of n in decimal or binary notation. All known procedures for prime checking blew up like this on various inputs. Finally in 2002, an amazingly simple, new method was discovered by Agrawal, Kayal, and Saxena, which showed that prime testing only required a polynomial number of steps. Their paper began with a quote from Gauss emphasizing the importance and antiquity of the problem even in his time—two centuries ago. So prime testing is definitely not in the category of infeasible problems requiring an exponentially growing number of steps in bad cases.

Factoring Given the product of two large primes $n = pq$, there is no efficient way to recover the primes p and q. The best known algorithm is the "number field sieve", which runs in time proportional to:

$$e^{1.9(\ln n)^{1/3}(\ln \ln n)^{2/3}}$$

This is infeasible when n has 300 digits or more.

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 negative numbers strangely.

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?
Samuel: Right, then what?
Bruce: All right. We take the 3-gallon jug and fill it a third of the way...
Samuel: No! He said, "Be precise." Exactly 4 gallons.
Bruce: Sh—. Every cop within 50 miles is running his a— off and I'm out here playing kids' games in the park.
Samuel: Hey, you want to focus on the problem at hand?

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: Live Free or Die Hard*, 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?

That's where number theory comes in handy.

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:

$$\begin{aligned} j_1 &= s_1 \cdot a + t_1 \cdot b \\ j_2 &= s_2 \cdot a + t_2 \cdot b \end{aligned}$$

  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. ■

So we have established that the jug problem has an invariant property, namely that the amount of water in every jug is always a linear combination of the capacities of the jugs. This lemma 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 part 3 of Lemma 4.1.1, 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 a lot of 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 $\text{gcd}(a, b)$. For example, $\text{gcd}(18, 24) = 6$. 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 $m = \text{gcd}(a, b)$ by showing both $\text{gcd}(a, b) \leq m$ and $m \leq \text{gcd}(a, b)$.

First, we show that $\text{gcd}(a, b) \leq m$. 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 $sa + tb$ for any s and t. The $\text{gcd}(a, b)$ is by definition a common divisor of a and b, so

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

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

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

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

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

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

$$\begin{aligned} a &= q \cdot (sa + tb) + r, \qquad \text{so} \\ r &= (1 - qs)a + (-qt)b. \end{aligned}$$

We've just expressed r as a linear combination of a and b. However, m is the *smallest positive* linear combination and $0 \le r < m$. 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. $\gcd(a, b) = \gcd(b, \text{rem}(a, b))$.

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:

$$\begin{aligned} sa + tb &= 1 \\ ua + vc &= 1 \end{aligned}$$

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):

$$\begin{aligned} \gcd(1147, 899) &= \gcd\bigl(899, \underbrace{\text{rem}(1147, 899)}_{=248}\bigr) \\ &= \gcd\bigl(248, \underbrace{\text{rem}(899, 248)}_{=155}\bigr) \\ &= \gcd\bigl(155, \underbrace{\text{rem}(248, 155)}_{=93}\bigr) \\ &= \gcd\bigl(93, \underbrace{\text{rem}(155, 93)}_{=62}\bigr) \\ &= \gcd\bigl(62, \underbrace{\text{rem}(93, 62)}_{=31}\bigr) \\ &= \gcd\bigl(31, \underbrace{\text{rem}(62, 31)}_{=0}\bigr) \\ &= \gcd(31, 0) \\ &= 31 \end{aligned}$$

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. You can prove that the algorithm always eventually terminates by using induction and the fact that the numbers in each step keep getting smaller until the remainder is 0, whereupon you have computed the GCD. In fact, the numbers are getting smaller quickly (by at least a factor of 2 every two steps) and so Euler’s Algorithm is quite fast. The fact that Euclid’s Algorithm actually produces the GCD (and not something different) can also be proved by an inductive invariant argument.

The calculation that $\gcd(1147, 899) = 31$ together with Corollary 4.2.3 implies that there is no way to measure out 2 gallons of water using jugs with capacities 1147 and 899, since we can only obtain multiples of 31 gallons with these jugs. This is good news—Bruce won’t even survive *Die Hard 6*!

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:

$$\gcd(26, 21) = \gcd(21, 5) = \gcd(5, 1) = 1.$$

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

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

1. Fill the smaller jug.
2. Pour all the water in the smaller jug into the larger jug. 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:

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

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

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

The Pulverizer goes through the same steps, but requires some extra bookkeeping along the way: as we compute $\text{gcd}(a, b)$, 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.

You can prove that the Pulverizer always works and that it terminates by using induction. Indeed, you can "pulverize" very large numbers very quickly by using this algorithm. As we will soon see, its speed makes the Pulverizer a very useful tool in the field of cryptography.

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:*

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

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. ■

The Prime Number Theorem

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

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

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

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

In late 2004 a billboard appeared in various locations around the country:

$$\left\{\begin{array}{c}\text{first 10-digit prime found}\\ \text{in consecutive digits of } e\end{array}\right\}.\,\mathbf{com}$$

Substituting the correct number for the expression in curly-braces produced the URL for a Google employment page. The idea was that Google was interested in hiring the sort of people that could and would solve such a problem.

How hard is this problem? Would you have to look through thousands or millions or billions of digits of e to find a 10-digit prime? The rule of thumb derived from the Prime Number Theorem says that among 10-digit numbers, about 1 in

$$\ln 10^{10} \approx 23$$

is prime. This suggests that the problem isn't really so hard! Sure enough, the first 10-digit prime in consecutive digits of e appears quite early:

$$\begin{aligned} e =\,&2.718281828459045235360287471352662497757247093699959574966\\ &96762772407663035354759457138217852516642\mathbf{7427466391}9320030\\ &599218174135966290435729003342952605956307381323286279434\ldots \end{aligned}$$

A routine induction argument extends this statement to:

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

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

Proof. Theorem 3.1.2 showed, using the Well Ordering Principle, that every positive integer can be expressed as a product of primes. So we just have to prove this expression is unique. We will use Well Ordering to prove this too.

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

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

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

4.4 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

Photograph of Alan Turing removed due to copyright restrictions.
Please see: http://en.wikipedia.org/wiki/File:Alan_Turing_photo.jpg

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)**.**

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

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

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

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 $29 \equiv_{\text{mod } 7} 15$ for example, but the notation with the modulus at the end is firmly entrenched and we’ll stick to it.

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

Corollary 4.5.2.

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

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

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

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. ■

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.

4.6 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 $0, 1, \dots, p-1$, 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:*

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

is a permutation[4] *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. ■

[4] 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 $2 \log p$ 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.

4.7 Arithmetic with an Arbitrary Modulus

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

In 1977, Ronald Rivest, Adi Shamir, and Leonard Adleman at MIT proposed a highly secure cryptosystem (called **RSA**) based on number theory. Despite decades of attack, no significant weakness has been found. Moreover, RSA has a major advantage over traditional codes: the sender and receiver of an encrypted mes-

The Riemann Hypothesis

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

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

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

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

Multiplying together all the left sides and all the right sides gives:

$$\sum_{n=1}^{\infty} \frac{1}{n^s} = \prod_{p \in \text{primes}} \left(\frac{1}{1 - 1/p^s} \right)$$

The sum on the left is obtained by multiplying out all the infinite series and applying the Fundamental Theorem of Arithmetic. For example, the term $1/300^s$ in the sum is obtained by multiplying $1/2^{2s}$ from the first equation by $1/3^s$ in the second and $1/5^{2s}$ in the third. Riemann noted that every prime appears in the expression on the right. So he proposed to learn about the primes by studying the equivalent, but simpler expression on the left. In particular, he regarded s as a complex number and the left side as a function, $\zeta(s)$. Riemann found that the distribution of primes is related to values of s for which $\zeta(s) = 0$, which led to his famous conjecture:

Definition 4.6.5. *The Riemann Hypothesis*: Every nontrivial zero of the zeta function $\zeta(s)$ lies on the line $s = 1/2 + ci$ in the complex plane.

A proof would immediately imply, among other things, a strong form of the Prime Number Theorem.
Researchers continue to work intensely to settle this conjecture, as they have for over a century. It is another of the Millennium Problems whose solver will earn $1,000,000 from the Clay Institute.

sage 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 4.6.3.

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

$$\text{rem}(k_1 \cdot k, n), \quad \text{rem}(k_2 \cdot k, n), \quad \text{rem}(k_3 \cdot k, n), \quad \dots \quad, \text{rem}(k_r \cdot k, n)$$

is a permutation of the sequence:

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

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 $\{1, 2, \dots, n\}$ (denoted $[1, n]$) 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 in $[1, 12]$ that are relatively prime to 12.[5]

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

[5] Recall that $\gcd(n, n) = n$ and so n is never relatively prime to itself.

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. *For any number n, if $p_1, p_2, \dots, p_j$ are the (distinct) prime factors of n, then*

$$\phi(n) = n\left(1 - \frac{1}{p_1}\right)\left(1 - \frac{1}{p_2}\right)\dots\left(1 - \frac{1}{p_j}\right).$$

For example,

$$\begin{aligned}\phi(300) &= \phi(2^2 \cdot 3 \cdot 5^2)\\ &= 300\left(1 - \frac{1}{2}\right)\left(1 - \frac{1}{3}\right)\left(1 - \frac{1}{5}\right)\\ &= 300\left(\frac{1}{2}\right)\left(\frac{2}{3}\right)\left(\frac{4}{5}\right)\\ &= 80.\end{aligned}$$

Corollary 4.7.5. *Let $n = pq$ where p and q are different primes. Then $\phi(n) = (p-1)(q-1)$.*

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

Proof of Corollary 4.7.5. 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, ..., pq, there are precisely q multiples of p and p multiples of q. Since p and q are relatively prime, the only number in $[1, pq]$ that is a multiple of both p and q is pq. Hence, there are $p + q - 1$ numbers in $[1, pq]$ that are *not* relatively prime to n. This means that

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

as claimed.[6] ■

We can now prove Euler's Theorem:

[6] This proof provides a brief preview of the kinds of counting arguments that we will explore more fully in Part III.

Theorem 4.7.6 (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 $k_1, \dots, k_r$ 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,

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

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 $\phi(n)$ is easy (using Theorem 4.7.4) 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 modulo n.

4.8 The RSA Algorithm

Finally, we are ready to see how the *RSA public key encryption scheme* works. The details are in the box on the next page.

It is not immediately clear from the description of the RSA cryptosystem that the decoding of the encrypted message is, in fact, the original unencrypted message. In order to check that this is the case, 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 4.5.2. That is,

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

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

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

The RSA Cryptosystem

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 $\gcd(m, n) = 1$.[a] 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:

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

[a] It would be very bad if $\gcd(m, n)$ equals p or q since then it would be easy for someone to use the encoded message to compute the secret key If $\gcd(m, n) = n$, then the encoded message would be 0, which is fairly useless. For very large values of n, it is extremely unlikely that $\gcd(m, n) \neq 1$. If this does happen, you should get a new set of keys or, at the very least, add some bits to m so that the resulting message is relatively prime to n.

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 in Equation 4.8, 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 4.7.5, 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 the 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. . . .

II Structures

Introduction

Structure is fundamental in computer science. Whether you are writing code, solving an optimization problem, or designing a network, you will be dealing with structure. The better you can understand the structure, the better your results will be. And if you can reason about structure, then you will be in a good position to convince others (and yourself) that your results are worthy.

The most important structure in computer science is a *graph*, also known as a *network*). Graphs provide an excellent mechanism for modeling associations between pairs of objects; for example, two exams that cannot be given at the same time, two people that like each other, or two subroutines that can be run independently. In Chapter 5, we study graphs that represent *symmetric* relationships, like those just mentioned. In Chapter 6, we consider graphs where the relationship is *one-way*; that is, a situation where you can go from x to y but not necessarily vice-versa.

In Chapter 7, we consider the more general notion of a *relation* and we examine important classes of relations such as partially ordered sets. Partially ordered sets arise frequently in scheduling problems.

We conclude in Chapter 8 with a discussion of *state machines*. State machines can be used to model a variety of processes and are a fundamental tool in proving that an algorithm terminates and that it produces the correct output.

5 Graph Theory

Informally, a graph is a bunch of dots and lines where the lines connect some pairs of dots. An example is shown in Figure 5.1. The dots are called *nodes* (or *vertices*) and the lines are called *edges*.

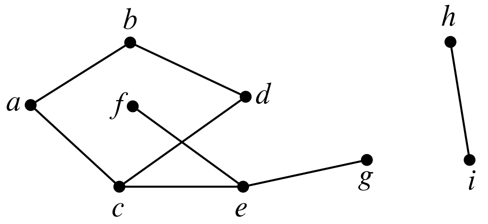


Figure 5.1 An example of a graph with 9 nodes and 8 edges.

Graphs are ubiquitous in computer science because they provide a handy way to represent a relationship between pairs of objects. The objects represent items of interest such as programs, people, cities, or web pages, and we place an edge between a pair of nodes if they are related in a certain way. For example, an edge between a pair of people might indicate that they like (or, in alternate scenarios, that they don't like) each other. An edge between a pair of courses might indicate that one needs to be taken before the other.

In this chapter, we will focus our attention on simple graphs where the relationship denoted by an edge is symmetric. Afterward, in Chapter 6, we consider the situation where the edge denotes a one-way relationship, for example, where one web page points to the other.[1]

5.1 Definitions

5.1.1 Simple Graphs

Definition 5.1.1. A *simple graph* G consists of a nonempty set V, called the *vertices* (aka *nodes*[2]) of G, and a set E of two-element subsets of V. The members of E are called the *edges* of G, and we write $G = (V, E)$.

[1] Two Stanford students analyzed such a graph to become multibillionaires. So, pay attention to graph theory, and who knows what might happen!

[2] We will use the terms vertex and node interchangeably.

The vertices correspond to the dots in Figure 5.1, and the edges correspond to the lines. The graph in Figure 5.1 is expressed mathematically as $G = (V, E)$, where:

$$\begin{aligned} V &= \{a, b, c, d, e, f, g, h, i\} \\ E &= \{\{a, b\}, \{a, c\}, \{b, d\}, \{c, d\}, \{c, e\}, \{e, f\}, \{e, g\}, \{h, i\}\}. \end{aligned}$$

Note that $\{a, b\}$ and $\{b, a\}$ are different descriptions of the same edge, since sets are unordered. In this case, the graph $G = (V, E)$ has 9 nodes and 8 edges.

Definition 5.1.2. 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 v is called the *degree* of the vertex and is denoted by $\deg(v)$; equivalently, the degree of a vertex is equals the number of vertices adjacent to it.

For example, in the simple graph shown in Figure 5.1, vertex 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 $\deg(e) = 3$. It is possible for a vertex to have degree 0, in which case it is not adjacent to any other vertices. A simple graph does not need to have any edges at all —in which case, the degree of every vertex is zero and $|E| = 0$[3] —but it does need to have at least one vertex, that is, $|V| \geq 1$.

Note that simple graphs do *not* have any *self-loops* (that is, an edge of the form $\{a, a\}$) since an edge is defined to be a set of *two* vertices. In addition, there is at most one edge between any pair of vertices in a simple graph. In other words, a simple graph does not contain *multiedges* or *multiple edges*. That is because E is a set. Lastly, and most importantly, simple graphs do not contain *directed edges* (that is, edges of the form (a, b) instead of $\{a, b\}$).

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. We will consider graphs with directed edges (called *directed graphs* or *digraphs*) at length in Chapter 6. Since we'll only be considering simple graphs in this chapter, we'll just call them "graphs" from now on.

5.1.2 Some Common Graphs

Some graphs come up so frequently that they have names. The *complete graph* on n vertices, denoted K_n, has an edge between every two vertices, for a total of $n(n-1)/2$ edges. For example, K_5 is shown in Figure 5.2.

The *empty graph* has no edges at all. For example, the empty graph with 5 nodes is shown in Figure 5.3.

[3] The *cardinality*, $|E|$, of the set E is the number of elements in E.

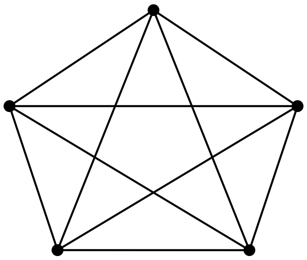


Figure 5.2 The complete graph on 5 nodes, K_5.

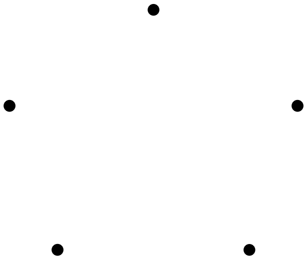


Figure 5.3 The empty graph with 5 nodes.

The n-node graph containing $n - 1$ edges in sequence is known as the *line graph* L_n. More formally, $L_n = (V, E)$ where

$$V = \{v_1, v_2, \ldots, v_n\}$$

and

$$E = \{\, \{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{n-1}, v_n\} \,\}$$

For example, L_5 is displayed in Figure 5.4.

If we add the edge $\{v_n, v_1\}$ to the line graph L_n, we get the graph C_n consisting of a simple cycle. For example, C_5 is illustrated in Figure 5.5.

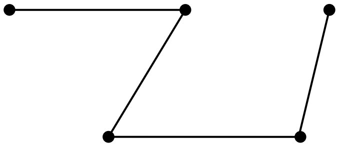


Figure 5.4 The 5-node line graph L_5.

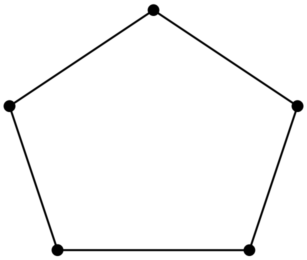


Figure 5.5 The 5-node cycle graph C_5.

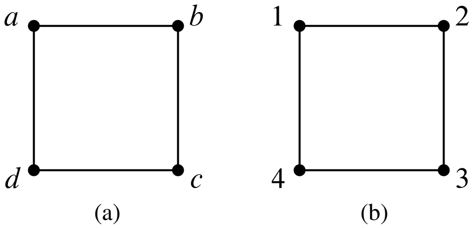


Figure 5.6 Two graphs that are isomorphic to C_4.

5.1.3 Isomorphism

Two graphs that look the same might actually be different in a formal sense. For example, the two graphs in Figure 5.6 are both simple cycles with 4 vertices, but one graph has vertex set $\{a, b, c, d\}$ while the other has vertex set $\{1, 2, 3, 4\}$. Strictly speaking, these graphs are different mathematical objects, but this is a frustrating distinction since the graphs *look the same*!

Fortunately, we can neatly capture the idea of "looks the same" through the notion of graph isomorphism.

Definition 5.1.3. If $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are two graphs, then we say that G_1 is *isomorphic* to G_2 iff there exists a *bijection*[4] $f : V_1 \to V_2$ 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.

In other words, two graphs are isomorphic if they are the same up to a relabeling of their vertices. For example, here is an isomorphism between vertices in the two

[4] A bijection $f : V_1 \to V_2$ is a function that associates every node in V_1 with a unique node in V_2 and vice-versa. We will study bijections more deeply in Part III.

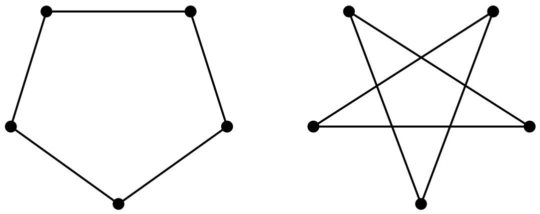


Figure 5.7 Two ways of drawing C_5.

graphs shown in Figure 5.6:

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, we have shown two different ways of drawing C_5 in Figure 5.7.

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, 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. This means that 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 in polynomial time[5] in $|V|$.

Having such a procedure would be useful. For example, it would make it easy to search for a particular molecule in a database given the molecular bonds. On

[5] *I.e.*, in an amount of time that is upper-bounded by $|V|^c$ where c is a fixed number independent of $|V|$.

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.

5.1.4 Subgraphs

Definition 5.1.4. A graph $G_1 = (V_1, E_1)$ is said to be a *subgraph* of a graph $G_2 = (V_2, E_2)$ if $V_1 \subseteq V_2$ and $E_1 \subseteq E_2$.

For example, the empty graph on n nodes is a subgraph of L_n, L_n is a subgraph of C_n, and C_n is a subgraph of K_n. Also, the graph $G = (V, E)$ where

$$V = \{g, h, i\} \quad \text{and} \quad E = \{\{h, i\}\}$$

is a subgraph of the graph in Figure 5.1. On the other hand, any graph containing an edge $\{g, h\}$ would not be a subgraph of the graph in Figure 5.1 because the graph in Figure 5.1 does not contain this edge.

Note that since a subgraph is itself a graph, the endpoints of any edge in a subgraph must also be in the subgraph. In other words if $G' = (V', E')$ is a subgraph of some graph G, and $\{v_i, v_j\} \in E'$, then it must be the case that $v_i \in V'$ and $v_j \in V'$.

5.1.5 Weighted Graphs

Sometimes, we will use edges to denote a connection between a pair of nodes where the connection has a *capacity* or *weight*. For example, we might be interested in the capacity of an Internet fiber between a pair of computers, the resistance of a wire between a pair of terminals, the tension of a spring connecting a pair of devices in a dynamical system, the tension of a bond between a pair of atoms in a molecule, or the distance of a highway between a pair of cities.

In such cases, it is useful to represent the system with an *edge-weighted* graph (aka a *weighted graph*). A weighted graph is the same as a simple graph except that we associate a real number (that is, the weight) with each edge in the graph. Mathematically speaking, a weighted graph consists of a graph $G = (V, E)$ and a weight function $w : E \to \mathbb{R}$. For example, Figure 5.8 shows a weighted graph where the weight of edge $\{a, b\}$ is 5.

5.1.6 Adjacency Matrices

There are many ways to represent a graph. We have already seen two ways: you can draw it, as in Figure 5.8 for example, or you can represent it with sets —as in $G = (V, E)$. Another common representation is with an adjacency matrix.

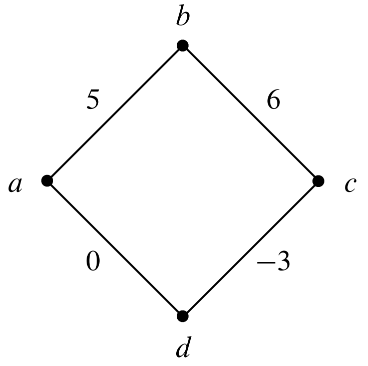


Figure 5.8 A 4-node weighted graph where the edge $\{a, b\}$ has weight 5.

$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} \qquad \begin{pmatrix} 0 & 5 & 0 & 0 \\ 5 & 0 & 6 & 0 \\ 0 & 6 & 0 & -3 \\ 0 & 0 & -3 & 0 \end{pmatrix}$$

(a) (b)

Figure 5.9 Examples of adjacency matrices. (a) shows the adjacency matrix for the graph in Figure 5.6(a) and (b) shows the adjacency matrix for the weighted graph in Figure 5.8. In each case, we set $v_1 = a$, $v_2 = b$, $v_3 = c$, and $v_4 = d$ to construct the matrix.

Definition 5.1.5. Given an n-node graph $G = (V, E)$ where $V = \{v_1, v_2, \ldots, v_n\}$, the *adjacency matrix* for G is the $n \times n$ matrix $A_G = \{a_{ij}\}$ where

$$a_{ij} = \begin{cases} 1 & \text{if } \{v_i, v_j\} \in E \\ 0 & \text{otherwise.} \end{cases}$$

If G is a weighted graph with edge weights given by $w : E \to \mathbb{R}$, then the adjacency matrix for G is $A_G = \{a_{ij}\}$ where

$$a_{ij} = \begin{cases} w(\{v_i, v_j\}) & \text{if } \{v_i, v_j\} \in E \\ 0 & \text{otherwise.} \end{cases}$$

For example, Figure 5.9 displays the adjacency matrices for the graphs shown in Figures 5.6(a) and 5.8 where $v_1 = a$, $v_2 = b$, $v_3 = c$, and $v_4 = d$.

5.2 Matching Problems

We begin our study of graph theory by considering the scenario where the nodes in a graph represent people and the edges represent a relationship between pairs of people such as “likes”, “marries”, and so on. Now, you may be wondering what marriage has to do with computer science, and with good reason. It turns out that the techniques we will develop apply to much more general scenarios where instead of matching men to women, we need to match packets to paths in a network, applicants to jobs, or Internet traffic to web servers. And, as we will describe later, these techniques are widely used in practice.

In our first example, we will show how graph theory can be used to debunk an urban legend about sexual practices in America. Yes, you read correctly. So, fasten your seat belt—who knew that math might actually be interesting!

5.2.1 Sex in America

On average, who 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 Americans 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.” 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 for Health Statistics?—don't answer; this is a setup question like "When did you stop beating your wife?" Using a little graph theory, we will now explain why none of these findings can be anywhere near the truth.

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.[6] We'll put an edge between a male and a female iff they have been sexual partners. A possible subgraph of this graph is illustrated in Figure 5.10 with males on the left and females on the right.

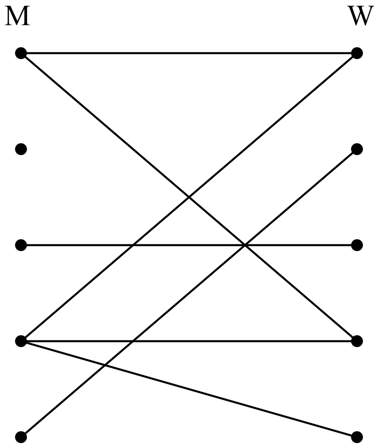


Figure 5.10 A possible subgraph of the sex partners graph.

Actually, G is a pretty hard graph to figure out, let alone draw. The graph is *enormous*: the US population is about 300 million, so $|V| \approx 300M$. In the United States, approximately 50.8% of the populatin is female and 49.2% is male, and 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 to debunk the sex surveys—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 and one F vertex (remember, we're only considering male-female relationships); so the sum of the degrees of the M vertices equals the number of edges, and the sum of the degrees of the F vertices equals the

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

number of edges. So these sums are equal:

$$\sum_{x \in M} \deg(x) = \sum_{y \in F} \deg(y).$$

If we divide both sides of this equation by the product of the sizes of the two sets, $|M| \cdot |F|$, we obtain

$$\left(\frac{\sum_{x \in M} \deg(x)}{|M|} \right) \cdot \frac{1}{|F|} = \left(\frac{\sum_{y \in F} \deg(y)}{|F|} \right) \cdot \frac{1}{|M|} \tag{5.1}$$

Notice that

$$\frac{\sum_{x \in M} \deg(x)}{|M|}$$

is simply the average degree of a node in M. This is the average number of opposite-gender partners for a male in America. Similarly,

$$\frac{\sum_{x \in F} \deg(x)}{|F|}$$

is the average degree of a node in F, which is the average number of opposite-gender partners for a female in America. Hence, Equation 5.1 implies that on average, an American male has $|F|/|M|$ times as many opposite-gender partners as the average American female.

From the Census Bureau reports, we know 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. Of course, this statistic really says nothing about any sex's promiscuity or selectivity. Remarkably, promiscuity is completely irrelevant in this analysis. That is because the ratio of the average number of partners is completely determined by 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 few 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.

The 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 quantities in any graph:

Lemma 5.2.1 (The Handshaking Lemma)**.** *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 5.2.1 is called the *Handshake Lemma* because 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.

5.2.2 Bipartite Matchings

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 that they have earned a special name—they are called *bipartite graphs*.

Definition 5.2.2. 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.

The bipartite matching problem is related to the sex-in-America problem that we just studied; only now the goal is to get everyone happily married. As you might imagine, this is not possible for a variety of reasons, not the least of which is the fact that there are more women in America than men. So, it is simply not possible to marry every woman to a man so that every man is married only once.

But what about getting a mate for every man so that every woman is married only once? Is it possible to do this so that each man is paired with a woman that he likes? The answer, of course, depends on the bipartite graph that represents who

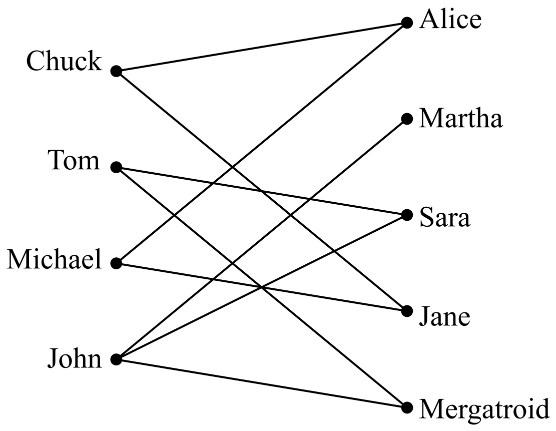


Figure 5.11 A graph where an edge between a man and woman denotes that the man likes the woman.

likes who, but the good news is that it is possible to find natural properties of the who-likes-who graph that completely determine the answer to this question.

In general, suppose that we have a set of men and an equal-sized or larger set of women, and there is a graph with an edge between a man and a woman if the man likes the woman. Note that in this scenario, the "likes" relationship need not be symmetric, since for the time being, we will only worry about finding a mate for each man that he likes.[7] (Later, we will consider the "likes" relationship from the female perspective as well.) For example, we might obtain the graph in Figure 5.11.

In this problem, a *matching* will mean a way of assigning every man to a woman so that different men are assigned to different women, and a man is always assigned to a woman that he likes. For example, one possible matching for the men is shown in Figure 5.12.

The Matching Condition

A famous result known as Hall's Matching Theorem gives necessary and sufficient conditions for the existence of a matching in a bipartite graph. It turns out to be a remarkably useful mathematical tool.

We'll state and prove Hall's Theorem using man-likes-woman terminology. Define *the set of women liked by a given set of men* to consist of all women liked by at least one of those men. For example, the set of women liked by Tom and John in

[7] By the way, we do not mean to imply that marriage should or should not be of a heterosexual nature. Nor do we mean to imply that men should get their choice instead of women. It's just that with bipartite graphs, the edges only connected male nodes to female nodes and there are fewer men in America. So please don't take offense.

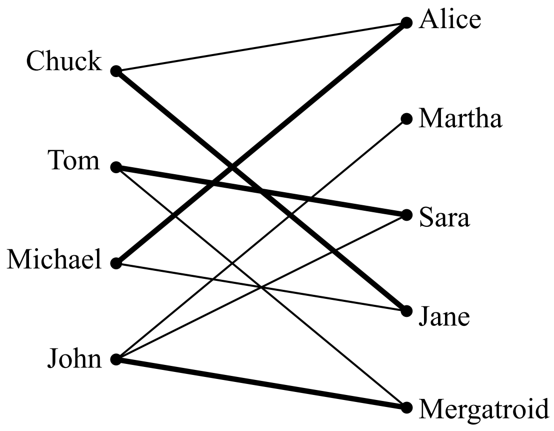


Figure 5.12 One possible matching for the men is shown with bold edges. For example, John is matched with Jane.

Figure 5.11 consists of Martha, Sarah, and Mergatroid. For us to have any chance at all of matching up the men, the following *matching condition* must hold:

Every subset of men likes at least as large a set of women.

For example, we can not find a matching if some set of 4 men like only 3 women. Hall's Theorem says that this necessary condition is actually sufficient; if the matching condition holds, then a matching exists.

Theorem 5.2.3. *A matching for a set of men M with a set of women W 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 men. Each man likes at least the woman he is matched with. Therefore, every subset of men likes at least as large a set of women. 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 $|M|$, the number of men, on the predicate:

$$\begin{aligned} P(m) ::= {} & \text{for any set of } m \text{ men } M\text{, if the matching condition holds} \\ & \text{for } M\text{, then there is a matching for } M. \end{aligned}$$

Base Case ($|M| = 1$): If $|M| = 1$, then the matching condition implies that the lone man likes at least one woman, and so a matching exists.

Inductive Step: We need to show that $P(m)$ IMPLIES $P(m + 1)$. Suppose that $|M| = m + 1 \geq 2$.

Case 1: Every proper subset[8] of men likes a *strictly larger* set of women. In this case, we have some latitude: we pair an arbitrary man with a woman he likes and send them both away. The matching condition still holds for the remaining men and women since we have removed only one woman, so we can match the rest of the men by induction.

Case 2: Some proper subset of men $X \subset M$ likes an *equal-size* set of women $Y \subset W$. We match the men in X with the women in Y by induction and send them all away. We can also match the rest of the men by induction if we show that the matching condition holds for the remaining men and women. To check the matching condition for the remaining people, consider an arbitrary subset of the remaining men $X' \subseteq (M - X)$, and let Y' be the set of remaining women that they like. We must show that $|X'| \leq |Y'|$. Originally, the combined set of men $X \cup X'$ liked the set of women $Y \cup Y'$. So, by the matching condition, we know:

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

We sent away $|X|$ men from the set on the left (leaving X') and sent away an equal number of women from the set on the right (leaving Y'). Therefore, it must be that $|X'| \leq |Y'|$ as claimed.

So in both cases, there is a matching for the men, which completes the proof of the Inductive step. The theorem follows by induction. ■

The proof of Theorem 5.2.3 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 can be solved from a computational perspective.

A Formal Statement

Let's restate Theorem 5.2.3 in abstract terms so that you'll not always be condemned to saying, "Now this group of men likes at least as many women..."

Definition 5.2.4. 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. A matching is said to be *perfect* if every node in the graph is incident to an edge in the matching. In any graph, the set $N(S)$, of *neighbors* of some set, S, of vertices is the set of all vertices adjacent to some vertex in S. That is,

$$N(S) ::= \{\, r \mid \{s, r\} \text{ is an edge for some } s \in S \,\}.$$

[8] Recall that a subset A of B is *proper* if $A \neq B$.

S is called a *bottleneck* if

$$|S| > |N(S)|.$$

Theorem 5.2.5 (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 men has a certain property. In general, verifying that every subset has some property, even if it's easy to check any particular subset for the property, quickly becomes overwhelming because the number of subsets of even relatively small sets is enormous—over a billion subsets for a set of size 30. However, there is a simple property of vertex degrees in a bipartite graph that guarantees the existence of a matching. 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 5.2.6. A bipartite graph G with vertex partition L, R where $|L| \leq |R|$ is *degree-constrained* if $\deg(l) \geq \deg(r)$ for every $l \in L$ and $r \in R$.

For example, the graph in Figure 5.11 is degree constrained since every node on the left is adjacent to at least two nodes on the right while every node on the right is incident to at most two nodes on the left.

Theorem 5.2.7. *Let G be a bipartite graph with vertex partition L, R where $|L| \leq |R|$. If G is degree-constrained, then there is a matching that covers L.*

Proof. The proof is by contradiction. Suppose that G is degree constrained but that there is no matching that covers L. By Theorem 5.2.5, this means that there must be a bottleneck $S \subseteq L$.

Let d be a value such that $\deg(l) \geq x \geq \deg(r)$ for every $l \in L$ and $r \in R$. Since every edge incident to a node in S is incident to a node in $N(S)$, we know that

$$|N(S)|x \geq |S|x$$

and thus that

$$|N(S)| \geq |S|.$$

This means that S is not a bottleneck, which is a contradiction. Hence G has a matching that covers L. ■

Regular graphs provide a large class of graphs that often arise in practice that are degree constrained. Hence, we can use Theorem 5.2.7 to prove that every regular bipartite graph has a perfect matching. This turns out to be a surprisingly useful result in computer science

Definition 5.2.8. A graph is said to be *regular* if every node has the same degree.

Theorem 5.2.9. *Every regular bipartite graph has a perfect matching.*

Proof. Let G be a regular bipartite graph with vertex partition L, R where $|L| \leq |R|$. Since regular graphs are degree-constrained, we know by Theorem 5.2.7 that there must be a matching in G that covers L. Since G is regular, we also know that $|L| = |R|$ and thus the matching must also cover R. This means that every node in G is incident to an edge in the matching and thus G has a perfect matching. ■

5.2.3 The Stable Marriage Problem

We next consider a version of the bipartite matching problem where there are an equal number of men and women, and where each person has preferences about who they would like to marry. In fact, we assume that each man has a complete list of all the women ranked according to his preferences, with no ties. Likewise, each woman has a ranked list of all of the men.

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 everyone: every man must marry exactly one woman and vice-versa—no polygamy. Moreover, we would like to find a matching between men and women that is *stable* in the sense that there is no pair of people that prefer each other to their spouses.

For example, suppose *every* man likes Angelina best, and every woman 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 together working on problem sets!

This unfortunate situation is illustrated in Figure 5.13, where the digits "1" and "2" near a man shows which of the two women he ranks first second, respectively, and similarly for the women.

More generally, in any matching, a man and woman who are not married to each other and who like each other better than their spouses, is called a *rogue couple*. In the situation shown in Figure 5.13, 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 man and woman who are not married to each other, at least one likes their spouse better than the other, and so they won't be tempted to start an affair.

Definition 5.2.10. 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 in Figure 5.13, we

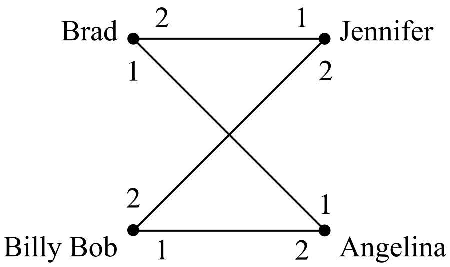


Figure 5.13 Preferences for four people. Both men like Angelina best and both women like Brad best.

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, and so there is a stable matching.

Surprisingly, there is always a stable matching among a group of men and women. The surprise springs in part from considering the apparently similar “buddy” matching problem. That is, if people can be paired off as buddies, regardless of gender, then a stable matching *may not* be possible. For example, Figure 5.14 shows a situation with a love triangle and a fourth person who is everyone’s last choice. In this figure Mergatroid’s preferences aren’t shown because they don’t even matter. Let’s see why there is no stable matching.

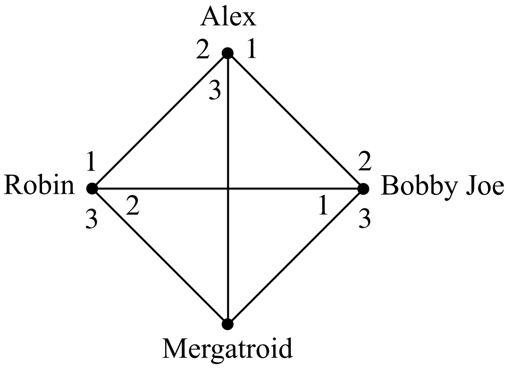


Figure 5.14 Some preferences with no stable buddy matching.

Lemma 5.2.11. *There is no stable buddy matching among the four people in Figure 5.14.*

Proof. We'll prove this by contradiction.

Assume, for the purposes of contradiction, that there is a stable matching. Then there are two members of the love triangle that are matched. Since preferences in the triangle are symmetric, we may assume in particular, that Robin and Alex are matched. Then the other pair must be Bobby-Joe matched with Mergatroid.

But then there is a rogue couple: Alex likes Bobby-Joe best, and Bobby-Joe prefers Alex to his buddy Mergatroid. 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 mens are only allowed to marry women, and vice versa, then it turns out that a stable matching can always be found.[9]

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 woman stands on her balcony. Each man stands under the balcony of his favorite among the women on his list, and he serenades her. If a man has no women left on his list, he stays home and does his math homework.

Afternoon: Each woman 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 man who is told by a woman to take a hike, crosses that woman off his list.

Termination condition: When a day arrives in which every woman has at most one suitor, the ritual ends with each woman 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 eventually reaches the termination condition.
- Everybody ends up married.
- The resulting marriages are stable.

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 man crosses a woman off his list. (If the ritual hasn't terminated, there must be some woman serenaded by at least two men, and at least one of them will have to cross her off his

[9] Once again, we disclaim any political statement here—its just the way that the math works out.

list). If we start with n men and n women, then each of the n men's lists initially has n women on it, for a total of n^2 list entries. Since no women 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.

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 woman has a favorite 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 man 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. This sounds like an invariant, and it is.

Definition 5.2.12. Let P be the predicate: For every woman, w, and every man, m, if w is crossed off m's list, then w has a suitor whom she prefers over m.

Lemma 5.2.13. *P is an invariant for The Mating Ritual.*

Proof. By induction on the number of days.

Base Case: In the beginning (that is, at the end of day 0), every woman is on every list—no one has been crossed off and so P is vacuously true.

Inductive Step: Assume P is true at the end of day d and let w be a woman that has been crossed off a man m's list by the end of day $d + 1$.

Case 1: w was crossed off m's list on day $d + 1$. Then, w must have a suitor she prefers on day $d + 1$.

Case 2: w was crossed off m's list prior to day $d + 1$. Since P is true at the end of day d, this means that w has a suitor she prefers to m on day d. She therefore has the same suitor or someone she prefers better at the end of day $d + 1$.

In both cases, P is true at the end of day $d + 1$ and so P must be an invariant. ■

With Lemma 5.2.13 in hand, we can now prove:

Theorem 5.2.14. *Everyone is married by the Mating Ritual.*

Proof. By contradiction. Assume that it is the last day of the Mating Ritual and someone does not get married. Since there are an equal number of men and women, and since bigamy is not allowed, this means that at least one man (call him Bob) and at least one woman do not get married.

Since Bob is not married, he can’t be serenading anybody and so his list must be empty. This means that Bob has crossed every woman off his list and so, by invariant P, every woman has a suitor whom she prefers to Bob. Since it is the last day and every woman still has a suitor, this means that every woman gets married. This is a contradiction since we already argued that at least one woman is *not* married. Hence our assumption must be false and so everyone must be married. ■

Theorem 5.2.15. *The Mating Ritual produces a stable matching.*

Proof. Let Brad and Jen be any man and woman, respectively, that are *not* married to each other on the last day of the Mating Ritual. We will prove that Brad and Jen are not a rogue couple, and thus that all marriages on the last day are stable. There are two cases to consider.

Case 1: Jen is not on Brad’s list by the end. Then by invariant P, we know that Jen has a suitor (and hence a husband) that she prefers to Brad. So she’s not going to run off with Brad—Brad and Jen cannot be a rogue couple.

Case 2: 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—once again, Brad and Jenn are not a rogue couple. ■

. . . Especially the Men

Who is favored by the Mating Ritual, the men or the women? The women *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 man keeps serenading the woman he most prefers among those on his list until he must cross her off, at which point he serenades the next most preferred woman on his list. So from the man’s perspective, the woman he is serenading can only change for the worse. Sounds like a good deal for the women.

But it’s not! The fact is that from the beginning, the men are serenading their first choice woman, and the desirability of the woman being serenaded decreases only enough to ensure overall stability. The Mating Ritual actually does as well as possible for all the men and does the worst possible job for the women.

To explain all this we need some definitions. Let’s begin by observing that while The Mating Ritual produces one stable matching, there may be other stable matchings among the same set of men and women. For example, reversing the roles of men and women 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, given the preferences shown in Figure 5.13, 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.

Definition 5.2.16. Given a set of preference lists for all men and women, one person is in another person's *realm of possible spouses* if there is a stable matching in which the two people are married. A person's *optimal spouse* is their most preferred person within their realm of possibility. A person's *pessimal spouse* is their least preferred person in their realm of possibility.

Everybody has an optimal and a pessimal spouse, since we know there is at least one stable matching, namely, the one produced by the Mating Ritual. Now here is the shocking truth about the Mating Ritual:

Theorem 5.2.17. *The Mating Ritual marries every man to his optimal spouse.*

Proof. By contradiction. Assume for the purpose of contradiction that some man does not get his optimal spouse. Then there must have been a day when he crossed off his optimal spouse—otherwise he would still be serenading (and would ultimately marry) her or some even more desirable woman.

By the Well Ordering Principle, there must be a *first* day when a man (call him "Keith") crosses off his optimal spouse (call her Nicole). According to the rules of the Ritual, Keith crosses off Nicole because Nicole has a preferred suitor (call him Tom), so

$$\text{Nicole prefers Tom to Keith.} \tag{$*$}$$

Since this is the first day an optimal woman gets crossed off, we know that Tom had not previously crossed off his optimal spouse, and so

$$\text{Tom ranks Nicole at least as high as his optimal spouse.} \tag{$**$}$$

By the definition of an optimal spouse, there must be some stable set of marriages in which Keith gets his optimal spouse, Nicole. But then the preferences given in ($*$) and ($**$) imply that Nicole and Tom are a rogue couple within this supposedly stable set of marriages (think about it). This is a contradiction. ■

Theorem 5.2.18. *The Mating Ritual marries every woman to her pessimal spouse.*

Proof. By contradiction. Assume that the theorem is not true. Hence there must be a stable set of marriages $\mathcal{M}$ where some woman (call her Nicole) is married to a man (call him Tom) that she likes less than her spouse in The Mating Ritual (call him Keith). This means that

$$\text{Nicole prefers Keith to Tom.} \tag{+}$$

By Theorem 5.2.17 and the fact that Nicole and Keith are married in the Mating Ritual, we know that

$$\text{Keith prefers Nicole to his spouse in } \mathcal{M}. \qquad (++)$$

This means that Keith and Nicole form a rogue couple in $\mathcal{M}$, which contradicts the stability of $\mathcal{M}$. ■

Applications

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 published, and unknown by them, a similar algorithm was being used to assign residents to hospitals by the National Resident Matching Program (NRMP)[10]. 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 men and women. (In this case, there may be multiple women married to one man, a scenario we consider in the problem section at the end of the chapter.). Before the Ritual-like algorithm 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.[11]

The Internet infrastructure company, Akamai, also uses a variation of the Mating Ritual to assign web traffic to its servers. In the early days, Akamai used other combinatorial optimization algorithms that got to be too slow as the number of servers (over 65,000 in 2010) and requests (over 800 billion per day) increased. Akamai switched to a Ritual-like approach since it is fast and can be run in a distributed manner. In this case, web requests correspond to women and web servers correspond to men. The web requests have preferences based on latency and packet loss, and the web servers have preferences based on cost of bandwidth and collocation.

Not surprisingly, the Mating Ritual is also used by at least one large online dating agency. Even here, there is no serenading going on—everything is handled by computer.

[10] Of course, there is no serenading going on in the hospitals—the preferences are submitted to a program and the whole process is carried out by a computer.

[11] 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.

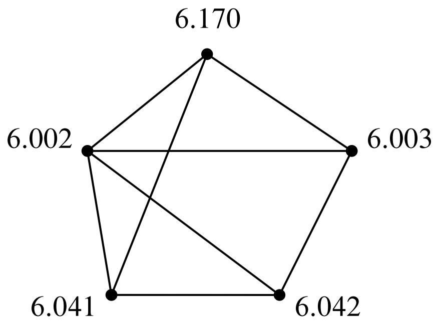


Figure 5.15 A scheduling graph for five exams. Exams connected by an edge cannot be given at the same time.

5.3 Coloring

In Section 5.2, we used edges to indicate an affinity between a pair of nodes. We now consider situations where it is useful to use edges to represent a *conflict* between a pair of nodes. For example, consider the following exam scheduling problem.

5.3.1 An Exam Scheduling Problem

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 (even at MIT) 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 the 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 appear as in Figure 5.15.

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).

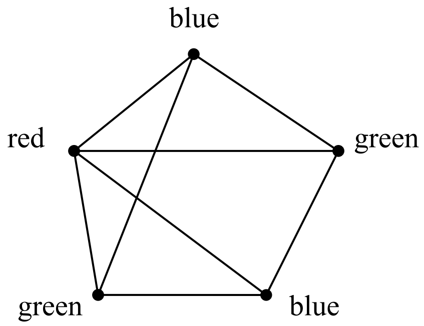


Figure 5.16 A 3-coloring of the exam graph from Figure 5.15.

We 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 then 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*. As shown in Figure 5.16, three colors suffice for our example.

The coloring in Figure 5.16 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 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 5.3.1. The minimum value of k for which a graph G has a valid k-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. 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.)

5.3.2 Degree-Bounded Coloring

There are some simple graph properties that give useful upper bounds on the chromatic number. For example, if the graph is bipartite, then we can color it with 2 colors (one color for the nodes in the "left" set and a second color for the nodes in the "right" set). In fact, if the graph has any edges at all, then being bipartite is equivalent to being 2-colorable.

Alternatively, if the graph is planar, then the famous 4-Color Theorem says that the graph is 4-colorable. This is a hard result to prove, but we will come close in Section 5.8 where we define planar graphs and prove that they are 5-colorable.

The chromatic number of a graph can also be shown to be small if the vertex degrees of the graph are small. In particular, if we have an upper 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 5.3.2. *A graph with maximum degree at most k is $(k+1)$-colorable.*

The natural way to try to prove this theorem is to use induction on k. Unfortunately, this approach leads to disaster. It is not that it is impossible, just that it is extremely painful and would ruin your week if you tried it on an exam. When you encounter such a disaster when using induction on graphs, it is usually best to change what you are inducting on. In graphs, typical good choices for the induction parameter are n, the number of nodes, or e, the number of edges.

Proof of Theorem 5.3.2. 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 (from the set of $k+1$ colors) that is different from all its adjacent vertices, since there are at most k vertices adjacent to v and so at least one of the $k+1$ colors is still 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 $\chi(K_n) = n$. In this case, every node has degree $k = n-1$ and so this is an example where Theorem 5.3.2 gives the best possible bound. By a similar argument, we can show that Theorem 5.3.2 gives the best possible bound for *any* graph with degree bounded by k that has K_{k+1} as a subgraph.

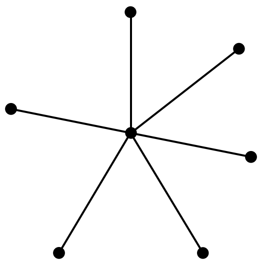


Figure 5.17 A 7-node star graph.

But sometimes $k + 1$ colors is far from the best that you can do. For example, the n-node *star graph* shown in Figure 5.17 has maximum degree $n - 1$ but can be colored using just 2 colors.

5.3.3 Why coloring?

One reason coloring problems frequently arise in practice is because scheduling conflicts are so common. For example, at Akamai, a new version of software is deployed over each of 75,000 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 75,000-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 connect 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. Registers can 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. Once again, the goal is to minimize the number of colors needed to color the graph.

Finally, there's the famous map coloring problem stated in Proposition 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 *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). Surprisingly, 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. (It's always easy to tell if an *arbitrary* graph is 2-colorable.) In Section 5.8, we'll develop enough planar graph theory to present an easy proof that all planar graphs are 5-colorable.

5.4 Getting from A to B in a Graph

5.4.1 Paths and Walks

Definition 5.4.1. A *walk*[12] in a graph, G, is a sequence of vertices

$$v_0, v_1, \dots, v_k$$

and edges

$$\{v_0, v_1\}, \{v_1, v_2\}, \dots, \{v_{k-1}, v_k\}$$

such that $\{v_i, v_{i+1}\}$ is an edge of G for all i where $0 \le i < k$. The walk is said to *start* at v_0 and to *end* at v_k, and the *length* of the walk is defined to be k. An edge, $\{u, v\}$, is *traversed n* times by the walk if there are n different values of i such that $\{v_i, v_{i+1}\} = \{u, v\}$. A *path* is a walk where all the v_i's are different, that is, $i \ne j$ implies $v_i \ne v_j$. For simplicity, we will refer to paths and walks by the sequence of vertices.[13]

For example, the graph in Figure 5.18 has a length 6 path a, b, c, d, e, f, g. This is the longest path in the graph. Of course, the graph has walks with arbitrarily large lengths; for example, $a, b, a, b, a, b, \dots$.

The length of a walk or 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 contains a sequence of 7 vertices.

[12] Some texts use the word *path* for our definition of walk and the term *simple path* for our definition of path.

[13] This works fine for simple graphs since the edges in a walk are completely determined by the sequence of vertices and there is no ambiguity. For graphs with multiple edges, we would need to specify the edges as well as the nodes.

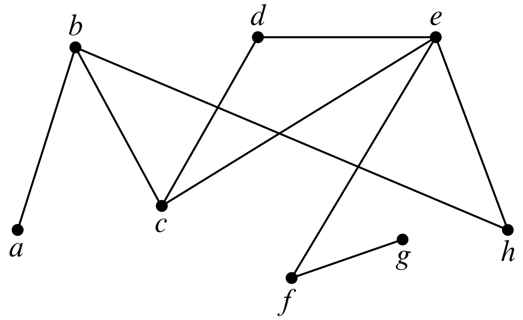


Figure 5.18 A graph containing a path a, b, c, d, e, f, g of length 6.

5.4.2 Finding a Path

Where there's a walk, there's a path. This is sort of obvious, but it's easy enough to prove rigorously using the Well Ordering Principle.

Lemma 5.4.2. *If there is a walk from a vertex u to a vertex v in a graph, then there is a path from u to v.*

Proof. Since there is a walk from u to v, there must, by the Well-ordering Principle, be a minimum length walk from u to v. If the minimum length is zero or one, this minimum length walk is itself a path from u to v. Otherwise, there is a minimum length walk

$$v_0, v_1, \ldots, v_k$$

from $u = v_0$ to $v = v_k$ where $k \geq 2$. We claim this walk must be a path. To prove the claim, suppose to the contrary that the walk is not a path; that is, some vertex on the walk occurs twice. This means that there are integers i, j such that $0 \leq i < j \leq k$ with $v_i = v_j$. Then deleting the subsequence

$$v_{i+1}, \ldots, v_j$$

yields a strictly shorter walk

$$v_0, v_1, \ldots, v_i, v_{j+1}, v_{j+2}, \ldots, v_k$$

from u to v, contradicting the minimality of the given walk. ■

Actually, we proved something stronger:

Corollary 5.4.3. *For any walk of length k in a graph, there is a path of length* at most *k with the same endpoints. Moreover, the shortest walk between a pair of vertices is, in fact, a path.*

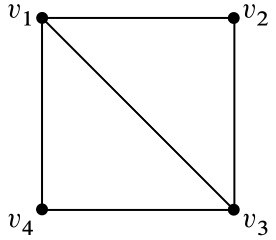


Figure 5.19 A graph for which there are 5 walks of length 3 from v_1 to v_4. The walks are (v_1, v_2, v_1, v_4), (v_1, v_3, v_1, v_4), (v_1, v_4, v_1, v_4), (v_1, v_2, v_3, v_4), and (v_1, v_4, v_3, v_4).

5.4.3 Numbers of Walks

Given a pair of nodes that are connected by a walk of length k in a graph, there are often many walks that can be used to get from one node to the other. For example, there are 5 walks of length 3 that start at v_1 and end at v_4 in the graph shown in Figure 5.19.

There is a surprising relationship between the number of walks of length k between a pair of nodes in a graph G and the kth power of the adjacency matrix A_G for G. The relationship is captured in the following theorem.

Theorem 5.4.4. *Let $G = (V, E)$ be an n-node graph with $V = \{v_1, v_2, \dots, v_n\}$ and let $A_G = \{a_{ij}\}$ denote the adjacency matrix for G. Let $a_{ij}^{(k)}$ denote the (i, j)-entry of the kth power of A_G. Then the number of walks of length k between v_i and v_j is $a_{ij}^{(k)}$.*

In other words, we can determine the number of walks of length k between any pair of nodes simply by computing the kth power of the adjacency matrix! That's pretty amazing.

For example, the first three powers of the adjacency matrix for the graph in Figure 5.19 are:

$$A = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} \qquad A^2 = \begin{pmatrix} 3 & 1 & 2 & 1 \\ 1 & 2 & 1 & 2 \\ 2 & 1 & 3 & 1 \\ 1 & 2 & 1 & 2 \end{pmatrix} \qquad A^3 = \begin{pmatrix} 4 & 5 & 5 & 5 \\ 5 & 2 & 5 & 2 \\ 5 & 5 & 4 & 5 \\ 5 & 2 & 5 & 2 \end{pmatrix}$$

Sure enough, the $(1, 4)$ coordinate of A^3 is $a_{14}^{(3)} = 5$, which is the number of length 3 walks from v_1 to v_4. And $a_{24}^{(3)} = 2$, which is the number of length 3 walks from v_2 to v_4. By proving the theorem, we'll discover why it is true and thereby uncover the relationship between matrix multiplication and numbers of walks.

Proof of Theorem 5.4.4. The proof is by induction on k. We will let $P(k)$ be the predicate that the theorem is true for k. Let $P_{ij}^{(k)}$ denote the number of walks of length k between v_i and v_j. Then $P(k)$ is the predicate

$$\forall i, j \in [1, n].\ P_{ij}^{(k)} = a_{ij}^{(k)}. \tag{5.2}$$

Base Case ($k = 1$): There are two cases to consider:

Case 1: $\{v_i, v_j\} \in E$. Then $P_{ij}^{(1)} = 1$ since there is precisely one walk of length 1 between v_i and v_j. Moreover, $\{v_i, v_j\} \in E$ means that $a_{ij}^{(1)} = a_{ij} = 1$. So, $P_{ij}^{(1)} = a_{ij}^{(1)}$ in this case.

Case 2: $\{v_i, v_j\} \notin E$. Then $P_{ij}^{(1)} = 0$ since there cannot be any walks of length 1 between v_i and v_j. Moreover, $\{v_i, v_j\} \notin E$ means that $a_{ij} = 0$. So, $P_{ij}^{(1)} = a_{ij}^{(1)}$ in this case as well.

Hence, $P(1)$ must be true.

Inductive Step: Assume $P(k)$ is true. In other words, assume that equation 5.2 holds.

We can group (and thus count the number of) walks of length $k+1$ from v_i to v_j according to the first edge in the walk (call it $\{v_i, v_t\}$). This means that

$$P_{ij}^{(k+1)} = \sum^{t:\{v_i,v_t\}\in E} P_{tj}^{(k)} \tag{5.3}$$

where the sum is over all t such that $\{v_i, v_t\}$ is an edge. Using the fact that $a_{ij} = 1$ if $\{v_i, v_t\} \in E$ and $a_{it} = 0$ otherwise, we can rewrite Equation 5.3 as follows:

$$P_{ij}^{(k+1)} = \sum_{t=1}^{n} a_{it} P_{tj}^{(k)}.$$

By the inductive hypothesis, $P_{tj}^{(k)} = a_{tj}^{(k)}$ and thus

$$P_{ij}^{(k+1)} = \sum_{t=1}^{n} a_{it} a_{tj}^{(k)}.$$

But the formula for matrix multiplication gives that

$$a_{ij}^{(k+1)} = \sum_{t=1}^{n} a_{it} a_{tj}^{(k)}.$$

and so we must have $P_{ij}^{(k+1)} = a_{ij}^{(k+1)}$ for all $i, j \in [1, n]$. Hence $P(k+1)$ is true and the induction is complete. ■

5.4.4 Shortest Paths

Although the connection between the power of the adjacency matrix and the number of walks is cool (at least if you are a mathematician), the problem of counting walks does not come up very often in practice. Much more important is the problem of finding the shortest path between a pair of nodes in a graph.

There is good news and bad news to report on this front. The good news is that it is not very hard to find a shortest path. The bad news is that you can't win one of those million dollar prizes for doing it.

In fact, there are several good algorithms known for finding a Shortest Path between a pair of nodes. The simplest to explain (but not the fastest) is to compute the powers of the adjacency matrix one by one until the value of $a_{ij}^{(k)}$ exceeds 0. That's because Theorem 5.4.4 and Corollary 5.4.3 imply that the length of the shortest path between v_i and v_j will be the smallest value of k for which $a_{ij}^{(k)} > 0$.

Paths in Weighted Graphs

The problem of computing shortest paths in a weighted graph frequently arises in practice. For example, when you drive home for vacation, you usually would like to take the shortest route.

Definition 5.4.5. Given a weighted graph, the length of a path in the graph is the sum of the weights of the edges in the path.

Finding shortest paths in weighted graphs is not a lot harder than finding shortest paths in unweighted graphs. We won't show you how to do it here, but you will study algorithms for finding shortest paths if you take an algorithms course. Not surprisingly, the proof of correctness will use induction.

5.5 Connectivity

Definition 5.5.1. Two vertices in a graph are said to be *connected* if there is a path that begins at one and ends at the other. By convention, every vertex is considered to be connected to itself by a path of length zero.

Definition 5.5.2. A graph is said to be *connected* when every pair of vertices are connected.

5.5.1 Connected Components

Being connected is usually a good property for a graph to have. For example, it could mean that it is possible to get from any node to any other node, or that it is possible to communicate between any pair of nodes, depending on the application.

But not all graphs are connected. For example, the graph where nodes represent cities and edges represent highways might be connected for North American cities, but would surely not be connected if you also included cities in Australia. The same is true for communication networks like the Internet—in order to be protected from viruses that spread on the Internet, some government networks are completely isolated from the Internet.

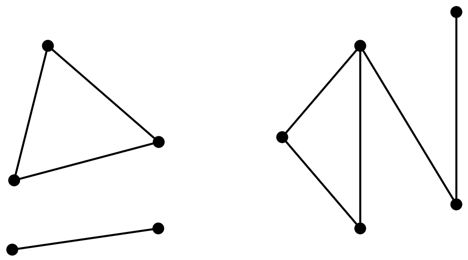


Figure 5.20 One graph with 3 connected components.

For example, the diagram in Figure 5.20 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. These connected pieces of a graph are called its *connected components*.

Definition 5.5.3. A *connected component* is a subgraph of a graph consisting of some vertex and every node and edge that is connected to that vertex.

So a graph is connected iff it has exactly one connected component. At the other extreme, the empty graph on n vertices has n connected components.

5.5.2 k-Connected Graphs

If we think of a graph as modeling 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 5.5.4. Two vertices in a graph are *k-edge connected* if they remain connected in every subgraph obtained by deleting $k - 1$ edges. A graph with at least two vertices is k-edge connected[14] if every two of its vertices are k-edge connected.

[14] 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."

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 5.18, vertices c and e are 3-edge connected, b and e are 2-edge connected, g and e are 1-edge connected, and no vertices are 4-edge connected. The graph as a whole is only 1-edge connected. 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 case $k = 2$.

5.5.3 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 5.5.5. *Every graph with v vertices and e edges has at least $v - e$ connected components.*

Of course for Theorem 5.5.5 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 v 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.

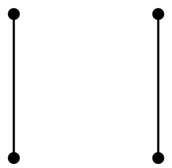


Figure 5.21 A counterexample graph to the False Claim.

Otherwise, if a and b were in different connected components of G', then these two components are merged into one component 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 5.5.6. *Every connected graph with v vertices has at least $v - 1$ edges.*

A couple of points about the proof of Theorem 5.5.5 are worth noticing. First, we used induction on the number of edges in the graph. This is very common in proofs involving graphs, as 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*.

5.5.4 Build-Up Error

False Claim. *If every vertex in a graph has degree at least 1, then the graph is connected.*

There are many counterexamples; for example, see Figure 5.21.

False proof. We use induction. Let $P(n)$ be the proposition that if every vertex in an n-vertex graph has degree at least 1, then the graph is connected.

Base case: There is only one graph with a single vertex and has degree 0. Therefore, $P(1)$ is vacuously true, since the if-part is false.

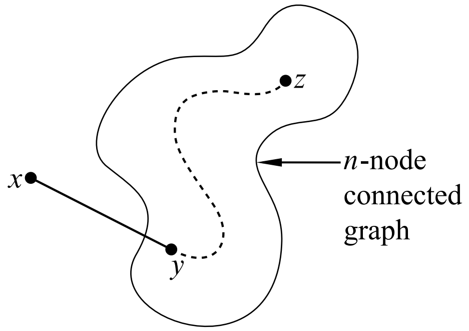


Figure 5.22 Adding a vertex x with degree at least 1 to a connected n-node graph.

Inductive step: We must show that $P(n)$ implies $P(n+1)$ for all $n \geq 1$. Consider an n-vertex graph in which every vertex has degree at least 1. By the assumption $P(n)$, this graph is connected; that is, there is a path between every pair of vertices. Now we add one more vertex x to obtain an $(n+1)$-vertex graph as shown in Figure 5.22.

All that remains is to check that there is a path from x to every other vertex z. Since x has degree at least one, there is an edge from x to some other vertex; call it y. Thus, we can obtain a path from x to z by adjoining the edge $\{x, y\}$ to the path from y to z. This proves $P(n+1)$.

By the principle of induction, $P(n)$ is true for all $n \geq 1$, which proves the theorem ■

Uh-oh. . . this proof looks fine! Where is the bug? It turns out that the faulty assumption underlying this argument is that *every* $(n+1)$*-vertex graph with minimum degree 1 can be obtained from an* n*-vertex graph with minimum degree 1 by adding 1 more vertex*. Instead of starting by considering an arbitrary $(n+1)$-node graph, this proof only considered $(n+1)$-node graphs that you can make by starting with an n-node graph with minimum degree 1.

The counterexample in Figure 5.21 shows that this assumption is false; there is no way to build the 4-vertex graph in Figure 5.21 from a 3-vertex graph with minimum degree 1. Thus the first error in the proof is the statement "This proves $P(n+1)$."

This kind of flaw is known as "build-up error." Usually, build-up error arises from a faulty assumption that every size $n+1$ graph with some property can be "built up" from a size n graph with the same property. (This assumption is correct for some properties, but incorrect for others—such as the one in the argument above.)

One way to avoid an accidental build-up error is to use a "shrink down, grow back" process in the inductive step; that is, start with a size $n + 1$ graph, remove a vertex (or edge), apply the inductive hypothesis $P(n)$ to the smaller graph, and then add back the vertex (or edge) and argue that $P(n + 1)$ holds. Let's see what would have happened if we'd tried to prove the claim above by this method:

Revised inductive step: We must show that $P(n)$ implies $P(n + 1)$ for all $n \geq 1$. Consider an $(n + 1)$-vertex graph G in which every vertex has degree at least 1. Remove an arbitrary vertex v, leaving an n-vertex graph G' in which every vertex has degree… uh oh!

The reduced graph G' might contain a vertex of degree 0, making the inductive hypothesis $P(n)$ inapplicable! We are stuck—and properly so, since the claim is false!

Always use shrink-down, grow-back arguments and you'll never fall into this trap.

5.6 Around and Around We Go

5.6.1 Cycles and Closed Walks

Definition 5.6.1. A *closed walk*[15] in a graph G is a sequence of vertices

$$v_0, v_1, \ldots, v_k$$

and edges

$$\{v_0, v_1\}, \{v_1, v_2\}, \ldots, \{v_{k-1}, v_k\}$$

where v_0 is the same node as v_k and $\{v_i, v_{i+1}\}$ is an edge of G for all i where $0 \leq i < k$. The *length* of the closed walk is k. A closed walk is said to be a *cycle* if $k \geq 3$ and $v_0, v_1, \ldots, v_{k-1}$ are all different.

For example, b, c, d, e, c, b is a closed walk of length 5 in the graph shown in Figure 5.18. It is not a cycle since it contains node c twice. On the other hand, c, d, e, c is a cycle of length 3 in this graph since every node appears just once.

There are many ways to represent the same closed walk or cycle. For example, b, c, d, e, c, b is the same as c, d, e, c, b, c (just starting at node c instead of node b) and the same as b, c, e, d, c, b (just reversing the direction).

[15] Some texts use the word *cycle* for our definition of closed walk and *simple cycle* for our definition of cycle.

Cycles are similar to paths, except that the last node is the first node and the notion of first and last does not matter. Indeed, there are many possible vertex orders that can be used to describe cycles and closed walks, whereas walks and paths have a prescribed beginning, end, and ordering.

5.6.2 Odd Cycles and 2-Colorability

We have already seen that determining the chromatic number of a graph is a challenging problem. There is a special case where this problem is very easy; namely, the case where every cycle in the graph has even length. In this case, the graph is 2-colorable! Of course, this is optimal if the graph has any edges at all. More generally, we will prove

Theorem 5.6.2. *The following properties of a graph are equivalent (that is, if the graph has any one of the properties, then it has all of the properties):*

1. *The graph is bipartite.*
2. *The graph is 2-colorable.*
3. *The graph does not contain any cycles with odd length.*
4. *The graph does not contain any closed walks with odd length.*

Proof. We will show that property 1 IMPLIES property 2, property 2 IMPLIES property 3, property 3 IMPLIES property 4, and property 4 IMPLIES property 1. This will show that all four properties are equivalent by repeated application of Rule 2.1.2 in Section 2.1.2.

1 IMPLIES 2 Assume that $G = (V, E)$ is a bipartite graph. Then V can be partitioned into two sets L and R so that no edge connects a pair of nodes in L nor a pair of nodes in R. Hence, we can use one color for all the nodes in L and a second color for all the nodes in R. Hence $\chi(G) = 2$.

2 IMPLIES 3 Let $G = (V, E)$ be a 2-colorable graph and

$$C ::= v_0, v_1, \ldots, v_k$$

be any cycle in G. Consider any 2-coloring for the nodes of G. Since $\{v_i, v_{i+1}\} \in E$, v_i and v_{i+1} must be differently colored for $0 \le i < k$. Hence v_0, v_2, v_4, ..., have one color and v_1, v_3, v_5, ..., have the other color. Since C is a cycle, v_k is the same node as v_0, which means they must have the same color, and so k must be an even number. This means that C has even length.

3 IMPLIES 4 The proof is by contradiction. Assume for the purposes of contradiction that G is a graph that does not contain any cycles with odd length (that is, G satisfies Property 3) but that G *does* contain a closed walk with odd length (that is, G does not satisfy Property 4).

Let

$$w ::= v_0, v_1, v_2, \ldots, v_k$$

be the *shortest* closed walk with odd length in G. Since G has no odd-length cycles, w cannot be a cycle. Hence $v_i = v_j$ for some $0 \leq i < j < k$. This means that w is the union of two closed walks:

$$v_0, v_1, \ldots, v_i, v_{j+1}, v_{j+2}, \ldots, v_k$$

and

$$v_i, v_{i+1}, \ldots, v_j.$$

Since w has odd length, one of these two closed walks must also have odd length and be shorter than w. This contradicts the minimality of w. Hence 3 IMPLIES 4.

4 IMPLIES 1 Once again, the proof is by contradiction. Assume for the purposes of contradictin that G is a graph without any closed walks with odd length (that is, G satisfies Property 4) but that G is *not* bipartite (that is, G does not satisfy Property 1).

Since G is not bipartite, it must contain a connected component $G' = (V', E')$ that is not bipartite. Let v be some node in V'. For every node $u \in V'$, define

$$\begin{aligned}\text{dist}(u) ::= &\text{ the length of the shortest path from } u \text{ to } v \text{ in } G'. \\ &\text{If } u = v, \text{ the distance is zero.}\end{aligned}$$

Partition V' into sets L and R so that

$$\begin{aligned} L &= \{\, u \mid \text{dist}(u) \text{ is even} \,\}, \\ R &= \{\, u \mid \text{dist}(u) \text{ is odd} \,\}. \end{aligned}$$

Since G' is not bipartite, there must be a pair of adjacent nodes u_1 and u_2 that are both in L or both in R. Let e denote the edge incident to u_1 and u_2.

Let P_i denote a shortest path in G' from u_i to v for $i = 1, 2$. Because u_1 and u_2 are both in L or both in R, it must be the case that P_1 and P_2 both have even length or they both have odd length. In either case, the union of P_1, P_2, and e forms a closed walk with odd length, which is a contradiction. Hence 4 IMPLIES 1. ■

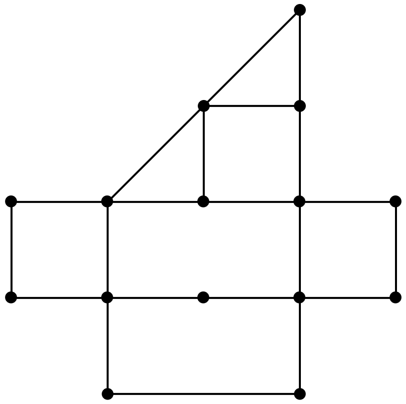


Figure 5.23 A possible floor plan for a museum. Can you find a walk that traverses every edge exactly once?

Theorem 5.6.2 turns out to be useful since bipartite graphs come up fairly often in practice. We'll see examples when we talk about planar graphs in Section 5.8 and when we talk about packet routing in communication networks in Chapter 6.

5.6.3 Euler Tours

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 the floor plan in Figure 5.23?

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 walk 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. Graphs with Euler tours and Euler walks both have simple characterizations.

Theorem 5.6.3. *A connected graph has an Euler tour if and only if every vertex has even degree.*

Proof. We first show that if a graph has an Euler tour, then every vertex has even degree. Assume that a graph $G = (V, E)$ has an Euler tour $v_0, v_1, \ldots, v_k$ where $v_k = v_0$. Since every edge is traversed once in the tour, $k = |E|$ and the degree of a node u in G is the number of times that node appears in the sequence $v_0, v_1, \ldots,$ v_{k-1} times two. We multiply by two since if $u = v_i$ for some i where $0 < i < k$, then both $\{v_{i-1}, v_i\}$ and $\{v_i, v_{i+1}\}$ are edges incident to u in G. If $u = v_0 = v_k$, then both $\{v_{k-1}, v_k\}$ and $\{v_0, v_1\}$ are edges incident to u in G. Hence, the degree of every node is even.

We next show that if the degree of every node is even in a graph $G = (V, E)$, then there is an Euler tour. Let

$$W ::= v_0, v_1, \dots, v_k$$

be the longest walk in G that traverses *no edge more than once*[16]. W must traverse every edge incident to v_k; otherwise the walk could be extended and W would not be the longest walk that traverses all edges at most once. Moreover, it must be that $v_k = v_0$ and that W is a closed walk, since otherwise v_k would have odd degree in W (and hence in G), which is not possible by assumption.

We conclude the argument with a proof by contradiction. 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 walk W' that is longer than W but that still uses no edge more than once:

$$W' ::= u, v_i, v_{i+1}, \dots, v_k, v_1, v_2, \dots, v_i.$$

This contradicts the definition of W, so W must be an Euler tour after all. ■

It is not difficult to extend Theorem 5.6.3 to prove that a connected graph G has an Euler walk if and only if precisely 0 or 2 nodes in G have odd degree. Hence, we can conclude that the graph shown in Figure 5.23 has an Euler walk but not an Euler tour since the graph has precisely two nodes with odd degree.

Although the proof of Theorem 5.6.3 does not explicitly define a method for finding an Euler tour when one exists, it is not hard to modify the proof to produce such a method. The idea is to grow a tour by continually splicing in closed walks until all the edges are consumed.

5.6.4 Hamiltonian Cycles

Hamiltonian cycles are the unruly cousins of Euler tours.

Definition 5.6.4. A *Hamiltonian cycle* in a graph G is a cycle that visits every *node* in G exactly once. Similarly, a *Hamiltonian* path is a path in G that visits every node exactly once.

[16] Did you notice that we are using a variation of the Well Ordering Principle here when we implicitly assume that a longest walk exists? This is ok since the length of a walk where no edge is used more than once is at most $|E|$.

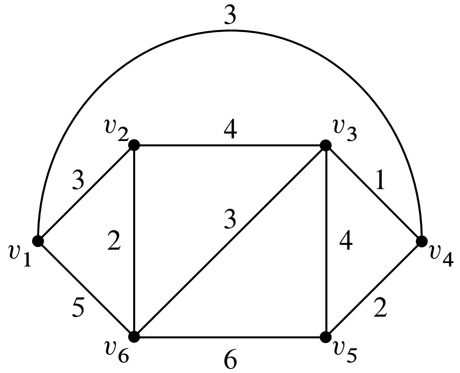


Figure 5.24 A weighted graph. Can you find a cycle with weight 15 that visits every node exactly once?

Although Hamiltonian cycles sound similar to Euler tours—one visits every node once while the other visits every edge once—finding a Hamiltonian cycle can be a lot harder than finding an Euler tour. The same is true for Hamiltonian paths. This is because no one has discovered a simple characterization of all graphs with a Hamiltonian cycle. In fact, determining whether a graph has a Hamiltonian cycle is the same category of problem as the SAT problem of Section 1.5 and the coloring problem in Section 5.3; you get a million dollars for finding an efficient way to determine when a graph has a Hamiltonian cycle—or proving that no procedure works efficiently on all graphs.

5.6.5 The Traveling Salesperson Problem

As if the problem of finding a Hamiltonian cycle is not hard enough, when the graph is weighted, we often want to find a Hamiltonian cycle that has least possible weight. This is a very famous optimization problem known as the Traveling Salesperson Problem.

Definition 5.6.5. Given a weighted graph G, the *weight* of a cycle in G is defined as the sum of the weights of the edges in the cycle.

For example, consider the graph shown in Figure 5.24 and suppose that you would like to visit every node once and finish at the node where you started. Can you find way to do this by traversing a cycle with weight 15?

Needless to say, if you can figure out a fast procedure that finds the optimal cycle for the traveling salesperson, let us know so that we can win a million dollars.

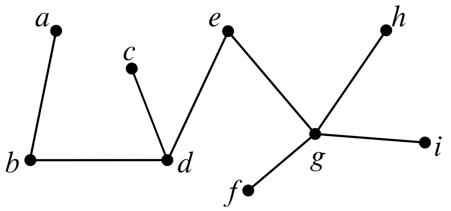


Figure 5.25 A 9-node tree.

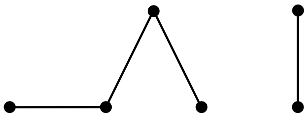


Figure 5.26 A 6-node forest consisting of 2 component trees. Note that this 6-node graph is not itself a tree since it is not connected.

5.7 Trees

As we have just seen, finding good cycles in a graph can be trickier than you might first think. But what if a graph has no cycles at all? Sounds pretty dull. But graphs without cycles (called *acyclic graphs*) are probably the most important graphs of all when it comes to computer science.

5.7.1 Definitions

Definition 5.7.1. A connected acyclic graph is called a *tree*.

For example, Figure 5.25 shows an example of a 9-node tree.

The graph shown in Figure 5.26 is not a tree since it is not connected, but it is a forest. That's because, of course, it consists of a collection of trees.

Definition 5.7.2. If every connected component of a graph G is a tree, then G is a *forest*.

One of the first things you will notice about trees is that they tend to have a lot of nodes with degree one. Such nodes are called *leaves*.

Definition 5.7.3. A *leaf* is a node with degree 1 in a tree (or forest).

For example, the tree in Figure 5.25 has 5 leaves and the forest in Figure 5.26 has 4 leaves.

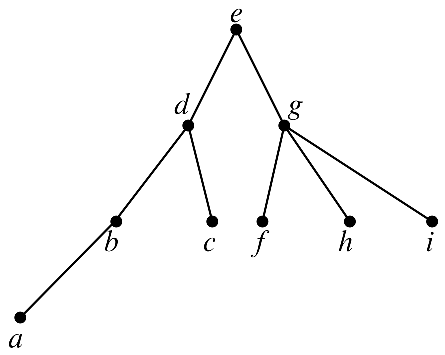


Figure 5.27 The tree from Figure 5.25 redrawn in a leveled fashion, with node E as the root.

Trees are a fundamental data structure in computer science. For example, information is often stored in tree-like data structures and the execution of many recursive programs can be modeled as the traversal of a tree. In such cases, it is often useful to draw the tree in a leveled fashion where the node in the top level is identified as the *root*, and where every edge joins a *parent* to a *child*. For example, we have redrawn the tree from Figure 5.25 in this fashion in Figure 5.27. In this example, node d is a child of node e and a parent of nodes b and c.

In the special case of *ordered binary trees*, every node is the parent of at most 2 children and the children are labeled as being a left-child or a right-child.

5.7.2 Properties

Trees have many unique properties. We have listed some of them in the following theorem.

Theorem 5.7.4. *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 nonadjacent nodes in a tree creates a graph with a cycle.*
4. *Removing any edge disconnects the graph.*
5. *If the tree has at least two vertices, then it has at least two leaves.*
6. *The number of vertices in a tree is one larger than the number of edges.*

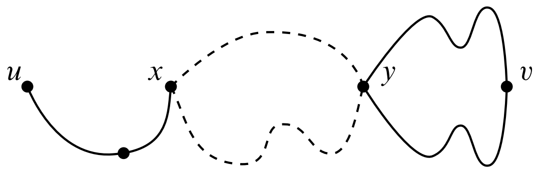


Figure 5.28 If there are two paths between u and v, the graph must contain a cycle.

Proof. 1. A cycle in a subgraph is also a 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. Since a tree is connected, there is at least one path between every pair of vertices. Suppose for the purposes of contradiction, that there are two different paths between some pair of 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. (For example, see Figure 5.28.) Then there are two paths from x to y with no common edges, which defines a cycle. This is a contradiction, since trees are acyclic. Therefore, there is exactly one path between every pair of vertices.

3. An additional edge $\{u, v\}$ together with the unique path between u and v forms a 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 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 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 proposition $P(n) ::=$ there are $n - 1$ edges in any n-vertex tree.

   Base Case ($n = 1$): $P(1)$ is true since a tree with 1 node has 0 edges and $1 - 1 = 0$.

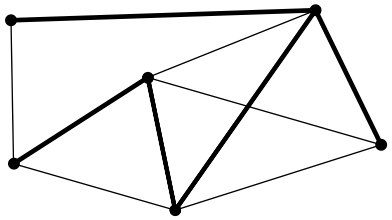


Figure 5.29 A graph where the edges of a spanning tree have been thickened.

> **Inductive step**: Now suppose that $P(n)$ is true 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 part 1 of Theorem 5.7.4, deleting v and its incident edge gives a smaller tree, and this smaller tree has $n-1$ edges by induction. If we re-attach the vertex v and its incident edge, then we find that T has $n = (n+1) - 1$ edges. Hence, $P(n+1)$ is true, and the induction proof is complete. ■

Various subsets of properties in Theorem 5.7.4 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.

5.7.3 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, Figure 5.29 is a connected graph with a spanning tree highlighted.

Theorem 5.7.5. *Every connected graph contains a spanning tree.*

Proof. By contradiction. Assume there is some connected graph G that has no spanning tree and 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. By the assumption, T is not a spanning tree and so it contains some cycle:

$$\{v_0, v_1\}, \{v_1, v_2\}, \ldots, \{v_k, v_0\}$$

Suppose that we remove the last edge, $\{v_k, v_0\}$. If a pair of vertices x and y was joined by a path not containing $\{v_k, v_0\}$, then they remain joined by that path. On the other hand, if x and y were joined by a path containing $\{v_k, v_0\}$, then they

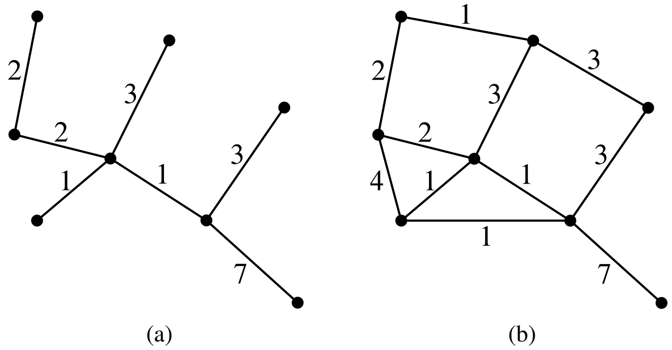


Figure 5.30 A spanning tree (a) with weight 19 for a graph (b).

remain joined by a walk containing the remainder of the cycle. By Lemma 5.4.2, they must also then be joined by a path. 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 the theorem must be true. ■

5.7.4 Minimum Weight Spanning Trees

Spanning trees are interesting because they connect all the nodes of a graph using the smallest possible number of edges. For example the spanning tree for the 6-node graph shown in Figure 5.29 has 5 edges.

Spanning trees are very useful in practice, but in the real world, not all spanning trees are equally desirable. That’s because, in practice, there are often costs associated with the edges of the graph.

For example, suppose the nodes of a graph represent buildings or towns and edges represent connections between buildings or towns. The cost to actually make a connection may vary a lot from one pair of buildings or towns to another. The cost might depend on distance or topography. For example, the cost to connect LA to NY might be much higher than that to connect NY to Boston. Or the cost of a pipe through Manhattan might be more than the cost of a pipe through a cornfield.

In any case, we typically represent the cost to connect pairs of nodes with a weighted edge, where the weight of the edge is its cost. The weight of a spanning tree is then just the sum of the weights of the edges in the tree. For example, the weight of the spanning tree shown in Figure 5.30 is 19.

The goal, of course, is to find the spanning tree with minimum weight, called the min-weight spanning tree (MST for short).

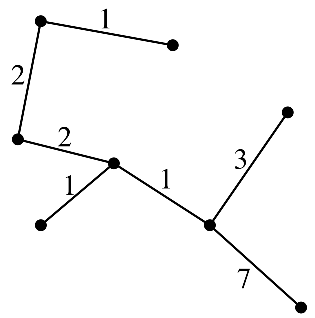


Figure 5.31 An MST with weight 17 for the graph in Figure 5.30(b).

Definition 5.7.6. The *min-weight spanning tree* (MST) of an edge-weighted graph G is the spanning tree of G with the smallest possible sum of edge weights.

Is the spanning tree shown in Figure 5.30(a) an MST of the weighted graph shown in Figure 5.30(b)? Actually, it is not, since the tree shown in Figure 5.31 is also a spanning tree of the graph shown in Figure 5.30(b), and this spanning tree has weight 17.

What about the tree shown in Figure 5.31? Is it an MST? It seems to be, but how do we prove it? In general, how do we find an MST? We could, of course, enumerate all trees, but this could take forever for very large graphs.

Here are two possible algorithms:

Algorithm 1. *Grow a tree one edge at a time by adding the minimum weight edge possible to the tree, making sure that you have a tree at each step.*

Algorithm 2. *Grow a subgraph one edge at a time by adding the minimum-weight edge possible to the subgraph, making sure that you have an acyclic subgraph at each step.*

For example, in the weighted graph we have been considering, we might run Algorithm 1 as follows. We would start by choosing one of the weight 1 edges, since this is the smallest weight in the graph. Suppose we chose the weight 1 edge on the bottom of the triangle of weight 1 edges in our graph. This edge is incident to two weight 1 edges, a weight 4 edge, a weight 7 edge, and a weight 3 edge. We would then choose the incident edge of minimum weight. In this case, one of the two weight 1 edges. At this point, we cannot choose the third weight 1 edge since this would form a cycle, but we can continue by choosing a weight 2 edge. We might end up with the spanning tree shown in Figure 5.32, which has weight 17, the smallest we've seen so far.

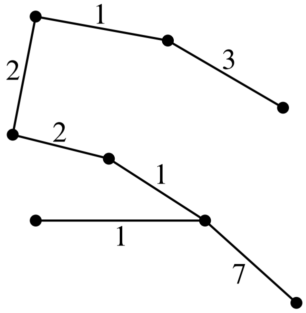


Figure 5.32 A spanning tree found by Algorithm 1.

Now suppose we instead ran Algorithm 2 on our graph. We might again choose the weight 1 edge on the bottom of the triangle of weight 1 edges in our graph. Now, instead of choosing one of the weight 1 edges it touches, we might choose the weight 1 edge on the top of the graph. Note that this edge still has minimum weight, and does not cause us to form a cycle, so Algorithm 2 can choose it. We would then choose one of the remaining weight 1 edges. Note that neither causes us to form a cycle. Continuing the algorithm, we may end up with the same spanning tree in Figure 5.32, though this need not always be the case.

It turns out that both algorithms work, but they might end up with different MSTs. The MST is not necessarily unique—indeed, if all edges of an n-node graph have the same weight ($= 1$), then all spanning trees have weight $n - 1$.

These are examples of greedy approaches to optimization. Sometimes it works and sometimes it doesn't. The good news is that it works to find the MST. In fact, both variations work. It's a little easier to prove it for Algorithm 2, so we'll do that one here.

Theorem 5.7.7. *For any connected, weighted graph G, Algorithm 2 produces an MST for G.*

Proof. The proof is a bit tricky. We need to show the algorithm terminates, that is, that if we have selected fewer than $n - 1$ edges, then we can always find an edge to add that does not create a cycle. We also need to show the algorithm creates a tree of minimum weight.

The key to doing all of this is to show that the algorithm never gets stuck or goes in a bad direction by adding an edge that will keep us from ultimately producing an MST. The natural way to prove this is to show that the set of edges selected at any point is contained in some MST—that is, we can always get to where we need to be. We'll state this as a lemma.

Lemma 5.7.8. *For any $m \geq 0$, let S consist of the first m edges selected by Algorithm 2. Then there exists some MST $T = (V, E)$ for G such that $S \subseteq E$, that is, the set of edges that we are growing is always contained in some MST.*

We'll prove this momentarily, but first let's see why it helps to prove the theorem. Assume the lemma is true. Then how do we know Algorithm 2 can always find an edge to add without creating a cycle? Well, as long as there are fewer than $n - 1$ edges picked, there exists some edge in $E - S$ and so there is an edge that we can add to S without forming a cycle. Next, how do we know that we get an MST at the end? Well, once $m = n - 1$, we know that S is an MST.

Ok, so the theorem is an easy corollary of the lemma. To prove the lemma, we'll use induction on the number of edges chosen by the algorithm so far. This is very typical in proving that an algorithm preserves some kind of invariant condition—induct on the number of steps taken, that is, the number of edges added.

Our inductive hypothesis $P(m)$ is the following: for any G and any set S of m edges initially selected by Algorithm 2, there exists an MST $T = (V, E)$ of G such that $S \subseteq E$.

For the base case, we need to show $P(0)$. In this case, $S = \emptyset$, so $S \subseteq E$ trivially holds for any MST $T = (V, E)$.

For the inductive step, we assume $P(m)$ holds and show that it implies $P(m+1)$. Let e denote the $(m+1)$st edge selected by Algorithm 2, and let S denote the first m edges selected by Algorithm 2. Let $T^* = (V^*, E^*)$ be the MST such that $S \subseteq E^*$, which exists by the inductive hypothesis. There are now two cases:

Case 1: $e \in E^*$, in which case $S \cup \{e\} \subseteq E^*$, and thus $P(m + 1)$ holds.

Case 2: $e \notin E^*$, as illustrated in Figure 5.33. Now we need to find a different MST that contains S and e.

What happens when we add e to T^*? Since T^* is a tree, we get a cycle. (Here we used part 3 of Theorem 5.7.4.) Moreover, the cycle cannot only contains edges in S, since e was chosen so that together with the edges in S, it does not form a cycle. This implies that $\{e\} \cup T^*$ contains a cycle that contains an edge e' of $E^* - S$. For example, such an e' is shown in Figure 5.33.

Note that the weight of e is at most that of e'. This is because Algorithm 2 picks the minimum weight edge that does not make a cycle with S. Since $e' \in T^*$, e' cannot make a cycle with S and if the weight of e were greater than the weight of e', Algorithm 2 would not have selected e ahead of e'.

Okay, we're almost done. Now we'll make an MST that contains $S \cup \{e\}$. Let $T^{**} = (V, E^{**})$ where $E^{**} = (E^* - \{e'\}) \cup \{e\}$, that is, we swap e and e' in T^*.

Claim 5.7.9. *T^{**} is an MST.*

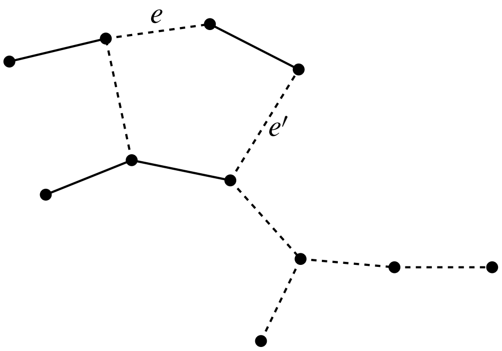


Figure 5.33 The graph formed by adding e to T^*. Edges of S are denoted with solid lines and edges of $E^* - S$ are denoted with dashed lines.

Proof of claim. We first show that T^{**} is a spanning tree. T^{**} is acyclic because it was produced by removing an edge from the only cycle in $T^* \cup \{e\}$. T^{**} is connected since the edge we deleted from $T^* \cup \{e\}$ was on a cycle. Since T^{**} contains all the nodes of G, it must be a spanning tree for G.

Now let's look at the weight of T^{**}. Well, since the weight of e was at most that of e', the weight of T^{**} is at most that of T^*, and thus T^{**} is an MST for G. ■

Since $S \cup \{e\} \subseteq E^{**}$, $P(m + 1)$ holds. Thus, Algorithm 2 must eventually produce an MST. This will happens when it adds $n - 1$ edges to the subgraph it builds. ■

So now we know for sure that the MST for our example graph has weight 17 since it was produced by Algorithm 2. And we have a fast algorithm for finding a minimum-weight spanning tree for any graph.

5.8 Planar Graphs

5.8.1 Drawing Graphs in the Plane

Suppose there are three dog houses and three human houses, as shown in Figure 5.34. Can you find a route from each dog house to each human house such that no route crosses any other route?

A *quadrapus* is a little-known animal similar to an octopus, but with four arms. Suppose there are five quadrapi resting on the sea floor, as shown in Figure 5.35.

Figure 5.34 Three dog houses and and three human houses. Is there a route from each dog house to each human house so that no pair of routes cross each other?

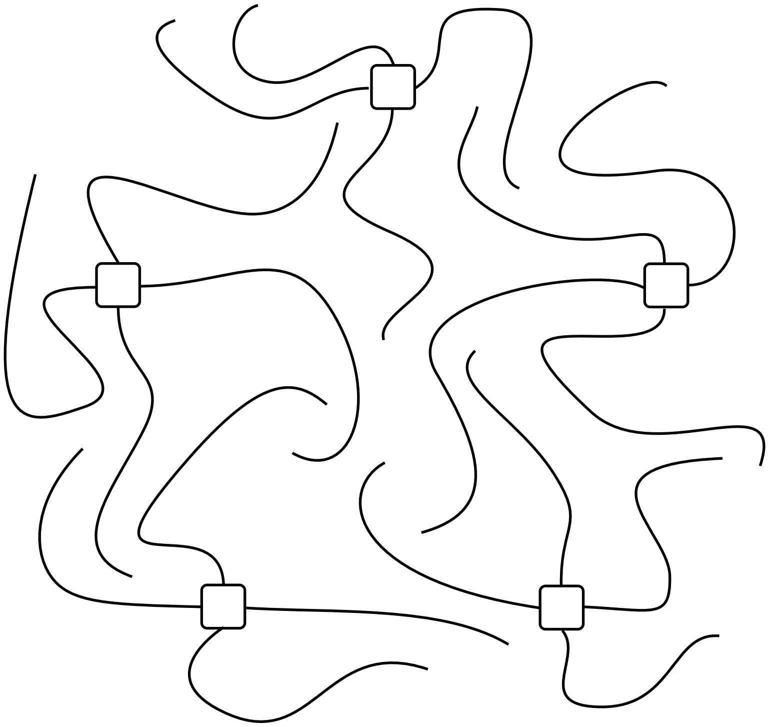


Figure 5.35 Five quadrapi (4-armed creatures).

Can each quadrapus simultaneously shake hands with every other in such a way that no arms cross?

Definition 5.8.1. A *drawing of a graph in the plane* consists of an assignment of vertices to distinct points in the plane and an assignment of edges to smooth, non-self-intersecting curves in the plane (whose endpoints are the nodes incident to the edge). The drawing is *planar* (that is, it is a *planar drawing*) if none of the curves "cross"—that is, if the only points that appear on more than one curve are the vertex points. A *planar graph* is a graph that has a planar drawing.

Thus, these two puzzles are asking whether the graphs in Figure 5.36 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, if you remove an edge from either of them, then the resulting graphs *can* be redrawn in the plane so that no edges cross. For example, we have illustrated the planar drawings for each resulting graph in Figure 5.37.

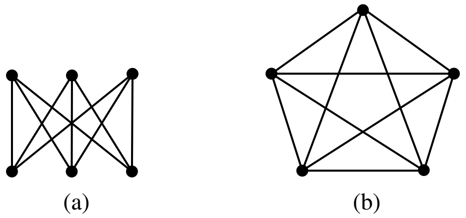


Figure 5.36 $K_{3,3}$ (a) and K_5 (b). Can you redraw these graphs so that no pairs of edges cross?

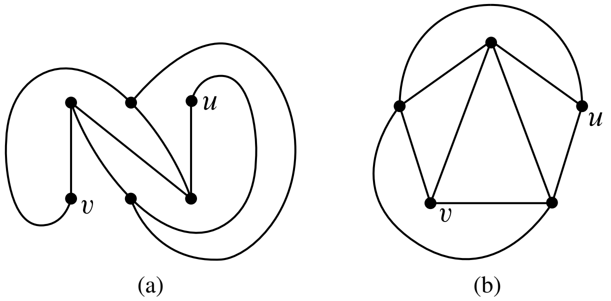


Figure 5.37 Planar drawings of $K_{3,3} - \{u, v\}$ (a) and $K_5 - \{u, v\}$ (b).

Planar drawings have applications in circuit layout and are helpful in displaying graphical data such as program flow charts, organizational charts, and scheduling conflicts. For these applications, the goal is to draw the graph in the plane with as few edge crossings as possible. (See the box on the following page for one such example.)

5.8.2 A Recursive Definition for Planar Graphs

Definition 5.8.1 is perfectly precise but has the challenge that it requires us to work with concepts such as a "smooth curve" when trying to prove results about planar graphs. The trouble is that we have not really laid the groundwork from geometry and topology to be able to reason carefully about such concepts. For example, we haven't really defined what it means for a curve to be smooth—we just drew a simple picture (for example, Figure 5.37) and hoped you would get the idea.

Relying on pictures to convey new concepts is generally not a good idea and can sometimes lead to disaster (or, at least, false proofs). Indeed, it is because of this issue that there have been so many false proofs relating to planar graphs over time.[18] Such proofs usually rely way too heavily on pictures and have way too many statements like,

> As you can see from Figure ABC, it must be that property XYZ holds for all planar graphs.

The good news is that there is another way to define planar graphs that uses only discrete mathematics. In particular, we can define the class of planar graphs as a recursive data type. In order to understand how it works, we first need to understand the concept of a *face* in a planar drawing.

Faces

In a planar drawing of a graph. the curves corresponding to the edges divide up the plane into connected regions. These regions are called the *continuous faces*[19] of the drawing. For example, the drawing in Figure 5.38 has four continuous faces. Face IV, which extends off to infinity in all directions, is called the *outside face*.

Notice that the vertices along the boundary of each of the faces in Figure 5.38 form a cycle. For example, labeling the vertices as in Figure 5.39, the cycles for the face boundaries are

$$abca \qquad abda \qquad bcdb \qquad acda. \tag{5.4}$$

[18] The false proof of the 4-Color Theorem for planar graphs is not the only example.

[19] Most texts drop the word *continuous* from the definition of a face. We need it to differentiate the connected region in the plane from the closed walk in the graph that bounds the region, which we will call a *discrete face*.

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 mightily 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.' "[17]

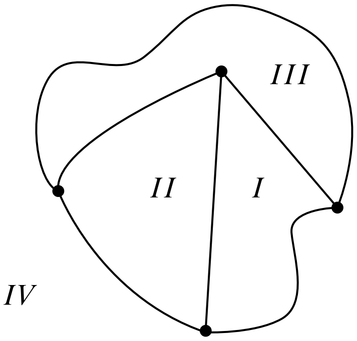


Figure 5.38 A planar drawing with four faces.

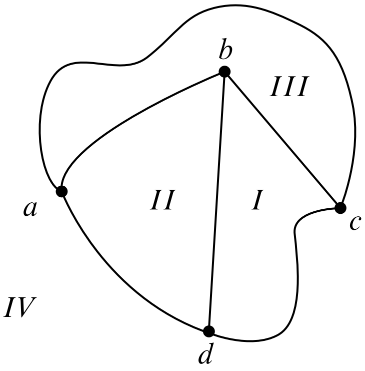


Figure 5.39 The drawing with labeled vertices.

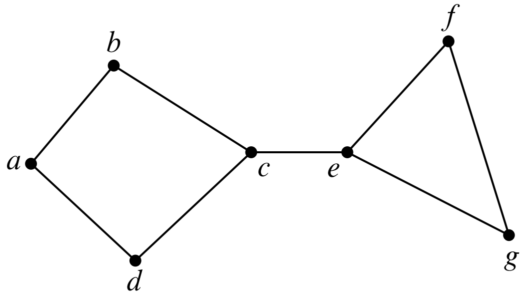


Figure 5.40 A planar drawing with a *bridge*, namely the edge $\{c, e\}$.

These four cycles correspond nicely to the four continuous faces in Figure 5.39. So nicely, in fact, that we can identify each of the faces in Figure 5.39 by its cycle. For example, the cycle $abca$ identifies face III. Hence, we say that the cycles in Equation 5.4 are the *discrete faces* of the graph in Figure 5.39. We use the term "discrete" since cycles in a graph are a discrete data type (as opposed to a region in the plane, which is a continuous data type).

Unfortunately, continuous faces in planar drawings are not always bounded by cycles in the graph—things can get a little more complicated. For example, consider the planar drawing in Figure 5.40. This graph has what we will call a *bridge* (namely, the edge $\{c, e\}$) and the outer face is

$$abcefgecda.$$

This is not a cycle, since it has to traverse the bridge $\{c, e\}$ twice, but it is a closed walk.

As another example, consider the planar drawing in Figure 5.41. This graph has what we will call a *dongle* (namely, the nodes v, x, y, and w, and the edges incident

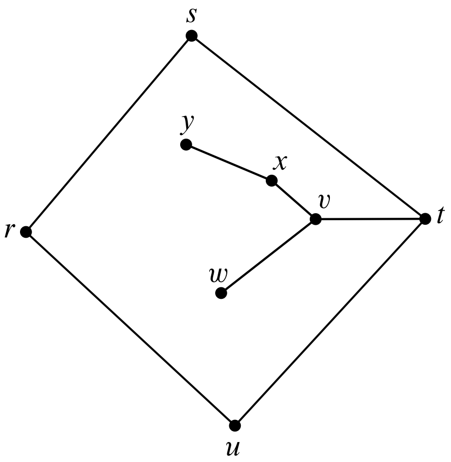


Figure 5.41 A planar drawing with a *dongle*, namely the subgraph with nodes v, w, x, y.

to them) and the inner face is

$$rstvxyxvwvtur.$$

This is not a cycle because it has to traverse *every* edge of the dongle twice—once "coming" and once "going," but once again, it is a closed walk.

It turns out that bridges and dongles are the only complications, at least for connected graphs. In particular, every continuous face in a planar drawing corresponds to a closed walk in the graph. We refer to such closed walks as the *discrete faces* of the drawing.

A Recursive Definition for Planar Embeddings

The association between the continuous faces of a planar drawing and closed walks will allow us to characterize a planar drawing in terms of the closed walks that bound the continuous faces. In particular, it 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 closed walks that we could get by drawing one edge after another.

Definition 5.8.2. A *planar embedding* of a *connected* graph consists of a nonempty set of closed walks 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 closed walk v.

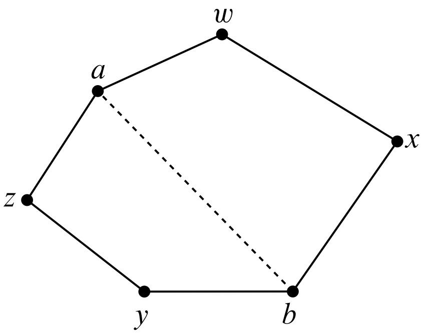


Figure 5.42 The "split a face" case.

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 closed walk of the form

$$a \ldots b \ldots a.$$

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[20]

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

as illustrated in Figure 5.42.

Constructor Case (add a bridge): Suppose G and H are connected graphs with planar embeddings and disjoint sets of vertices. Let a be a vertex on a discrete face, γ, in the embedding of G. That is, γ is of the form

$$a \ldots a.$$

Similarly, let b be a vertex on a discrete face, δ, in the embedding of H. So δ is of the form

$$b \cdots b.$$

Then the graph obtained by connecting G and H with a new edge, $\{a, b\}$, has a planar embedding whose discrete faces are the union of the discrete faces of G and

[20] There is a special case of 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.

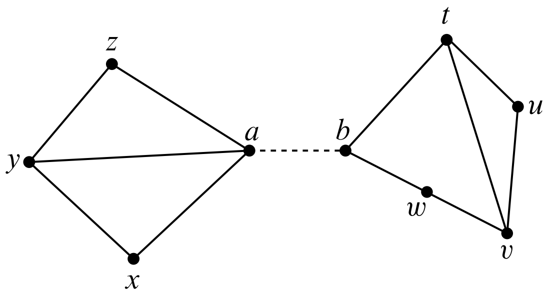


Figure 5.43 The "add a bridge" case.

H, except that faces γ and δ are replaced by one new face

$$a \ldots ab \cdots ba.$$

This is illustrated in Figure 5.43, where the faces of G and H are:

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

and after adding the bridge $\{a, b\}$, there is a single connected graph with faces

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

Does It Work?

Yes! In general, a graph is planar if and only if each of its connected components has a planar embedding as defined in Definition 5.8.2. Unfortunately, proving this fact requires a bunch of mathematics that we don't cover in this text—stuff like geometry and topology. Of course, that is why we went to the trouble of including Definition 5.8.2—we don't want to deal with that stuff in this text and now that we have a recursive definition for planar graphs, we won't need to. That's the good news.

The bad news is that Definition 5.8.2 looks a lot more complicated than the intuitively simple notion of a drawing where edges don't cross. It seems like it would be easier to stick to the simple notion and give proofs using pictures. Perhaps so, but your proofs are more likely to be complete and correct if you work from the discrete Definition 5.8.2 instead of the continuous Definition 5.8.1.

Where Did the Outer Face Go?

Every planar drawing has an immediately-recognizable outer face—its the one that goes to infinity in all directions. But where is the outer face in a planar embedding?

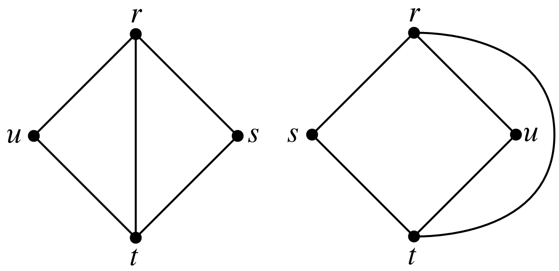


Figure 5.44 Two illustrations of the same embedding.

There isn't one! That's because 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. For example, the two embeddings shown in Figure 5.44 are really the same.

This is what justifies the "add a bridge" case in Definition 5.8.2: 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.

5.8.3 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. For example, we will now use Definition 5.8.2 and structural induction to establish one of the most basic properties of a connected planar graph; namely, the number of vertices and edges completely determines the number of faces in every possible planar embedding of the graph.

Theorem 5.8.3 (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 5.38, $|V| = 4$, $|E| = 6$, 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. Since

$$\begin{aligned}
(v_G + v_H) &- (e_G + e_H + 1) + (f_G + f_H - 1) \\
&= (v_G - e_G + f_G) + (v_H - e_H + f_H) - 2 \\
&= (2) + (2) - 2 && \text{(by structural induction hypothesis)} \\
&= 2,
\end{aligned}$$

$v - e + f$ remains equal to 2 for the constructed embedding. That is, $P(E)$ also holds in this case.

This completes the proof of the constructor cases, and the theorem follows by structural induction. ■

5.8.4 Bounding the Number of Edges in a Planar Graph

Like Euler's formula, the following lemmas follow by structural induction from Definition 5.8.2.

Lemma 5.8.4. *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 5.8.5. *In a planar embedding of a connected graph with at least three vertices, each face is of length at least three.*

Combining Lemmas 5.8.4 and 5.8.5 with Euler's Formula, we can now prove that planar graphs have a limited number of edges:

Theorem 5.8.6. *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 5.8.4, 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 5.8.5, when $v \geq 3$, each face boundary is of length at least three, so this sum is at least $3f$. This implies that

$$3f \leq 2e. \tag{5.5}$$

But $f = e - v + 2$ by Euler's formula, and substituting into (5.5) gives

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

■

5.8.5 Returning to K_5 and $K_{3,3}$

Theorem 5.8.6 lets us prove that the quadrapi can't all shake hands without crossing. Representing quadrapi 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 Theorem 5.8.6 required for K_5 to be planar, which proves

Corollary 5.8.7. *K_5 is not planar.*

We can also use Euler's Formula to show that $K_{3,3}$ is not planar. The proof is similar to that of Theorem 5.8.6 except that we use the additional fact that $K_{3,3}$ is a bipartite graph.

Theorem 5.8.8. *$K_{3,3}$ is not planar.*

Proof. By contradiction. Assume $K_{3,3}$ is planar and consider any planar embedding of $K_{3,3}$ with f faces. Since $K_{3,3}$ is bipartite, we know by Theorem 5.6.2 that $K_{3,3}$ does not contain any closed walks of odd length. By Lemma 5.8.5, every face has length at least 3. This means that every face in any embedding of $K_{3,3}$ must have length at least 4. Plugging this fact into the proof of Theorem 5.8.6, we find that the sum of the lengths of the face boundaries is exactly $2e$ and at least $4f$. Hence,

$$4f \leq 2e$$

for any bipartite graph.

Plugging in $e = 9$ and $v = 6$ for $K_{3,3}$ in Euler's Formula, we find that

$$f = 2 + e - v = 5.$$

But

$$4 \cdot 5 \not\leq 2 \cdot 9,$$

and so we have a contradiction. Hence $K_{3,3}$ must not be planar. ■

5.8.6 Another Characterization for Planar Graphs

We did not choose to pick on K_5 and $K_{3,3}$ because of their application to dog houses or quadrapi shaking hands. Rather, we selected these graphs as examples because they provide another way to characterize the set of planar graphs.

Theorem 5.8.9 (Kuratowski)**.** *A graph is not planar if and only if it contains K_5 or $K_{3,3}$ as a minor.*

Definition 5.8.10. A *minor* of a graph G is a graph that can be obtained by repeatedly[21] deleting vertices, deleting edges, and merging *adjacent* vertices of G. *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 5.45.

For example, Figure 5.46 illustrates why C_3 is a minor of the graph in Figure 5.46(a). In fact C_3 is a minor of a connected graph G if and only if G is not a tree.

We will not prove Theorem 5.8.9 here, nor will we prove the following handy facts, which are obvious given the continuous Definition 5.8.1, and which can be proved using the recursive Definition 5.8.2.

Lemma 5.8.11. *Deleting an edge from a planar graph leaves another planar graph.*

Corollary 5.8.12. *Deleting a vertex from a planar graph, along with all its incident edges, leaves another planar graph.*

Theorem 5.8.13. *Any subgraph of a planar graph is planar.*

Theorem 5.8.14. *Merging two adjacent vertices of a planar graph leaves another planar graph.*

[21] The three operations can be performed in any order and in any quantities, or not at all.

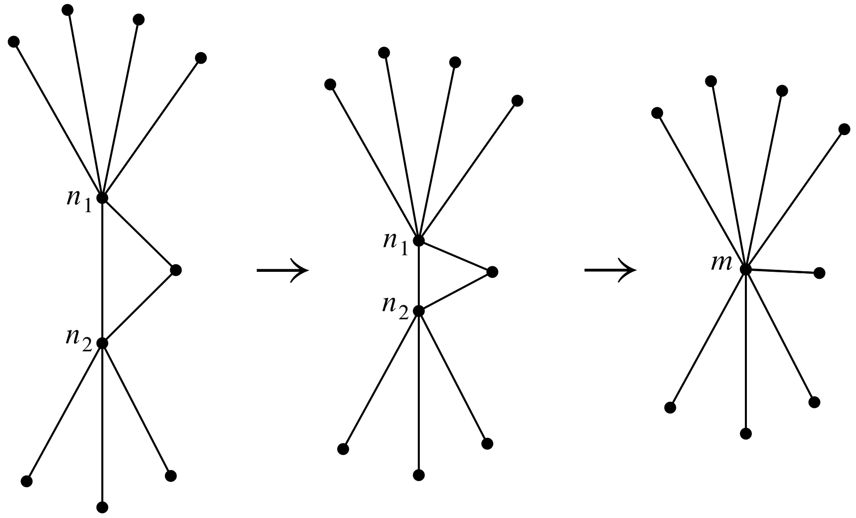


Figure 5.45 Merging adjacent vertices n_1 and n_2 into new vertex, m.

5.8.7 Coloring Planar Graphs

We've covered a lot of ground with planar graphs, but not nearly enough to prove the famous 4-color theorem. But we can get awfully close. Indeed, we have done almost enough work to prove that every planar graph can be colored using only 5 colors. We need only one more lemma:

Lemma 5.8.15. *Every planar graph has a vertex of degree at most five.*

Proof. By contradiction. If every vertex had degree at least 6, then the sum of the vertex degrees is at least $6v$, but since the sum of the vertex degrees equals $2e$, by the Handshake Lemma (Lemma 5.2.1), we have $e \geq 3v$ contradicting the fact that $e \leq 3v - 6 < 3v$ by Theorem 5.8.6. ■

Theorem 5.8.16. *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.

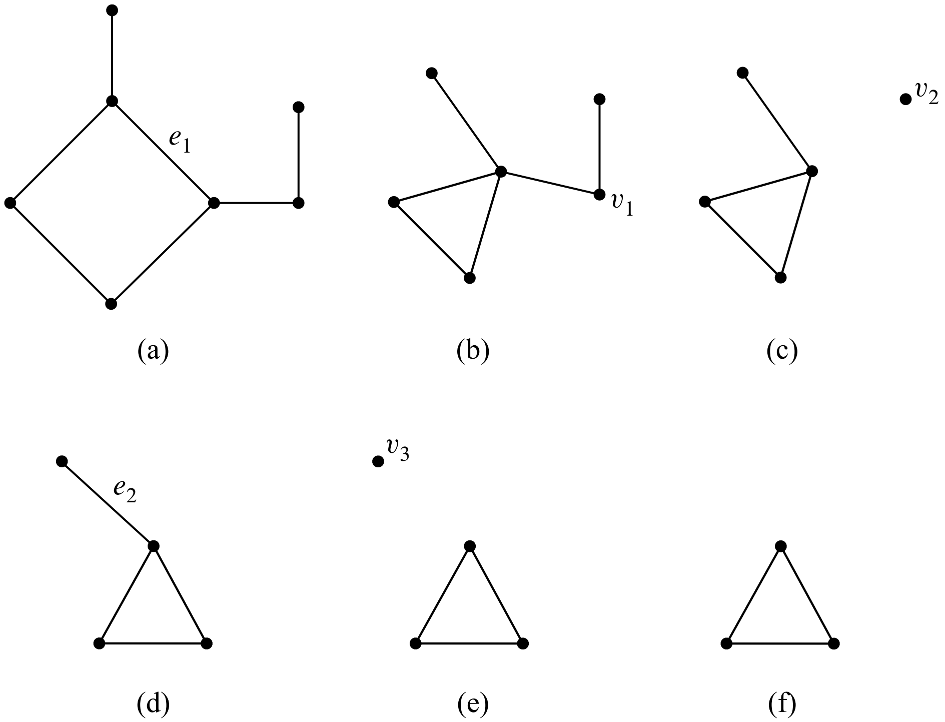


Figure 5.46 One method by which the graph in (a) can be reduced to C_3 (f), thereby showing that C_3 is a minor of the graph. The steps are: merging the nodes incident to e_1 (b), deleting v_1 and all edges incident to it (c), deleting v_2 (d), deleting e_2, and deleting v_3 (f).

First, choose a vertex, g, of G with degree at most 5; Lemma 5.8.15 guarantees there will be such a vertex.

Case 1: ($\deg(g) < 5$): Deleting g from G leaves a graph, H, that is planar by Corollary 5.8.12, 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 5.8.13 (since K_5 is not planar). 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. In this new graph, n_2 is adjacent to m, and the graph is planar by Theorem 5.8.14. So we can then merge m and n_2 into a another new vertex, m', resulting in a new graph, G', which by Theorem 5.8.14 is also planar. Since G' has $v - 1$ vertices, it is five-colorable by the induction hypothesis.

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.

■

5.8.8 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 in Figure 5.34: 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

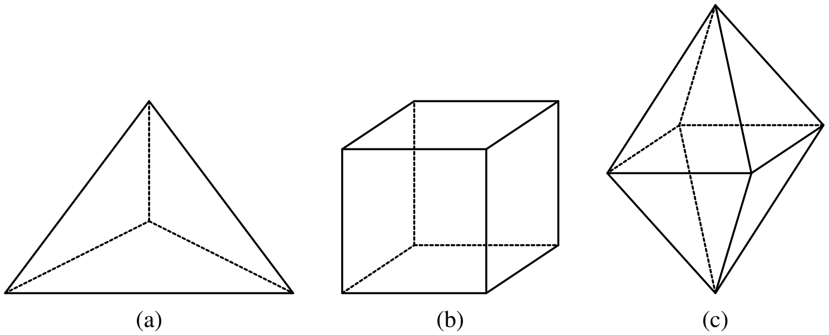


Figure 5.47 The tetrahedron (a), cube (b), and octahedron (c).

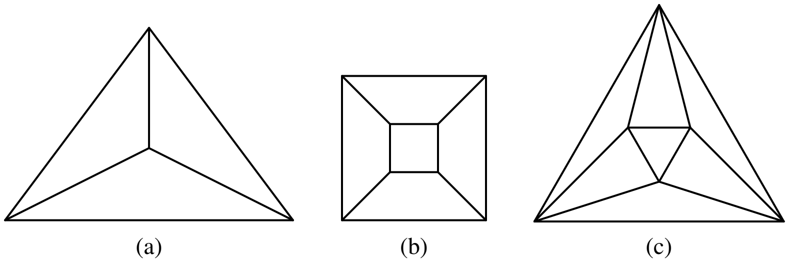


Figure 5.48 Planar embeddings of the tetrahedron (a), cube (b, and octahedron (c).

corners of the polyhedron corresponding to vertices of the graph. 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 in Figure 5.34 are shown in Figure 5.48.

Let m be the number of faces that meet at each corner of a polyhedron, and let n be the number of edges on each face. In the corresponding planar graph, there are m edges incident to each of the v vertices. By the Handshake Lemma 5.2.1, 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

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

Figure 5.49 The only possible regular polyhedra.

gives:

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

which simplifies to

$$\frac{1}{m} + \frac{1}{n} = \frac{1}{e} + \frac{1}{2} \tag{5.6}$$

Equation 5.6 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, as shown in Figure 5.49. 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. The largest polyhedron, the dodecahedron, was the other great mathematical secret of the Pythagorean sect.

The 5 polyhedra in Figure 5.49 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!

6 Directed Graphs

6.1 Definitions

So far, we have been working with graphs with undirected edges. A *directed edge* is an edge where the endpoints are distinguished—one is the *head* and one is the *tail*. In particular, a directed edge is specified as an ordered pair of vertices u, v and is denoted by (u, v) or $u \to v$. In this case, u is the *tail* of the edge and v is the *head*. For example, see Figure 6.1.

A graph with directed edges is called a *directed graph* or *digraph*.

Definition 6.1.1. A directed graph $G = (V, E)$ consists of a nonempty set of nodes V and a set of directed edges E. Each edge e of E is specified by an ordered pair of vertices $u, v \in V$. A directed graph is *simple* if it has no *loops* (that is, edges of the form $u \to u$) and no multiple edges.

Since we will focus on the case of simple directed graphs in this chapter, we will generally omit the word *simple* when referring to them. Note that such a graph can contain an edge $u \to v$ as well as the edge $v \to u$ since these are different edges (for example, they have a different tail).

Directed graphs arise in applications where the relationship represented by an edge is 1-way or asymmetric. Examples include: a 1-way street, one person likes another but the feeling is not necessarily reciprocated, a communication channel such as a cable modem that has more capacity for downloading than uploading, one entity is larger than another, and one job needs to be completed before another job can begin. We'll see several such examples in this chapter and also in Chapter 7.

Most all of the definitions for undirected graphs from Chapter 5 carry over in a natural way for directed graphs. For example, two directed graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are *isomorphic* if there exists a bijection $f : V_1 \to V_2$ such that for every pair of vertices $u, v \in V_1$,

$$u \to v \in E_1 \qquad \text{IFF} \qquad f(u) \to f(v) \in E_2.$$

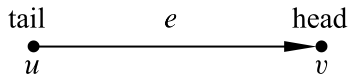


Figure 6.1 A directed edge $e = (u, v)$. u is the tail of e and v is the head of e.

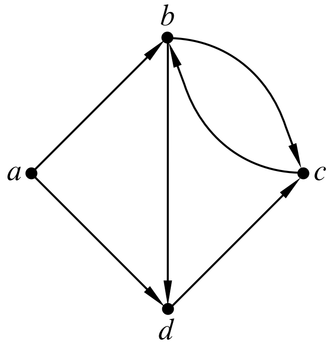


Figure 6.2 A 4-node directed graph with 6 edges.

Directed graphs have adjacency matrices just like undirected graphs. In the case of a directed graph $G = (V, E)$, the adjacency matrix $A_G = \{a_{ij}\}$ is defined so that

$$a_{ij} = \begin{cases} 1 & \text{if } i \to j \in E \\ 0 & \text{otherwise.} \end{cases}$$

The only difference is that the adjacency matrix for a directed graph is not necessarily symmetric (that is, it may be that $A_G^T \neq A_G$).

6.1.1 Degrees

With directed graphs, the notion of degree splits into *indegree* and *outdegree*. For example, indegree$(c) = 2$ and outdegree$(c) = 1$ for the graph in Figure 6.2. If a node has outdegree 0, it is called a *sink*; if it has indegree 0, it is called a *source*. The graph in Figure 6.2 has one source (node a) and no sinks.

6.1.2 Directed Walks, Paths, and Cycles

The definitions for (directed) walks, paths, and cycles in a directed graph are similar to those for undirected graphs except that the direction of the edges need to be consistent with the order in which the walk is traversed.

Definition 6.1.2. A *directed walk* (or more simply, a *walk*) in a directed graph G is a sequence of vertices $v_0, v_1, \ldots, v_k$ and edges

$$v_0 \to v_1, v_1 \to v_2, \ldots, v_{k-1} \to v_k$$

such that $v_{i-1} \to v_i$ is an edge of G for all i where $0 \leq i < k$. A *directed path* (or *path*) in a directed graph is a walk where the nodes in the walk are all different. A *directed closed walk* (or *closed walk*) in a directed graph is a walk where $v_0 = v_k$. A *directed cycle* (or *cycle*) in a directed graph is a closed walk where all the vertices v_i are different for $0 \leq i < k$.

As with undirected graphs, we will typically refer to a walk in a directed graph by a sequence of vertices. For example, for the graph in Figure 6.2,

- a, b, c, b, d is a walk,
- a, b, d is a path,
- d, c, b, c, b, d is a closed walk, and
- b, d, c, b is a cycle.

Note that b, c, b is also a cycle for the graph in Figure 6.2. This is a cycle of length 2. Such cycles are not possible with undirected graphs.

Also note that

$$c, b, a, d$$

is *not* a walk in the graph shown in Figure 6.2, since $b \to a$ is not an edge in this graph. (You are *not* allowed to traverse edges in the wrong direction as part of a walk.)

A path or cycle in a directed graph is said to be *Hamiltonian* if it visits every node in the graph. For example, a, b, d, c is the only Hamiltonian path for the graph in Figure 6.2. The graph in Figure 6.2 does not have a Hamiltonian cycle.

A walk in a directed graph is said to be *Eulerian* if it contains every edge. The graph shown in Figure 6.2 does not have an Eulerian walk. Can you see why not? (Hint: Look at node a.)

6.1.3 Strong Connectivity

The notion of being connected is a little more complicated for a directed graph than it is for an undirected graph. For example, should we consider the graph in Figure 6.2 to be connected? There is a path from node a to every other node so on that basis, we might answer "Yes." But there is no path from nodes b, c, or d to node a, and so on that basis, we might answer "No." For this reason, graph theorists have come up with the notion of *strong* connectivity for directed graphs.

Definition 6.1.3. A directed graph $G = (V, E)$ is said to be *strongly connected* if for every pair of nodes $u, v \in V$, there is a directed path from u to v (and vice-versa) in G.

For example, the graph in Figure 6.2 is not strongly connected since there is no directed path from node b to node a. But if node a is removed, the resulting graph would be strongly connected.

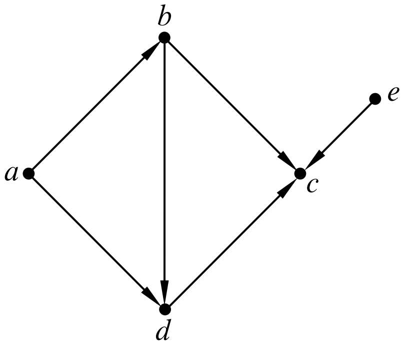


Figure 6.3 A 4-node directed acyclic graph (DAG).

A directed graph is said to be *weakly connected* (or, more simply, *connected*) if the corresponding undirected graph (where directed edges $u \to v$ and/or $v \to u$ are replaced with a single undirected edge $\{u, v\}$ is connected. For example, the graph in Figure 6.2 is weakly connected.

6.1.4 DAGs

If an undirected graph does not have any cycles, then it is a tree or a forest. But what does a directed graph look like if it has no cycles? For example, consider the graph in Figure 6.3. This graph is weakly connected and has no directed cycles but it certainly does not look like a tree.

Definition 6.1.4. A directed graph is called a *directed acyclic graph* (or, *DAG*) if it does not contain any directed cycles.

A first glance, DAGs don't appear to be particularly interesting. But first impressions are not always accurate. In fact, DAGs arise in many scheduling and optimization problems and they have several interesting properties. We will study them extensively in Chapter 7.

6.2 Tournament Graphs

Suppose that n players compete in a round-robin tournament and that for every pair of players u and v, either u beats v or v beats u. Interpreting the results of a round-robin tournament can be problematic—there might be all sorts of cycles where x beats y and y beats z, yet z beats x. Who is the best player? Graph theory does not solve this problem but it can provide some interesting perspectives.

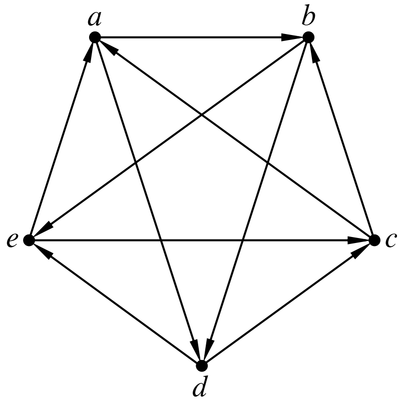


Figure 6.4 A 5-node tournament graph.

The results of a round-robin tournament can be represented with a *tournament graph*. This is a directed graph in which the vertices represent players and the edges indicate the outcomes of games. In particular, an edge from u to v indicates that player u defeated player v. In a round-robin tournament, every pair of players has a match. Thus, in a tournament graph there is either an edge from u to v or an edge from v to u (but not both) for *every* pair of distinct vertices u and v. An example of a tournament graph is shown in Figure 6.4.

6.2.1 Finding a Hamiltonian Path in a Tournament Graph

We're going to prove that in every round-robin tournament, there exists a ranking of the players such that each player lost to the player one position higher. For example, in the tournament corresponding to Figure 6.4, the ranking

$$a > b > d > e > c$$

satisfies this criterion, because b lost to a, d lost to b, e lost to d, and c lost to e. In graph terms, proving the existence of such a ranking amounts to proving that every tournament graph has a Hamiltonian path.

Theorem 6.2.1. *Every tournament graph contains a directed Hamiltonian path.*

Proof. We use strong induction. Let $P(n)$ be the proposition that every tournament graph with n vertices contains a directed Hamiltonian path.

Base case: $P(1)$ is trivially true; every graph with a single vertex has a Hamiltonian path consisting of only that vertex.

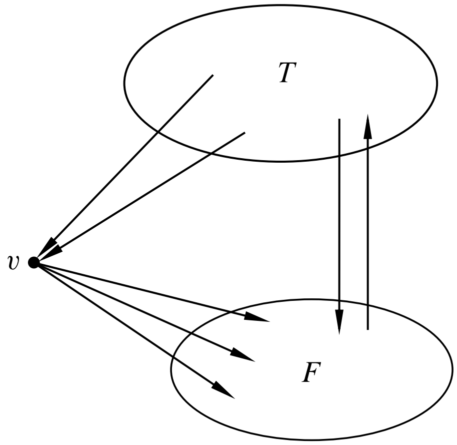


Figure 6.5 The sets T and F in a tournament graph.

Inductive step: For $n \geq 1$, we assume that $P(1), \ldots, P(n)$ are all true and prove $P(n+1)$. Consider a tournament graph $G = (V, E)$ with $n + 1$ players. Select one vertex v arbitrarily. Every other vertex in the tournament either has an edge *to* vertex v or an edge *from* vertex v. Thus, we can partition the remaining vertices into two corresponding sets, T and F, each containing at most n vertices, where $T = \{ u \mid u \to v \in E \}$ and $F = \{ u \mid v \to u \in E \}$. For example, see Figure 6.5.

The vertices in T together with the edges that join them form a smaller tournament. Thus, by strong induction, there is a Hamiltonian path within T. Similarly, there is a Hamiltonian path within the tournament on the vertices in F. Joining the path in T to the vertex v followed by the path in F gives a Hamiltonian path through the whole tournament. As special cases, if T or F is empty, then so is the corresponding portion of the path. ■

The ranking defined by a Hamiltonian path is not entirely satisfactory. For example, in the tournament associated with Figure 6.4, notice that the lowest-ranked player, c, actually defeated the highest-ranked player, a.

In practice, players are typically ranked according to how many victories they achieve. This makes sense for several reasons. One not-so-obvious reason is that if the player with the most victories does not beat some other player v, he is guaranteed to have at least beaten a third player who beat v. We'll prove this fact shortly. But first, let's talk about chickens.

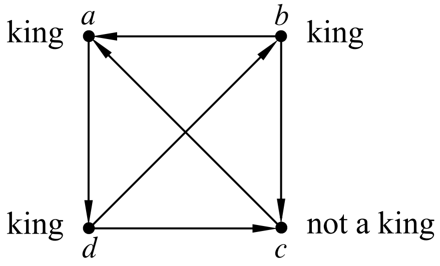


Figure 6.6 A 4-chicken tournament in which chickens a, b, and d are kings.

.

6.2.2 The King Chicken Theorem

Suppose that there are n chickens in a farmyard. Chickens are rather aggressive birds that tend to establish dominance in relationships by pecking. (Hence the term "pecking order.") In particular, for each pair of distinct chickens, either the first pecks the second or the second pecks the first, but not both. We say that chicken u *virtually pecks* chicken v if either:

- Chicken u directly pecks chicken v, or
- Chicken u pecks some other chicken w who in turn pecks chicken v.

A chicken that virtually pecks every other chicken is called a *king chicken*.

We can model this situation with a tournament digraph. The vertices are chickens, and an edge $u \to v$ indicates that chicken u pecks chicken v. In the tournament shown in Figure 6.6, three of the four chickens are kings. Chicken c is not a king in this example since it does not peck chicken b and it does not peck any chicken that pecks chicken b. Chicken a *is* a king since it pecks chicken d, who in turn pecks chickens b and c.

Theorem 6.2.2 (King Chicken Theorem)**.** *The chicken with the largest outdegree in an n-chicken tournament is a king.*

Proof. By contradiction. Let u be a node in a tournament graph $G = (V, E)$ with maximum outdegree and suppose that u is not a king. Let $Y = \{ v \mid u \to v \in E \}$ be the set of chickens that chicken u pecks. Then $\text{outdegree}(u) = |Y|$.

Since u is not a king, there is a chicken $x \notin Y$ (that is, x is not pecked by chicken u) and that is not pecked by any chicken in Y. Since for any pair of chickens, one pecks the other, this means that x pecks u as well as every chicken in Y. This means that

$$\text{outdegree}(x) = |Y| + 1 > \text{outdegree}(u).$$

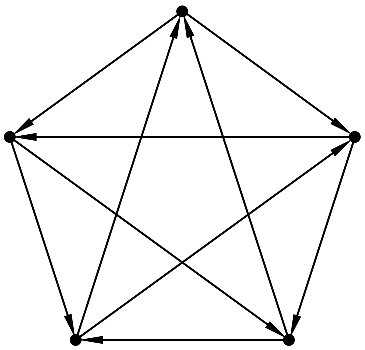


Figure 6.7 A 5-chicken tournament in which every chicken is a king.

But u was assumed to be the node with the largest degree in the tournament, so we have a contradiction. Hence, u must be a king. ■

Theorem 6.2.2 means that if the player with the most victories is defeated by another player x, then at least he/she defeats some third player that defeats x. In this sense, the player with the most victories has some sort of bragging rights over every other player. Unfortunately, as Figure 6.6 illustrates, there can be many other players with such bragging rights, even some with fewer victories. Indeed, for some tournaments, it is possible that every player is a "king." For example, consider the tournament illustrated in Figure 6.7.

6.3 Communication Networks

While reasoning about chickens pecking each other may be amusing (to mathematicians, at least), the use of directed graphs to model communication networks is very serious business. In the context of communication problems, vertices represent computers, processors, or switches, and edges 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, such as an array or butterfly, by contrast, find application in telephone switching systems and the communication hardware inside parallel computers.

6.3.1 Packet Routing

Whatever architecture is chosen, the goal of a communication network is to get data from *inputs* to *outputs*. In this text, we will focus on a model in which the data to be communicated is in the form of a *packet*. In practice, a packet would consist of a fixed amount of data, and a message (such as a web page or a movie) would consist of many packets.

For simplicity, we will restrict our attention to the scenario where there is just one packet at every input and where there is just one packet destined for each output. We will denote the number of inputs and output by N and we will often assume that N is a power of two.

We will specify the desired destinations of the packets by a permutation[1] of 0, 1, ..., $N - 1$. So a permutation, π, defines a *routing problem*: get a packet that starts at input i to output $\pi(i)$ for $0 \leq i < N$. 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 paths, P_i, for $i = 0, \ldots, N - 1$, where P_i goes from input i to output $\pi(i)$.

Of course, the goal is to get all the packets to their destinations as quickly as possible using as little hardware as possible. The time needed to get the packages to their destinations depends on several factors, such as how many switches they need to go through and how many packets will need to cross the same wire. We will assume that only one packet can cross a wire at a time. The complexity of the hardware depends on factors such as the number of switches needed and the size of the switches.

Let's see how all this works with an example—routing packets on a complete binary tree.

6.3.2 The Complete Binary Tree

One of the simplest structured communications networks is a *complete binary tree*. A complete binary tree with 4 inputs and 4 outputs is shown in Figure 6.8.

In this diagram and many that follow, the squares represent *terminals* (that is, the inputs and outputs), and 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 terminal in the complete binary tree is along the corresponding directed path. For

[1] A permutation of a sequence is a reordering of the sequence.

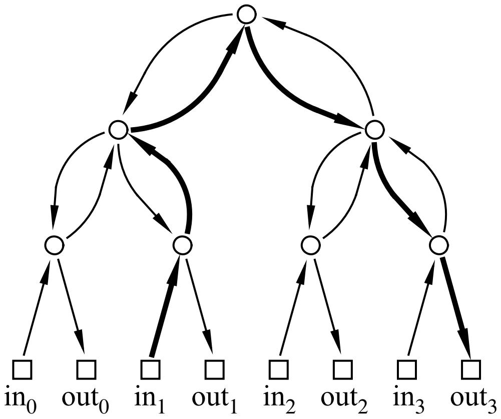


Figure 6.8 A 4-input, 4-output complete binary tree. The squares represent terminals (input and output registers) and the circles represent switches. Directed edges represent communication channels in the network through which data packets can move. The unique path from input 1 to output 3 is shown in bold.

example, the route of a packet traveling from input 1 to output 3 is shown in bold in Figure 6.8.

6.3.3 Network Diameter

The delay between the time that a packet arrives at an input and the time that it reaches its designated output is referred to as *latency* and it is a critical issue in communication networks. If congestion is not a factor, then this delay is generally proportional to the length of the path a packet follows. Assuming it takes one time unit to travel across a wire, 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.[2]

Generally a packet is routed from input to output using the shortest path possible. The length of this shortest path is the *distance* between the input and output. With a shortest path routing, the worst possible 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[3] is the maximum length of any shortest

[2] Latency can also be 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, which is 1 less than the number of edges traversed.

[3] 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.

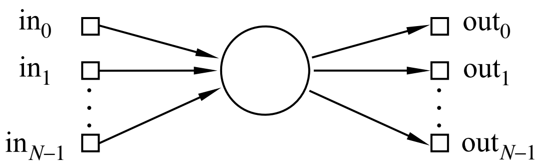


Figure 6.9 A monster $N \times N$ switch.

path between an input and an output. For example, in the complete binary tree shown in Figure 6.8, 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 $2^{20} = 1{,}048{,}576$ inputs and outputs using a complete binary tree and the worst input-output delay for any packet would be this diameter, namely, $2 \log(2^{20}) + 2 = 42$.

6.3.4 Switch Size

One way to reduce the diameter of a network (and hence the latency needed to route packets) 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, as shown in Figure 6.9. In this case, the "network" would consist of a single switch and the latency would be 2.

This isn't very productive, however, since we've just concealed the original network design problem inside this abstract monster 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.

6.3.5 Switch Count

Another goal in designing a communication network is to use as few switches as possible. The number of switches in a complete binary tree is $1 + 2 + 4 + 8 + \cdots + N = 2N - 1$, since there is 1 switch at the top (the "root switch"), 2 below it, 4 below those, and so forth. This is nearly the best possible with 3×3 switches, since at least one switch will be needed for each pair of inputs and outputs.

6.3.6 Congestion

The complete binary tree has a fatal drawback: the root switch is a bottleneck. At best, this switch must handle an enormous amount of traffic: every packet traveling from the left side of the network to the right or vice-versa. Passing all these packets through a single switch could take a long time. At worst, if this switch fails, the network is broken into two equal-sized pieces.

The traffic through the root depends on the routing problem. For example, if the routing problem is given by the identity permutation, $\pi(i) ::= i$, 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 P for π, each path P_i beginning at input i must eventually loop all the way up through the root switch and then travel back down to output $(N-1) - i$.

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. 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 "maxi-min" congestion is called the *congestion of the network*.

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. Then, your

network	diameter	switch size	# switches	congestion
complete binary tree	$2\log N + 2$	3×3	$2N - 1$	N

Table 6.1 A summary of the attributes of the complete binary tree.

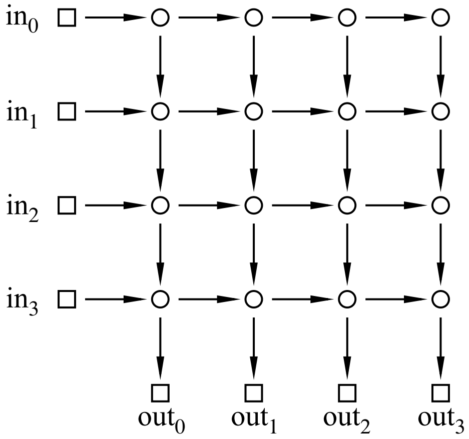


Figure 6.10 A 4 × 4 2-dimensional array.

enemy would choose the root switch and achieve a score of N. In other words, the max congestion of the complete binary tree is N—which is horrible!

We have summarized the results of our analysis of the complete binary tree in Table 6.1. Overall, the complete binary tree does well in every category except the last—congestion, and that is a killer in practice. Next, we will look at a network that solves the congestion problem, but at a very high cost.

6.3.7 The 2-d Array

An illustration of the $N \times N$ 2-d *array* (also known as the *grid* or *crossbar*) is shown in Figure 6.10 for the case when $N = 4$.

The diameter of the 4 × 4 2-d array is 8, which is the number of edges between input 0 and output 3. More generally, the diameter of a 2-d array with N inputs and outputs is $2N$, which is much worse than the diameter of the complete binary tree ($2\log N + 2$). On the other hand, replacing a complete binary tree with a 2-d array almost eliminates congestion.

Theorem 6.3.1. *The congestion of an N-input 2-d 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

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

Table 6.2 Comparing the N-input 2-d array to the N-input complete binary tree.

from input i to column $\pi(i)$ and then goes down to output $\pi(i)$. In this solution, the switch in row i and column j encounters 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. ■

The characteristics of the 2-d array are recorded in Table 6.2. The crucial entry in this table is the number of switches, which is N^2. This is a major defect of the 2-d array; a network with $N = 1000$ inputs 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.

6.3.8 The Butterfly

The Holy Grail of switching networks would combine the best properties of the complete binary tree (low diameter, few switches) and the array (low congestion). The *butterfly* is a widely-used compromise between the two. A butterfly network with $N = 8$ inputs is shown in Figure 6.11.

The structure of the butterfly is certainly more complicated than that of the complete binary or 2-d array. Let's see how it is constructed.

All the terminals and switches in the network are in N rows. In particular, input i is at the left end of row i, and output i is at the right end of row i. Now let's label the rows in *binary* so that the label on row i is the binary number $b_1 b_2 \ldots b_{\log N}$ that represents the integer i.

Between the inputs and outputs, there are $\log(N) + 1$ levels of switches, numbered from 0 to $\log N$. Each level consists of a column of N switches, one per row. Thus, each switch in the network is uniquely identified by a sequence $(b_1, b_2, \ldots, b_{\log N}, l)$, where $b_1 b_2 \ldots b_{\log N}$ is the switch's row in binary and l is the switch's level.

All that remains is to describe how the switches are connected up. The basic

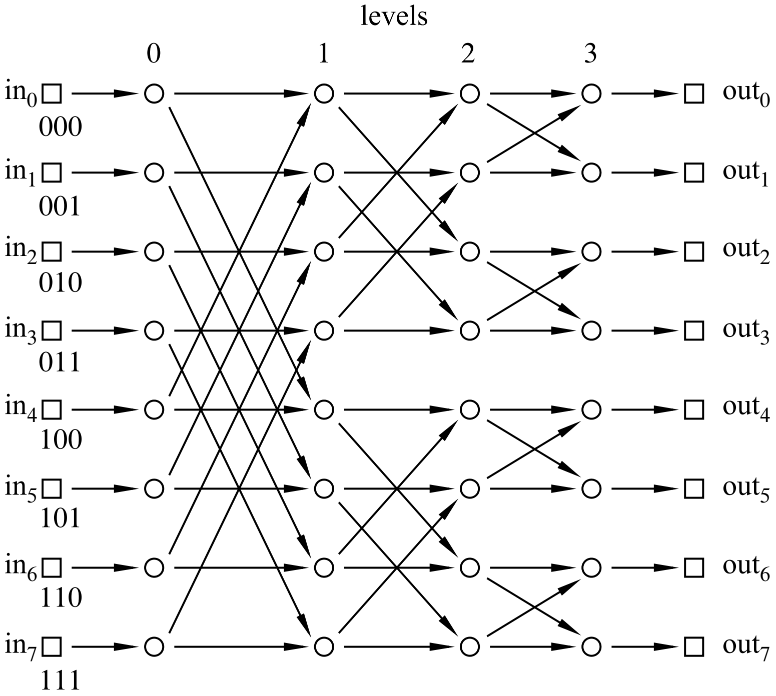


Figure 6.11 An 8-input/output butterfly.

connection pattern is expressed below in a compact notation:

$$\begin{tikzcd}
 & (b_1, b_2, \dots b_{l+1}, \dots b_{\log N}, l+1) \\
(b_1, b_2, \dots b_{l+1}, \dots b_{\log N}, l) \arrow[ur] \arrow[dr] & \\
 & (b_1, b_2, \dots \overline{b_{l+1}}, \dots b_{\log N}, l+1)
\end{tikzcd}$$

This says that there are directed edges from switch $(b_1, b_2, \dots, b_{\log N}, l)$ to two switches in the next level. One edges leads to the switch in the *same* row, and the other edge leads to the switch in the row obtained by *inverting* the $(l+1)$st bit b_{l+1}. For example, referring back to the illustration of the size $N = 8$ butterfly, there is an edge from switch $(0, 0, 0, 0)$ to switch $(0, 0, 0, 1)$, which is in the same row, and to switch $(1, 0, 0, 1)$, which is in the row obtained by inverting bit $l + 1 = 1$.

The butterfly network has a recursive structure; specifically, a butterfly of size $2N$ consists of two butterflies of size N and one additional level of switches. Each switch in the additional level has directed edges to a corresponding switch in each of the smaller butterflies. For example, see Figure 6.12.

Despite the relatively complicated structure of the butterfly, there is a simple way to route packets through its switches. In particular, suppose that we want to send a packet from input $x_1 x_2 \dots x_{\log N}$ to output $y_1 y_2 \dots y_{\log N}$. (Here we are specifying the input and output numbers in binary.) Roughly, the plan is to "correct" the first bit on the first level, correct the second bit on the second level, and so forth. Thus, the sequence of switches visited by the packet is:

$$\begin{aligned}
(x_1, x_2, x_3, \dots, x_{\log N}, 0) &\to (y_1, x_2, x_3, \dots, x_{\log N}, 1) \\
&\to (y_1, y_2, x_3, \dots, x_{\log N}, 2) \\
&\to (y_1, y_2, y_3, \dots, x_{\log N}, 3) \\
&\to \quad \dots \\
&\to (y_1, y_2, y_3, \dots, y_{\log N}, \log N)
\end{aligned}$$

In fact, this is the *only* path from the input to the output!

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. The task of proving this fact has been left to the problem section.[4]

A comparison of the butterfly with the complete binary tree and the 2-d array is provided in Table 6.3. As you can see, the butterfly has lower congestion than the complete binary tree. And it uses fewer switches and has lower diameter than the

[4] The routing problems that result in $\sqrt{N}$ congestion do arise in practice, but for most routing problems, the congestion is much lower (around $\log N$), which is one reason why the butterfly is useful in practice.

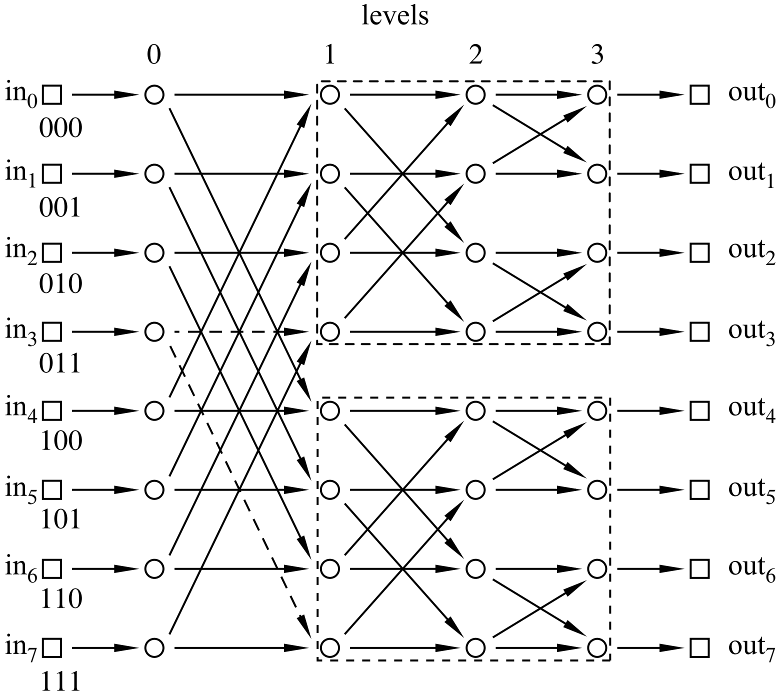


Figure 6.12 An N-input butterfly contains two $N/2$-input butterflies (shown in the dashed boxes). Each switch on the first level is adjacent to a corresponding switch in each of the sub-butterflies. For example, we have used dashed lines to show these edges for the node $(0, 1, 1, 0)$.

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 $\sqrt{N/2}$

Table 6.3 A comparison of the N-input butterfly with the N-input complete binary tree and the N-input 2-d array.

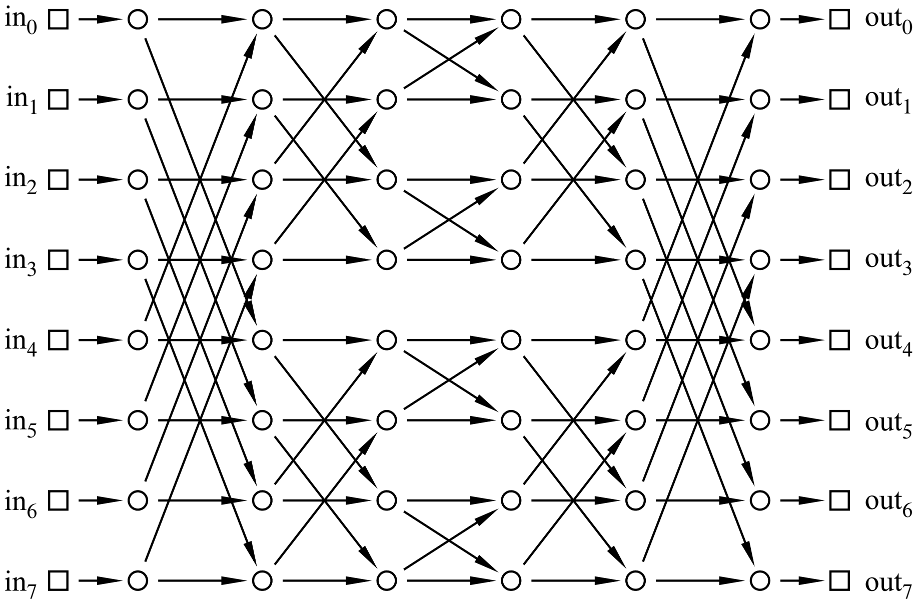


Figure 6.13 The 8-input Beneš network.

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.

6.3.9 Beneš Network

In the 1960's, a researcher at Bell Labs named Václav Beneš had a remarkable idea. He obtained a marvelous communication network with congestion 1 by placing *two* butterflies back-to-back. For example, the 8-input Beneš network is shown in Figure 6.13.

Putting two butterflies back-to-back roughly doubles the number of switches and the diameter of a single butterfly, but it completely eliminates congestion problems! The proof of this fact relies on a clever induction argument that we'll come to in a

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

Table 6.4 A comparison of the N-input Beneš network with the N-input complete binary tree, 2-d array, and butterfly.

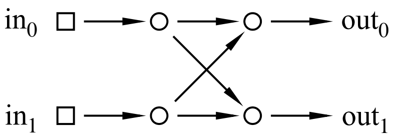


Figure 6.14 The 2-input Beneš network.

moment. Let's first see how the Beneš network stacks up against the other networks we have been studying. As you can see in Table 6.4, the Beneš network has small size and diameter, and completely eliminates congestion. The Holy Grail of routing networks is in hand!

Theorem 6.3.2. *The congestion of the N-input Beneš network is 1 for any N that is a power of 2.*

Proof. We use induction. Let $P(a)$ be the proposition that the congestion of the 2^a-input Beneš network is 1.

Base case ($a = 1$): We must show that the congestion of the 2^1-input Beneš network is 1. The network is shown in Figure 6.14.

There are only two possible permutation routing problems for a 2-input network. If $\pi(0) = 0$ and $\pi(1) = 1$, then we can route both packets along the straight edges. On the other hand, if $\pi(0) = 1$ and $\pi(1) = 0$, then we can route both packets along the diagonal edges. In both cases, a single packet passes through each switch.

Inductive step: We must show that $P(a)$ implies $P(a+1)$ where $a \geq 1$. Thus, we assume that the congestion of a 2^a-input Beneš network is 1 in order to prove that the congestion of a 2^{a+1}-input Beneš network is also 1.

Digression

Time out! Let's work through an example, develop some intuition, and then complete the proof. Notice that inside a Beneš network of size $2N$ lurk two Beneš subnetworks of size N. This follows from our earlier observation that a butterfly

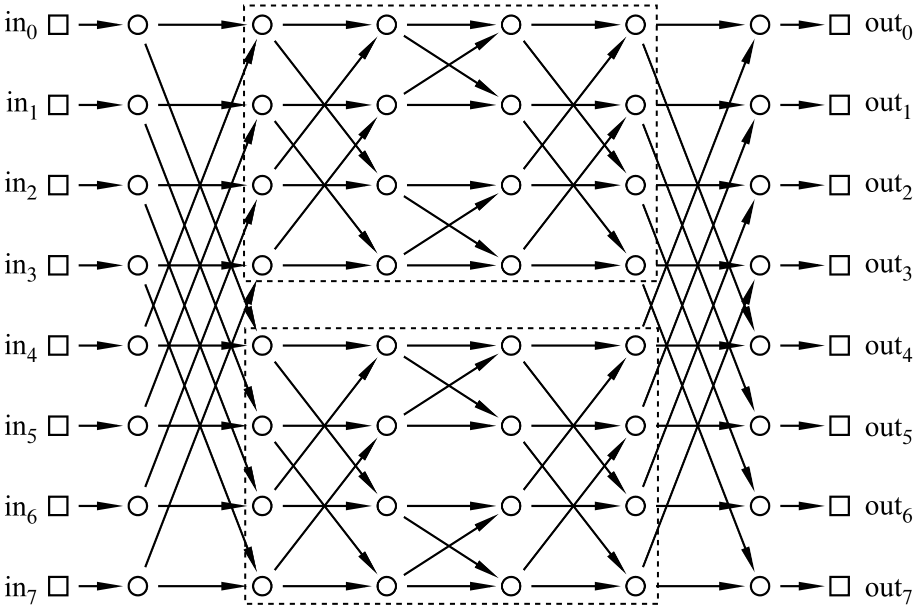


Figure 6.15 A $2N$-input Beneš network contains two N-input Beneš networks—shown here for $N = 4$.

of size $2N$ contains two butterflies of size N. In the Beneš network shown in Figure 6.15 with $N = 8$ inputs and outputs, the two 4-input/output subnetworks are shown in dashed boxes.

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

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

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 the packets at inputs 0 and 4 both 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, the packets at inputs 1 and 5,

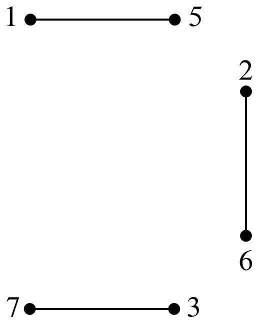


Figure 6.16 The beginnings of a constraint graph for our packet routing problem. Adjacent packets cannot be routed using the same sub-Beneš network.

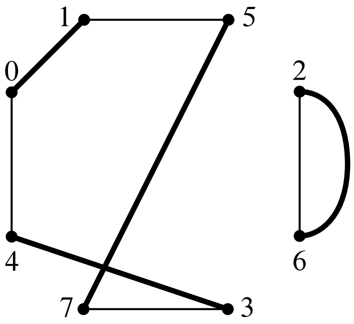


Figure 6.17 The updated constraint graph.

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 (labeled according to their input position). If two packets must pass through different networks, then there is an edge between them. The resulting constraint graph is illustrated in Figure 6.16. 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 since 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 constraint graph with gray edges, as is illustrated in Figure 6.17.

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. Fortunately, this is easy to verify:

Lemma 6.3.3. *If the edges of an undirected graph G can be grouped into two sets such that every vertex is incident to at most 1 edge from each set, 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*. Note that 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. This means that doubled edges form connected components with 2 nodes. Such connected components are easily colored with 2 colors and so we can henceforth ignore them and focus on the remaining nodes and edges, which form a simple graph.

By Theorem 5.6.2, we know that if a simple graph has no odd cycles, then it is 2-colorable. So all we need to do is show that every cycle in G has even length. This is easy since any cycle in G must traverse successive edges that alternate from one set to the other. In particular, a closed walk must traverse a path of alternating edges that begins and ends with edges from different sets. This means that the cycle has to be of even length. ■

For example, a 2-coloring of the constraint graph in Figure 6.17 is shown in Figure 6.18. 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. With this insight in hand, the digression is over and we can now complete the proof of Theorem 6.3.2.

Proof of Theorem 6.3.2 (continued). Let π be an arbitrary permutation of $0, 1, \ldots, N-1$. Let G be the graph whose vertices are packet numbers $0, 1, \ldots, N-1$ and whose edges come from the union of these two sets:

$$E_1 ::= \{\, \{u, v\} \mid |u - v| = N/2 \,\}, \text{ and}$$
$$E_2 ::= \{\, \{u, w\} \mid |\pi(u) - \pi(w)| = N/2 \,\}.$$

Now any vertex, u, is incident to at most two edges: a unique edge $\{u, v\} \in E_1$ and a unique edge $\{u, w\} \in E_2$. So according to Lemma 6.3.3, there is a 2-coloring for

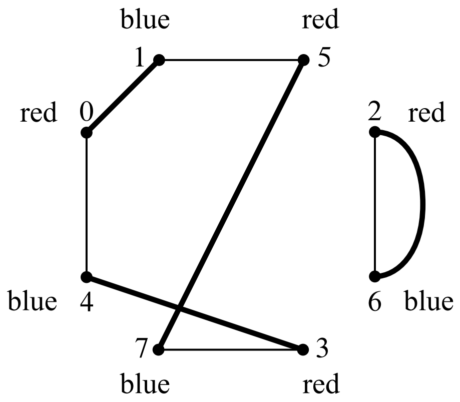


Figure 6.18 A 2-coloring of the constraint graph in Figure 6.17.

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 $P(n)$. ■

7 Relations and Partial Orders

A relation is a mathematical tool for describing associations between elements of sets. Relations are widely used in computer science, especially in databases and scheduling applications. A relation can be defined across many items in many sets, but in this text, we will focus on *binary* relations, which represent an association between two items in one or two sets.

7.1 Binary Relations

7.1.1 Definitions and Examples

Definition 7.1.1. Given sets A and B, a *binary relation* $R : A \to B$ *from*[1] A to B is a subset of $A \times B$. The sets A and B are called the *domain* and *codomain* of R, respectively. We commonly use the notation aRb or $a \sim_R b$ to denote that $(a, b) \in R$.

A relation is similar to a function. In fact, every function $f : A \to B$ is a relation. In general, the difference between a function and a relation is that a relation might associate multiple elements of$B$ with a single element of$A$, whereas a function can only associate at most one element of B (namely, $f(a)$) with each element $a \in A$.

We have already encountered examples of relations in earlier chapters. For example, in Section 5.2, we talked about a relation between the set of men and the set of women where mRw if man m likes woman w. In Section 5.3, we talked about a relation on the set of MIT courses where $c_1 R c_2$ if the exams for c_1 and c_2 cannot be given at the same time. In Section 6.3, we talked about a relation on the set of switches in a network where $s_1 R s_2$ if s_1 and s_2 are directly connected by a wire that can send a packet from s_1 to s_2. We did not use the formal definition of a relation in any of these cases, but they are all examples of relations.

As another example, we can define an "in-charge-of" relation T from the set of MIT faculty F to the set of subjects in the 2010 MIT course catalog. This relation contains pairs of the form

$$(\langle\text{instructor-name}\rangle, \langle\text{subject-num}\rangle)$$

[1] We also say that the relationship is *between* A and B, or *on* A if $B = A$.

(Meyer,	6.042),
(Meyer,	18.062),
(Meyer,	6.844),
(Leighton,	6.042),
(Leighton,	18.062),
(Freeman,	6.011),
(Freeman,	6.881)
(Freeman,	6.882)
(Freeman,	6.UAT)
(Eng,	6.UAT)
(Guttag,	6.00)

Figure 7.1 Some items in the “in-charge-of” relation T between faculty and subject numbers.

where the faculty member named $\langle$instructor-name$\rangle$ is in charge of the subject with number $\langle$subject-num$\rangle$. So T contains pairs like those shown in Figure 7.1.

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 jointly 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 jointly 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-of subjects are Fall term only.

7.1.2 Representation as a Bipartite Graph

Every relation $R : A \to B$ can be easily represented as a bipartite graph $G = (V, E)$ by creating a “left” node for each element of A and a “right” node for each element of B. We then create an edge between a left node u and a right node v whenever aRb. Similarly, every bipartite graph (and every partition of the nodes into a “left” and “right” set for which no edge connects a pair of left nodes or a pair of right nodes) determines a relation between the nodes on the left and the nodes on the right.

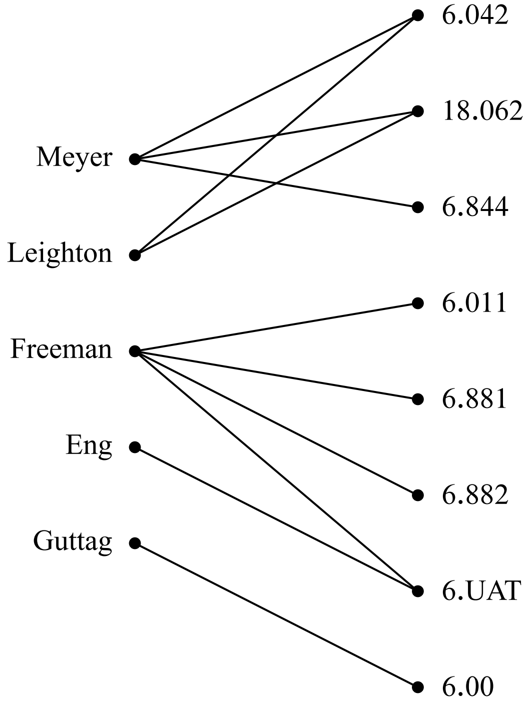


Figure 7.2 Part of the bipartite graph for the "in charge of" relation T from Figure 7.1.

For example, we have shown part of the bipartite graph for the in-charge-of relation from Figure 7.1 in Figure 7.2. In this case, there is an edge between ⟨instructor-name⟩ and ⟨subject-number⟩ if ⟨instructor-name⟩ is in charge of ⟨subject-number⟩.

A relation $R : A \to B$ between finite sets can also be represented as a matrix $A = \{a_{ij}\}$ where

$$a_{ij} = \begin{cases} 1 & \text{if the } i\text{th element of } A \text{ is related to the } j\text{th element of } B \\ 0 & \text{otherwise} \end{cases}$$

for $1 \leq i \leq |A|$ and $1 \leq j \leq |B|$. For example, the matrix for the relation in Figure 7.2 (but restricted to the five faculty and eight subject numbers shown in Figure 7.2, ordering them as they appear top-to-bottom in Figure 7.2) is shown in Figure 7.3.

7.1.3 Relational Images

The idea of the image of a set under a function extends directly to relations.

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Figure 7.3 The matrix for the "in charge of" relation T restricted to the five faculty and eight subject numbers shown in Figure 7.2. The $(3, 4)$ entry of this matrix is 1 since the third professor (Freeman) is in charge of the fourth subject number (6.011).

Definition 7.1.2. The *image* of a set Y under a relation $R : A \to B$, written $R(Y)$, is the set of elements that are related to some element in Y, namely,

$$R(Y) ::= \{ b \in B \mid yRb \text{ for some } y \in Y \}.$$

The image of the domain, $R(A)$, is called the *range* of R.

For example, to find the subject numbers that Meyer is in charge of, we can look for all the pairs of the form

$$(\text{Meyer}, \langle\text{subject-number}\rangle)$$

in the graph of the teaching relation T, and then just list the right-hand sides of these pairs. These right-hand sides are exactly the image $T(\text{Meyer})$, which happens to be $\{6.042, 18.062, 6.844\}$. Similarly, since the domain F is the set of all in-charge faculty, $T(F)$, the range of T, is exactly the set of *all* subjects being taught.

7.1.4 Inverse Relations and Images

Definition 7.1.3. The *inverse* R^{-1} of a relation $R : A \to B$ is the relation from B to A defined by the rule

$$bR^{-1}a \text{ if and only if } aRb.$$

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-of example above, we can find the faculty in charge of 6.UAT by taking the pairs of the form

$$(\langle\text{instructor-name}\rangle, \text{6.UAT})$$

for 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.

7.1.5 Combining Relations

There are at least two natural ways to combine relations to form new relations. For example, given relations $R : B \to C$ and $S : A \to B$, the *composition* of R with S is the relation $(R \circ S) : A \to C$ defined by the rule

$$a(R \circ S)c \ \text{ IFF } \ \exists b \in B.\ (bRc) \text{ AND } (aSb)$$

where $a \in A$ and $c \in C$.

As a special case, the composition of two functions $f : B \to C$ and $g : A \to B$ is the function $f \circ g : A \to C$ defined by

$$(f \circ g)(a) = f(g(a))$$

for all $a \in A$. For example, if $A = B = C = \mathbb{R}$, $g(x) = x + 1$ and $f(x) = x^2$, then

$$\begin{aligned} (f \circ g)(x) &= (x+1)^2 \\ &= x^2 + 2x + 1. \end{aligned}$$

One can also define the *product* of two relations $R_1 : A_1 \to B_1$ and $R_2 : A_2 \to B_2$ to be the relation $S = R_1 \times R_2$ where

$$S : A_1 \times A_2 \to B_1 \times B_2$$

and

$$(a_1, a_2)S(b_1, b_2) \quad \text{iff} \quad a_1 R_1 b_1 \text{ and } a_2 R_2 b_2.$$

7.2 Relations and Cardinality

7.2.1 Surjective and Injective Relations

There are some properties of relations that will be useful when we take up the topic of counting in Part III because they imply certain relations between the *sizes* of domains and codomains. In particular, we say that a binary relation $R : A \to B$ is

- *surjective* if every element of B is assigned to at least one element of A. More concisely, R is surjective iff $R(A) = B$ (that is, if the range of R is the codomain of R),

- *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 *at most once*, and
- *bijective* if R is total, surjective, injective, and a function[2].

We can illustrate these properties of a relation $R : A \to B$ in terms of the corresponding bipartite graph G for the relation, where nodes on the left side of G correspond to elements of A and nodes on the right side of G correspond to elements of B. For example:

- "R is a function" means that every node on the left is incident to *at most* one edge.
- "R is total" means that *every* node on the left is incident to *at least* one edge. So if R is a function, being total means that every node on the left is incident to exactly one edge.
- "R is surjective" means that *every* node on the right is incident to *at least* one edge.
- "R is injective" means that *every* node on the right is incident to *at most* one edge.
- "R is bijective" means that every node on both sides is incident to *precisely* one edge (that is, there is a perfect matching between A and B).

For example, consider the relations R_1 and R_2 shown in Figure 7.4. R_1 is a total surjective function (every node in the left column is incident to exactly one edge, and every node in the right column is incident to at least one edge), but not injective (node 3 is incident to 2 edges). R_2 is a total injective function (every node in the left column is incident to exactly one edge, and every node in the right column is incident to at most one edge), but not surjective (node 4 is not incident to any edges).

Notice that we need to know what the domain is to determine whether a relation is total, and we need to know the codomain to determine whether it's surjective. For example, 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 that includes 0. It is bijective if its domain and codomain are both $\mathbb{R}^+$, but neither injective nor surjective it is domain and codomain are both $\mathbb{R}$.

[2] 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.

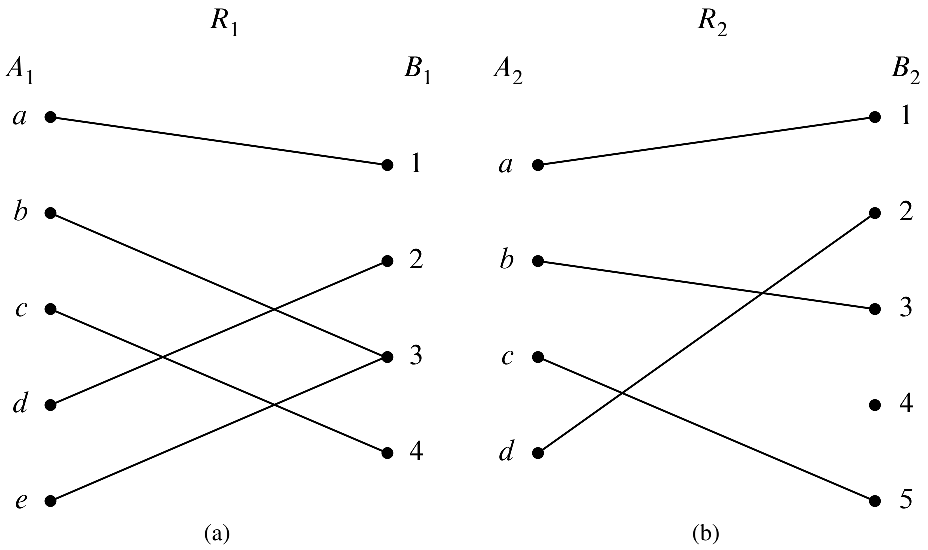


Figure 7.4 Relation $R_1 : A_1 \to B_1$ is shown in (a) and relation $R_2 : A_2 \to B_2$ is shown in (b).

7.2.2 Cardinality

The relational properties in Section 7.2.1 are useful in figuring out the relative sizes of domains and codomains.

If A is a finite set, we use $|A|$ to denote the number of elements in A. This is called the *cardinality* of A. In general, a finite set may have no elements (the empty set), or one element, or two elements, ..., or any nonnegative integer number of elements, so for any finite set, $|A| \in \mathbb{N}$.

Now suppose $R : A \to B$ is a function. Then every edge in the bipartite graph $G = (V, E)$ for R is incident to exactly one element of A, so the number of edges is at most the number of elements of A. That is, if R is a function, then

$$|E| \leq |A|.$$

Similarly, if R is surjective, then every element of B is incident to an edge, so there must be at least as many edges in the graph as the size of B. That is

$$|E| \geq |B|.$$

Combining these inequalities implies that $R : A \to B$ is a surjective function, then $|A| \geq |B|$. This fact and two similar rules relating domain and codomain size to relational properties are captured in the following theorem.

Theorem 7.2.1 (Mapping Rules)**.** *Let A and B be finite sets.*

1. *If there is a surjection from A to B, then $|A| \geq |B|$.*
2. *If there is an injection from A to B, then $|A| \leq |B|$.*
3. *If there is a bijection between A and B, then $|A| = |B|$.*

Mapping rule 2 can be explained by the same kind of reasoning we used for rule 1. Rule 3 is an immediate consequence of the first two mapping rules.

We will see many examples where Theorem 7.2.1 is used to determine the cardinality of a finite set. Later, in Chapter 13, we will consider the case when the sets are infinite and we'll use surjective and injective relations to prove that some infinite sets are "bigger" than other infinite sets.

7.3 Relations on One Set

For the rest of this chapter, we are going to focus on relationships between elements of a single set; that is, relations from a set A to a set B where $A = B$. Thus, a relation on a set A is a subset $R \subseteq A \times A$. Here are some examples:

- Let A be a set of people and the relation R describe who likes whom: that is, $(x, y) \in R$ if and only if x likes y.
- Let A be a set of cities. Then we can define a relation R such that xRy if and only if there is a nonstop flight from city x to city y.
- Let $A = \mathbb{Z}$ and let xRy hold if and only if $x \equiv y \pmod 5$.
- Let $A = \mathbb{N}$ and let xRy if and only if $x \mid y$.
- Let $A = \mathbb{N}$ and let xRy if and only if $x \leq y$.

The last examples clarify the reason for using xRy or $x \sim_R y$ to indicate that the relation R holds between x and y: many common relations ($<$, $\leq$, $=$, $|$, $\equiv$) are expressed with the relational symbol in the middle.

7.3.1 Representation as a Digraph

Every relation on a single set A can be modeled as a directed graph (albeit one that may contain loops). For example, the graph in Figure 7.5 describes the "likes" relation for a particular set of 3 people.

In this case, we see that:

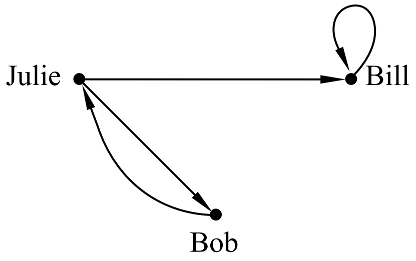


Figure 7.5 The directed graph for the "likes" relation on the set {Bill, Bob, Julie}.

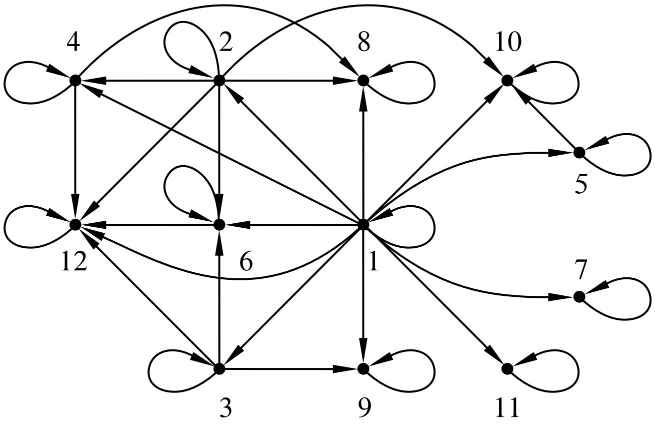


Figure 7.6 The digraph for divisibility on $\{1, 2, \ldots, 12\}$.

- Julie likes Bill and Bob, but not herself.
- Bill likes only himself.
- Bob likes Julie, but not Bill nor himself.

Everything about the relationship is conveyed by the directed graph and nothing more. This is no coincidence; a set A together with a relation R is precisely the same thing as directed graph $G = (V, E)$ with vertex set $V = A$ and edge set $E = R$ (where E may have loops).

As another example, we have illustrated the directed graph for the divisibility relationship on the set $\{1, 2, \ldots, 12\}$ in Figure 7.6. In this graph, every node has a loop (since every positive number divides itself) and the composite numbers are the nodes with indegree more than 1 (not counting the loop).

Relations on a single set can also be represented as a 0, 1-matrix. In this case, the matrix is identical to the adjacency matrix for the corresponding digraph. For example, the matrix for the relation shown in Figure 7.5 is simply

$$\begin{pmatrix} 0 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

where $v_1 = \text{Julie}$, $v_2 = \text{Bill}$, and $v_3 = \text{Bob}$.

7.3.2 Symmetry, Transitivity, and Other Special Properties

Many relations on a single set that arise in practice possess one or more noteworthy properties. These properties are summarized in the box on the following page. In each case, we provide the formal of the definition of the property, explain what the property looks like in a digraph G for the relation, and give an example of what the property means for the "likes" relation.

For example, the congruence relation modulo 5 on $\mathbb{Z}$ is reflexive symmetric, and transitive, but not irreflexive, antisymmetric, or asymmetric. The same is true for the "connected" relation $R : V \to V$ on an undirected graph $G = (V, E)$ where uRv if u and v are in the same connected component of graph G. In fact, relations that have these three properties are so common that we give them a special name: *equivalence relations*. We will discuss them in greater detail in just a moment.

As another example, the "divides" relation on $\mathbb{Z}^+$ is reflexive, antisymmetric, and transitive, but not irreflexive, symmetric, or asymmetric. The same is true for the "$\leq$" relation on $\mathbb{R}$. Relations that have these three properties are also very common and they fall into a special case of relations called a *partial order*. We will discuss partial orders at length in Sections 7.5–7.9.

As a final example, consider the "likes" relation on the set $\{\text{Julie}, \text{Bill}, \text{Bob}\}$ illustrated in Figure 7.5. This relation has *none* of the six properties described in the box.

7.4 Equivalence Relations

A relation is an *equivalence relation* if it is reflexive, symmetric, and transitive. Congruence modulo n is an excellent example of an equivalence relation:

- It is reflexive because $x \equiv x \pmod{n}$.
- It is symmetric because $x \equiv y \pmod{n}$ implies $y \equiv x \pmod{n}$.
- It is transitive because $x \equiv y \pmod{n}$ and $y \equiv z \pmod{n}$ imply that $x \equiv z \pmod{n}$.

Properties of a Relation $R : A \to A$

Reflexivity R is *reflexive* if

$\forall x \in A.\ xRX.$

"Everyone likes themselves."

Every node in G has a loop.

Irreflexivity R is *irreflexive* if

$\neg\exists x \in A.\ xRx.$

"No one likes themselves."

There are no loops in G.

Symmetry R is *symmetric* if

$\forall x, y \in A.\ xRy$ IMPLIES yRx.

"If x likes y, then y likes x."

If there is an edge from x to y in G, then there is an edge from y to x in G as well.

Antisymmetry R is *antisymmetric* if

$\forall x, y \in A\ (xRy$ AND $yRx)$ IMPLIES $x = y$.

"No pair of distinct people like each other."

There is at most one directed edge between any pair of distinct nodes.

Asymmetry R is *asymmetric* if

$\neg\exists x, y \in A.\ xRy$ AND yRx.

"No one likes themselves and no pair of people like each other."

There are no loops and there is at most one directed edge between any pair of nodes.

Transitivity R is *transitive* if

$\forall x, y, z \in A.\ (xRy$ AND $yRz)$ IMPLIES xRz.

"If x likes y and y likes z, then x likes z too."

For any walk $v_0, v_1, \ldots, v_k$ in G where $k \geq 2$, $v_0 \to v_k$ is in G (and, hence, $v_i \to v_j$ is also in G for all $i < j$.

There is an even more well-known example of an equivalence relation: equality itself. Thus, an equivalence relation is a relation that shares some key properties with "$=$".

7.4.1 Partitions

There is another way to think about equivalence relations, but we'll need a couple of definitions to understand this alternative perspective.

Definition 7.4.1. Given an equivalence relation $R : A \to A$, the *equivalence class* of an element $x \in A$ is the set of all elements of A related to x by R. The equivalence class of x is denoted $[x]$. Thus, in symbols:

$$[x] = \{\, y \mid xRy \,\}.$$

For example, suppose that $A = \mathbb{Z}$ and xRy means that $x \equiv y \pmod{5}$. Then

$$[7] = \{\dots, -3, 2, 7, 12, 22, \dots\}.$$

Notice that 7, 12, 17, etc., all have the same equivalence class; that is, $[7] = [12] = [17] = \cdots$.

Definition 7.4.2. A *partition* of a finite set A is a collection of disjoint, nonempty subsets A_1, A_2, ..., A_n whose union is all of A. The subsets are usually called the *blocks* of the partition.[3] For example, one possible partition of $A = \{a, b, c, d, e\}$ is

$$A_1 = \{a, c\} \qquad A_2 = \{b, e\} \qquad A_3 = \{d\}.$$

Here's the connection between all this stuff: there is an exact correspondence between *equivalence relations on A* and *partitions of A*. We can state this as a theorem:

Theorem 7.4.3. *The equivalence classes of an equivalence relation on a set A form a partition of A.*

We won't prove this theorem (too dull even for us!), but let's look at an example.

[3] We think they should be called the *parts* of the partition. Don't you think that makes a lot more sense?

The congruent-mod-5 relation partitions the integers into five equivalence classes:

$$\begin{gathered}
\{\ldots,-5,0,5,10,15,20,\ldots\} \\
\{\ldots,-4,1,6,11,16,21,\ldots\} \\
\{\ldots,-3,2,7,12,17,22,\ldots\} \\
\{\ldots,-2,3,8,13,18,23,\ldots\} \\
\{\ldots,-1,4,9,14,19,24,\ldots\}
\end{gathered}$$

In these terms, $x \equiv y \pmod 5$ is equivalent to the assertion that x and y are both in the same block of this partition. For example, $6 \equiv 16 \pmod 5$, because they're both in the second block, but $2 \not\equiv 9 \pmod 5$ because 2 is in the third block while 9 is in the last block.

In social terms, if "likes" were an equivalence relation, then everyone would be partitioned into cliques of friends who all like each other and no one else.

7.5 Partial Orders

7.5.1 Strong and Weak Partial Orders

Definition 7.5.1. A relation R on a set A is a *weak partial order* if it is transitive, antisymmetric, and reflexive. The relation is said to be a *strong partial order* if it is transitive, antisymmetric, and irreflexive.[4]

Some authors defined partial orders to be what we call weak partial orders, but we'll use the phrase *partial order* to mean either a weak or a strong partial order. The difference between a weak partial order and a strong one has to do with the reflexivity property: in a weak partial order, *every* element is related to itself, but in a strong partial order, *no* element is related to itself. Otherwise, they are the same in that they are both transitive and antisymmetric.

Examples of weak partial orders include "$\leq$" on $\mathbb{R}$, "$\subseteq$" on the set of subsets of (say) $\mathbb{Z}$, and the "divides" relation on $\mathbb{N}^+$. Examples of strict partial orders include "$<$" on $\mathbb{R}$, and "$\subset$" on the set of subsets of $\mathbb{Z}$.[5]

[4] Equivalently, the relation is transitive and asymmetric, but stating it this way might have obscured the irreflexivity property.

[5] If you are not feeling comfortable with all the definitions that we've been throwing at you, it's probably a good idea to verify that each of these relations are indeed partial orders by checking that they have the transitivity and antisymmetry properties.

We often denote a weak partial order with a symbol such as $\preceq$ or $\sqsubseteq$ instead of a letter such as R. This makes sense from one perspective since the symbols call to mind $\leq$ and $\subseteq$, which define common partial orders. On the other hand, a partial order is really a set of related pairs of items, and so a letter like R would be more normal.

Likewise, we will often use a symbol like $\prec$ or $\sqsubset$ to denote a strong partial order.

7.5.2 Total Orders

A partial order is "partial" because there can be two elements with no relation between them. For example, in the "divides" partial order on $\{1, 2, \ldots, 12\}$, there is no relation between 3 and 5 (since neither divides the other).

In general, we say that two elements a and b are *incomparable* if neither $a \preceq b$ nor $b \preceq a$. Otherwise, if $a \preceq b$ or $b \preceq a$, then we say that a and b are *comparable*.

Definition 7.5.2. A *total order* is a partial order in which every pair of distinct elements is comparable.

For example, the "$\leq$" partial order on $\mathbb{R}$ is a total order because for any pair of real numbers x and y, either $x \leq y$ or $y \leq x$. The "divides" partial order on $\{1, 2, \ldots, 12\}$ is not a total order because $3 \nmid 5$ and $5 \nmid 3$.

7.6 Posets and DAGs

7.6.1 Partially Ordered Sets

Definition 7.6.1. Given a partial order $\preceq$ on a set A, the pair $(A, \preceq)$ is called a *partially ordered set* or *poset*.

In terms of graph theory, a poset is simply the directed graph $G = (A, \preceq)$ with vertex set A and edge set $\preceq$. For example, Figure 7.6 shows the graph form of the poset for the "divides" relation on $\{1, 2, \ldots, 12\}$. We have shown the graph form of the poset for the "$<$"-relation on $\{1, 2, 3, 4\}$ in Figure 7.7.

7.6.2 Posets Are Acyclic

Did you notice anything that is common to Figures 7.6 and 7.7? Of course, they both exhibit the transitivity and antisymmetry properties. And, except for the loops in Figure 7.6, they both do not contain any cycles. This is not a coincidence. In fact, the combination of the transitivity and asymmetry properties imply that the digraph

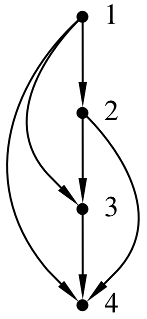


Figure 7.7 Representing the poset for the "$<$"-relation on $\{1, 2, 3, 4\}$ as a digraph.

for any poset is an acyclic graph (that is, a DAG), at least if you don't count loops as cycles. We prove this fact in the following theorem.

Theorem 7.6.2. *A poset has no directed cycles other than self-loops.*

Proof. We use proof by contradiction. Let $(A, \preceq)$ be a poset. Suppose that there exist $n \geq 2$ distinct elements $a_1, a_2, \dots, a_n$ such that

$$a_1 \preceq a_2 \preceq a_3 \preceq \cdots \preceq a_{n-1} \preceq a_n \preceq a_1.$$

Since $a_1 \preceq a_2$ and $a_2 \preceq a_3$, transitivity implies $a_1 \preceq a_3$. Another application of transitivity shows that $a_1 \preceq a_4$ and a routine induction argument establishes that $a_1 \preceq a_n$. Since we know that $a_n \preceq a_1$, antisymmetry implies $a_1 = a_n$, contradicting the supposition that $a_1, \dots, a_n$ are distinct and $n \geq 2$. Thus, there is no such directed cycle. ■

Thus, deleting the self-loops from a poset leaves a directed graph without cycles, which makes it a *directed acyclic graph* or *DAG*.

7.6.3 Transitive Closure

Theorem 7.6.2 tells us that every poset corresponds to a DAG. Is the reverse true? That is, does every DAG correspond to a poset? The answer is "Yes," but we need to modify the DAG to make sure that it satisfies the transitivity property. For example, consider the DAG shown in Figure 7.8. As any DAG must, this graph satisfies the antisymmetry property[6] but it does not satisfy the transitivity property because $v_1 \to v_2$ and $v_2 \to v_3$ are in the graph but $v_1 \to v_3$ is not in the graph.

[6] If $u \to v$ and $v \to u$ are in a digraph G, then G would have a cycle of length 2 and it could not be a DAG.

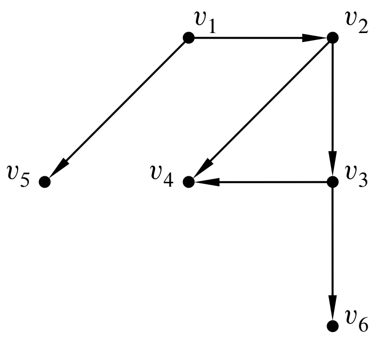


Figure 7.8 A 6-node digraph that does not satisfy the transitivity property.

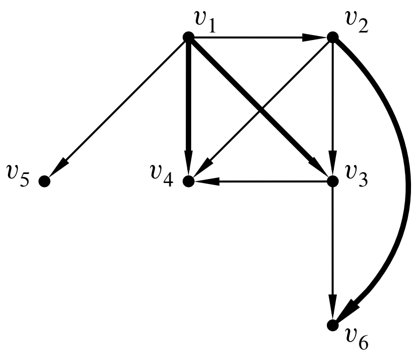


Figure 7.9 The transitive closure for the digraph in Figure 7.8. The edges that were added to form the transitive closure are shown in bold.

Definition 7.6.3. Given a digraph $G = (V, E)$, the *transitive closure* of G is the digraph $G^+ = (V, E^+)$ where

$$E^+ = \{\, u \to v \mid \text{there is a directed path of positive length from } u \text{ to } v \text{ in } G \,\}.$$

Similarly, if R is the relation corresponding to G, the *transitive closure* of R (denoted R^+) is the relation corresponding to G^+.

For example, the transitive closure for the graph in Figure 7.8 is shown in Figure 7.9.

If G is a DAG, then the transitive closure of G is a strong partial order. The proof of this fact is left as an exercise in the problem section.

7.6.4 The Hasse Diagram

One problem with viewing a poset as a digraph is that there tend to be lots of edges due to the transitivity property. Fortunately, we do not necessarily have to draw

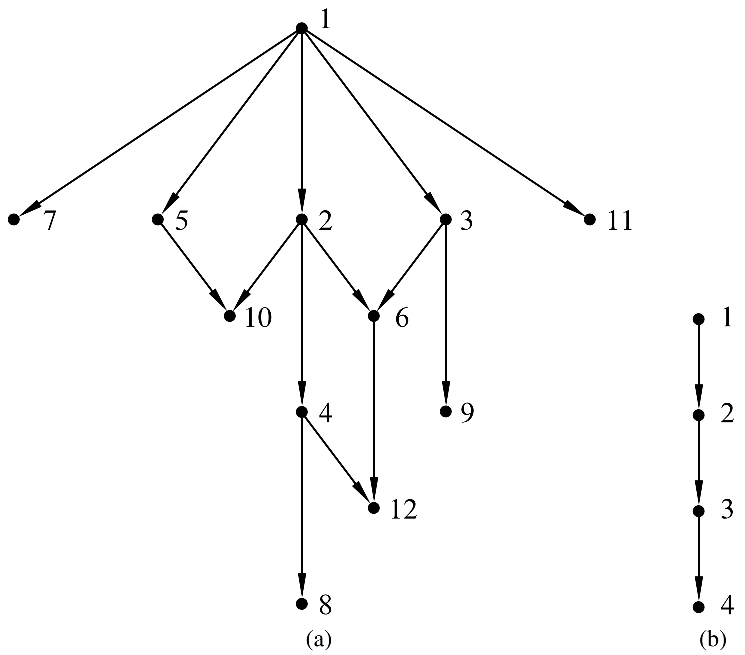


Figure 7.10 The Hasse diagrams for the posets in Figure 7.6 and 7.7.

all the edges if we know that the digraph corresponds to a poset. For example, we could choose not to draw any edge which would be implied by the transitivity property, knowing that it is really there by implication. In general, a *Hasse diagram* for a poset $(A, \preceq)$ is a digraph with vertex set A and edge set $\preceq$ minus all self-loops and edges implied by transitivity. For example, the Hasse diagrams of the posets shown in Figures 7.6 and 7.7 are shown in Figure 7.10.

7.7 Topological Sort

A total order that is consistent with a partial order is called a topological sort. More precisely,

Definition 7.7.1. A *topological sort* of a poset $(A, \preceq)$ is a total order $(A, \preceq_T)$ such that

$$x \preceq y \text{ IMPLIES } x \preceq_T y.$$

For example, consider the poset that describes how a guy might get dressed for a formal occasion. The Hasse diagram for such a poset is shown in Figure 7.11.

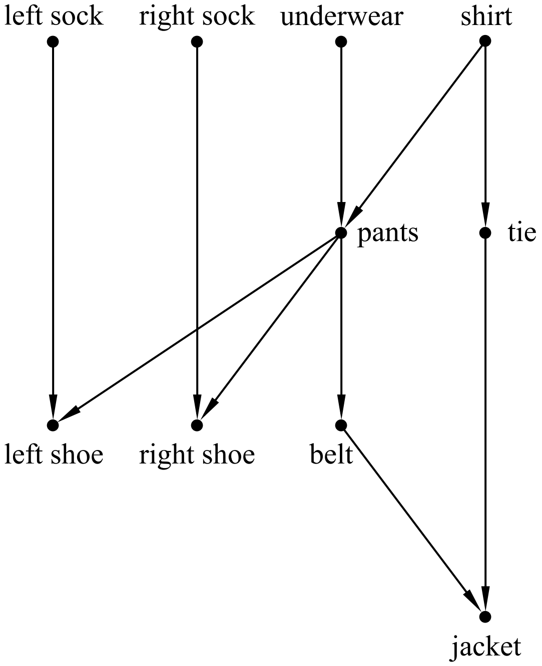


Figure 7.11 The Hasse diagram for a poset that describes which items much precede others when getting dressed.

In this poset, the *set* is all the garments and the *partial order* specifies which items much precede others when getting dressed.

There are several total orders that are consistent with the partial order shown in Figure 7.11. We have shown two of them in list form in Figure 7.12. Each such list is a topological sort for the partial order in Figure 7.11. In what follows, we will prove that every *finite* poset has a topological sort. You can think of this as a mathematical proof that you *can* get dressed in the morning (and then show up for math lecture).

Theorem 7.7.2. *Every finite poset has a topological sort.*

We'll prove the theorem constructively. The basic idea is to pull the "smallest" element a out of the poset, find a topological sort of the remainder recursively, and then add a back into the topological sort as an element smaller than all the others.

The first hurdle is that "smallest" is not such a simple concept in a set that is only partially ordered. In a poset $(A, \preceq)$, an element $x \in A$ is *minimal* if there is no other element $y \in A$ such that $y \preceq x$. For example, there are *four* minimal elements in the getting-dressed poset: left sock, right sock, underwear, and shirt. (It may seem

underwear	left sock
pants	shirt
belt	tie
shirt	underwear
tie	right sock
jacket	pants
left sock	right shoe
right sock	belt
left shoe	jacket
right shoe	left shoe
(a)	(b)

Figure 7.12 Two possible topological sorts of the poset shown in Figure 7.11. In each case, the elements are listed so that $x \preceq y$ iff x is above y in the list.

odd that the minimal elements are at the top of the Hasse diagram rather than the bottom. Some people adopt the opposite convention. If you’re not sure whether minimal elements are on the top or bottom in a particular context, ask.) Similarly, an element $x \in A$ is *maximal* if there is no other element $y \in A$ such that $x \preceq y$.

Proving that every poset *has* a minimal element is extremely difficult, because it is not true. For example, the poset $(\mathbb{Z}, \leq)$ has no minimal element. However, there is at least one minimal element in every *finite* poset.

Lemma 7.7.3. *Every finite poset has a minimal element.*

Proof. Let $(A, \preceq)$ be an arbitrary poset. Let $a_1, a_2, \ldots, a_n$ be a maximum-length sequence of distinct elements in A such that

$$a_1 \preceq a_2 \preceq \cdots \preceq a_n.$$

The existence of such a maximum-length sequence follows from the Well Ordering Principle and the fact that A is finite. Now $a_0 \preceq a_1$ cannot hold for any element $a_0 \in A$ not in the chain, since the chain already has maximum length. And $a_i \preceq a_1$ cannot hold for any $i \geq 2$, since that would imply a cycle

$$a_i \preceq a_1 \preceq a_2 \preceq \cdots \preceq a_i$$

and no cycles exist in a poset by Theorem 7.6.2. Therefore a_1 is a minimal element. ■

Now we’re ready to prove Theorem 7.7.2, which says that every finite poset has a topological sort. The proof is rather intricate; understanding the argument requires a clear grasp of all the mathematical machinery related to posets and relations!

Proof of Theorem 7.7.2. We use induction. Let $P(n)$ be the proposition that every n-element poset has a topological sort.

Base case: Every 1-element poset is already a total order and thus is its own topological sort. So $P(1)$ is true.

Inductive step: Now we assume $P(n)$ in order to prove $P(n+1)$ where $n \geq 1$. Let $(A, \preceq)$ be an $(n+1)$-element poset. By Lemma 7.7.3, there exists a minimal element in $a \in A$. Remove a and all pairs in $\preceq$ involving a to obtain an n-element poset $(A', \preceq')$. This has a topological sort $(A', \preceq'_T)$ by the assumption $P(n)$. Now we construct a total order $(A, \preceq_T)$ by adding a back as an element smaller than all the others. Formally, let

$$\preceq_T = \preceq'_T \cup \{\, (a, z) \mid z \in A \,\}.$$

All that remains is to check that this total order is consistent with the original partial order $(A, \preceq)$; that is, we must show that

$$x \preceq y \text{ IMPLIES } x \preceq_T y.$$

We assume that the left side is true and show that the right side follows. There are two cases.

Case 1 If $x = a$, then $a \preceq_T y$ holds, because $a \preceq_T z$ for all $z \in A$.

Case 2 if $x \neq a$, then y can not equal a either, since a is a minimal element in the partial order $\preceq$. Thus, both x and y are in A' and so $x \preceq' y$. This means $x \preceq'_T y$, since $\preceq'_T$ is a topological sort of the partial order $\preceq'$. And this implies $x \preceq_T y$ since $\preceq_T$ contains $\preceq'_T$.

Thus, $(A, \preceq_T)$ is a topological sort of $(A, \preceq)$. This shows that $P(n)$ implies $P(n+1)$ for all $n \geq 1$. Therefore $P(n)$ is true for all $n \geq 1$ by the principle of induction, which proves the theorem. ■

7.8 Parallel Task Scheduling

When items of a poset are tasks that need to be done and the partial order is a precedence constraint, topological sorting provides us with a way to execute the tasks sequentially without violating the precedence constraints.

But what if we have the ability to execute more than one task at the same time? For example, suppose that the tasks are programs, the partial order indicates data

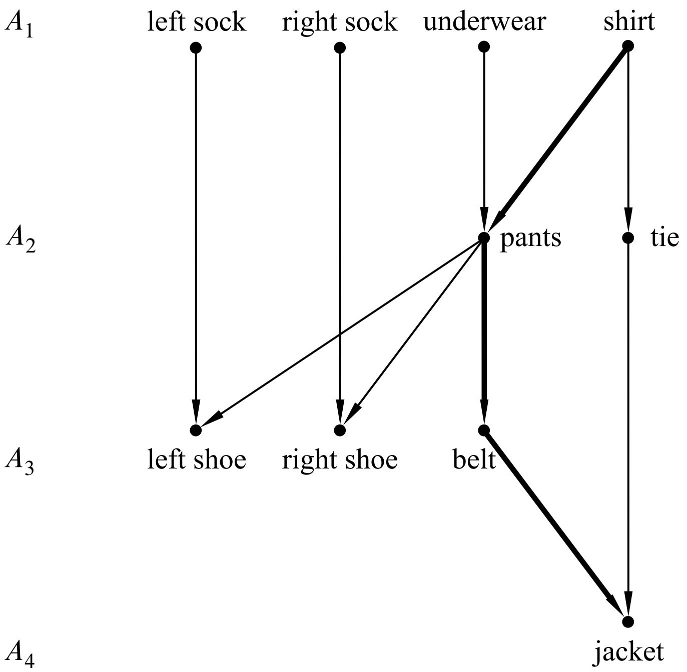


Figure 7.13 A parallel schedule for the tasks-in-getting-dressed poset in Figure 7.11. The tasks in A_i can be performed in step i for $1 \leq i \leq 4$. A chain of length 4 (the critical path in this example) is shown with bold edges.

dependence, and we have a parallel machine with lots of processors instead of a sequential machine with only one processor. How should we schedule the tasks so as to minimize the total time used?

For simplicity, assume all tasks take 1 unit of time and we have an unlimited number of identical processors. For example, in the clothes example in Figure 7.11, how long would it take to handle all the garments?

In the first unit of time, we should do all minimal items, so we would put on our left sock, our right sock, our underwear, and our shirt.[7] In the second unit of time, we should put on our pants and our tie. Note that we cannot put on our left or right shoe yet, since we have not yet put on our pants. In the third unit of time, we should put on our left shoe, our right shoe, and our belt. Finally, in the last unit of time, we can put on our jacket. This schedule is illustrated in Figure 7.13.

The total time to do these tasks is 4 units. We cannot do better than 4 units of

[7] Yes, we know that you can't actually put on both socks at once, but imagine you are being dressed by a bunch of robot processors and you are in a big hurry. Still not working for you? Ok, forget about the clothes and imagine they are programs with the precedence constraints shown in Figure 7.11.

time because there is a sequence of 4 tasks, each needing to be done before the next, of length 4. For example, we must put on our shirt before our pants, our pants before our belt, and our belt before our jacket. Such a sequence of items is known as a *chain*

Definition 7.8.1. A *chain* is a sequence $a_1 \preceq a_2 \preceq \cdots \preceq a_t$, where $a_i \neq a_j$ for all $i \neq j$, such that each item is comparable to the next in the chain, and it is smaller with respect to $\preceq$. The *length* of the chain is t, the number of elements in the chain.

Thus, the time it takes to schedule tasks, even with an unlimited number of processors, is at least the length of the longest chain. Indeed, if we used less time, then two items from a longest chain would have to be done at the same time, which contradicts the precedence constraints. For this reason, a longest chain is also known as a *critical path*. For example, Figure 7.13 shows the critical path for the getting-dressed poset.

In this example, we were in fact able to schedule all the tasks in t steps, where t is the length of the longest chain. The really nice thing about posets is that this is always possible! In other words, for any poset, there is a legal parallel schedule that runs in t steps, where t is the length of the longest chain.

There are lots of ways to prove this fact. Our proof will also give us the corresponding schedule in t time steps, and allow us to obtain some nice corollaries.

Theorem 7.8.2. *Given any finite poset $(A, \preceq)$ for which the longest chain has length t, it is possible to partition A into t subsets A_1, A_2, ..., A_t such that for all $i \in \{1, 2, \dots, t\}$ and for all $a \in A_i$, we have that all $b \preceq a$ appear in the set $A_1 \cup \ldots \cup A_{i-1}$.*

Before proving this theorem, first note that for each i, all items in A_i can be scheduled in time step i. This is because all preceding tasks are scheduled in preceding time steps, and thus are already completed. So the theorem implies that

Corollary 7.8.3. *The total amount of parallel time needed to complete the tasks is the same as the length of the longest chain.*

Proof of Theorem 7.8.2. For all $a \in A_i$, put a in A_i, where i is the length of the longest chain ending at a. For example, the A_i for the getting-dressed poset are shown in Figure 7.13. In what follows, we show that for all i, for all $a \in A_i$ and for all $b \preceq a$ with $b \neq a$, we have $b \in A_1 \cup A_2 \cup \ldots \cup A_{i-1}$.

We prove this by contradiction. Assume there is some i, $a \in A_i$, and $b \preceq a$ with $b \neq a$ and $b \notin A_1 \cup A_2 \cup \ldots \cup A_{i-1}$. By the way we defined A_i, this implies there is a chain of length at least i ending at b. Since $b \preceq a$ and $b \neq a$, we can extend this chain to a chain of length at least $i + 1$, ending at a. But then a could not be in A_i. This is a contradiction. ■

If we have an unlimited number of processors, then the time to complete all tasks is equal to the length of the longest chain of dependent tasks. The case where there are only a limited number of processors is very useful in practice and it is covered in the Problems section.

7.9 Dilworth's Lemma

Definition 7.9.1. An *antichain* in a poset is a set of elements such that any two elements in the set are incomparable.

For example, each A_i in the proof of Theorem 7.8.2 and in Figure 7.13 is an antichain since its elements have no dependencies between them (which is why they could be executed at the same time).

Our conclusions about scheduling also tell us something about antichains.

Corollary 7.9.2. *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 $A_1, A_2, \dots, A_t$ defined in Theorem 7.8.2. ■

Corollary 7.9.2 implies a famous result[8] about partially ordered sets:

Lemma 7.9.3 (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. By contradiction. Assume that the longest chain has length at most t and the longest antichain has size less than n/t. Then by Corollary 7.9.2, the n elements can be partitioned into at most t antichains. Hence, there are fewer than $t \cdot n/t = n$ elements in the poset, which is a contradiction. Hence there must be a chain longer than t or an antichain with at least n/t elements. ■

Corollary 7.9.4. *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 7.9.3. ■

As an application, consider a permutation of the numbers from 1 to n arranged as a sequence from left to right on a line. Corollary 7.9.4 can be used to show that there must be a $\sqrt{n}$-length subsequence of these numbers that is completely

[8] Lemma 7.9.3 also follows from a more general result known as Dilworth's Theorem that we will not discuss.

increasing or completely decreasing as you move from left to right. For example, the sequence

$$7, 8, 9, 4, 5, 6, 1, 2, 3$$

has an increasing subsequence of length 3 (for example, 7, 8, 9) and a decreasing subsequence of length 3 (for example, 9, 6, 3). The proof of this result is left as an exercise that will test your ability to find the right partial order on the numbers in the sequence.

8 State Machines

This chapter needs to be reworked.

III Counting

Introduction

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.

Perhaps surprisingly, but certainly not coincidentally, the number in each of these two situations is the same: 1820.

Counting is useful in computer science for several reasons:

- Determining the time and storage required to solve a computational problem—a central objective in computer science—often comes down to solving a counting problem.
- Counting is the basis of probability theory, which plays a central role in all sciences, including computer science.
- Two remarkable proof techniques, the "pigeonhole principle" and "combinatorial proof," rely on counting. These lead to a variety of interesting and useful insights.

In the next several chapters, we're going to present a lot of rules for counting. These rules are actually theorems, and we will prove some of them, but our focus won't be on the proofs *per se*—our objective is to teach you simple counting as a practical skill, like integration.

We begin our study of counting in Chapter 9 with a collection of rules and methods for finding closed-form expressions for commonly-occurring sums and products such as $\sum_{i=1}^{n} x^i$ and $n! = \prod_{i=1}^{n} i$. We also introduce asymptotic notations such as $\sim$, O, and Θ that are commonly used in computer science to express the how a quantity such as the running time of a program grows with the size of the input.

In Chapter 10, we show how to solve a variety of recurrences that arise in computational problems. These methods are especially useful when you need to design or analyze recursive programs.

In Chapters 11 and 12, we describe the most basic rules for determining the cardinality of a set. This material is simple yet powerful, and it provides a great tool set for use in your future career.

We conclude in Chapter 13 with a brief digression into the final frontier of counting—infinity. We'll define what it means for a set to be countable and show you some examples of sets that are really big—bigger even than the set of real numbers.

9 Sums and Asymptotics

Sums and products arise regularly in the analysis of algorithms, financial applications, physical problems, and probabilistic systems. For example, we have already encountered the sum $1 + 2 + 4 + \cdots + N$ when counting the number of nodes in a complete binary tree with N inputs. Although such a sum can be represented compactly using the sigma notation

$$\sum_{i=0}^{\log N} 2^i, \tag{9.1}$$

it is a lot easier and more helpful to express the sum by its *closed form* value

$$2N - 1.$$

By *closed form*, we mean an expression that does not make use of summation or product symbols or otherwise need those handy (but sometimes troublesome) dots.... Expressions in closed form are usually easier to evaluate (it doesn't get much simpler than $2N - 1$, for example) and it is usually easier to get a feel for their magnitude than expressions involving large sums and products.

But how do you find a closed form for a sum or product? Well, it's part math and part art. And it is the subject of this chapter.

We will start the chapter with a motivating example involving annuities. Figuring out the value of the annuity will involve a large and nasty-looking sum. We will then describe several methods for finding closed forms for all sorts of sums, including the annuity sums. In some cases, a closed form for a sum may not exist and so we will provide a general method for finding good upper and lower bounds on the sum (which are closed form, of course).

The methods we develop for sums will also work for products since you can convert any product into a sum by taking a logarithm of the product. As an example, we will use this approach to find a good closed-form approximation to

$$n! ::= 1 \cdot 2 \cdot 3 \cdots n.$$

We conclude the chapter with a discussion of asymptotic notation. Asymptotic notation is often used to bound the error terms when there is no exact closed form expression for a sum or product. It also provides a convenient way to express the growth rate or order of magnitude of a sum or product.

9.1 The Value of an Annuity

Would you prefer a million dollars today or $50,000 a year for the rest of your life? On the one hand, instant gratification is nice. On the other hand, the *total dollars* received at $50K per year is much larger if you live long enough.

Formally, this is a question about the value of an annuity. An *annuity* is a financial instrument that pays out a fixed amount of money at the beginning of every year for some specified number of years. In particular, an n-year, m-payment annuity pays m dollars at the start of each year for n years. In some cases, n is finite, but not always. Examples include lottery payouts, student loans, and home mortgages. There are even Wall Street people who specialize in trading annuities.[1]

A key question is, “What is an annuity 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.

9.1.1 The Future Value of Money

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[2] 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 is 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.

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

[1] Such trading ultimately led to the subprime mortgage disaster in 2008–2009. We’ll talk more about that in Section 19.5.3.

[2] 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.

payment values. This gives:

$$\begin{aligned} V &= \sum_{i=1}^{n} \frac{m}{(1+p)^{i-1}} \\ &= m \cdot \sum_{j=0}^{n-1} \left(\frac{1}{1+p}\right)^{j} && \text{(substitute } j = i-1\text{)} \\ &= m \cdot \sum_{j=0}^{n-1} x^{j} && \text{(substitute } x = 1/(1+p)\text{).} \qquad (9.2) \end{aligned}$$

The goal of the preceding substitutions was to get the summation into a simple special form so that we can solve it with a general formula. In particular, the terms of the sum

$$\sum_{j=0}^{n-1} x^{j} = 1 + x + x^2 + x^3 + \cdots + x^{n-1}$$

form a *geometric series*, which means that the ratio of consecutive terms is always the same and it is a positive value less than one. In this case, the ratio is always x, and $0 < x < 1$ since we assumed that $p > 0$. It turns out that there is a nice closed-form expression for any geometric series; namely

$$\sum_{i=0}^{n-1} x^{i} = \frac{1-x^n}{1-x}. \qquad (9.3)$$

Equation 9.3 can be verified by induction, but, as is often the case, the proof by induction gives no hint about how the formula was found in the first place. So we'll take this opportunity to describe a method that you could use to figure it out for yourself. It is called the *Perturbation Method*.

9.1.2 The Perturbation Method

Given a sum that has a nice structure, it is often useful to "perturb" the sum so that we can somehow combine the sum with the perturbation to get something much simpler. For example, suppose

$$S = 1 + x + x^2 + \cdots + x^{n-1}.$$

An example of a perturbation would be

$$xS = x + x^2 + \cdots + x^{n}.$$

The difference between S and xS is not so great, and so if we were to subtract xS from S, there would be massive cancellation:

$$\begin{aligned} S &= 1 + x + x^2 + x^3 + \cdots + x^{n-1} \\ -xS &= \quad\; - x - x^2 - x^3 - \cdots - x^{n-1} - x^n. \end{aligned}$$

The result of the subtraction is

$$S - xS = 1 - x^n.$$

Solving for S gives the desired closed-form expression in Equation 9.3:

$$S = \frac{1 - x^n}{1 - x}.$$

We’ll see more examples of this method when we introduce *generating functions* in Chapter 12.

9.1.3 A Closed Form for the Annuity Value

Using Equation 9.3, we can derive a simple formula for V, the value of an annuity that pays m dollars at the start of each year for n years.

$$\begin{aligned} V &= m\left(\frac{1 - x^n}{1 - x}\right) && \text{(by Equations 9.2 and 9.3)} \qquad (9.4) \\ &= m\left(\frac{1 + p - (1/(1+p))^{n-1}}{p}\right) && \text{(substituting } x = 1/(1+p)\text{)}. \qquad (9.5) \end{aligned}$$

Equation 9.5 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.

9.1.4 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 Equation 9.3 as n tends to infinity.

Theorem 9.1.1. *If $|x| < 1$, then*

$$\sum_{i=0}^{\infty} x^i = \frac{1}{1-x}.$$

Proof.

$$\begin{aligned}
\sum_{i=0}^{\infty} x^i &::= \lim_{n\to\infty} \sum_{i=0}^{n-1} x^i \\
&= \lim_{n\to\infty} \frac{1-x^n}{1-x} && \text{(by Equation 9.3)} \\
&= \frac{1}{1-x}.
\end{aligned}$$

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 9.1.1 applies, and we get

$$\begin{aligned}
V &= m \cdot \sum_{j=0}^{\infty} x^j && \text{(by Equation 9.2)} \\
&= m \cdot \frac{1}{1-x} && \text{(by Theorem 9.1.1)} \\
&= m \cdot \frac{1+p}{p} && (x = 1/(1+p)).
\end{aligned}$$

Plugging in $m = \$50{,}000$ and $p = 0.08$, we see that 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.

9.1.5 Examples

Equation 9.3 and Theorem 9.1.1 are incredibly useful in computer science. In fact, we already used Equation 9.3 implicitly when we claimed in Chapter 6 than an N-input complete binary tree has

$$1 + 2 + 4 + \cdots + N = 2N - 1$$

nodes. Here are some other common sums that can be put into closed form using Equation 9.3 and Theorem 9.1.1:

$$1 + 1/2 + 1/4 + \cdots = \sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^i = \frac{1}{1-(1/2)} = 2 \tag{9.6}$$

$$0.99999\cdots = 0.9 \sum_{i=0}^{\infty} \left(\frac{1}{10}\right)^i = 0.9\left(\frac{1}{1-1/10}\right) = 0.9\left(\frac{10}{9}\right) = 1 \tag{9.7}$$

$$1 - 1/2 + 1/4 - \cdots = \sum_{i=0}^{\infty} \left(\frac{-1}{2}\right)^i = \frac{1}{1-(-1/2)} = \frac{2}{3} \tag{9.8}$$

$$1 + 2 + 4 + \cdots + 2^{n-1} = \sum_{i=0}^{n-1} 2^i = \frac{1-2^n}{1-2} = 2^n - 1 \tag{9.9}$$

$$1 + 3 + 9 + \cdots + 3^{n-1} = \sum_{i=0}^{n-1} 3^i = \frac{1-3^n}{1-3} = \frac{3^n - 1}{2} \tag{9.10}$$

If the terms in a geometric sum grow smaller, as in Equation 9.6, then the sum is said to be *geometrically decreasing*. If the terms in a geometric sum grow progressively larger, as in Equations 9.9 and 9.10, then the sum is said to be *geometrically increasing*. In either case, the sum is usually approximately equal to the term in the sum with the greatest absolute value. For example, in Equations 9.6 and 9.8, the largest term is equal to 1 and the sums are 2 and 2/3, both relatively close to 1. In Equation 9.9, the sum is about twice the largest term. In Equation 9.10, 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. You can see why this rule of thumb works by looking carefully at Equation 9.3 and Theorem 9.1.1.

9.1.6 Variations of Geometric Sums

We now know all about geometric sums—if you have one, life is easy. 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:

$$\sum_{i=1}^{n-1} ix^i = x + 2x^2 + 3x^3 + \cdots + (n-1)x^{n-1}$$

This is not a geometric sum, since the ratio between successive terms is not fixed, and so our formula for the sum of a geometric sum cannot be directly applied. But suppose that we differentiate Equation 9.3:

$$\frac{d}{dx}\left(\sum_{i=0}^{n-1} x^i\right) = \frac{d}{dx}\left(\frac{1-x^n}{1-x}\right). \tag{9.11}$$

The left-hand side of Equation 9.11 is simply

$$\sum_{i=0}^{n-1} \frac{d}{dx}(x^i) = \sum_{i=0}^{n-1} i x^{i-1}.$$

The right-hand side of Equation 9.11 is

$$\begin{aligned}\frac{-nx^{n-1}(1-x)-(-1)(1-x^n)}{(1-x)^2} &= \frac{-nx^{n-1}+nx^n+1-x^n}{(1-x)^2}\\ &= \frac{1-nx^{n-1}+(n-1)x^n}{(1-x)^2}.\end{aligned}$$

Hence, Equation 9.11 means that

$$\sum_{i=0}^{n-1} i x^{i-1} = \frac{1-nx^{n-1}+(n-1)x^n}{(1-x)^2}.$$

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 $\sum i x^{i-1}$, but we want a formula for the series $\sum i x^i$. The solution is simple: multiply by x. This gives:

$$\sum_{i=1}^{n-1} i x^i = \frac{x-nx^n+(n-1)x^{n+1}}{(1-x)^2} \tag{9.12}$$

and we have the desired closed-form expression for our sum[3]. It's a little complicated looking, but it's easier to work with than the sum.

Notice that if $|x| < 1$, then this series converges to a finite value even if there are infinitely many terms. Taking the limit of Equation 9.12 as n tends infinity gives the following theorem:

[3] Since we could easily have made a mistake in the calculation, it is always a good idea to go back and validate a formula obtained this way with a proof by induction.

Theorem 9.1.2. *If* $|x| < 1$, *then*

$$\sum_{i=1}^{\infty} i x^i = \frac{x}{(1-x)^2}.$$

As a consequence, suppose that 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 Theorem 9.1.2 to compute the value:

$$\begin{aligned} V &= \sum_{i=1}^{\infty} \frac{im}{(1+p)^i} \\ &= m \cdot \frac{1/(1+p)}{(1-\frac{1}{1+p})^2} \\ &= m \cdot \frac{1+p}{p^2}. \end{aligned}$$

The second line follows by an application of Theorem 9.1.2. 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 the 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!

9.2 Power Sums

In Chapter 3, we verified the formula

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}. \tag{9.13}$$

But the source of this formula is still a mystery. Sure, we can prove it is true using well ordering or induction, but where did the expression on the right come from in the first place? Even more inexplicable is the closed form expression for the sum of consecutive squares:

$$\sum_{i=1}^{n} i^2 = \frac{(2n+1)(n+1)n}{6}. \tag{9.14}$$

It turns out that there is a way to derive these expressions, but before we explain it, we thought it would be fun[4] to show you how Gauss proved Equation 9.13 when he was a young boy.[5]

Gauss's idea is related to the perturbation method we used in Section 9.1.2. Let

$$S = \sum_{i=1}^{n} i.$$

Then we can write the sum in two orders:

$$\begin{aligned} S &= 1 + \quad 2 \quad + \ldots + (n-1) + n, \\ S &= n + (n-1) + \ldots + \quad 2 \quad + 1. \end{aligned}$$

Adding these two equations gives

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

Hence,

$$S = \frac{n(n+1)}{2}.$$

Not bad for a young child. Looks like Gauss had some potential.. . .

Unfortunately, the same trick does not work for summing consecutive squares. However, we can observe that the result might be a third-degree polynomial in n, since the sum contains n terms that average out to a value that grows quadratically in n. So we might guess that

$$\sum_{i=1}^{n} i^2 = an^3 + bn^2 + cn + d.$$

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,

[4] Remember that we are mathematicians, so our definition of "fun" may be different than yours.

[5] We suspect that Gauss was probably not an ordinary boy.

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:

$$\begin{aligned}
n = 0 &\quad \text{implies} \quad 0 = d \\
n = 1 &\quad \text{implies} \quad 1 = a + b + c + d \\
n = 2 &\quad \text{implies} \quad 5 = 8a + 4b + 2c + d \\
n = 3 &\quad \text{implies} \quad 14 = 27a + 9b + 3c + d.
\end{aligned}$$

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 9.14.

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, 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![6]

9.3 Approximating Sums

Unfortunately, it is not always possible to find a closed-form expression for a sum. For example, consider the sum

$$S = \sum_{i=1}^{n} \sqrt{i}.$$

No closed form expression is known for S.

In such cases, we need to resort to approximations for S if we want to have a closed form. The good news is that there is a general method to find closed-form upper and lower bounds that work for most any sum. Even better, the method is simple and easy to remember. It works by replacing the sum by an integral and then adding either the first or last term in the sum.

[6] Alternatively, you can use the method based on generating functions described in Chapter 12, which does not require any guessing at all.

Theorem 9.3.1. *Let $f : \mathbb{R}^+ \to \mathbb{R}^+$ be a nondecreasing*[7] *continuous function and let*

$$S = \sum_{i=1}^{n} f(i)$$

and

$$I = \int_1^n f(x)\,dx.$$

Then

$$I + f(1) \leq S \leq I + f(n).$$

Similarly, if f is nonincreasing, then

$$I + f(n) \leq S \leq I + f(1).$$

Proof. Let $f : \mathbb{R}^+ \to \mathbb{R}^+$ be a nondecreasing function. For example, $f(x) = \sqrt{x}$ is such a function.

Consider the graph shown in Figure 9.1. The value of

$$S = \sum_{i=1}^{n} f(i)$$

is represented by the shaded area in this figure. This is because the ith rectangle in the figure (counting from left to right) has width 1 and height $f(i)$.

The value of

$$I = \int_1^n f(x)\,dx$$

is the shaded area under the curve of $f(x)$ from 1 to n shown in Figure 9.2.

Comparing the shaded regions in Figures 9.1 and 9.2, we see that S is at least I plus the area of the leftmost rectangle. Hence,

$$S \geq I + f(1) \tag{9.15}$$

This is the lower bound for S. We next derive the upper bound.

Figure 9.3 shows the curve of $f(x)$ from 1 to n shifted left by 1. This is the same as the curve $f(x + 1)$ from 0 to $n - 1$ and it has the same area I.

Comparing the shaded regions in Figures 9.1 and 9.3, we see that S is at most I plus the area of the rightmost rectangle. Hence,

$$S \leq I + f(n). \tag{9.16}$$

[7] A function f is *nondecreasing* if $f(x) \geq f(y)$ whenever $x \geq y$. It is *nonincreasing* if $f(x) \leq f(y)$ whenever $x \geq y$.

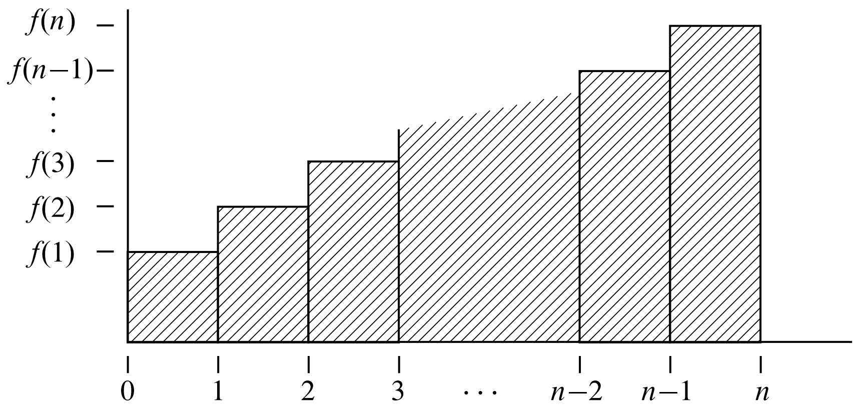


Figure 9.1 The area of the ith rectangle is $f(i)$. The shaded region has area $\sum_{i=1}^{n} f(i)$.

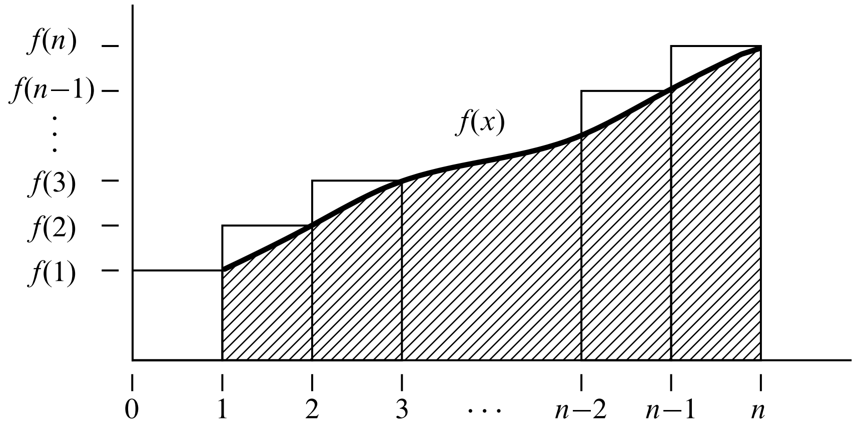


Figure 9.2 The shaded area under the curve of $f(x)$ from 1 to n (shown in bold) is $I = \int_1^n f(x)\,dx$.

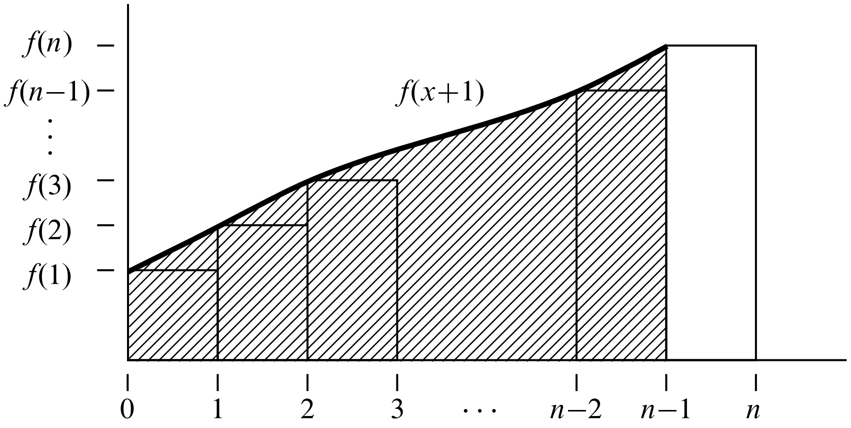


Figure 9.3 The shaded area under the curve of $f(x+1)$ from 0 to $n-1$ is I, the same as the area under the curve of $f(x)$ from 1 to n. This curve is the same as the curve in Figure 9.2 except that has been shifted left by 1.

Combining Equations 9.15 and 9.16, we find that

$$I + f(1) \le S \le I + f(n),$$

for any nondecreasing function f, as claimed

The argument for the case when f is nonincreasing is very similar. The analogous graphs to those shown in Figures 9.1–9.3 are provided in Figure 9.4. As you can see by comparing the shaded regions in Figures 9.4(a) and 9.4(b),

$$S \le I + f(1).$$

Similarly, comparing the shaded regions in Figures 9.4(a) and 9.4(c) reveals that

$$S \ge I + f(n).$$

Hence, if f is nonincreasing,

$$I + f(n) \le S \le I + f(1).$$

as claimed. ■

Theorem 9.3.1 provides good bounds for most sums. At worst, the bounds will be off by the largest term in the sum. For example, we can use Theorem 9.3.1 to bound the sum

$$S = \sum_{i=1}^{n} \sqrt{i}$$

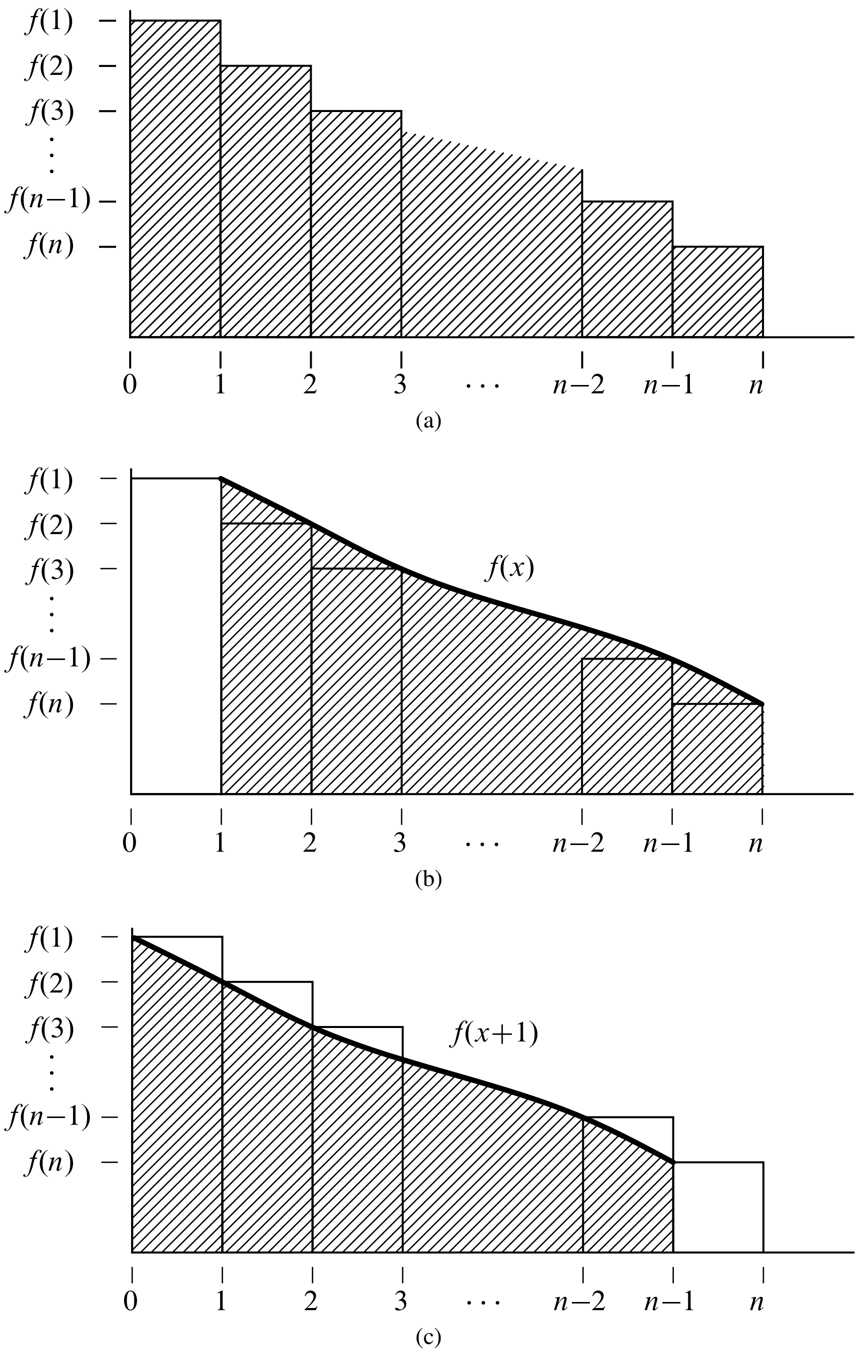


Figure 9.4 The area of the shaded region in (a) is $S = \sum_{i=1}^{n} f(i)$. The area in the shaded regions in (b) and (c) is $I = \int_{1}^{n} f(x)\,dx$.

as follows.

We begin by computing

$$\begin{aligned} I &= \int_1^n \sqrt{x}\,dx \\ &= \left.\frac{x^{3/2}}{3/2}\right|_1^n \\ &= \frac{2}{3}(n^{3/2} - 1). \end{aligned}$$

We then apply Theorem 9.3.1 to conclude that

$$\frac{2}{3}(n^{3/2} - 1) + 1 \;\le\; S \;\le\; \frac{2}{3}(n^{3/2} - 1) + \sqrt{n}$$

and thus that

$$\frac{2}{3}n^{3/2} + \frac{1}{3} \;\le\; S \;\le\; \frac{2}{3}n^{3/2} + \sqrt{n} - \frac{2}{3}.$$

In other words, the sum is very close to $\frac{2}{3}n^{3/2}$.

We'll be using Theorem 9.3.1 extensively going forward. At the end of this chapter, we will also introduce some notation that expresses phrases like "the sum is very close to" in a more precise mathematical manner. But first, we'll see how Theorem 9.3.1 can be used to resolve a classic paradox in structural engineering.

9.4 Hanging Out Over the Edge

Suppose that you have n identical blocks[8] and that you stack them one on top of the next on a table as shown in Figure 9.5. Is there some value of n for which it is possible to arrange the stack so that one of the blocks hangs out completely over the edge of the table without having the stack fall over? (You are not allowed to use glue or otherwise hold the stack in position.)

Most people's first response to this question—sometimes also their second and third responses—is "No. No block will ever get completely past the edge of the table." But in fact, if n is large enough, you can get the top block to stick out as far as you want: one block-length, two block-lengths, any number of block-lengths!

[8] We will assume that the blocks are rectangular, uniformly weighted and of length 1.

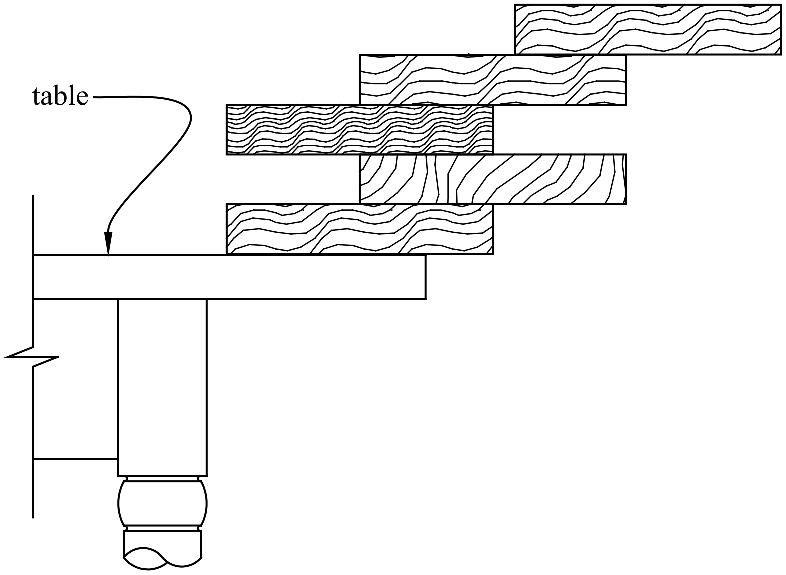


Figure 9.5 A stack of 5 identical blocks on a table. The top block is hanging out over the edge of the table, but if you try stacking the blocks this way, the stack will fall over.

9.4.1 Stability

A stack of blocks is said to be *stable* if it will not fall over of its own accord. For example, the stack illustrated in Figure 9.5 is not stable because the top block is sure to fall over. This is because the center or mass of the top block is hanging out over air.

In general, a stack of n blocks will be stable if and only if the center of mass of the top i blocks sits over the $(i+1)$st block for $i = 1, 2, \ldots, n-1$, and over the table for $i = n$.

We define the *overhang* of a stable stack to be the distance between the edge of the table and the rightmost end of the rightmost block in the stack. Our goal is thus to maximize the overhang of a stable stack.

For example, the maximum possible overhang for a single block is $1/2$. That is because the center of mass of a single block is in the middle of the block (which is distance $1/2$ from the right edge of the block). If we were to place the block so that its right edge is more than $1/2$ from the edge of the table, the center of mass would be over air and the block would tip over. But we can place the block so the center of mass is at the edge of the table, thereby achieving overhang $1/2$. This position is illustrated in Figure 9.6.

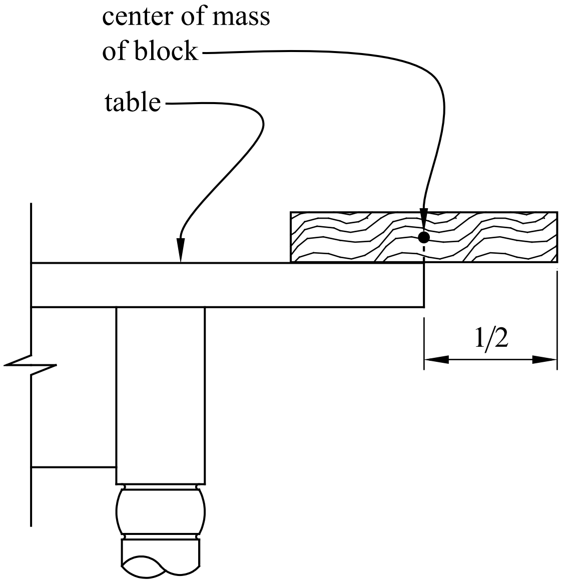


Figure 9.6 One block can overhang half a block length.

In general, the overhang of a stack of blocks is maximized by sliding the entire stack rightward until its center of mass is at the edge of the table. The overhang will then be equal to the distance between the center of mass of the stack and the rightmost edge of the rightmost block. We call this distance the *spread* of the stack. Note that the spread does not depend on the location of the stack on the table—it is purely a property of the blocks in the stack. Of course, as we just observed, the maximum possible overhang is equal to the maximum possible spread. This relationship is illustrated in Figure 9.7.

9.4.2 A Recursive Solution

Our goal is to find a formula for the maximum possible spread S_n that is achievable with a stable stack of n blocks.

We already know that $S_1 = 1/2$ since the right edge of a single block with length 1 is always distance $1/2$ from its center of mass. Let's see if we can use a recursive approach to determine S_n for all n. This means that we need to find a formula for S_n in terms of S_i where $i < n$.

Suppose we have a stable stack $\mathcal{S}$ of n blocks with maximum possible spread S_n. There are two cases to consider depending on where the rightmost block is in the stack.

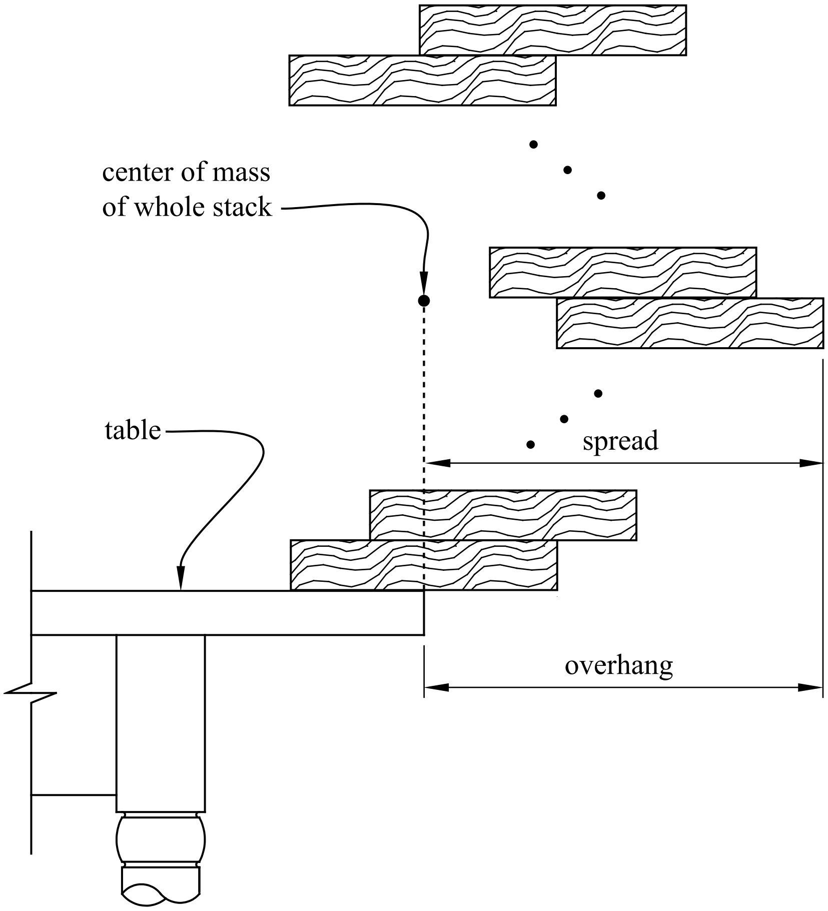


Figure 9.7 The overhang is maximized by maximizing the spread and then placing the stack so that the center of mass is at the edge of the table.

Case 1: *The rightmost block in $\mathcal{S}$ is the bottom block.* Since the center of mass of the top $n-1$ blocks must be over the bottom block for stability, the spread is maximized by having the center of mass of the top $n-1$ blocks be directly over the *left* edge of the bottom block. In this case the center of mass of $\mathcal{S}$ is[9]

$$\frac{(n-1)\cdot 1 + (1)\cdot \frac{1}{2}}{n} = 1 - \frac{1}{2n}$$

to the left of the right edge of the bottom block and so the spread for $\mathcal{S}$ is

$$1 - \frac{1}{2n}. \tag{9.17}$$

For example, see Figure 9.8.

In fact, the scenario just described is easily achieved by arranging the blocks as shown in Figure 9.9, in which case we have the spread given by Equation 9.17. For example, the spread is $3/4$ for 2 blocks, $5/6$ for 3 blocks, $7/8$ for 4 blocks, etc.

Can we do any better? The best spread in Case 1 is always less than 1, which means that we cannot get a block fully out over the edge of the table in this scenario. Maybe our intuition was right that we can't do better. Before we jump to any false conclusions, however, let's see what happens in the other case.

Case 2: *The rightmost block in $\mathcal{S}$ is among the top $n-1$ blocks.* In this case, the spread is maximized by placing the top $n-1$ blocks so that their center of mass is directly over the *right* end of the bottom block. This means that the center of mass for $\mathcal{S}$ is at location

$$\frac{(n-1)\cdot C + 1\cdot\left(C - \frac{1}{2}\right)}{n} = C - \frac{1}{2n}$$

where C is the location of the center of mass of the top $n-1$ blocks. In other words, the center of mass of $\mathcal{S}$ is $1/2n$ to the left of the center of mass of the top $n-1$ blocks. (The difference is due to the effect of the bottom block, whose center of mass is $1/2$ unit to the left of C.) This means that the spread of $\mathcal{S}$ is $1/2n$ greater than the spread of the top $n-1$ blocks (because we are in the case where the rightmost block is among the top $n-1$ blocks.)

Since the rightmost block is among the top $n-1$ blocks, the spread for $\mathcal{S}$ is maximized by maximizing the spread for the top $n-1$ blocks. Hence the maximum spread for $\mathcal{S}$ in this case is

$$S_{n-1} + \frac{1}{2n} \tag{9.18}$$

[9] The center of mass of a stack of blocks is the average of the centers of mass of the individual blocks.

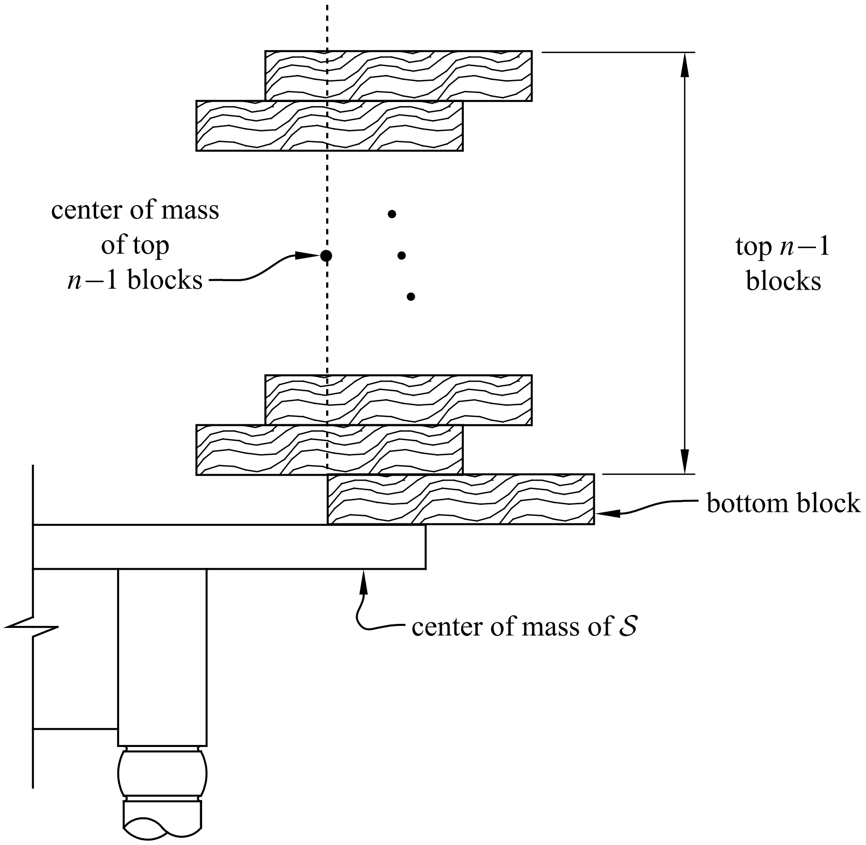


Figure 9.8 The scenario where the bottom block is the rightmost block. In this case, the spread is maximized by having the center of mass of the top $n-1$ blocks be directly over the left edge of the bottom block.

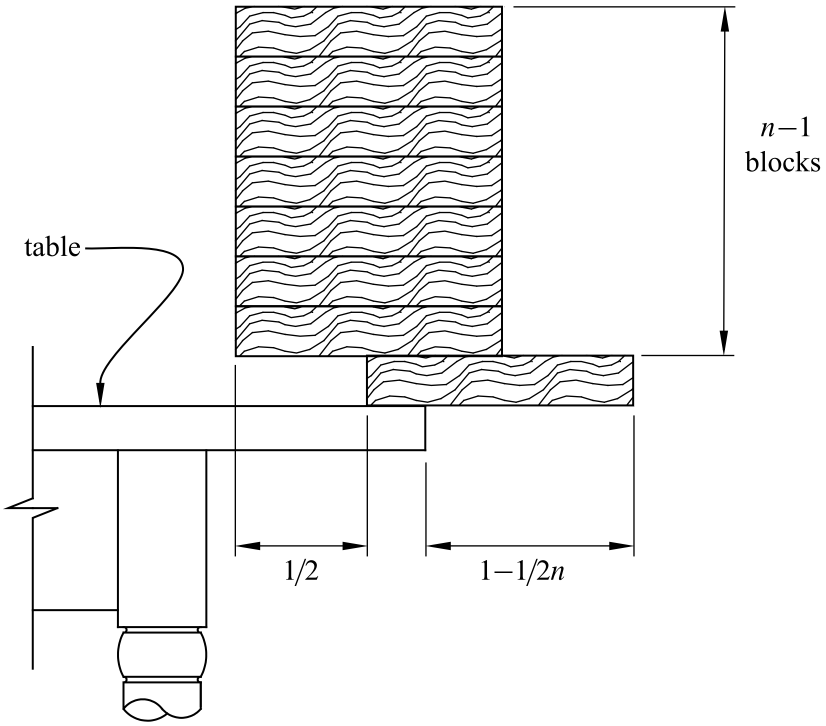


Figure 9.9 A method for achieving spread (and hence overhang) $1 - 1/2n$ with n blocks, where the bottom block is the rightmost block.

where S_{n-1} is the maximum possible spread for $n-1$ blocks (using any strategy).

We are now almost done. There are only two cases to consider when designing a stack with maximum spread and we have analyzed both of them. This means that we can combine Equation 9.17 from Case 1 with Equation 9.18 from Case 2 to conclude that

$$S_n = \max\left\{1 - \frac{1}{2n},\ S_{n-1} + \frac{1}{2n}\right\} \tag{9.19}$$

for any $n > 1$.

Uh-oh. This looks complicated. Maybe we are not almost done after all!

Equation 9.19 is an example of a *recurrence*. We will describe numerous techniques for solving recurrences in Chapter 10, but, fortunately, Equation 9.19 is simple enough that we can solve it without waiting for all the hardware in Chapter 10.

One of the first things to do when you have a recurrence is to get a feel for it by computing the first few terms. This often gives clues about a way to solve the recurrence, as it will in this case.

We already know that $S_1 = 1/2$. What about S_2? From Equation 9.19, we find that

$$\begin{aligned} S_2 &= \max\left\{1 - \frac{1}{4},\ \frac{1}{2} + \frac{1}{4}\right\} \\ &= 3/4. \end{aligned}$$

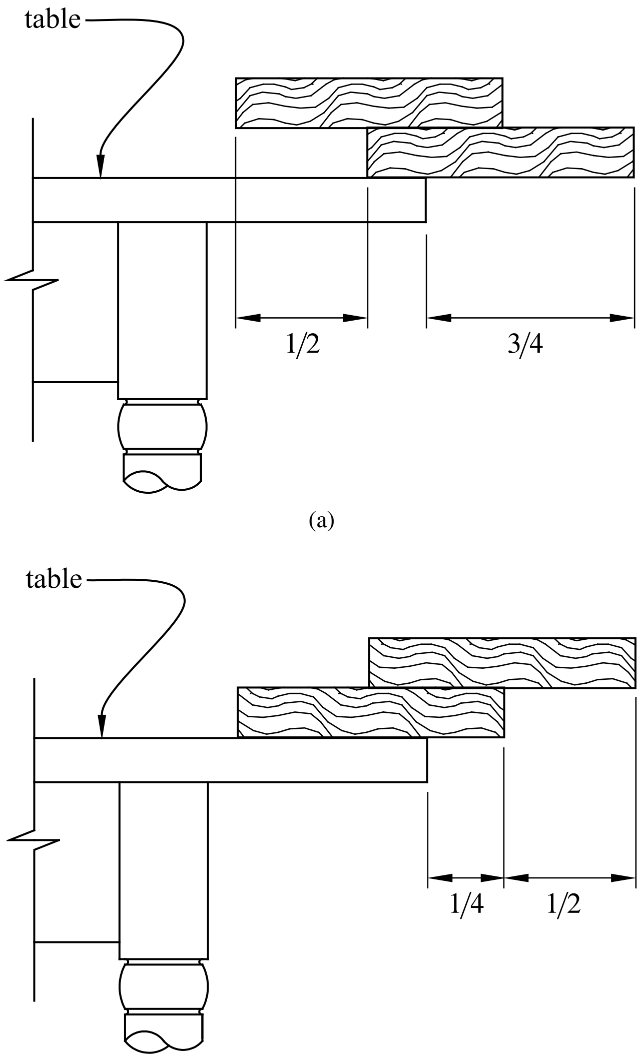
Both cases give the same spread, albeit by different approaches. For example, see Figure 9.10.

That was easy enough. What about S_3?

$$\begin{aligned} S_3 &= \max\left\{1 - \frac{1}{6},\ \frac{3}{4} + \frac{1}{6}\right\} \\ &= \max\left\{\frac{5}{6},\ \frac{11}{12}\right\} \\ &= \frac{11}{12}. \end{aligned}$$

As we can see, the method provided by Case 2 is the best. Let's check $n = 4$.

$$\begin{aligned} S_4 &= \max\left\{1 - \frac{1}{8},\ \frac{11}{12} + \frac{1}{8}\right\} \\ &= \frac{25}{24}. \end{aligned} \tag{9.20}$$



(a)

(b)

Figure 9.10 Two ways to achieve spread (and hence overhang) $3/4$ with $n = 2$ blocks. The first way (a) is from Case 1 and the second (b) is from Case 2.

Wow! This is a breakthrough—for two reasons. First, Equation 9.20 tells us that by using only 4 blocks, we can make a stack so that one of the blocks is hanging out completely over the edge of the table. The two ways to do this are shown in Figure 9.11.

The second reason that Equation 9.20 is important is that we now know that $S_4 > 1$, which means that we no longer have to worry about Case 1 for $n > 4$ since Case 1 never achieves spread greater than 1. Moreover, even for $n \leq 4$, we have now seen that the spread achieved by Case 1 never exceeds the spread achieved by Case 2, and they can be equal only for $n = 1$ and $n = 2$. This means that

$$S_n = S_{n-1} + \frac{1}{2n} \tag{9.21}$$

for all $n > 1$ since we have shown that the best spread can always be achieved using Case 2.

The recurrence in Equation 9.21 is much easier to solve than the one we started with in Equation 9.19. We can solve it by expanding the equation as follows:

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

and so on. This suggests that

$$S_n = \sum_{i=1}^{n} \frac{1}{2i}, \tag{9.22}$$

which is, indeed, the case.

Equation 9.22 can be verified by induction. The base case when $n = 1$ is true since we know that $S_1 = 1/2$. The inductive step follows from Equation 9.21.

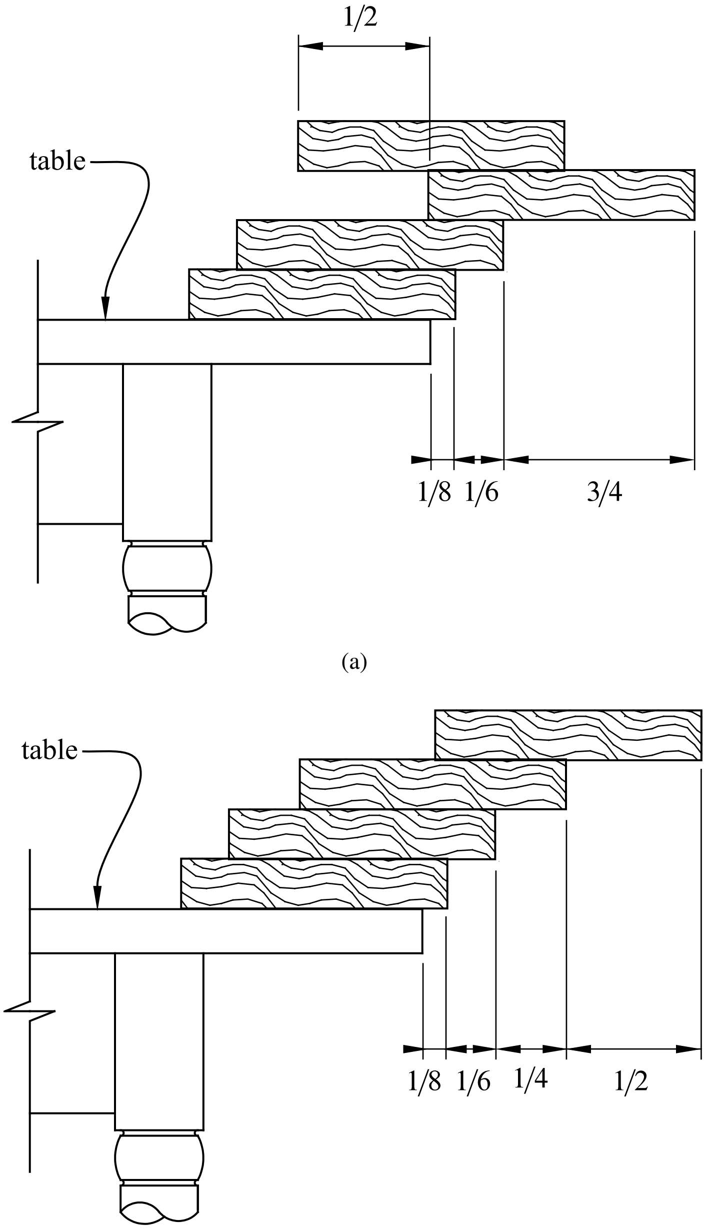
So we now know the maximum possible spread and hence the maximum possible overhang for any stable stack of books. Are we done? Not quite. Although we know that $S_4 > 1$, we still don't know how big the sum $\sum_{i=1}^{n} \frac{1}{2i}$ can get.

It turns out that S_n is very close to a famous sum known as the nth Harmonic number H_n.

9.4.3 Harmonic Numbers

Definition 9.4.1. The *nth Harmonic number* is

$$H_n ::= \sum_{i=1}^{n} \frac{1}{i}.$$



(a)

(b)

Figure 9.11 The two ways to achieve spread (and overhang) 25/24. The method in (a) uses Case 1 for the top 2 blocks and Case 2 for the others. The method in (b) uses Case 2 for every block that is added to the stack.

So Equation 9.22 means that

$$S_n = \frac{H_n}{2}. \tag{9.23}$$

The first few Harmonic numbers are easy to compute. For example,

$$H_4 = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} = \frac{25}{12}.$$

There is good news and bad news about Harmonic numbers. The bad news is that there is no closed-form expression known for the Harmonic numbers. The good news is that we can use Theorem 9.3.1 to get close upper and lower bounds on H_n. In particular, since

$$\int_1^n \frac{1}{x}\,dx = \ln(x)\,\Big|_1^n = \ln(n),$$

Theorem 9.3.1 means that

$$\ln(n) + \frac{1}{n} \le H_n \le \ln(n) + 1. \tag{9.24}$$

In other words, the nth Harmonic number is very close to $\ln(n)$.

Because the Harmonic numbers frequently arise in practice, mathematicians have worked hard to get even better approximations for them. In fact, it is now known that

$$H_n = \ln(n) + \gamma + \frac{1}{2n} + \frac{1}{12n^2} + \frac{\epsilon(n)}{120n^4} \tag{9.25}$$

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.

We are now finally done with our analysis of the block stacking problem. Plugging the value of H_n into Equation 9.23, we find that the maximum overhang for n blocks is very close to $\frac{1}{2}\ln(n)$. Since $\ln(n)$ grows to infinity as n increases, this means that if we are given enough blocks (in theory anyway), we can get a block to hang out arbitrarily far over the edge of the table. Of course, the number of blocks we need will grow as an exponential function of the overhang, so it will probably take you a long time to achieve an overhang of 2 or 3, never mind an overhang of 100.

9.4.4 Asymptotic Equality

For cases like Equation 9.25 where we understand the growth of a function like H_n up to some (unimportant) error terms, we use a special notation, $\sim$, to denote the leading term of the function. For example, we say that $H_n \sim \ln(n)$ to indicate that the leading term of H_n is $\ln(n)$. More precisely:

Definition 9.4.2. 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

$$\lim_{x \to \infty} f(x)/g(x) = 1.$$

Although it is tempting to write $H_n \sim \ln(n) + \gamma$ to indicate the two leading terms, this is not really right. According to Definition 9.4.2, $H_n \sim \ln(n) + c$ 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:

$$|H_n - \ln(n) - \gamma| \leq \left| \frac{1}{200} - \frac{1}{120000} + \frac{1}{120 \cdot 100^4} \right| < \frac{1}{200}.$$

We will spend a lot more time talking about asymptotic notation at the end of the chapter. But for now, let's get back to sums.

9.5 Double Trouble

Sometimes we have to evaluate sums of sums, otherwise known as *double summations*. This sounds hairy, and sometimes it is. But usually, it is straightforward—you just 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,[10]

$$\begin{aligned}
\sum_{n=0}^{\infty}\left(y^n\sum_{i=0}^{n}x^i\right) &= \sum_{n=0}^{\infty}\left(y^n\frac{1-x^{n+1}}{1-x}\right) && \text{Equation 9.3}\\
&= \left(\frac{1}{1-x}\right)\sum_{n=0}^{\infty}y^n - \left(\frac{1}{1-x}\right)\sum_{n=0}^{\infty}y^n x^{n+1}\\
&= \frac{1}{(1-x)(1-y)} - \left(\frac{x}{1-x}\right)\sum_{n=0}^{\infty}(xy)^n && \text{Theorem 9.1.1}\\
&= \frac{1}{(1-x)(1-y)} - \frac{x}{(1-x)(1-xy)} && \text{Theorem 9.1.1}\\
&= \frac{(1-xy)-x(1-y)}{(1-x)(1-y)(1-xy)}\\
&= \frac{1-x}{(1-x)(1-y)(1-xy)}\\
&= \frac{1}{(1-y)(1-xy)}.
\end{aligned}$$

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 first n Harmonic numbers

$$\sum_{k=1}^{n} H_k = \sum_{k=1}^{n}\sum_{j=1}^{k}\frac{1}{j} \tag{9.26}$$

For intuition about this sum, we can apply Theorem 9.3.1 to Equation 9.24 to conclude that the sum is close to

$$\int_1^n \ln(x)\,dx = x\ln(x) - x\,\Big|_1^n = n\ln(n) - n + 1.$$

Now let’s look for an exact answer. If we think about the pairs (k, j) over which

[10] Ok, so maybe this one is a little hairy, but it is also fairly straightforward. Wait till you see the next one!

we are summing, they form a triangle:

		j						
		1	2	3	4	5	$\dots$	n
k	1	1						
	2	1	1/2					
	3	1	1/2	1/3				
	4	1	1/2	1/3	1/4			
		$\dots$						
	n	1	1/2		$\dots$			$1/n$

The summation in Equation 9.26 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

$$\begin{aligned}
\sum_{k=1}^{n} H_k &= \sum_{k=1}^{n} \sum_{j=1}^{k} \frac{1}{j} \\
&= \sum_{j=1}^{n} \sum_{k=j}^{n} \frac{1}{j} \\
&= \sum_{j=1}^{n} \frac{1}{j} \sum_{k=j}^{n} 1 \\
&= \sum_{j=1}^{n} \frac{1}{j} (n - j + 1) \\
&= \sum_{j=1}^{n} \frac{n+1}{j} - \sum_{j=1}^{n} \frac{j}{j} \\
&= (n+1) \sum_{j=1}^{n} \frac{1}{j} - \sum_{j=1}^{n} 1 \\
&= (n+1) H_n - n. \qquad (9.27)
\end{aligned}$$

9.6 Products

We've covered several techniques for finding closed forms for sums but no methods for dealing with products. Fortunately, we do not need to develop an entirely new set of tools when we encounter a product such as

$$n! ::= \prod_{i=1}^{n} i. \tag{9.28}$$

That's because we can convert any product into a sum by taking a logarithm. For example, if

$$P = \prod_{i=1}^{n} f(i),$$

then

$$\ln(P) = \sum_{i=1}^{n} \ln(f(i)).$$

We can then apply our summing tools to find a closed form (or approximate closed form) for $\ln(P)$ and then exponentiate at the end to undo the logarithm.

For example, let's see how this works for the factorial function $n!$ We start by taking the logarithm:

$$\begin{aligned}\ln(n!) &= \ln(1 \cdot 2 \cdot 3 \cdots (n-1) \cdot n)\\ &= \ln(1) + \ln(2) + \ln(3) + \cdots + \ln(n-1) + \ln(n)\\ &= \sum_{i=1}^{n} \ln(i).\end{aligned}$$

Unfortunately, no closed form for this sum is known. However, we can apply Theorem 9.3.1 to find good closed-form bounds on the sum. To do this, we first compute

$$\begin{aligned}\int_1^n \ln(x)\,dx &= x\ln(x) - x\Big|_1^n\\ &= n\ln(n) - n + 1.\end{aligned}$$

Plugging into Theorem 9.3.1, this means that

$$n\ln(n) - n + 1 \;\le\; \sum_{i=1}^{n} \ln(i) \;\le\; n\ln(n) - n + 1 + \ln(n).$$

Exponentiating then gives

$$\frac{n^n}{e^{n-1}} \leq n! \leq \frac{n^{n+1}}{e^{n-1}}. \tag{9.29}$$

This means that $n!$ is within a factor of n of n^n/e^{n-1}.

9.6.1 Stirling's Formula

$n!$ is probably the most commonly used product in discrete mathematics, and so mathematicians have put in the effort to find much better closed-form bounds on its value. The most useful bounds are given in Theorem 9.6.1.

Theorem 9.6.1 (Stirling's Formula)**.** *For all $n \geq 1$,*

$$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{\epsilon(n)}$$

where

$$\frac{1}{12n+1} \leq \epsilon(n) \leq \frac{1}{12n}.$$

Theorem 9.6.1 can be proved by induction on n, but the details are a bit painful (even for us) and so we will not go through them here.

There are several important things to notice about Stirling's Formula. First, $\epsilon(n)$ is always positive. This means that

$$n! > \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \tag{9.30}$$

for all $n \in \mathbb{N}^+$.

Second, $\epsilon(n)$ tends to zero as n gets large. This means that[11]

$$n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n, \tag{9.31}$$

which is rather surprising. After all, who would expect both π and e to show up in a closed-form expression that is asymptotically equal to $n!$?

Third, $\epsilon(n)$ is small even for small values of n. This means that Stirling's Formula provides good approximations for $n!$ for most all values of n. For example, if we use

$$\sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

[11] The $\sim$ notation was defined in Section 9.4.4.

Approximation	$n \geq 1$	$n \geq 10$	$n \geq 100$	$n \geq 1000$
$\sqrt{2\pi n}\left(\frac{n}{e}\right)^n$	$< 10\%$	$< 1\%$	$< 0.1\%$	$< 0.01\%$
$\sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{1/12n}$	$< 1\%$	$< 0.01\%$	$< 0.0001\%$	$< 0.000001\%$

Table 9.1 Error bounds on common approximations for $n!$ from Theorem 9.6.1. For example, if $n \geq 100$, then $\sqrt{2\pi n}\left(\frac{n}{e}\right)^n$ approximates $n!$ to within 0.1%.

as the approximation for $n!$, as many people do, we are guaranteed to be within a factor of

$$e^{\epsilon(n)} \leq e^{\frac{1}{12n}}$$

of the correct value. For $n \geq 10$, this means we will be within 1% of the correct value. For $n \geq 100$, the error will be less than 0.1%.

If we need an even closer approximation for $n!$, then we could use either

$$\sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{1/12n}$$

or

$$\sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{1/(12n+1)}$$

depending on whether we want an upper bound or a lower bound, respectively. By Theorem 9.6.1, we know that both bounds will be within a factor of

$$e^{\frac{1}{12n} - \frac{1}{12n+1}} = e^{\frac{1}{144n^2+12n}}$$

of the correct value. For $n \geq 10$, this means that either bound will be within 0.01% of the correct value. For $n \geq 100$, the error will be less than 0.0001%.

For quick future reference, these facts are summarized in Corollary 9.6.2 and Table 9.1.

Corollary 9.6.2. *For $n \geq 1$,*

$$n! < 1.09\sqrt{2\pi n}\left(\frac{n}{e}\right)^n.$$

For $n \geq 10$,

$$n! < 1.009\sqrt{2\pi n}\left(\frac{n}{e}\right)^n.$$

For $n \geq 100$,

$$n! < 1.0009\sqrt{2\pi n}\left(\frac{n}{e}\right)^n.$$

9.7 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. For example, the asymptotic notation $\sim$ of Definition 9.4.2 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.

9.7.1 Little Oh

Definition 9.7.1. 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 9.7.2. *$x^a = o(x^b)$ for all nonnegative constants $a < b$.*

Using the familiar fact that $\log x < x$ for all $x > 1$, we can prove

Lemma 9.7.3. *$\log x = o(x^\epsilon)$ for all $\epsilon > 0$.*

Proof. Choose $\epsilon > \delta > 0$ and let $x = z^\delta$ in the inequality $\log x < x$. This implies

$$\log z < z^\delta/\delta = o(z^\epsilon) \qquad \text{by Lemma 9.7.2.} \tag{9.32}$$

■

Corollary 9.7.4. *$x^b = o(a^x)$ for any $a, b \in \mathbb{R}$ with $a > 1$.*

Lemma 9.7.3 and Corollary 9.7.4 can also be proved using l'Hôpital's Rule or the McLaurin Series for $\log x$ and e^x. Proofs can be found in most calculus texts.

9.7.2 Big Oh

Big Oh is the most frequently used asymptotic notation. It is used to give an upper bound on the growth of a function, such as the running time of an algorithm.

Definition 9.7.5. 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[12] makes it clear that

Lemma 9.7.6. *If $f = o(g)$ or $f \sim g$, then $f = O(g)$.*

Proof. $\lim f/g = 0$ or $\lim f/g = 1$ implies $\lim f/g < \infty$. ■

It is easy to see that the converse of Lemma 9.7.6 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 lim sup without mentioning it. Namely, here is an equivalent definition:

Definition 9.7.7. Given functions $f, g : \mathbb{R} \to \mathbb{R}$, we say that

$$f = O(g)$$

iff there exists a constant $c \geq 0$ and an x_0 such that for all $x \geq x_0$, $|f(x)| \leq cg(x)$.

This definition is rather complicated, but the idea is simple: $f(x) = O(g(x))$ means $f(x)$ is less than or equal to $g(x)$, except that we're willing to ignore a constant factor, namely, c, and to allow exceptions for small x, namely, $x < x_0$.

We observe,

Lemma 9.7.8. *If $f = o(g)$, then it is* not *true that $g = O(f)$.*

[12] We can't simply use the limit as $x \to \infty$ in the definition of $O()$, because if $f(x)/g(x)$ oscillates between, say, 3 and 5 as x grows, then $f = O(g)$ because $f \leq 5g$, but $\lim_{x \to \infty} f(x)/g(x)$ does not exist. So instead of limit, we use the technical notion of lim sup. In this oscillating case, $\limsup_{x \to \infty} f(x)/g(x) = 5$.

The precise definition of lim sup is

$$\limsup_{x \to \infty} h(x) ::= \lim_{x \to \infty} \text{lub}_{y \geq x} h(y),$$

where "lub" abbreviates "least upper bound."

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 9.7.9. $100x^2 = O(x^2)$.

Proof. Choose $c = 100$ and $x_0 = 1$. Then the proposition holds, since for all $x \geq 1$, $\left|100x^2\right| \leq 100x^2$. ■

Proposition 9.7.10. $x^2 + 100x + 10 = O(x^2)$.

Proof. $(x^2+100x+10)/x^2 = 1+100/x+10/x^2$ and so its limit as x approaches infinity is $1+0+0 = 1$. So in fact, $x^2+100x+10 \sim x^2$, and therefore $x^2+100x+10 = O(x^2)$. Indeed, it's conversely true that $x^2 = O(x^2+100x+10)$. ■

Proposition 9.7.10 generalizes to an arbitrary polynomial:

Proposition 9.7.11. $a_k x^k + a_{k-1}x^{k-1} + \cdots + a_1 x + a_0 = O(x^k)$.

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 uses a number of operations proportional to n^3 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. It turns out that there is another, ingenious matrix multiplication procedure that uses $O(n^{2.55})$ 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 in practice because it happens to be less efficient than the usual $O(n^3)$ procedure on matrices of practical size.[13]

9.7.3 Omega

Suppose you want to make a statement of the form “the running time of the algorithm is a least...”. Can you say it is “at least $O(n^2)$”? No! This statement is meaningless since big-oh can only be used for *upper* bounds. For lower bounds, we use a different symbol, called “big-Omega.”

[13] It is even conceivable that there is an $O(n^2)$ matrix multiplication procedure, but none is known.

Definition 9.7.12. Given functions $f, g : \mathbb{R} \to \mathbb{R}$, we say that

$$f = \Omega(g)$$

iff there exists a constant $c > 0$ and an x_0 such that for all $x \geq x_0$, we have $f(x) \geq c|g(x)|$.

In other words, $f(x) = \Omega(g(x))$ means that $f(x)$ is greater than or equal to $g(x)$, except that we are willing to ignore a constant factor and to allow exceptions for small x.

If all this sounds a lot like big-Oh, only in reverse, that's because big-Omega is the opposite of big-Oh. More precisely,

Theorem 9.7.13. *$f(x) = O(g(x))$ if and only if $g(x) = \Omega(f(x))$.*

Proof.

$$\begin{aligned}
&f(x) = O(g(x)) \\
&\qquad \text{iff} \quad \exists c > 0, x_0.\ \forall x \geq x_0.\ |f(x)| \leq cg(x) && \text{(Definition 9.7.7)} \\
&\qquad \text{iff} \quad \exists c > 0, x_0.\ \forall x \geq x_0.\ g(x) \geq \frac{1}{c}|f(x)| \\
&\qquad \text{iff} \quad \exists c' > 0, x_0.\ \forall x \geq x_0.\ g(x) \geq c'|f(x)| && (\text{set } c' = 1/c) \\
&\qquad \text{iff} \quad g(x) = \Omega(f(x)) && \text{(Definition 9.7.12)} \quad \blacksquare
\end{aligned}$$

For example, $x^2 = \Omega(x)$, $2^x = \Omega(x^2)$, and $x/100 = \Omega(100x + \sqrt{x})$.

So if the running time of your algorithm on inputs of size n is $T(n)$, and you want to say it is at least quadratic, say

$$T(n) = \Omega(n^2).$$

Little Omega

There is also a symbol called little-omega, analogous to little-oh, to denote that one function grows strictly faster than another function.

Definition 9.7.14. For functions $f, g : \mathbb{R} \to \mathbb{R}$ with f nonnegative, we say that

$$f(x) = \omega(g(x))$$

iff

$$\lim_{x \to \infty} \frac{g(x)}{f(x)} = 0.$$

In other words,

$$f(x) = \omega(g(x))$$

iff

$$g(x) = o(f(x)).$$

For example, $x^{1.5} = \omega(x)$ and $\sqrt{x} = \omega(\ln^2(x))$.

The little-omega symbol is not as widely used as the other asymptotic symbols we have been discussing.

9.7.4 Theta

Sometimes we want to specify that a running time $T(n)$ is precisely quadratic up to constant factors (both upper bound *and* lower bound). We could do this by saying that $T(n) = O(n^2)$ and $T(n) = \Omega(n^2)$, but rather than say both, mathematicians have devised yet another symbol, Θ, to do the job.

Definition 9.7.15.

$$f = \Theta(g) \quad \text{iff} \quad f = O(g) \text{ and } g = O(f).$$

The statement $f = \Theta(g)$ can be paraphrased intuitively as “f and g are equal to within a constant factor.” Indeed, by Theorem 9.7.13, we know that

$$f = \Theta(g) \quad \text{iff} \quad f = O(g) \text{ and } f = \Omega(g).$$

The Theta notation allows us to 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 we can more simply write

$$T(n) = \Theta(n^3).$$

In this case, we would say that T *is of order* n^3 or that $T(n)$ *grows cubically*, which is probably what we really want to know. 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*[14] 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.

[14] 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 $T(n) \sim n^3$.

9.7.5 Pitfalls with Asymptotic Notation

There is a long list of ways to make mistakes with asymptotic notation. This section presents some of the ways that Big Oh notation can lead to ruin and despair. With minimal effort, you can cause just as much chaos with the other symbols.

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; 4^x actually grows as the square of 2^x.

Constant Confusion

Every constant is $O(1)$. For example, $17 = O(1)$. This is true because if we let $f(x) = 17$ and $g(x) = 1$, then there exists a $c > 0$ and an x_0 such that $|f(x)| \leq cg(x)$. In particular, we could choose $c = 17$ and $x_0 = 1$, since $|17| \leq 17 \cdot 1$ for all $x \geq 1$. We can construct a false theorem that exploits this fact.

False Theorem 9.7.16.

$$\sum_{i=1}^{n} i = O(n)$$

Bogus 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—it 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[15] "$T(n) = \Omega(n^2)$."

[15] This can also be correctly expressed as $n^2 = O(T(n))$, but such notation is rare.

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 "$O(f) = g$."

Similarly, you will often see statements like

$$H_n = \ln(n) + \gamma + O\left(\frac{1}{n}\right)$$

or

$$n! = (1 + o(1))\sqrt{2\pi n}\left(\frac{n}{e}\right)^n.$$

In such cases, the true meaning is

$$H_n = \ln(n) + \gamma + f(n)$$

for some $f(n)$ where $f(n) = O(1/n)$, and

$$n! = (1 + g(n))\sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

where $g(n) = o(1)$. These transgressions are OK as long as you (and you reader) know what you mean.

10 Recurrences

A recurrence describes a sequence of numbers. Early terms are specified explicitly and later terms are expressed as a function of their predecessors. As a trivial example, this recurrence describes the sequence 1, 2, 3, etc.:

$$\begin{aligned} T_1 &= 1 \\ T_n &= T_{n-1} + 1 \qquad \text{(for } n \geq 2\text{).} \end{aligned}$$

Here, the first term is defined to be 1 and each subsequent term is one more than its predecessor.

Recurrences turn out to be a powerful tool. In this chapter, we'll emphasize using recurrences to analyze the performance of recursive algorithms. However, recurrences have other applications in computer science as well, such as enumeration of structures and analysis of random processes. And, as we saw in Section 9.4, they also arise in the analysis of problems in the physical sciences.

A recurrence in isolation is not a very useful description of a sequence. One can not easily answer simple questions such as, "What is the hundredth term?" or "What is the asymptotic growth rate?" So one typically wants to *solve* a recurrence; that is, to find a closed-form expression for the nth term.

We'll first introduce two general solving techniques: guess-and-verify and plug-and-chug. These methods are applicable to every recurrence, but their success requires a flash of insight—sometimes an unrealistically brilliant flash. So we'll also introduce two big classes of recurrences, linear and divide-and-conquer, that often come up in computer science. Essentially all recurrences in these two classes are solvable using cookbook techniques; you follow the recipe and get the answer. A drawback is that calculation replaces insight. The "Aha!" moment that is essential in the guess-and-verify and plug-and-chug methods is replaced by a "Huh" at the end of a cookbook procedure.

At the end of the chapter, we'll develop rules of thumb to help you assess many recurrences without any calculation. These rules can help you distinguish promising approaches from bad ideas early in the process of designing an algorithm.

Recurrences are one aspect of a broad theme in computer science: reducing a big problem to progressively smaller problems until easy base cases are reached. This same idea underlies both induction proofs and recursive algorithms. As we'll see, all three ideas snap together nicely. For example, one might describe the running time of a recursive algorithm with a recurrence and use induction to verify the solution.

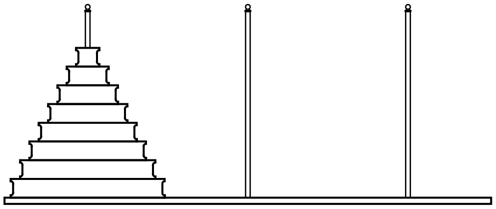


Figure 10.1 The initial configuration of the disks in the Towers of Hanoi problem.

10.1 The Towers of Hanoi

According to legend, there is a temple in Hanoi with three posts and 64 gold disks of different sizes. Each disk has a hole through the center so that it fits on a post. In the misty past, all the disks were on the first post, with the largest on the bottom and the smallest on top, as shown in Figure 10.1.

Monks in the temple have labored through the years since to move all the disks to one of the other two posts according to the following rules:

- The only permitted action is removing the top disk from one post and dropping it onto another post.
- A larger disk can never lie above a smaller disk on any post.

So, for example, picking up the whole stack of disks at once and dropping them on another post is illegal. That’s good, because the legend says that when the monks complete the puzzle, the world will end!

To clarify the problem, suppose there were only 3 gold disks instead of 64. Then the puzzle could be solved in 7 steps as shown in Figure 10.2.

The questions we must answer are, “Given sufficient time, can the monks succeed?” If so, “How long until the world ends?” And, most importantly, “Will this happen before the final exam?”

10.1.1 A Recursive Solution

The Towers of Hanoi problem can be solved recursively. As we describe the procedure, we’ll also analyze the running time. To that end, let T_n be the minimum number of steps required to solve the n-disk problem. For example, some experimentation shows that $T_1 = 1$ and $T_2 = 3$. The procedure illustrated above shows that T_3 is at most 7, though there might be a solution with fewer steps.

The recursive solution has three stages, which are described below and illustrated in Figure 10.3. For clarity, the largest disk is shaded in the figures.

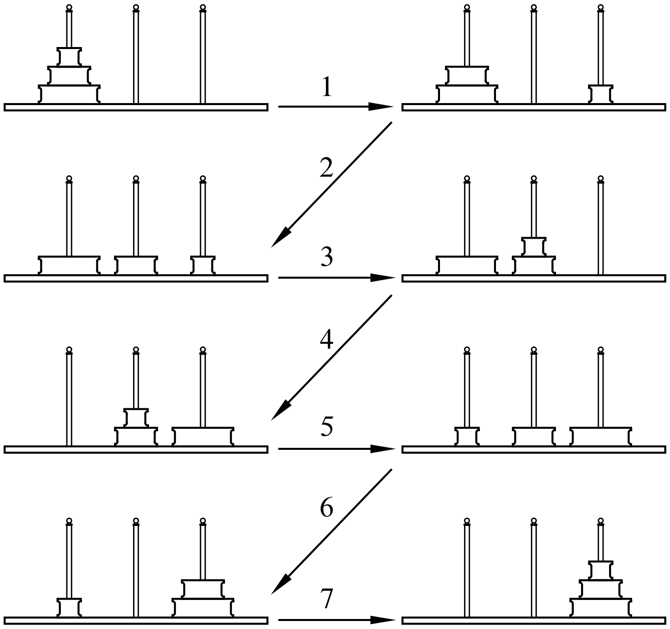


Figure 10.2 The 7-step solution to the Towers of Hanoi problem when there are $n = 3$ disks.

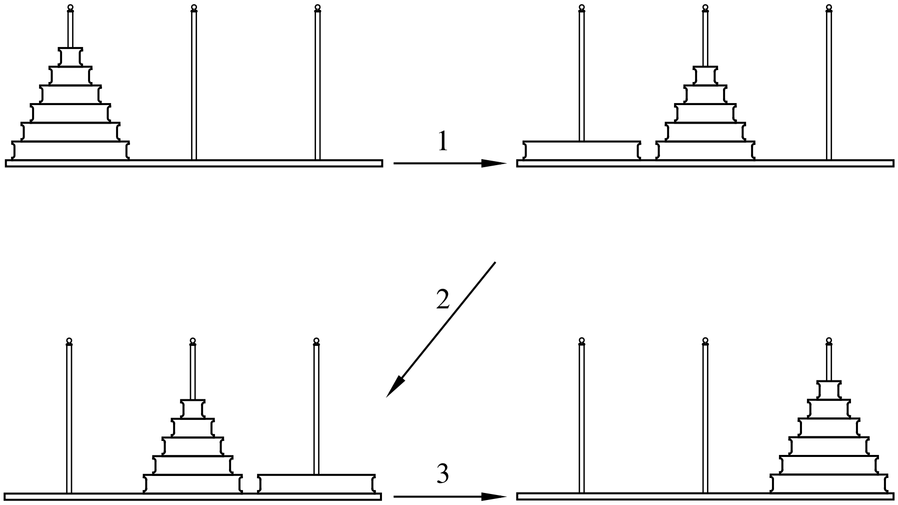


Figure 10.3 A recursive solution to the Towers of Hanoi problem.

Stage 1. Move the top $n-1$ disks from the first post to the second using the solution for $n-1$ disks. This can be done in T_{n-1} steps.

Stage 2. Move the largest disk from the first post to the third post. This takes just 1 step.

Stage 3. Move the $n-1$ disks from the second post to the third post, again using the solution for $n-1$ disks. This can also be done in T_{n-1} steps.

This algorithm shows that T_n, the minimum number of steps required to move n disks to a different post, is at most $T_{n-1} + 1 + T_{n-1} = 2T_{n-1} + 1$. We can use this fact to upper bound the number of operations required to move towers of various heights:

$$\begin{aligned} T_3 &\le 2 \cdot T_2 + 1 = 7 \\ T_4 &\le 2 \cdot T_3 + 1 \le 15 \end{aligned}$$

Continuing in this way, we could eventually compute an upper bound on T_{64}, the number of steps required to move 64 disks. So this algorithm answers our first question: given sufficient time, the monks can finish their task and end the world. This is a shame. After all that effort, they’d probably want to smack a few high-fives and go out for burgers and ice cream, but nope—world’s over.

10.1.2 Finding a Recurrence

We can not yet compute the exact number of steps that the monks need to move the 64 disks, only an upper bound. Perhaps, having pondered the problem since the beginning of time, the monks have devised a better algorithm.

In fact, there is no better algorithm, and here is why. At some step, the monks must move the largest disk from the first post to a different post. For this to happen, the $n-1$ smaller disks must all be stacked out of the way on the only remaining post. Arranging the $n-1$ smaller disks this way requires at least T_{n-1} moves. After the largest disk is moved, at least another T_{n-1} moves are required to pile the $n-1$ smaller disks on top.

This argument shows that the number of steps required is at least $2T_{n-1} + 1$. Since we gave an algorithm using exactly that number of steps, we can now write an expression for T_n, the number of moves required to complete the Towers of Hanoi problem with n disks:

$$\begin{aligned} T_1 &= 1 \\ T_n &= 2T_{n-1} + 1 \qquad \text{(for } n \ge 2\text{).} \end{aligned}$$

This is a typical recurrence. These two lines define a sequence of values, $T_1, T_2, T_3, \ldots$. The first line says that the first number in the sequence, T_1, is equal to 1. The second line defines every other number in the sequence in terms of its predecessor. So we can use this recurrence to compute any number of terms in the sequence:

$$\begin{aligned} T_1 &= 1 \\ T_2 &= 2 \cdot T_1 + 1 = 3 \\ T_3 &= 2 \cdot T_2 + 1 = 7 \\ T_4 &= 2 \cdot T_3 + 1 = 15 \\ T_5 &= 2 \cdot T_4 + 1 = 31 \\ T_6 &= 2 \cdot T_5 + 1 = 63. \end{aligned}$$

10.1.3 Solving the Recurrence

We could determine the number of steps to move a 64-disk tower by computing T_7, T_8, and so on up to T_{64}. But that would take a lot of work. It would be nice to have a closed-form expression for T_n, so that we could quickly find the number of steps required for any given number of disks. (For example, we might want to know how much sooner the world would end if the monks melted down one disk to purchase burgers and ice cream *before* the end of the world.)

There are several methods for solving recurrence equations. The simplest is to *guess* the solution and then *verify* that the guess is correct with an induction proof. As a basis for a good guess, let's look for a pattern in the values of T_n computed above: 1, 3, 7, 15, 31, 63. A natural guess is $T_n = 2^n - 1$. But whenever you guess a solution to a recurrence, you should always verify it with a proof, typically by induction. After all, your guess might be wrong. (But why bother to verify in this case? After all, if we're wrong, its not the end of the... no, let's check.)

Claim 10.1.1. $T_n = 2^n - 1$ *satisfies the recurrence:*

$$\begin{aligned} T_1 &= 1 \\ T_n &= 2T_{n-1} + 1 \qquad \textit{(for } n \geq 2\textit{)}. \end{aligned}$$

Proof. The proof is by induction on n. The induction hypothesis is that $T_n = 2^n - 1$. This is true for $n = 1$ because $T_1 = 1 = 2^1 - 1$. Now assume that $T_{n-1} = 2^{n-1} - 1$ in order to prove that $T_n = 2^n - 1$, where $n \geq 2$:

$$\begin{aligned} T_n &= 2T_{n-1} + 1 \\ &= 2(2^{n-1} - 1) + 1 \\ &= 2^n - 1. \end{aligned}$$

The first equality is the recurrence equation, the second follows from the induction assumption, and the last step is simplification. ■

Such verification proofs are especially tidy because recurrence equations and induction proofs have analogous structures. In particular, the base case relies on the first line of the recurrence, which defines T_1. And the inductive step uses the second line of the recurrence, which defines T_n as a function of preceding terms.

Our guess is verified. So we can now resolve our remaining questions about the 64-disk puzzle. Since $T_{64} = 2^{64} - 1$, the monks must complete more than 18 billion billion steps before the world ends. Better study for the final.

10.1.4 The Upper Bound Trap

When the solution to a recurrence is complicated, one might try to prove that some simpler expression is an upper bound on the solution. For example, the exact solution to the Towers of Hanoi recurrence is $T_n = 2^n - 1$. Let's try to prove the "nicer" upper bound $T_n \leq 2^n$, proceeding exactly as before.

Proof. (Failed attempt.) The proof is by induction on n. The induction hypothesis is that $T_n \leq 2^n$. This is true for $n = 1$ because $T_1 = 1 \leq 2^1$. Now assume that $T_{n-1} \leq 2^{n-1}$ in order to prove that $T_n \leq 2^n$, where $n \geq 2$:

$$\begin{aligned} T_n &= 2T_{n-1} + 1 \\ &\leq 2(2^{n-1}) + 1 \\ &\not\leq 2^n \qquad \leftarrow \text{Uh-oh!} \end{aligned}$$

The first equality is the recurrence relation, the second follows from the induction hypothesis, and the third step is a flaming train wreck. ■

The proof doesn't work! As is so often the case with induction proofs, the argument only goes through with a *stronger* hypothesis. This isn't to say that upper bounding the solution to a recurrence is hopeless, but this is a situation where induction and recurrences do not mix well.

10.1.5 Plug and Chug

Guess-and-verify is a simple and general way to solve recurrence equations. But there is one big drawback: you have to *guess right*. That was not hard for the Towers of Hanoi example. But sometimes the solution to a recurrence has a strange form that is quite difficult to guess. Practice helps, of course, but so can some other methods.

Plug-and-chug is another way to solve recurrences. This is also sometimes called "expansion" or "iteration". As in guess-and-verify, the key step is identifying a pattern. But instead of looking at a sequence of *numbers*, you have to spot a pattern in a sequence of *expressions*, which is sometimes easier. The method consists of three steps, which are described below and illustrated with the Towers of Hanoi example.

Step 1: Plug and Chug Until a Pattern Appears

The first step is to expand the recurrence equation by alternately "plugging" (applying the recurrence) and "chugging" (simplifying the result) until a pattern appears. Be careful: too much simplification can make a pattern harder to spot. The rule to remember—indeed, a rule applicable to the whole of college life—is *chug in moderation*.

$$\begin{aligned}
T_n &= 2T_{n-1} + 1 \\
&= 2(2T_{n-2} + 1) + 1 && \text{plug} \\
&= 4T_{n-2} + 2 + 1 && \text{chug} \\
&= 4(2T_{n-3} + 1) + 2 + 1 && \text{plug} \\
&= 8T_{n-3} + 4 + 2 + 1 && \text{chug} \\
&= 8(2T_{n-4} + 1) + 4 + 2 + 1 && \text{plug} \\
&= 16T_{n-4} + 8 + 4 + 2 + 1 && \text{chug}
\end{aligned}$$

Above, we started with the recurrence equation. Then we replaced T_{n-1} with $2T_{n-2} + 1$, since the recurrence says the two are equivalent. In the third step, we simplified a little—but not too much! After several similar rounds of plugging and chugging, a pattern is apparent. The following formula seems to hold:

$$\begin{aligned}
T_n &= 2^k T_{n-k} + 2^{k-1} + 2^{k-2} + \ldots + 2^2 + 2^1 + 2^0 \\
&= 2^k T_{n-k} + 2^k - 1
\end{aligned}$$

Once the pattern is clear, simplifying is safe and convenient. In particular, we've collapsed the geometric sum to a closed form on the second line.

Step 2: Verify the Pattern

The next step is to verify the general formula with one more round of plug-and-chug.

$$\begin{aligned} T_n &= 2^k T_{n-k} + 2^k - 1 \\ &= 2^k (2T_{n-(k+1)} + 1) + 2^k - 1 && \text{plug} \\ &= 2^{k+1} T_{n-(k+1)} + 2^{k+1} - 1 && \text{chug} \end{aligned}$$

The final expression on the right is the same as the expression on the first line, except that k is replaced by $k + 1$. Surprisingly, this effectively *proves* that the formula is correct for all k. Here is why: we know the formula holds for $k = 1$, because that's the original recurrence equation. And we've just shown that if the formula holds for some $k \geq 1$, then it also holds for $k + 1$. So the formula holds for all $k \geq 1$ by induction.

Step 3: Write T_n Using Early Terms with Known Values

The last step is to express T_n as a function of early terms whose values are known. Here, choosing $k = n - 1$ expresses T_n in terms of T_1, which is equal to 1. Simplifying gives a closed-form expression for T_n:

$$\begin{aligned} T_n &= 2^{n-1} T_1 + 2^{n-1} - 1 \\ &= 2^{n-1} \cdot 1 + 2^{n-1} - 1 \\ &= 2^n - 1. \end{aligned}$$

We're done! This is the same answer we got from guess-and-verify.

Let's compare guess-and-verify with plug-and-chug. In the guess-and-verify method, we computed several terms at the beginning of the sequence, T_1, T_2, T_3, etc., until a pattern appeared. We generalized to a formula for the nth term, T_n. In contrast, plug-and-chug works backward from the nth term. Specifically, we started with an expression for T_n involving the preceding term, T_{n-1}, and rewrote this using progressively earlier terms, T_{n-2}, T_{n-3}, etc. Eventually, we noticed a pattern, which allowed us to express T_n using the very first term, T_1, whose value we knew. Substituting this value gave a closed-form expression for T_n. So guess-and-verify and plug-and-chug tackle the problem from opposite directions.

10.2 Merge Sort

Algorithms textbooks traditionally claim that sorting is an important, fundamental problem in computer science. Then they smack you with sorting algorithms until life as a disk-stacking monk in Hanoi sounds delightful. Here, we'll cover just *one* well-known sorting algorithm, Merge Sort. The analysis introduces another kind of recurrence.

Here is how Merge Sort works. The input is a list of n numbers, and the output is those same numbers in nondecreasing order. There are two cases:

- If the input is a single number, then the algorithm does nothing, because the list is already sorted.
- Otherwise, the list contains two or more numbers. The first half and the second half of the list are each sorted recursively. Then the two halves are merged to form a sorted list with all n numbers.

Let's work through an example. Suppose we want to sort this list:

$$10, 7, 23, 5, 2, 8, 6, 9.$$

Since there is more than one number, the first half (10, 7, 23, 5) and the second half (2, 8, 6, 9) are sorted recursively. The results are 5, 7, 10, 23 and 2, 6, 8, 9. All that remains is to merge these two lists. This is done by repeatedly emitting the smaller of the two leading terms. When one list is empty, the whole other list is emitted. The example is worked out below. In this table, underlined numbers are about to be emitted.

First Half	Second Half	Output
5, 7, 10, 23	$\underline{2}$, 6, 8, 9	
$\underline{5}$, 7, 10, 23	6, 8, 9	2
7, 10, 23	$\underline{6}$, 8, 9	2, 5
$\underline{7}$, 10, 23	8, 9	2, 5, 6
10, 23	$\underline{8}$, 9	2, 5, 6, 7
10, 23	$\underline{9}$	2, 5, 6, 7, 8
$\underline{10}$, $\underline{23}$		2, 5, 6, 7, 8, 9
		2, 5, 6, 7, 8, 9, 10, 23

The leading terms are initially 5 and 2. So we output 2. Then the leading terms are 5 and 6, so we output 5. Eventually, the second list becomes empty. At that point, we output the whole first list, which consists of 10 and 23. The complete output consists of all the numbers in sorted order.

10.2.1 Finding a Recurrence

A traditional question about sorting algorithms is, “What is the maximum number of comparisons used in sorting n items?” This is taken as an estimate of the running time. In the case of Merge Sort, we can express this quantity with a recurrence. Let T_n be the maximum number of comparisons used while Merge Sorting a list of n numbers. For now, assume that n is a power of 2. This ensures that the input can be divided in half at every stage of the recursion.

- If there is only one number in the list, then no comparisons are required, so $T_1 = 0$.
- Otherwise, T_n includes comparisons used in sorting the first half (at most $T_{n/2}$), in sorting the second half (also at most $T_{n/2}$), and in merging the two halves. The number of comparisons in the merging step is at most $n - 1$. This is because at least one number is emitted after each comparison and one more number is emitted at the end when one list becomes empty. Since n items are emitted in all, there can be at most $n - 1$ comparisons.

Therefore, the maximum number of comparisons needed to Merge Sort n items is given by this recurrence:

$$\begin{aligned} T_1 &= 0 \\ T_n &= 2T_{n/2} + n - 1 \qquad \text{(for } n \geq 2 \text{ and a power of 2).} \end{aligned}$$

This fully describes the number of comparisons, but not in a very useful way; a closed-form expression would be much more helpful. To get that, we have to solve the recurrence.

10.2.2 Solving the Recurrence

Let’s first try to solve the Merge Sort recurrence with the guess-and-verify technique. Here are the first few values:

$$\begin{aligned} T_1 &= 0 \\ T_2 &= 2T_1 + 2 - 1 = 1 \\ T_4 &= 2T_2 + 4 - 1 = 5 \\ T_8 &= 2T_4 + 8 - 1 = 17 \\ T_{16} &= 2T_8 + 16 - 1 = 49. \end{aligned}$$

We’re in trouble! Guessing the solution to this recurrence is hard because there is no obvious pattern. So let’s try the plug-and-chug method instead.

Step 1: Plug and Chug Until a Pattern Appears

First, we expand the recurrence equation by alternately plugging and chugging until a pattern appears.

$$\begin{aligned}
T_n &= 2T_{n/2} + n - 1 & \\
&= 2(2T_{n/4} + n/2 - 1) + (n - 1) & \text{plug} \\
&= 4T_{n/4} + (n - 2) + (n - 1) & \text{chug} \\
&= 4(2T_{n/8} + n/4 - 1) + (n - 2) + (n - 1) & \text{plug} \\
&= 8T_{n/8} + (n - 4) + (n - 2) + (n - 1) & \text{chug} \\
&= 8(2T_{n/16} + n/8 - 1) + (n - 4) + (n - 2) + (n - 1) & \text{plug} \\
&= 16T_{n/16} + (n - 8) + (n - 4) + (n - 2) + (n - 1) & \text{chug}
\end{aligned}$$

A pattern is emerging. In particular, this formula seems holds:

$$\begin{aligned}
T_n &= 2^k T_{n/2^k} + (n - 2^{k-1}) + (n - 2^{k-2}) + \ldots + (n - 2^0) \\
&= 2^k T_{n/2^k} + kn - 2^{k-1} - 2^{k-2} \ldots - 2^0 \\
&= 2^k T_{n/2^k} + kn - 2^k + 1.
\end{aligned}$$

On the second line, we grouped the n terms and powers of 2. On the third, we collapsed the geometric sum.

Step 2: Verify the Pattern

Next, we verify the pattern with one additional round of plug-and-chug. If we guessed the wrong pattern, then this is where we'll discover the mistake.

$$\begin{aligned}
T_n &= 2^k T_{n/2^k} + kn - 2^k + 1 & \\
&= 2^k (2T_{n/2^{k+1}} + n/2^k - 1) + kn - 2^k + 1 & \text{plug} \\
&= 2^{k+1} T_{n/2^{k+1}} + (k + 1)n - 2^{k+1} + 1 & \text{chug}
\end{aligned}$$

The formula is unchanged except that k is replaced by $k + 1$. This amounts to the induction step in a proof that the formula holds for all $k \geq 1$.

Step 3: Write T_n Using Early Terms with Known Values

Finally, we express T_n using early terms whose values are known. Specifically, if we let $k = \log n$, then $T_{n/2^k} = T_1$, which we know is 0:

$$\begin{aligned} T_n &= 2^k T_{n/2^k} + kn - 2^k + 1 \\ &= 2^{\log n} T_{n/2^{\log n}} + n \log n - 2^{\log n} + 1 \\ &= nT_1 + n \log n - n + 1 \\ &= n \log n - n + 1. \end{aligned}$$

We're done! We have a closed-form expression for the maximum number of comparisons used in Merge Sorting a list of n numbers. In retrospect, it is easy to see why guess-and-verify failed: this formula is fairly complicated.

As a check, we can confirm that this formula gives the same values that we computed earlier:

n	T_n	$n \log n - n + 1$
1	0	$1 \log 1 - 1 + 1 = 0$
2	1	$2 \log 2 - 2 + 1 = 1$
4	5	$4 \log 4 - 4 + 1 = 5$
8	17	$8 \log 8 - 8 + 1 = 17$
16	49	$16 \log 16 - 16 + 1 = 49$

As a double-check, we could write out an explicit induction proof. This would be straightforward, because we already worked out the guts of the proof in step 2 of the plug-and-chug procedure.

10.3 Linear Recurrences

So far we've solved recurrences with two techniques: guess-and-verify and plug-and-chug. These methods require spotting a pattern in a sequence of numbers or expressions. In this section and the next, we'll give cookbook solutions for two large classes of recurrences. These methods require no flash of insight; you just follow the recipe and get the answer.

10.3.1 Climbing Stairs

How many different ways are there to climb n stairs, if you can either step up one stair or hop up two? For example, there are five different ways to climb four stairs:

1. step, step, step, step

2. hop, hop
3. hop, step, step
4. step, hop step
5. step, step, hop

Working through this problem will demonstrate the major features of our first cookbook method for solving recurrences. We'll fill in the details of the general solution afterward.

Finding a Recurrence

As special cases, there is 1 way to climb 0 stairs (do nothing) and 1 way to climb 1 stair (step up). In general, an ascent of n stairs consists of either a step followed by an ascent of the remaining $n - 1$ stairs or a hop followed by an ascent of $n - 2$ stairs. So the total number of ways to climb n stairs is equal to the number of ways to climb $n - 1$ plus the number of ways to climb $n - 2$. These observations define a recurrence:

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

Here, $f(n)$ denotes the number of ways to climb n stairs. Also, we've switched from subscript notation to functional notation, from T_n to f_n. Here the change is cosmetic, but the expressiveness of functions will be useful later.

This is the Fibonacci recurrence, the most famous of all recurrence equations. Fibonacci numbers arise in all sorts of applications and in nature. Fibonacci introduced the numbers in 1202 to study rabbit reproduction. Fibonacci numbers also appear, oddly enough, in the spiral patterns on the faces of sunflowers. And the input numbers that make Euclid's GCD algorithm require the greatest number of steps are consecutive Fibonacci numbers.

Solving the Recurrence

The Fibonacci recurrence belongs to the class of linear recurrences, which are essentially all solvable with a technique that you can learn in an hour. This is somewhat amazing, since the Fibonacci recurrence remained unsolved for almost six centuries!

In general, a *homogeneous linear recurrence* has the form

$$f(n) = a_1 f(n-1) + a_2 f(n-2) + \ldots + a_d f(n-d)$$

where $a_1, a_2, \ldots, a_d$ and d are constants. The *order* of the recurrence is d. Commonly, the value of the function f is also specified at a few points; these are called *boundary conditions*. For example, the Fibonacci recurrence has order $d = 2$ with coefficients $a_1 = a_2 = 1$ and $g(n) = 0$. The boundary conditions are $f(0) = 1$ and $f(1) = 1$. The word "homogeneous" sounds scary, but effectively means "the simpler kind". We'll consider linear recurrences with a more complicated form later.

Let's try to solve the Fibonacci recurrence with the benefit centuries of hindsight. In general, linear recurrences tend to have exponential solutions. So let's guess that

$$f(n) = x^n$$

where x is a parameter introduced to improve our odds of making a correct guess. We'll figure out the best value for x later. To further improve our odds, let's neglect the boundary conditions, $f(0) = 0$ and $f(1) = 1$, for now. Plugging this guess into the recurrence $f(n) = f(n-1) + f(n-2)$ gives

$$x^n = x^{n-1} + x^{n-2}.$$

Dividing both sides by x^{n-2} leaves a quadratic equation:

$$x^2 = x + 1.$$

Solving this equation gives *two* plausible values for the parameter x:

$$x = \frac{1 \pm \sqrt{5}}{2}.$$

This suggests that there are at least two different solutions to the recurrence, neglecting the boundary conditions.

$$f(n) = \left(\frac{1+\sqrt{5}}{2}\right)^n \quad \text{or} \quad f(n) = \left(\frac{1-\sqrt{5}}{2}\right)^n$$

A charming features of homogeneous linear recurrences is that any linear combination of solutions is another solution.

Theorem 10.3.1. *If $f(n)$ and $g(n)$ are both solutions to a homogeneous linear recurrence, then $h(n) = sf(n) + tg(n)$ is also a solution for all $s, t \in \mathbb{R}$.*

Proof.

$$\begin{aligned}
h(n) &= sf(n) + tg(n) \\
&= s\,(a_1 f(n-1) + \ldots + a_d f(n-d)) + t\,(a_1 g(n-1) + \ldots + a_d g(n-d)) \\
&= a_1(sf(n-1) + tg(n-1)) + \ldots + a_d(sf(n-d) + tg(n-d)) \\
&= a_1 h(n-1) + \ldots + a_d h(n-d)
\end{aligned}$$

The first step uses the definition of the function h, and the second uses the fact that f and g are solutions to the recurrence. In the last two steps, we rearrange terms and use the definition of h again. Since the first expression is equal to the last, h is also a solution to the recurrence. ■

The phenomenon described in this theorem—a linear combination of solutions is another solution—also holds for many differential equations and physical systems. In fact, linear recurrences are so similar to linear differential equations that you can safely snooze through that topic in some future math class.

Returning to the Fibonacci recurrence, this theorem implies that

$$f(n) = s\left(\frac{1+\sqrt{5}}{2}\right)^n + t\left(\frac{1-\sqrt{5}}{2}\right)^n$$

is a solution for all real numbers s and t. The theorem expanded two solutions to a whole spectrum of possibilities! Now, given all these options to choose from, we can find one solution that satisfies the boundary conditions, $f(0) = 1$ and $f(1) = 1$. Each boundary condition puts some constraints on the parameters s and t. In particular, the first boundary condition implies that

$$f(0) = s\left(\frac{1+\sqrt{5}}{2}\right)^0 + t\left(\frac{1-\sqrt{5}}{2}\right)^0 = s + t = 1.$$

Similarly, the second boundary condition implies that

$$f(1) = s\left(\frac{1+\sqrt{5}}{2}\right)^1 + t\left(\frac{1-\sqrt{5}}{2}\right)^1 = 1.$$

Now we have two linear equations in two unknowns. The system is not degenerate, so there is a unique solution:

$$s = \frac{1}{\sqrt{5}} \cdot \frac{1+\sqrt{5}}{2} \qquad t = -\frac{1}{\sqrt{5}} \cdot \frac{1-\sqrt{5}}{2}.$$

These values of s and t identify a solution to the Fibonacci recurrence that also satisfies the boundary conditions:

$$\begin{aligned} f(n) &= \frac{1}{\sqrt{5}} \cdot \frac{1+\sqrt{5}}{2}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}} \cdot \frac{1-\sqrt{5}}{2}\left(\frac{1-\sqrt{5}}{2}\right)^n \\ &= \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^{n+1} - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^{n+1}. \end{aligned}$$

It is easy to see why no one stumbled across this solution for almost six centuries! All Fibonacci numbers are integers, but this expression is full of square roots of five! Amazingly, the square roots always cancel out. This expression really does give the Fibonacci numbers if we plug in $n = 0, 1, 2$, etc.

This closed-form for Fibonacci numbers has some interesting corollaries. The first term tends to infinity because the base of the exponential, $(1 + \sqrt{5})/2 = 1.618\ldots$ is greater than one. This value is often denoted ϕ and called the “golden ratio”. The second term tends to zero, because $(1 - \sqrt{5})/2 = -0.618033988\ldots$ has absolute value less than 1. This implies that the nth Fibonacci number is:

$$f(n) = \frac{\phi^{n+1}}{\sqrt{5}} + o(1).$$

Remarkably, this expression involving irrational numbers is actually very close to an integer for all large n—namely, a Fibonacci number! For example:

$$\frac{\phi^{20}}{\sqrt{5}} = 6765.000029\ldots \approx f(19).$$

This also implies that the ratio of consecutive Fibonacci numbers rapidly approaches the golden ratio. For example:

$$\frac{f(20)}{f(19)} = \frac{10946}{6765} = 1.618033998\ldots.$$

10.3.2 Solving Homogeneous Linear Recurrences

The method we used to solve the Fibonacci recurrence can be extended to solve any homogeneous linear recurrence; that is, a recurrence of the form

$$f(n) = a_1 f(n-1) + a_2 f(n-2) + \ldots + a_d f(n-d)$$

where $a_1, a_2, \ldots, a_d$ and d are constants. Substituting the guess $f(n) = x^n$, as with the Fibonacci recurrence, gives

$$x^n = a_1 x^{n-1} + a_2 x^{n-2} + \ldots + a_d x^{n-d}.$$

Dividing by x^{n-d} gives

$$x^d = a_1 x^{d-1} + a_2 x^{d-2} + \ldots + a_{d-1} x + a_d.$$

This is called the *characteristic equation* of the recurrence. The characteristic equation can be read off quickly since the coefficients of the equation are the same as the coefficients of the recurrence.

The solutions to a linear recurrence are defined by the roots of the characteristic equation. Neglecting boundary conditions for the moment:

- If r is a nonrepeated root of the characteristic equation, then r^n is a solution to the recurrence.
- If r is a repeated root with multiplicity k then r^n, nr^n, n^2r^n, …, $n^{k-1}r^n$ are all solutions to the recurrence.

Theorem 10.3.1 implies that every linear combination of these solutions is also a solution.

For example, suppose that the characteristic equation of a recurrence has roots s, t, and u twice. These four roots imply four distinct solutions:

$$f(n) = s^n \qquad f(n) = t^n \qquad f(n) = u^n \qquad f(n) = nu^n.$$

Furthermore, every linear combination

$$f(n) = a \cdot s^n + b \cdot t^n + c \cdot u^n + d \cdot nu^n \tag{10.1}$$

is also a solution.

All that remains is to select a solution consistent with the boundary conditions by choosing the constants appropriately. Each boundary condition implies a linear equation involving these constants. So we can determine the constants by solving a system of linear equations. For example, suppose our boundary conditions were $f(0) = 0$, $f(1) = 1$, $f(2) = 4$, and $f(3) = 9$. Then we would obtain four equations in four unknowns:

$$\begin{array}{lcl}
f(0) = 0 & \Rightarrow & a \cdot s^0 + b \cdot t^0 + c \cdot u^0 + d \cdot 0u^0 = 0 \\
f(1) = 1 & \Rightarrow & a \cdot s^1 + b \cdot t^1 + c \cdot u^1 + d \cdot 1u^1 = 1 \\
f(2) = 4 & \Rightarrow & a \cdot s^2 + b \cdot t^2 + c \cdot u^2 + d \cdot 2u^2 = 4 \\
f(3) = 9 & \Rightarrow & a \cdot s^3 + b \cdot t^3 + c \cdot u^3 + d \cdot 3u^3 = 9
\end{array}$$

This looks nasty, but remember that s, t, and u are just constants. Solving this system gives values for a, b, c, and d that define a solution to the recurrence consistent with the boundary conditions.

10.3.3 Solving General Linear Recurrences

We can now solve all linear homogeneous recurrences, which have the form

$$f(n) = a_1 f(n-1) + a_2 f(n-2) + \ldots + a_d f(n-d).$$

Many recurrences that arise in practice do not quite fit this mold. For example, the Towers of Hanoi problem led to this recurrence:

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

The problem is the extra $+1$; that is not allowed in a homogeneous linear recurrence. In general, adding an extra function $g(n)$ to the right side of a linear recurrence gives an *inhomogeneous linear recurrence*:

$$f(n) = a_1 f(n-1) + a_2 f(n-2) + \ldots + a_d f(n-d) + g(n).$$

Solving inhomogeneous linear recurrences is neither very different nor very difficult. We can divide the whole job into five steps:

1. Replace $g(n)$ by 0, leaving a homogeneous recurrence. As before, find roots of the characteristic equation.

2. Write down the solution to the homogeneous recurrence, but do not yet use the boundary conditions to determine coefficients. This is called the *homogeneous solution*.

3. Now restore $g(n)$ and find a single solution to the recurrence, ignoring boundary conditions. This is called a *particular solution*. We'll explain how to find a particular solution shortly.

4. Add the homogeneous and particular solutions together to obtain the *general solution*.

5. Now use the boundary conditions to determine constants by the usual method of generating and solving a system of linear equations.

As an example, let's consider a variation of the Towers of Hanoi problem. Suppose that moving a disk takes time proportional to its size. Specifically, moving the smallest disk takes 1 second, the next-smallest takes 2 seconds, and moving the nth disk then requires n seconds instead of 1. So, in this variation, the time to complete the job is given by a recurrence with a $+n$ term instead of a $+1$:

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

Clearly, this will take longer, but how much longer? Let's solve the recurrence with the method described above.

In Steps 1 and 2, dropping the $+n$ leaves the homogeneous recurrence $f(n) = 2f(n-1)$. The characteristic equation is $x = 2$. So the homogeneous solution is $f(n) = c2^n$.

In Step 3, we must find a solution to the full recurrence $f(n) = 2f(n-1) + n$, without regard to the boundary condition. Let's guess that there is a solution of the form $f(n) = an + b$ for some constants a and b. Substituting this guess into the recurrence gives

$$\begin{aligned} an + b &= 2(a(n-1) + b) + n \\ 0 &= (a+1)n + (b - 2a). \end{aligned}$$

The second equation is a simplification of the first. The second equation holds for all n if both $a + 1 = 0$ (which implies $a = -1$) and $b - 2a = 0$ (which implies that $b = -2$). So $f(n) = an + b = -n - 2$ is a particular solution.

In the Step 4, we add the homogeneous and particular solutions to obtain the general solution

$$f(n) = c2^n - n - 2.$$

Finally, in step 5, we use the boundary condition, $f(1) = 1$, determine the value of the constant c:

$$\begin{aligned} f(1) = 1 \quad &\Rightarrow \quad c2^1 - 1 - 2 = 1 \\ &\Rightarrow \quad c = 2. \end{aligned}$$

Therefore, the function $f(n) = 2 \cdot 2^n - n - 2$ solves this variant of the Towers of Hanoi recurrence. For comparison, the solution to the original Towers of Hanoi problem was $2^n - 1$. So if moving disks takes time proportional to their size, then the monks will need about twice as much time to solve the whole puzzle.

10.3.4 How to Guess a Particular Solution

Finding a particular solution can be the hardest part of solving inhomogeneous recurrences. This involves guessing, and you might guess wrong.[1] However, some rules of thumb make this job fairly easy most of the time.

- Generally, look for a particular solution with the same form as the inhomogeneous term $g(n)$.

- If $g(n)$ is a constant, then guess a particular solution $f(n) = c$. If this doesn't work, try polynomials of progressively higher degree: $f(n) = bn + c$, then $f(n) = an^2 + bn + c$, etc.

- More generally, if $g(n)$ is a polynomial, try a polynomial of the same degree, then a polynomial of degree one higher, then two higher, etc. For example, if $g(n) = 6n + 5$, then try $f(n) = bn + c$ and then $f(n) = an^2 + bn + c$.

[1] In Chapter 12, we will show how to solve linear recurrences with generating functions—it's a little more complicated, but it does not require guessing.

- If $g(n)$ is an exponential, such as 3^n, then first guess that $f(n) = c3^n$. Failing that, try $f(n) = bn3^n + c3^n$ and then $an^2 3^n + bn3^n + c3^n$, etc.

The entire process is summarized on the following page.

10.4 Divide-and-Conquer Recurrences

We now have a recipe for solving general linear recurrences. But the Merge Sort recurrence, which we encountered earlier, is not linear:

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

In particular, $T(n)$ is not a linear combination of a fixed number of immediately preceding terms; rather, $T(n)$ is a function of $T(n/2)$, a term halfway back in the sequence.

Merge Sort is an example of a divide-and-conquer algorithm: it divides the input, "conquers" the pieces, and combines the results. Analysis of such algorithms commonly leads to *divide-and-conquer* recurrences, which have this form:

$$T(n) = \sum_{i=1}^{k} a_i T(b_i n) + g(n)$$

Here $a_1, \ldots a_k$ are positive constants, $b_1, \ldots, b_k$ are constants between 0 and 1, and $g(n)$ is a nonnegative function. For example, setting $a_1 = 2$, $b_1 = 1/2$, and $g(n) = n - 1$ gives the Merge Sort recurrence.

10.4.1 The Akra-Bazzi Formula

The solution to virtually all divide and conquer solutions is given by the amazing *Akra-Bazzi formula*. Quite simply, the asymptotic solution to the general divide-and-conquer recurrence

$$T(n) = \sum_{i=1}^{k} a_i T(b_i n) + g(n)$$

is

$$T(n) = \Theta\left(n^p \left(1 + \int_1^n \frac{g(u)}{u^{p+1}}\, du\right)\right) \tag{10.2}$$

Short Guide to Solving Linear Recurrences

A linear recurrence is an equation

$$f(n) = \underbrace{a_1 f(n-1) + a_2 f(n-2) + \ldots + a_d f(n-d)}_{\text{homogeneous part}} \quad \underbrace{+\, g(n)}_{\text{inhomogeneous part}}$$

together with boundary conditions such as $f(0) = b_0$, $f(1) = b_1$, etc. Linear recurrences are solved as follows:

1. Find the roots of the characteristic equation
$$x^n = a_1 x^{n-1} + a_2 x^{n-2} + \ldots + a_{k-1} x + a_k.$$

2. Write down the homogeneous solution. Each root generates one term and the homogeneous solution is their sum. A nonrepeated root r generates the term cr^n, where c is a constant to be determined later. A root r with multiplicity k generates the terms
$$d_1 r^n \qquad d_2 n r^n \qquad d_3 n^2 r^n \qquad \ldots \qquad d_k n^{k-1} r^n$$
where $d_1, \ldots d_k$ are constants to be determined later.

3. Find a particular solution. This is a solution to the full recurrence that need not be consistent with the boundary conditions. Use guess-and-verify. If $g(n)$ is a constant or a polynomial, try a polynomial of the same degree, then of one higher degree, then two higher. For example, if $g(n) = n$, then try $f(n) = bn + c$ and then $an^2 + bn + c$. If $g(n)$ is an exponential, such as 3^n, then first guess $f(n) = c3^n$. Failing that, try $f(n) = (bn + c)3^n$ and then $(an^2 + bn + c)3^n$, etc.

4. Form the general solution, which is the sum of the homogeneous solution and the particular solution. Here is a typical general solution:
$$f(n) = \underbrace{c2^n + d(-1)^n}_{\text{homogeneous solution}} + \underbrace{3n + 1}_{\text{inhomogeneous solution}}.$$

5. Substitute the boundary conditions into the general solution. Each boundary condition gives a linear equation in the unknown constants. For example, substituting $f(1) = 2$ into the general solution above gives
$$\begin{aligned} 2 &= c \cdot 2^1 + d \cdot (-1)^1 + 3 \cdot 1 + 1 \\ \Rightarrow \quad -2 &= 2c - d. \end{aligned}$$
Determine the values of these constants by solving the resulting system of linear equations.

where p satisfies

$$\sum_{i=1}^{k} a_i b_i^p = 1. \tag{10.3}$$

A rarely-troublesome requirement is that the function $g(n)$ must not grow or oscillate too quickly. Specifically, $|g'(n)|$ must be bounded by some polynomial. So, for example, the Akra-Bazzi formula is valid when $g(n) = x^2 \log n$, but not when $g(n) = 2^n$.

Let's solve the Merge Sort recurrence again, using the Akra-Bazzi formula instead of plug-and-chug. First, we find the value p that satisfies

$$2 \cdot (1/2)^p = 1.$$

Looks like $p = 1$ does the job. Then we compute the integral:

$$\begin{aligned}
T(n) &= \Theta\left(n\left(1 + \int_1^n \frac{u-1}{u^2}\,du\right)\right) \\
&= \Theta\left(n\left(1 + \left(\log u + \frac{1}{u}\right)_1^n\right)\right) \\
&= \Theta\left(n\left(\log n + \frac{1}{n}\right)\right) \\
&= \Theta\left(n \log n\right).
\end{aligned}$$

The first step is integration and the second is simplification. We can drop the $1/n$ term in the last step, because the $\log n$ term dominates. We're done!

Let's try a scary-looking recurrence:

$$T(n) = 2T(n/2) + 8/9T(3n/4) + n^2.$$

Here, $a_1 = 2$, $b_1 = 1/2$, $a_2 = 8/9$, and $b_2 = 3/4$. So we find the value p that satisfies

$$2 \cdot (1/2)^p + (8/9)(3/4)^p = 1.$$

Equations of this form don't always have closed-form solutions, so you may need to approximate p numerically sometimes. But in this case the solution is simple: $p = 2$. Then we integrate:

$$\begin{aligned}
T(n) &= \Theta\left(n^2\left(1 + \int_1^n \frac{u^2}{u^3}\,du\right)\right) \\
&= \Theta\left(n^2(1 + \log n)\right) \\
&= \Theta\left(n^2 \log n\right).
\end{aligned}$$

That was easy!

10.4.2 Two Technical Issues

Until now, we've swept a couple issues related to divide-and-conquer recurrences under the rug. Let's address those issues now.

First, the Akra-Bazzi formula makes no use of boundary conditions. To see why, let's go back to Merge Sort. During the plug-and-chug analysis, we found that

$$T_n = nT_1 + n\log n - n + 1.$$

This expresses the nth term as a function of the first term, whose value is specified in a boundary condition. But notice that $T_n = \Theta(n\log n)$ for *every* value of T_1. The boundary condition doesn't matter!

This is the typical situation: *the asymptotic solution to a divide-and-conquer recurrence is independent of the boundary conditions*. Intuitively, if the bottom-level operation in a recursive algorithm takes, say, twice as long, then the overall running time will at most double. This matters in practice, but the factor of 2 is concealed by asymptotic notation. There are corner-case exceptions. For example, the solution to $T(n) = 2T(n/2)$ is either $\Theta(n)$ or zero, depending on whether $T(1)$ is zero. These cases are of little practical interest, so we won't consider them further.

There is a second nagging issue with divide-and-conquer recurrences that does not arise with linear recurrences. Specifically, dividing a problem of size n may create subproblems of non-integer size. For example, the Merge Sort recurrence contains the term $T(n/2)$. So what if n is 15? How long does it take to sort seven-and-a-half items? Previously, we dodged this issue by analyzing Merge Sort only when the size of the input was a power of 2. But then we don't know what happens for an input of size, say, 100.

Of course, a practical implementation of Merge Sort would split the input *approximately* in half, sort the halves recursively, and merge the results. For example, a list of 15 numbers would be split into lists of 7 and 8. More generally, a list of n numbers would be split into approximate halves of size $\lceil n/2 \rceil$ and $\lfloor n/2 \rfloor$. So the maximum number of comparisons is actually given by this recurrence:

$$\begin{aligned} T(1) &= 0 \\ T(n) &= T\left(\lceil n/2 \rceil\right) + T\left(\lfloor n/2 \rfloor\right) + n - 1 \qquad \text{(for } n \geq 2\text{).} \end{aligned}$$

This may be rigorously correct, but the ceiling and floor operations make the recurrence hard to solve exactly.

Fortunately, *the asymptotic solution to a divide and conquer recurrence is unaffected by floors and ceilings*. More precisely, the solution is not changed by replacing a term $T(b_i n)$ with either $T(\lceil b_i n \rceil)$ or $T(\lfloor b_i n \rfloor)$. So leaving floors and ceilings out of divide-and-conquer recurrences makes sense in many contexts; those are complications that make no difference.

10.4.3 The Akra-Bazzi Theorem

The Akra-Bazzi formula together with our assertions about boundary conditions and integrality all follow from the *Akra-Bazzi Theorem*, which is stated below.

Theorem 10.4.1 (Akra-Bazzi)**.** *Suppose that the function $T : \mathbb{R} \to \mathbb{R}$ satisfies the recurrence*

$$T(x) = \begin{cases} \text{is nonnegative and bounded} & \text{for } 0 \le x \le x_0 \\ \sum\limits_{i=1}^{k} a_i T(b_i x + h_i(x)) + g(x) & \text{for } x > x_0 \end{cases}$$

where:

1. *$a_1, \ldots, a_k$ are positive constants.*
2. *$b_1, \ldots, b_k$ are constants between 0 and 1.*
3. *x_0 is large enough so that T is well-defined.*
4. *$g(x)$ is a nonnegative function such that $|g'(x)|$ is bounded by a polynomial.*
5. *$|h_i(x)| = O(x/\log^2 x)$.*

Then

$$T(x) = \Theta\left(x^p\left(1 + \int_1^x \frac{g(u)}{u^{p+1}}\,du\right)\right)$$

where p satisfies

$$\sum_{i=1}^{k} a_i b_i^p = 1.$$

The Akra-Bazzi theorem can be proved using a complicated induction argument, though we won't do that here. But let's at least go over the statement of the theorem.

All the recurrences we've considered were defined over the integers, and that is the common case. But the Akra-Bazzi theorem applies more generally to functions defined over the real numbers.

The Akra-Bazzi formula is lifted directed from the theorem statement, except that the recurrence in the theorem includes extra functions, h_i. These functions extend the theorem to address floors, ceilings, and other small adjustments to the sizes of subproblems. The trick is illustrated by this combination of parameters

$$
\begin{aligned}
a_1 &= 1 & b_1 &= 1/2 & h_1(x) &= \left\lceil \frac{x}{2} \right\rceil - \frac{x}{2} \\
a_2 &= 1 & b_2 &= 1/2 & h_2(x) &= \left\lfloor \frac{x}{2} \right\rfloor - \frac{x}{2} \\
& & g(x) &= x - 1 & &
\end{aligned}
$$

which corresponds the recurrence

$$
\begin{aligned}
T(x) &= 1 \cdot T\left(\frac{x}{2} + \left(\left\lceil \frac{x}{2} \right\rceil - \frac{x}{2}\right)\right) + \cdot T\left(\frac{x}{2} + \left(\left\lfloor \frac{x}{2} \right\rfloor - \frac{x}{2}\right)\right) + x - 1 \\
&= T\left(\left\lceil \frac{x}{2} \right\rceil\right) + T\left(\left\lfloor \frac{x}{2} \right\rfloor\right) + x - 1.
\end{aligned}
$$

This is the rigorously correct Merge Sort recurrence valid for all input sizes, complete with floor and ceiling operators. In this case, the functions $h_1(x)$ and $h_2(x)$ are both at most 1, which is easily $O(x/\log^2 x)$ as required by the theorem statement. These functions h_i do not affect—or even appear in—the asymptotic solution to the recurrence. This justifies our earlier claim that applying floor and ceiling operators to the size of a subproblem does not alter the asymptotic solution to a divide-and-conquer recurrence.

10.4.4 The Master Theorem

There is a special case of the Akra-Bazzi formula known as the Master Theorem that handles some of the recurrences that commonly arise in computer science. It is called the *Master* Theorem because it was proved long before Akra and Bazzi arrived on the scene and, for many years, it was the final word on solving divide-and-conquer recurrences. We include the Master Theorem here because it is still widely referenced in algorithms courses and you can use it without having to know anything about integration.

Theorem 10.4.2 (Master Theorem)**.** *Let T be a recurrence of the form*

$$T(n) = aT\left(\frac{n}{b}\right) + g(n).$$

Case 1: *If $g(n) = O\left(n^{\log_b(a) - \epsilon}\right)$ for some constant $\epsilon > 0$, then*

$$T(n) = \Theta\left(n^{\log_b(a)}\right).$$

Case 2: *If* $g(n) = \Theta\left(n^{\log_b(a)} \log^k(n)\right)$ *for some constant* $k \geq 0$*, then*

$$T(n) = \Theta\left(n^{\log_b(a)} \log^{k+1}(n)\right).$$

Case 3: *If* $g(n) = \Omega\left(n^{\log_b(a)+\epsilon}\right)$ *for some constant* $\epsilon > 0$ *and* $ag(n/b) < cg(n)$ *for some constant* $c < 1$ *and sufficiently large* n*, then*

$$T(n) = \Theta(g(n)).$$

The Master Theorem can be proved by induction on n or, more easily, as a corollary of Theorem 10.4.1. We will not include the details here.

10.4.5 Pitfalls with Asymptotic Notation and Induction

We've seen that asymptotic notation is quite useful, particularly in connection with recurrences. And induction is our favorite proof technique. But mixing the two is risky business; there is great potential for subtle errors and false conclusions!

False Claim. *If*

$$\begin{aligned} T(1) &= 1 \quad \textit{and} \\ T(n) &= 2T(n/2) + n, \end{aligned}$$

then $T(n) = O(n)$.

The Akra-Bazzi theorem implies that the correct solution is $T(n) = \Theta(n \log n)$ and so this claim is false. But where does the following "proof" go astray?

Bogus proof. The proof is by strong induction. Let $P(n)$ be the proposition that $T(n) = O(n)$.

Base case: $P(1)$ is true because $T(1) = 1 = O(1)$.

Inductive step: For $n \geq 2$, assume $P(1)$, $P(2)$, …, $P(n-1)$ to prove $P(n)$. We have

$$\begin{aligned} T(n) &= 2 \cdot T(n/2) + n \\ &= 2 \cdot O(n/2) + n \\ &= O(n). \end{aligned}$$

The first equation is the recurrence, the second uses the assumption $P(n/2)$, and the third is a simplification. ■

Where's the bug? The proof is already far off the mark in the second sentence, which defines the induction hypothesis. The statement "$T(n) = O(n)$" is either true or false; it's validity does not depend on a particular value of n. Thus the very idea of trying to prove that the statement holds for $n = 1, 2, \ldots,$ is wrong-headed.

The safe way to reason inductively about asymptotic phenomena is to *work directly with the definition of the asymptotic notation*. Let's try to prove the claim above in this way. Remember that $f(n) = O(n)$ means that there exist constants n_0 and $c > 0$ such that $|f(n)| \leq cn$ for all $n \geq n_0$. (Let's not worry about the absolute value for now.) If all goes well, the proof attempt should fail in some blatantly obvious way, instead of in a subtle, hard-to-detect way like the earlier argument. Since our perverse goal is to demonstrate that the proof won't work for *any* constants n_0 and c, we'll leave these as variables and assume only that they're chosen so that the base case holds; that is, $T(n_0) \leq cn$.

Proof Attempt. We use strong induction. Let $P(n)$ be the proposition that $T(n) \leq cn$.

Base case: $P(n_0)$ is true, because $T(n_0) \leq cn$.

Inductive step: For $n > n_0$, assume that $P(n_0), \ldots, P(n-1)$ are true in order to prove $P(n)$. We reason as follows:

$$\begin{aligned}
T(n) &= 2T(n/2) + n \\
&\leq 2c(n/2) + n \\
&= cn + n \\
&= (c+1)n \\
&\not\leq cn.
\end{aligned}$$ ■

The first equation is the recurrence. Then we use induction and simplify until the argument collapses!

In general, it is a good idea to stay away from asymptotic notation altogether while you are doing the induction. Once the induction is over and done with, then you can safely use big-Oh to simplify your result.

10.5 A Feel for Recurrences

We've guessed and verified, plugged and chugged, found roots, computed integrals, and solved linear systems and exponential equations. Now let's step back and look for some rules of thumb. What kinds of recurrences have what sorts of solutions?

Here are some recurrences we solved earlier:

	Recurrence	Solution
Towers of Hanoi	$T_n = 2T_{n-1} + 1$	$T_n \sim 2^n$
Merge Sort	$T_n = 2T_{n/2} + n - 1$	$T_n \sim n \log n$
Hanoi variation	$T_n = 2T_{n-1} + n$	$T_n \sim 2 \cdot 2^n$
Fibonacci	$T_n = T_{n-1} + T_{n-2}$	$T_n \sim (1.618\ldots)^{n+1}/\sqrt{5}$

Notice that the recurrence equations for Towers of Hanoi and Merge Sort are somewhat similar, but the solutions are radically different. Merge Sorting $n = 64$ items takes a few hundred comparisons, while moving $n = 64$ disks takes more than 10^{19} steps!

Each recurrence has one strength and one weakness. In the Towers of Hanoi, we broke a problem of size n into two subproblem of size $n - 1$ (which is large), but needed only 1 additional step (which is small). In Merge Sort, we divided the problem of size n into two subproblems of size $n/2$ (which is small), but needed $(n - 1)$ additional steps (which is large). Yet, Merge Sort is faster by a mile!

This suggests that *generating smaller subproblems is far more important to algorithmic speed than reducing the additional steps per recursive call.* For example, shifting to the variation of Towers of Hanoi increased the last term from $+1$ to $+n$, but the solution only doubled. And one of the two subproblems in the Fibonacci recurrence is just *slightly* smaller than in Towers of Hanoi (size $n - 2$ instead of $n - 1$). Yet the solution is exponentially smaller! More generally, linear recurrences (which have big subproblems) typically have exponential solutions, while divide-and-conquer recurrences (which have small subproblems) usually have solutions bounded above by a polynomial.

All the examples listed above break a problem of size n into two smaller problems. How does the number of subproblems affect the solution? For example, suppose we increased the number of subproblems in Towers of Hanoi from 2 to 3, giving this recurrence:

$$T_n = 3T_{n-1} + 1$$

This increases the root of the characteristic equation from 2 to 3, which raises the solution exponentially, from $\Theta(2^n)$ to $\Theta(3^n)$.

Divide-and-conquer recurrences are also sensitive to the number of subproblems. For example, for this generalization of the Merge Sort recurrence:

$$\begin{aligned} T_1 &= 0 \\ T_n &= aT_{n/2} + n - 1. \end{aligned}$$

the Akra-Bazzi formula gives:

$$T_n = \begin{cases} \Theta(n) & \text{for } a < 2 \\ \Theta(n \log n) & \text{for } a = 2 \\ \Theta(n^{\log a}) & \text{for } a > 2. \end{cases}$$

So the solution takes on three completely different forms as a goes from 1.99 to 2.01!

How do boundary conditions affect the solution to a recurrence? We've seen that they are almost irrelevant for divide-and-conquer recurrences. For linear recurrences, the solution is usually dominated by an exponential whose base is determined by the number and size of subproblems. Boundary conditions matter greatly only when they give the dominant term a zero coefficient, which changes the asymptotic solution.

So now we have a rule of thumb! The performance of a recursive procedure is usually dictated by the size and number of subproblems, rather than the amount of work per recursive call or time spent at the base of the recursion. In particular, if subproblems are smaller than the original by an additive factor, the solution is most often exponential. But if the subproblems are only a fraction the size of the original, then the solution is typically bounded by a polynomial.

11 Cardinality Rules

11.1 Counting One Thing by Counting Another

How do you count the number of people in a crowded room? You could count heads, since for each person there is exactly one head. Alternatively, you could count ears and divide by two. Of course, you might have to adjust the calculation if someone lost an ear in a pirate raid or someone was born with three ears. The point here is that you can often *count one thing by counting another*, though some fudge factors may be required. 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 S and it is denoted by $|S|$. 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 Theorem 7.2.1. Of particular interest here is part 3 of Theorem 7.2.1, where we state that if there is a bijection between two sets, then the sets have the same size. This important fact is commonly known as the *Bijection Rule*.

11.1.1 The Bijection Rule

Rule 11.1.1 (Bijection Rule)**.** *If there is a bijection $f : A \to B$ between A and B, then $|A| = |B|$.*

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, consider the two sets mentioned at the beginning of Part III:

A = all ways to select a dozen doughnuts when five varieties are available

B = all 16-bit sequences with exactly 4 ones

Let's consider a particular element of set A:

$$\underbrace{0\;0}_{\text{chocolate}} \quad \underbrace{}_{\text{lemon-filled}} \quad \underbrace{0\;0\;0\;0\;0\;0}_{\text{sugar}} \quad \underbrace{0\;0}_{\text{glazed}} \quad \underbrace{0\;0}_{\text{plain}}$$

We've depicted each doughnut with a 0 and left a gap between the different varieties. Thus, the selection above contains two chocolate doughnuts, no lemon-filled, six sugar, two glazed, and two plain. Now let's put a 1 into each of the four gaps:

$$\underbrace{0\,0}_{\text{chocolate}} \quad 1 \quad \underbrace{\qquad}_{\text{lemon-filled}} \quad 1 \quad \underbrace{0\,0\,0\,0\,0\,0}_{\text{sugar}} \quad 1 \quad \underbrace{0\,0}_{\text{glazed}} \quad 1 \quad \underbrace{0\,0}_{\text{plain}}$$

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:

$$c \text{ chocolate, } l \text{ lemon-filled, } s \text{ sugar, } g \text{ glazed, and } p \text{ plain}$$

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 example 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.

11.2 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!

11.2.1 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 11.2.1 (Product Rule)**.** *If $P_1, P_2, \ldots P_n$ are sets, then:*

$$|P_1 \times P_2 \times \ldots \times P_n| = |P_1| \cdot |P_2| \cdots |P_n|$$

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 where:

$$\begin{aligned}
B &= \{\text{pancakes, bacon and eggs, bagel, Doritos}\}\\
L &= \{\text{burger and fries, garden salad, Doritos}\}\\
D &= \{\text{macaroni, pizza, frozen burrito, pasta, Doritos}\}
\end{aligned}$$

Then $B \times L \times D$ is the set of all possible daily diets. Here are some sample elements:

$$\begin{gathered}
(\text{pancakes, burger and fries, pizza})\\
(\text{bacon and eggs, garden salad, pasta})\\
(\text{Doritos, Doritos, frozen burrito})
\end{gathered}$$

The Product Rule tells us how many different daily diets are possible:

$$\begin{aligned}
|B \times L \times D| &= |B| \cdot |L| \cdot |D|\\
&= 4 \cdot 3 \cdot 5\\
&= 60.
\end{aligned}$$

11.2.2 Subsets of an *n*-element Set

How many different subsets of an n-element set X are there? For example, the set $X = \{x_1, x_2, x_3\}$ has eight different subsets:

$$\begin{array}{cccc}
\emptyset & \{x_1\} & \{x_2\} & \{x_1, x_2\}\\
\{x_3\} & \{x_1, x_3\} & \{x_2, x_3\} & \{x_1, x_2, x_3\}.
\end{array}$$

There is a natural bijection from subsets of X to n-bit sequences. Let $x_1, x_2, \ldots, x_n$ be the elements of X. Then a particular subset of X maps to the sequence $(b_1, \ldots, b_n)$

where $b_i = 1$ if and only if x_i is in that subset. For example, if $n = 10$, then the subset $\{x_2, x_3, x_5, x_7, x_{10}\}$ maps to a 10-bit sequence as follows:

$$\begin{array}{rcccccccccccc}
\text{subset:} & \{ & & x_2, & x_3, & & x_5, & & x_7, & & & x_{10} & \} \\
\text{sequence:} & (& 0, & 1, & 1, & 0, & 1, & 0, & 1, & 0, & 0, & 1 &)
\end{array}$$

We just used a bijection to transform the original problem into a question about sequences—*exactly according to plan!* Now if we answer the sequence question, then we’ve solved our original problem as well.

But how many different n-bit sequences are there? For example, there are 8 different 3-bit sequences:

$$\begin{array}{cccc}
(0,0,0) & (0,0,1) & (0,1,0) & (0,1,1) \\
(1,0,0) & (1,0,1) & (1,1,0) & (1,1,1)
\end{array}$$

Well, we can write the set of all n-bit sequences as a product of sets:

$$\underbrace{\{0,1\} \times \{0,1\} \times \ldots \times \{0,1\}}_{n \text{ terms}} = \{0,1\}^n$$

Then Product Rule gives the answer:

$$\begin{aligned}
|\{0,1\}^n| &= |\{0,1\}|^n \\
&= 2^n
\end{aligned}$$

This means that the number of subsets of an n-element set X is also 2^n. We’ll put this answer to use shortly.

11.2.3 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 11.2.2 (Sum Rule)**.** *If $A_1, A_2, \ldots, A_n$ are* disjoint *sets, then:*

$$|A_1 \cup A_2 \cup \ldots \cup A_n| = |A_1| + |A_2| + \ldots + |A_n|$$

Thus, according to Linus’ budget, Lucy can be out-of-sorts for:

$$\begin{aligned}
|C \cup I \cup S| &= |C| + |I| + |S| \\
&= 20 + 40 + 60 \\
&= 120 \text{ days}
\end{aligned}$$

Notice that the Sum Rule holds only for a union of *disjoint* sets. Finding the size of a union of intersecting sets is a more complicated problem that we'll take up later.

11.2.4 Counting Passwords

Few counting problems can be solved with a single rule. More often, a solution is a flurry of sums, products, bijections, and other methods. For example, the sum and product rules together are useful for solving problems involving passwords, telephone numbers, and license plates. For example, on a certain computer system, a valid password is a sequence of between six and eight symbols. The first symbol must be a letter (which can be lowercase or uppercase), and the remaining symbols must be either letters or digits. How many different passwords are possible?

Let's define two sets, corresponding to valid symbols in the first and subsequent positions in the password.

$$\begin{aligned} F &= \{a, b, \ldots, z, A, B, \ldots, Z\} \\ S &= \{a, b, \ldots, z, A, B, \ldots, Z, 0, 1, \ldots, 9\} \end{aligned}$$

In these terms, the set of all possible passwords is:[1]

$$(F \times S^5) \cup (F \times S^6) \cup (F \times S^7)$$

Thus, the length-six passwords are in the 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:

$$\begin{aligned} &|(F \times S^5) \cup (F \times S^6) \cup (F \times S^7)| \\ &\quad = |F \times S^5| + |F \times S^6| + |F \times S^7| && \text{Sum Rule} \\ &\quad = |F| \cdot |S|^5 + |F| \cdot |S|^6 + |F| \cdot |S|^7 && \text{Product Rule} \\ &\quad = 52 \cdot 62^5 + 52 \cdot 62^6 + 52 \cdot 62^7 \\ &\quad \approx 1.8 \cdot 10^{14} \text{ different passwords.} \end{aligned}$$

11.3 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

[1] The notation S^5 means $S \times S \times S \times S \times S$.

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 $|P^3| = |P|^3 = n^3$, 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:

$$S = \{(x, y, z) \in P^3 \mid x, y, \text{ and } z \text{ are different people}\}$$

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 11.3.1 (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.

11.3.1 Defective Dollar Bills

A dollar bill 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 bills are all-too-common. In fact, how common are *nondefective* bills? Assuming that the digit portions of serial numbers all occur equally often, we could answer this question by computing

$$\text{fraction of nondefective bills} = \frac{|\{\text{serial \#'s with all digits different}\}|}{|\{\text{serial numbers}\}|}. \tag{11.1}$$

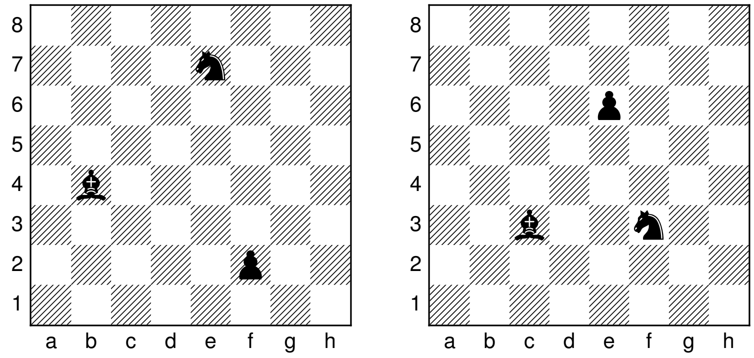
Let's first consider the denominator. Here there are no restrictions; there are are 10 possible first digits, 10 possible second digits, 10 third digits, and so on. Thus, the total number of 8-digit serial numbers is 10^8 by the Product Rule.

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

$$10 \cdot 9 \cdot 8 \cdot 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 = \frac{10!}{2} = 1{,}814{,}400$$

serial numbers with all digits different. Plugging these results into Equation 11.1, we find:

$$\text{fraction of nondefective bills} = \frac{1{,}814{,}400}{100{,}000{,}000} = 1.8144\%$$



(a) valid

(b) invalid

Figure 11.1 Two ways of placing a pawn (♙), a knight (♘), and a bishop (♗) on a chessboard. The configuration shown in (b) is invalid because the bishop and the knight are in the same row.

11.3.2 A Chess Problem

In how many different ways can we place a pawn (P), a knight (N), and a bishop (B) on a chessboard so that no two pieces share a row or a column? A valid configuration is shown in Figure 11.1(a), and an invalid configuration is shown in Figure 11.1(b).

First, we map this problem about chess pieces to a question about sequences. There is a bijection from configurations to sequences

$$(r_P, c_P, r_N, c_N, r_B, c_B)$$

where r_P, r_N, and r_B are distinct rows and c_P, c_N, and c_B are distinct columns. In particular, r_P is the pawn's row, c_P is the pawn's column, r_N 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_N is one of 7 rows (any one but r_P)
- c_N is one of 7 columns (any one but c_P)
- r_B is one of 6 rows (any one but r_P or r_N)
- c_B is one of 6 columns (any one but c_P or c_N)

Thus, the total number of configurations is $(8 \cdot 7 \cdot 6)^2$.

11.3.3 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 $3! = 6$ permutations 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:

$$n! \sim \sqrt{2\pi n} \left(\frac{n}{e}\right)^n.$$

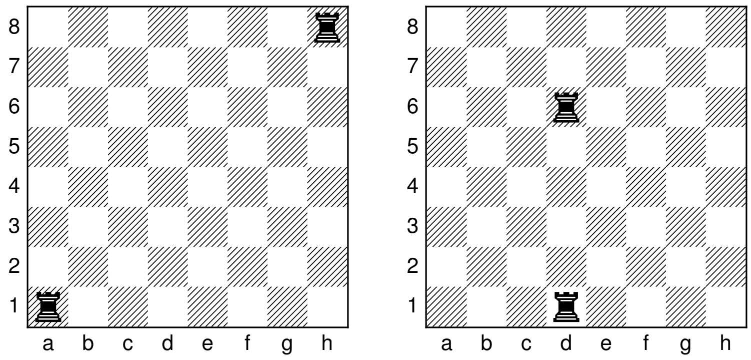
11.4 The Division Rule

Counting ears and dividing by two is a silly way to count the number of people in a room, but this approach is representative of a powerful counting principle.

A *k-to-1 function* maps exactly k elements of the domain to every element of the codomain. For example, the function mapping each ear to its owner is 2-to-1. Similarly, the function mapping each finger to its owner is 10-to-1, and the function mapping each finger and toe to its owner is 20-to-1. The general rule is:

Rule 11.4.1 (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, $|A| = 2 \cdot |B|$. 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.



(a) valid

(b) invalid

Figure 11.2 Two ways to place 2 rooks (♖) on a chessboard. The configuration in (b) is invalid because the rooks are in the same column.

11.4.1 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 in Figure 11.2(a), and an invalid configuration is shown in Figure 11.2(b).

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 in Figure 11.2(a).

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, $|A| = 2 \cdot |B|$.

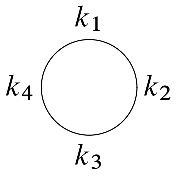
Rearranging terms gives:

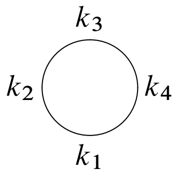
$$|B| = \frac{|A|}{2} = \frac{(8 \cdot 7)^2}{2}.$$

On the second line, we've computed the size of A using the General Product Rule just as in the earlier chess problem.

11.4.2 Knights of the Round Table

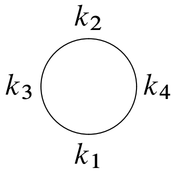
In how many ways can King Arthur seat n different knights at his round table? Two seatings are considered equivalent if one can be obtained from the other by rotation. For example, the following two arrangements are equivalent:





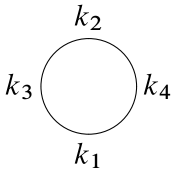
Let A be all the permutations of the knights, and let B be the set of all possible seating arrangements at the round table. We can map each permutation in set A to a circular seating arrangement in set B by seating the first knight in the permutation anywhere, putting the second knight to his left, the third knight to the left of the second, and so forth all the way around the table. For example:

$$(k_2, k_4, k_1, k_3) \quad \longrightarrow$$



This mapping is actually an n-to-1 function from A to B, since all n cyclic shifts of the original sequence map to the same seating arrangement. In the example, $n = 4$ different sequences map to the same seating arrangement:

$$\begin{matrix} (k_2, k_4, k_1, k_3) \\ (k_4, k_1, k_3, k_2) \\ (k_1, k_3, k_2, k_4) \\ (k_3, k_2, k_4, k_1) \end{matrix} \quad \longrightarrow$$



Therefore, by the division rule, the number of circular seating arrangements is:

$$|B| = \frac{|A|}{n} = \frac{n!}{n} = (n-1)!$$

Note that $|A| = n!$ since there are $n!$ permutations of n knights.

11.5 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.

11.5.1 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 $\{a_1, \dots, a_n\}$ into a k-element subset simply by taking the first k elements of the permutation. That is, the permutation $a_1a_2 \dots a_n$ will map to the set $\{a_1, a_2, \dots, a_k\}$.

Notice that any other permutation with the same first k elements $a_1, \dots, a_k$ *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 $\{a_1, a_2, \dots, a_k\}$ if its first k elements are the elements $a_1, \dots, a_k$ 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 11.5.1 (Subset Rule)**.** *The number of k-element subsets of an n-element set is*

$$\binom{n}{k} = \frac{n!}{k!\,(n-k)!}.$$

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[2] equals 1.

11.5.2 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 $\{x_1, x_2, \dots, x_8\}$ 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,

[2] We don't use it here, but a *sum* of zero terms equals 0.

$$\text{The number of } n\text{-bit sequences with exactly } k \text{ ones is } \binom{n}{k}.$$

11.6 Sequences with Repetitions

11.6.1 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 disjoint subsets of A and $|A_i| = k_i$ for $i = 1, \ldots, m$.

Rule 11.6.1 (Subset Split Rule)**.** *The number of* $(k_1, k_2, \ldots, k_m)$*-splits of an* n*-element set is*

$$\binom{n}{k_1, \ldots, k_m} ::= \frac{n!}{k_1!\, k_2! \cdots k_m!}$$

The proof of this Rule is essentially the same as for the Subset Rule. Namely, we map any permutation $a_1 a_2 \ldots a_n$ of an n-element set A into a $(k_1, k_2, \ldots, k_m)$-split by letting the 1st subset in the split be the first k_1 elements of the permutation, the 2nd subset of the split be the next k_2 elements, ..., and the mth subset of the split be the final k_m elements of the permutation. This map is a $k_1!\, k_2! \cdots k_m!$-to-1 function from the $n!$ permutations to the $(k_1, k_2, \ldots, k_m)$-splits of A, and the Subset Split Rule now follows from the Division Rule.

11.6.2 The Bookkeeper Rule

We can also generalize our count of n-bit sequences with k ones to counting sequences of n 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 $\{1, 2, \dots, 10\}$. Namely, map a permutation to the sequence of sets of positions where each of the different letters occur.

For example, in the permutation BOOKKEEPER itself, the B is in the 1st position, the O's occur in the 2nd and 3rd positions, K's in 4th and 5th, the E's in the 6th, 7th and 9th, P in the 8th, and R is in the 10th position. So BOOKKEEPER maps to

$$(\{1\}, \{2, 3\}, \{4, 5\}, \{6, 7, 9\}, \{8\}, \{10\}).$$

From this bijection and the Subset Split Rule, we conclude that the number of ways to rearrange the letters in the word BOOKKEEPER is:

$$\frac{\overbrace{10!}^{\text{total letters}}}{\underbrace{1!}_{\text{B's}}\ \underbrace{2!}_{\text{O's}}\ \underbrace{2!}_{\text{K's}}\ \underbrace{3!}_{\text{E's}}\ \underbrace{1!}_{\text{P's}}\ \underbrace{1!}_{\text{R's}}}$$

This example generalizes directly to an exceptionally useful counting principle which we will call the

Rule 11.6.2 (Bookkeeper Rule). *Let $l_1, \dots, l_m$ be distinct elements. The number of sequences with k_1 occurrences of l_1, and k_2 occurrences of l_2, ..., and k_m occurrences of l_m is*

$$\frac{(k_1 + k_2 + \ldots + k_m)!}{k_1!\, k_2!\, \ldots\, k_m!}$$

For example, suppose you are 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:

$$\frac{20!}{5!^4}.$$

11.6.3 The Binomial Theorem

Counting gives insight into one of the basic theorems of algebra. A *binomial* is a sum of two terms, such as $a + b$. Now consider its 4th power, $(a + b)^4$.

If we multiply out this 4th power expression completely, we get

$$\begin{aligned}(a+b)^4 \quad = \quad & \quad aaaa + aaab + aaba + aabb \\ & + abaa + abab + abba + abbb \\ & + baaa + baab + baba + babb \\ & + bbaa + bbab + bbba + bbbb\end{aligned}$$

Notice that there is one term for every sequence of a's and b's. So there are 2^4 terms, and the number of terms with k copies of b and $n-k$ copies of a is:

$$\frac{n!}{k!\,(n-k)!} = \binom{n}{k}$$

by the Bookkeeper Rule. Hence, the coefficient of $a^{n-k}b^k$ is $\binom{n}{k}$. So for $n = 4$, this means:

$$(a+b)^4 = \binom{4}{0} \cdot a^4b^0 + \binom{4}{1} \cdot a^3b^1 + \binom{4}{2} \cdot a^2b^2 + \binom{4}{3} \cdot a^1b^3 + \binom{4}{4} \cdot a^0b^4$$

In general, this reasoning gives the Binomial Theorem:

Theorem 11.6.3 (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 rearrangements 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 11.6.4. For $n, k_1, \dots, k_m \in \mathbb{N}$, 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 11.6.5 (Multinomial Theorem)**.** *For all $n \in \mathbb{N}$,*

$$(z_1 + z_2 + \cdots + z_m)^n = \sum_{\substack{k_1,\ldots,k_m \in \mathbb{N} \\ k_1+\cdots+k_m=n}} \binom{n}{k_1, k_2, \ldots, k_m} z_1^{k_1} z_2^{k_2} \cdots z_m^{k_m}.$$

You’ll be better off remembering the reasoning behind the Multinomial Theorem rather than this ugly formal statement.

11.6.4 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.

11.7 Counting Practice: Poker Hands

Five-Card Draw is a card game in which each player is initially dealt a *hand* consisting of 5 cards from a deck of 52 cards.[3] (Then the game gets complicated, but let’s not worry about that.) The number of different hands in Five-Card Draw is the

[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.

number of 5-element subsets of a 52-element set, which is

$$\binom{52}{5} = 2,598,960.$$

Let’s get some counting practice by working out the number of hands with various special properties.

11.7.1 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:

$$\begin{gathered}\{8\spadesuit,\ 8\diamondsuit,\ Q\heartsuit,\ 8\heartsuit,\ 8\clubsuit\}\\ \{A\clubsuit,\ 2\clubsuit,\ 2\heartsuit,\ 2\diamondsuit,\ 2\spadesuit\}\end{gathered}$$

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:

$$\begin{gathered}(8, Q, \heartsuit) \leftrightarrow \{\,8\spadesuit,\ 8\diamondsuit,\ 8\heartsuit,\ 8\clubsuit,\ Q\heartsuit\}\\ (2, A, \clubsuit) \leftrightarrow \{2\clubsuit,\ 2\heartsuit,\ 2\diamondsuit,\ 2\spadesuit,\ A\clubsuit\}\end{gathered}$$

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, Four-of-a-Kind is considered to be a very good poker hand!

11.7.2 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:

$$\{2\spadesuit,\ 2\clubsuit,\ 2\diamondsuit,\ J\clubsuit,\ J\diamondsuit\}$$
$$\{5\diamondsuit,\ 5\clubsuit,\ 5\heartsuit,\ 7\heartsuit,\ 7\clubsuit\}$$

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:

$$(2, \{\spadesuit, \clubsuit, \diamondsuit\}, J, \{\clubsuit, \diamondsuit\}) \leftrightarrow \{2\spadesuit,\ 2\clubsuit,\ 2\diamondsuit,\ J\clubsuit,\ J\diamondsuit\}$$
$$(5, \{\diamondsuit, \clubsuit, \heartsuit\}, 7, \{\heartsuit, \clubsuit\}) \leftrightarrow \{5\diamondsuit,\ 5\clubsuit,\ 5\heartsuit,\ 7\heartsuit,\ 7\clubsuit\}$$

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 speed bump.

11.7.3 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:

$$\{3\diamondsuit,\ 3\spadesuit,\ Q\diamondsuit,\ Q\heartsuit,\ A\clubsuit\}$$
$$\{9\heartsuit,\ 9\diamondsuit,\ 5\heartsuit,\ 5\clubsuit,\ K\spadesuit\}$$

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:

$$\begin{array}{rl}
(3, \{\diamondsuit, \spadesuit\}, Q, \{\diamondsuit, \heartsuit\}, A, \clubsuit) \;\searrow & \\
 & \{3\diamondsuit,\ 3\spadesuit,\ Q\diamondsuit,\ Q\heartsuit,\ A\clubsuit\} \\
(Q, \{\diamondsuit, \heartsuit\}, 3, \{\diamondsuit, \spadesuit\}, A, \clubsuit) \;\nearrow & \\
 & \\
(9, \{\heartsuit, \diamondsuit\}, 5, \{\heartsuit, \clubsuit\}, K, \spadesuit) \;\searrow & \\
 & \{9\heartsuit,\ 9\diamondsuit,\ 5\heartsuit,\ 5\clubsuit,\ K\spadesuit\} \\
(5, \{\heartsuit, \clubsuit\}, 9, \{\heartsuit, \diamondsuit\}, K, \spadesuit) \;\nearrow &
\end{array}$$

The problem is that nothing distinguishes the first pair from the second. A pair of 5's and a pair of 9's is the same as a pair of 9's and a pair of 5's. We avoided this difficulty in counting Full Houses because, for example, a pair of 6's and a triple of kings is different from a pair of kings and a triple of 6's.

We ran into precisely this difficulty last time, when we went from counting arrangements of *different* pieces on a chessboard to counting arrangements of two *identical* rooks. The solution then was to apply the Division Rule, and we can do the same here. In this case, the Division rule says there are twice as many sequences as hands, so the number of hands with Two Pairs is actually:

$$\frac{13 \cdot \binom{4}{2} \cdot 12 \cdot \binom{4}{2} \cdot 11 \cdot 4}{2}.$$

Another Approach

The preceding example was disturbing! One could easily overlook the fact that the mapping was 2-to-1 on an exam, fail the course, and turn to a life of crime. You can make the world a safer place in two ways:

1. Whenever you use a mapping $f : A \to B$ to translate one counting problem to another, check that the same number elements in A are mapped to each element in B. If k elements of A map to each of element of B, then apply the Division Rule using the constant k.

2. As an extra check, try solving the same problem in a different way. Multiple approaches are often available—and all had better give the same answer! (Sometimes different approaches give answers that *look* different, but turn out to be the same after some algebra.)

We already used the first method; let's try the second. There is a bijection between hands with two pairs and sequences that specify:

1. The ranks of the two pairs, which can be chosen in $\binom{13}{2}$ ways.

2. The suits of the lower-rank pair, which can be selected in $\binom{4}{2}$ ways.

3. The suits of the higher-rank pair, which can be selected in $\binom{4}{2}$ ways.

4. The rank of the extra card, which can be chosen in 11 ways.

5. The suit of the extra card, which can be selected in $\binom{4}{1} = 4$ ways.

For example, the following sequences and hands correspond:

$$(\{3, Q\}, \{\diamondsuit, \spadesuit\}, \{\diamondsuit, \heartsuit\}, A, \clubsuit) \leftrightarrow \{3\diamondsuit,\ 3\spadesuit,\ Q\diamondsuit,\ Q\heartsuit,\ A\clubsuit\}$$
$$(\{9, 5\}, \{\heartsuit, \clubsuit\}, \{\heartsuit, \diamondsuit\}, K, \spadesuit) \leftrightarrow \{9\heartsuit,\ 9\diamondsuit,\ 5\heartsuit,\ 5\clubsuit,\ K\spadesuit\}$$

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.

11.7.4 Hands with Every Suit

How many hands contain at least one card from every suit? Here is an example of such a hand:

$$\{7\diamondsuit,\ K\clubsuit,\ 3\diamondsuit,\ A\heartsuit,\ 2\spadesuit\}$$

Each such hand is described by a sequence that specifies:

1. The ranks of the diamond, the club, the heart, and the spade, which can be selected in $13 \cdot 13 \cdot 13 \cdot 13 = 13^4$ ways.

2. The suit of the extra card, which can be selected in 4 ways.

3. The rank of the extra card, which can be selected in 12 ways.

For example, the hand above is described by the sequence:

$$(7, K, A, 2, \diamondsuit, 3) \leftrightarrow \{7\diamondsuit,\ K\clubsuit,\ A\heartsuit,\ 2\spadesuit,\ 3\diamondsuit\}.$$

Are there other sequences that correspond to the same hand? There is one more! We could equally well regard either the $3\diamondsuit$ or the $7\diamondsuit$ as the extra card, so this is actually a 2-to-1 mapping. Here are the two sequences corresponding to the example hand:

$$\begin{tikzcd}
(7, K, A, 2, \diamondsuit, 3) \arrow[dr] & \\
& \{7\diamondsuit,\ K\clubsuit,\ A\heartsuit,\ 2\spadesuit,\ 3\diamondsuit\} \\
(3, K, A, 2, \diamondsuit, 7) \arrow[ur] &
\end{tikzcd}$$

Therefore, the number of hands with every suit is:

$$\frac{13^4 \cdot 4 \cdot 12}{2}.$$

11.8 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 if M, E, and P are disjoint, then the sum of their sizes is

$$|M \cup E \cup P| = |M| + |E| + |P|.$$

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 most-complicated 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.

[4] . . . though not at MIT anymore.

11.8.1 Union of Two Sets

For two sets, S_1 and S_2, the *Inclusion-Exclusion Rule* is that the size of their union is:

$$|S_1 \cup S_2| = |S_1| + |S_2| - |S_1 \cap S_2| \tag{11.2}$$

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.

11.8.2 Union of Three Sets

So how many students are there in the math, EECS, and physics departments? In other words, what is $|M \cup E \cup P|$ if:

$$\begin{aligned} |M| &= 60 \\ |E| &= 200 \\ |P| &= 40. \end{aligned}$$

The size of a union of three sets is given by a more complicated Inclusion-Exclusion formula:

$$\begin{aligned} |S_1 \cup S_2 \cup S_3| = {} & |S_1| + |S_2| + |S_3| \\ & - |S_1 \cap S_2| - |S_1 \cap S_3| - |S_2 \cap S_3| \\ & + |S_1 \cap S_2 \cap S_3|. \end{aligned}$$

Remarkably, the expression on the right accounts for each element in the union of S_1, S_2, and S_3 exactly once. For example, suppose that x is an element of all three sets. Then x is counted three times (by the $|S_1|$, $|S_2|$, and $|S_3|$ terms), subtracted off three times (by the $|S_1 \cap S_2|$, $|S_1 \cap S_3|$, and $|S_2 \cap S_3|$ terms), and then counted once more (by the $|S_1 \cap S_2 \cap S_3|$ term). The net effect is that x is counted just once.

If x is in two sets (say, S_1 and S_2), then x is counted twice (by the $|S_1|$ and $|S_2|$ terms) and subtracted once (by the $|S_1 \cap S_2|$ term). In this case, x does not factor into any of the other terms, since $x \notin S_3$.

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:

$$\begin{aligned}|M \cup E \cup P| &= |M| + |E| + |P| - |M \cap E| - |M \cap P| - |E \cap P| + |M \cap E \cap P| \\ &= 60 + 200 + 40 - 6 - 5 - 13 + 2 \\ &= 278\end{aligned}$$

11.8.3 Sequences with 42, 04, or 60

In how many permutations of the set $\{0, 1, 2, \dots, 9\}$ 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}, but not P_{42}. 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 $\{0, 1, 2, \dots 9\}$ containing 6 and 0 consecutively and permutations of:

$$\{60, 1, 2, 3, 4, 5, 7, 8, 9\}.$$

For example, the following two sequences correspond:

$$(7, 2, 5, \underline{6}, \underline{0}, 4, 3, 8, 1, 9) \longleftrightarrow (7, 2, 5, \underline{60}, 4, 3, 8, 1, 9).$$

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:

$$|P_{42} \cup P_{04} \cup P_{60}| = 9! + 9! + 9! - 8! - 8! - 8! + 7!.$$

11.8.4 Union of *n* Sets

The size of a union of n sets is given by the following rule.

Rule 11.8.1 (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

$$\sum_{1 \le i < j \le n} |S_i \cap S_j|.$$

Similarly, the sum of the sizes of the three-way intersections is

$$\sum_{1 \le i < j < k \le n} |S_i \cap S_j \cap S_k|.$$

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)**.**

$$\begin{aligned} |\bigcup_{i=1}^{n} S_i| &= \sum_{i=1}^{n} |S_i| \\ &\quad - \sum_{1 \le i < j \le n} |S_i \cap S_j| \\ &\quad + \sum_{1 \le i < j < k \le n} |S_i \cap S_j \cap S_k| + \cdots \\ &\quad + (-1)^{n-1} |\bigcap_{i=1}^{n} S_i|. \end{aligned}$$

11.8.5 Computing Euler's Function

As an example, let's 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. 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 the 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,

$$|C_1 \cap C_2 \cap C_3| = \frac{n}{p_1 p_2 p_3}.$$

Reasoning this way about all the intersections among the C_i's and applying Inclusion-Exclusion, we get

$$\begin{aligned}
|S| &= |\bigcup_{i=1}^{m} C_i| \\
&= \sum_{i=1}^{m} |C_i| - \sum_{1 \le i < j \le m} |C_i \cap C_j| + \sum_{1 \le i < j < k \le m} |C_i \cap C_j \cap C_k| - \cdots + (-1)^{m-1} |\bigcap_{i=1}^{m} C_i| \\
&= \sum_{i=1}^{m} \frac{n}{p_i} - \sum_{1 \le i < j \le m} \frac{n}{p_i p_j} + \sum_{1 \le i < j < k \le m} \frac{n}{p_i p_j p_k} - \cdots + (-1)^{m-1} \frac{n}{p_1 p_2 \cdots p_n} \\
&= n \left(\sum_{i=1}^{m} \frac{1}{p_i} - \sum_{1 \le i < j \le m} \frac{1}{p_i p_j} + \sum_{1 \le i < j < k \le m} \frac{1}{p_i p_j p_k} - \cdots + (-1)^{m-1} \frac{1}{p_1 p_2 \cdots p_n} \right)
\end{aligned}$$

But $\phi(n) = n - |S|$ by definition, so

$$\begin{aligned}
\phi(n) &= n \left(1 - \sum_{i=1}^{m} \frac{1}{p_i} + \sum_{1 \le i < j \le m} \frac{1}{p_i p_j} - \sum_{1 \le i < j < k \le m} \frac{1}{p_i p_j p_k} + \cdots + (-1)^{m} \frac{1}{p_1 p_2 \cdots p_n} \right) \\
&= n \prod_{i=1}^{m} \left(1 - \frac{1}{p_i} \right). \qquad (11.3)
\end{aligned}$$

Yikes! That was pretty hairy. Are you getting tired of all that nasty algebra? If so, then good news is on the way. In the next section, we will show you how to prove some heavy-duty formulas without using any algebra at all. Just a few words and you are done. No kidding.

11.9 Combinatorial Proofs

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.

Hmmm. . .

11.9.1 Pascal’s Identity

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:

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}.$$

This is called *Pascal's Identity*. And we proved it *without any algebra!* Instead, we relied purely on counting techniques.

11.9.2 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 way. 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 a less colorful example of a combinatorial argument.

Theorem 11.9.1.

$$\sum_{r=0}^{n} \binom{n}{r}\binom{2n}{n-r} = \binom{3n}{n}$$

Proof. We give a combinatorial proof. Let S be all n-card hands that can be dealt from a deck containing n red cards (numbered $1, \ldots, n$) and $2n$ black cards (numbered $1, \ldots, 2n$). First, note that every $3n$-element set has

$$|S| = \binom{3n}{n}$$

n-element subsets.

From another perspective, the number of hands with exactly r red cards is

$$\binom{n}{r}\binom{2n}{n-r}$$

since there are $\binom{n}{r}$ ways to choose the r red cards and $\binom{2n}{n-r}$ ways to choose the $n-r$ black cards. Since the number of red cards can be anywhere from 0 to n, the total number of n-card hands is:

$$|S| = \sum_{r=0}^{n} \binom{n}{r}\binom{2n}{n-r}.$$

Equating these two expressions for $|S|$ proves the theorem. ■

Combinatorial proofs are almost magical. Theorem 11.9.1 looks pretty scary, but we proved it without any algebraic manipulations at all. The key to constructing a combinatorial proof is choosing the set S properly, which can be tricky. Generally, the simpler side of the equation should provide some guidance. For example, the right side of Theorem 11.9.1 is $\binom{3n}{n}$, which suggests that it will be helpful to choose S to be all n-element subsets of some $3n$-element set.

11.10 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 of the Mapping Rule.

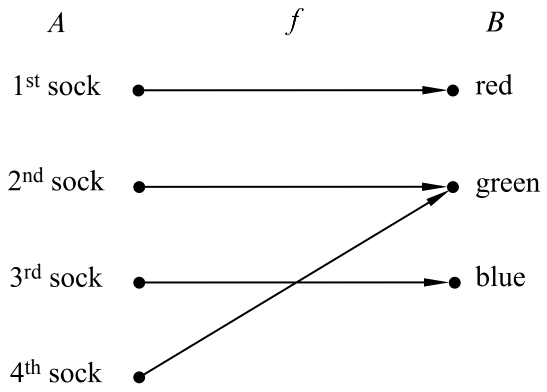


Figure 11.3 One possible mapping of four socks to three colors.

Rule 11.10.1 (Pigeonhole Principle)**.** *If $|X| > |Y|$, then for every total function*[5] *$f : X \to Y$, 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). Therefore, four socks are enough to ensure a matched pair. For example, one possible mapping of four socks to three colors is shown in Figure 11.3.

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:

[5] This Mapping Rule applies even if f is a total injective *relation*. Recall that a function is total if $\forall x \in X \, \exists y \in Y. \; f(x) = y$.

1. The set A (the pigeons).
2. The set B (the pigeonholes).
3. The function f (the rule for assigning pigeons to pigeonholes).

11.10.1 Hairs on Heads

There are a number of generalizations of the pigeonhole principle. For example:

Rule 11.10.2 (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!

11.10.2 Subsets with the Same Sum

For your reading pleasure, we have displayed ninety 25-digit numbers in Figure 11.4. Are there two different subsets of these 25-digit numbers that have the same sum? For example, maybe the sum of the last ten numbers in the first column is equal to the sum of the first eleven numbers in the second column?

Finding two subsets with the same sum may seem like a silly puzzle, but solving these sorts of problems turns out to be useful in diverse applications such as finding good ways to fit packages into shipping containers and decoding secret messages.

It turns out that it is hard to find different subsets with the same sum, which is why this problem arises in cryptography. But it is easy to prove that two such subsets *exist*. That's where the Pigeonhole Principle comes in.

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).

```
0020480135385502964448038
5763257331083479647409398
0489445991866915676240992
5800949123548989122628663
1082662032430379651370981
6042900801199280218026001
1178480894769706178994993
6116171789137737896701405
1253127351683239693851327
6144868973001582369723512
1301505129234077811069011
6247314593851169234746152
1311567111143866433882194
6814428944266874963488274
1470029452721203587686214
6870852945543886849147881
1578271047286257499433886
6914955508120950093732397
1638243921852176243192354
6949632451365987152423541
1763580219131985963102365
7128211143613619828415650
1826227795601842231029694
7173920083651862307925394
1843971862675102037201420
7215654874211755676220587
2396951193722134526177237
7256932847164391040233050
2781394568268599801096354
7332822657075235431620317
2796605196713610405408019
7426441829541573444964139
2931016394761975263190347
7632198126531809327186321
2933458058294405155197296
7712154432211912882310511
3075514410490975920315348
7858918664240262356610010
8149436716871371161932035
3111474985252793452860017
7898156786763212963178679
3145621587936120118438701
8147591017037573337848616
3148901255628881103198549
5692168374637019617423712
3171004832173501394113017
8247331000042995311646021
3208234421597368647019265
8496243997123475922766310
3437254656355157864869113
8518399140676002660747477
3574883393058653923711365
8543691283470191452333763
3644909946040480189969149
8675309258374137092461352
3790044132737084094417246
8694321112363996867296665
3870332127437971355322815
8772321203608477245851154
4080505804577801451363100
8791422161722582546341091
4167283461025702348124920
9062628024592126283973285
4235996831123777788211249
9137845566925526349897794
4670939445749439042111220
9153762966803189291934419
4815379351865384279613427
9270880194077636406984249
4837052948212922604442190
9324301480722103490379204
5106389423855018550671530
9436090832146695147140581
5142368192004769218069910
9475308159734538249013238
5181234096130144084041856
9492376623917486974923202
5198267398125617994391348
9511972558779880288252979
5317592940316231219758372
9602413424619187112552264
5384358126771794128356947
9631217114906129219461111
3157693105325111284321993
5439211712248901995423441
9908189853102753335981319
5610379826092838192760458
9913237476341764299813987
5632317555465228677676044
8176063831682536571306791
```

Figure 11.4 Ninety 25-digit numbers. Can you find two different subsets of these numbers that have the same sum?

We proved that an n-element set has 2^n different subsets in Section 11.2. Therefore:

$$|A| = 2^{90} \geq 1.237 \times 10^{27}$$

On the other hand:

$$|B| = 90 \cdot 10^{25} + 1 \leq 0.901 \times 10^{27}.$$

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*. To see if was possible to actually *find* two different subsets of the ninety 25-digit numbers with the same sum, we offered a $100 prize to the first student who did it. 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.

11.11 A 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.

Five audience members each select one card from the deck. The Assistant then gathers up the five cards and holds up four of them so the Magician can see them. The Magician concentrates for a short time and then correctly names the secret, fifth card!

Since we don’t really believe the Magician can read minds, we know the Assistant has somehow communicated the secret card to the Magician. Since real Magicians and Assistants are not to be trusted, we can expect that the Assistant would illegitimately signal the Magician with coded phrases or body language, but they don’t have to cheat in this way. In fact, the Magician and Assistant could be

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 mathematicians 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 in such a set must be greater than $c2^n$ for some constant $c > 0$. He offered $500 to anyone who could prove or disprove his conjecture, but the problem remains unsolved.

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).

11.11.1 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 has a second 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 5 cards, then the Assistant must reveal a sequence of 4 cards that is adjacent in the bipartite graph. Some edges are shown in the diagram in Figure 11.5.

For example,

$$\{8\heartsuit, K\spadesuit, Q\spadesuit, 2\diamondsuit, 6\diamondsuit\} \tag{11.4}$$

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 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 Figure 11.5. If the audience selects the set

$$\{8\heartsuit, K\spadesuit, Q\spadesuit, 9\clubsuit, 6\diamondsuit\}, \tag{11.5}$$

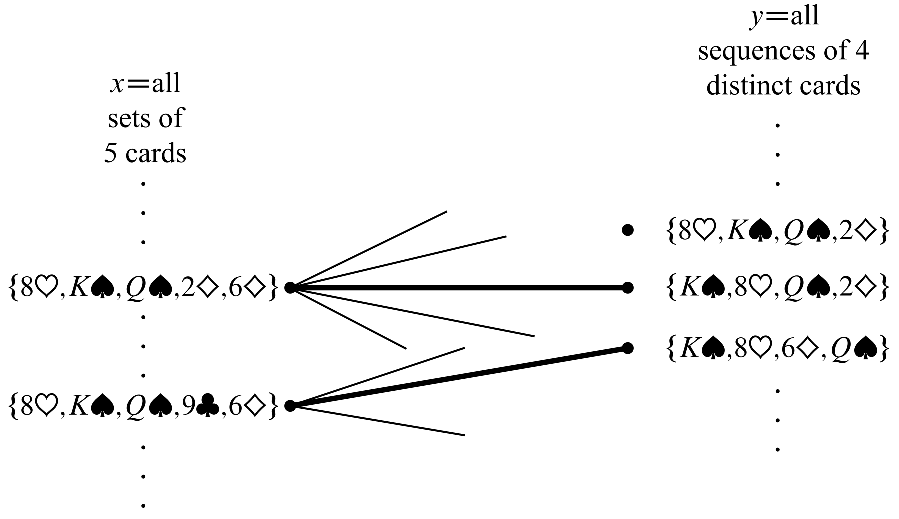


Figure 11.5 The bipartite graph where the nodes on the left correspond to *sets* of 5 cards and the nodes on the right correspond to *sequences* of 4 cards. There is an edge between a set and a sequence whenever all the cards in the sequence are contained in the set.

then the Assistant reveals the corresponding sequence

$$(K\spadesuit, 8\heartsuit, 6\diamondsuit, Q\spadesuit). \tag{11.6}$$

Using the matching, the Magician sees that the hand (11.5) is matched to the sequence (11.6), so he can name the one card in the corresponding set not already revealed, namely, the 9♣. 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 (11.4), it would be possible for the Assistant to reveal the same sequence (11.6), 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♣ or the 2♢.

So how can we be sure the needed matching can be found? The answer 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 5.2.6, and therefore satisfies Hall's matching condition.

In fact, this reasoning shows 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!

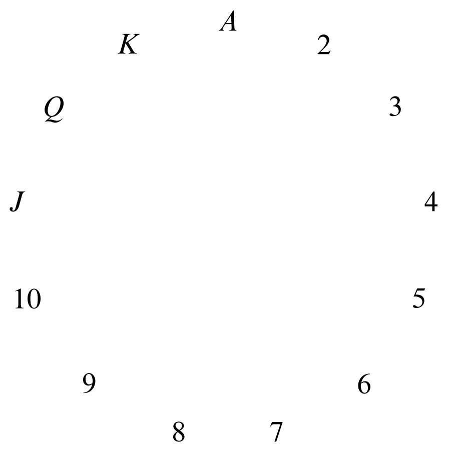


Figure 11.6 The 13 card ranks arranged in cyclic order.

11.11.2 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 $10\heartsuit$. This is always possible because of the Pigeonhole Principle—there are five cards and 4 suits so two cards must be in the same suit.

- The Assistant locates the ranks of these two cards on the cycle shown in Figure 11.6. For any two distinct ranks on this cycle, one is always between 1 and 6 hops clockwise from the other. For example, the $3\heartsuit$ is 6 hops clockwise from the $10\heartsuit$.

- The more counterclockwise of these two cards is revealed first, and the other becomes the secret card. Thus, in our example, the $10\heartsuit$ would be revealed, and the $3\heartsuit$ would be the secret card. Therefore:

  - The suit of the secret card is the same as the suit of the first card revealed.

  - The rank of the secret card is between 1 and 6 hops clockwise from the rank of the first card revealed.

- All that remains is to communicate a number between 1 and 6. The Magician and Assistant agree beforehand on an ordering of all the cards in the deck from smallest to largest such as:

  $$A\clubsuit\ A\diamondsuit\ A\heartsuit\ A\spadesuit\ 2\clubsuit\ 2\diamondsuit\ 2\heartsuit\ 2\spadesuit\ \ldots\ K\heartsuit\ K\spadesuit$$

  The order in which the last three cards are revealed communicates the number according to the following scheme:

  $$\begin{array}{lcccl} (& \text{small}, & \text{medium}, & \text{large} &) = 1 \\ (& \text{small}, & \text{large}, & \text{medium} &) = 2 \\ (& \text{medium}, & \text{small}, & \text{large} &) = 3 \\ (& \text{medium}, & \text{large}, & \text{small} &) = 4 \\ (& \text{large}, & \text{small}, & \text{medium} &) = 5 \\ (& \text{large}, & \text{medium}, & \text{small} &) = 6 \end{array}$$

  In the example, the Assistant wants to send 6 and so reveals the remaining three cards in large, medium, small order. Here is the complete sequence that the Magician sees:

  $$10\heartsuit \quad Q\spadesuit \quad J\diamondsuit \quad 9\diamondsuit$$

- The Magician starts with the first card, $10\heartsuit$, and hops 6 ranks clockwise to reach $3\heartsuit$, which is the secret card!

So that's how the trick can work with a standard deck of 52 cards. On the other hand, Hall's Theorem implies that the Magician and Assistant can *in principle* perform the trick with a deck of up to 124 cards. It turns out that there is a method which they could actually learn to use with a reasonable amount of practice for a 124-card deck, but we won't explain it here.[6]

11.11.3 The 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

$$|X| = \binom{52}{4} = 270,725$$

[6] See *The Best Card Trick* by Michael Kleber for more information.

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!

11.11.4 Never Say Never

No sooner than we finished proving that the Magician can’t pull off the trick with four cards instead of five, a student showed us a way that it might be doable after all. The idea is to place the three cards on a table one at a time instead of revealing them all at once. This provides the Magician with two completely independent sequences of three cards: one for the *temporal* order in which the cards are placed on the table, and one for the *spatial* order in which they appear once placed.

For example, suppose the audience selects

$$10\heartsuit \quad 9\diamondsuit \quad 3\heartsuit \quad Q\spadesuit$$

and the assistant decides to reveal

$$10\heartsuit \quad 9\diamondsuit \quad Q\spadesuit.$$

The assistant might decide to reveal the $Q\spadesuit$ first, the $10\heartsuit$ second, and the $9\diamondsuit$ third, thereby production the *temporal* sequence

$$(Q\spadesuit, 10\heartsuit, 9\diamondsuit).$$

If the $Q\spadesuit$ is placed in the middle position on the table, the $10\heartsuit$ is placed in the rightmost position on the table, and the $9\diamondsuit$ is placed in the leftmost position on the table, the *spatial* sequence would be

$$(9\diamondsuit, Q\spadesuit, 10\heartsuit).$$

In this version of the card trick, X consists of all sets of 4 cards and Y consists of all *pairs* of sequences of the same 3 cards. As before, we can create a bipartite graph where an edge connects a set S of 4 cards in X with a pair of sequences in Y if the 3 cards in the sequences are in S.

The degree of every node in X is then

$$4 \cdot 3! \cdot 3! = 144$$

since there are 4 choices for which card is not revealed and 3! orders for each sequence in the pair.

The degree of every node in Y is 49 since there are $52 - 3 = 49$ possible choices for the 4th card. Since $144 \geq 49$, we can use Hall's Theorem to establish the existing of a matching for X.

Hence, the magic trick *is* doable with 4 cards—the assistant just has to convey more information. Can you figure out a convenient way to pull off the trick on the fly?

So what about the 3-card version? Surely that is not doable.…

12 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 then apply all that machinery to problems about sequences. In this way, we can use generating functions to solve all sorts of counting problems. They can also be used to find closed-form expressions for sums and to solve recurrences. In fact, many of the problems we addressed in Chapters 9–11 can be formulated and solved using generating functions.

12.1 Definitions and Examples

The *ordinary generating function* for the sequence[1] $\langle g_0, g_1, g_2, g_3 \dots \rangle$ is the power series:

$$G(x) = g_0 + g_1 x + g_2 x^2 + g_3 x^3 + \cdots .$$

There are a few other kinds of generating functions in common use, but ordinary generating functions are enough to illustrate the power of the idea, so we'll stick to them and 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:

$$\langle g_0, g_1, g_2, g_3, \dots \rangle \longleftrightarrow g_0 + g_1 x + g_2 x^2 + g_3 x^3 + \cdots .$$

For example, here are some sequences and their generating functions:

$$\begin{aligned}
\langle 0, 0, 0, 0, \dots \rangle &\longleftrightarrow 0 + 0x + 0x^2 + 0x^3 + \cdots = 0 \\
\langle 1, 0, 0, 0, \dots \rangle &\longleftrightarrow 1 + 0x + 0x^2 + 0x^3 + \cdots = 1 \\
\langle 3, 2, 1, 0, \dots \rangle &\longleftrightarrow 3 + 2x + 1x^2 + 0x^3 + \cdots = 3 + 2x + x^2
\end{aligned}$$

[1] In this chapter, we'll put sequences in angle brackets to more clearly distinguish them from the many other mathematical expressions floating around.

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 won't worry about convergence issues for now. This formula gives closed form generating functions for a whole range of sequences. For example:

$$\langle 1, 1, 1, 1, \dots \rangle \longleftrightarrow 1 + x + x^2 + x^3 + x^4 + \cdots = \frac{1}{1-x}$$

$$\langle 1, -1, 1, -1, \dots \rangle \longleftrightarrow 1 - x + x^2 - x^3 + x^4 - \cdots = \frac{1}{1+x}$$

$$\left\langle 1, a, a^2, a^3, \dots \right\rangle \longleftrightarrow 1 + ax + a^2x^2 + a^3x^3 + \cdots = \frac{1}{1-ax}$$

$$\langle 1, 0, 1, 0, 1, 0, \dots \rangle \longleftrightarrow 1 + x^2 + x^4 + x^6 + x^8 + \cdots = \frac{1}{1-x^2}$$

12.2 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.

12.2.1 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:

$$\langle 2, 0, 2, 0, 2, 0, \ldots \rangle .$$

Rule 12.2.1 (Scaling Rule)**.** *If*

$$\langle f_0, f_1, f_2, \ldots \rangle \longleftrightarrow F(x),$$

then

$$\langle cf_0,\ cf_1,\ cf_2,\ \ldots \rangle \longleftrightarrow c \cdot F(x).$$

The idea behind this rule is that:

$$\begin{aligned}\langle cf_0, cf_1, cf_2, \ldots \rangle \longleftrightarrow{} & cf_0 + cf_1 x + cf_2 x^2 + \cdots \\ &= c \cdot (f_0 + f_1 x + f_2 x^2 + \cdots) \\ &= cF(x).\end{aligned}$$

12.2.2 Addition

Adding generating functions corresponds to adding the two sequences term by term. For example, adding two of our earlier examples gives:

$$\begin{array}{rcl} \langle 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ \ldots \rangle & \longleftrightarrow & \dfrac{1}{1-x} \\ + \langle 1, -1, 1, -1, 1, -1, \ldots \rangle & \longleftrightarrow & \dfrac{1}{1+x} \\ \hline \langle 2,\ 0,\ 2,\ 0,\ 2,\ 0,\ \ldots \rangle & \longleftrightarrow & \dfrac{1}{1-x} + \dfrac{1}{1+x} \end{array}$$

We've now derived two different expressions that both generate the sequence $\langle 2, 0, 2, 0, \ldots \rangle$. They are, of course, equal:

$$\frac{1}{1-x} + \frac{1}{1+x} = \frac{(1+x) + (1-x)}{(1-x)(1+x)} = \frac{2}{1-x^2}.$$

Rule 12.2.2 (Addition Rule)**.** *If*

$$\begin{aligned}\langle f_0, f_1, f_2, \ldots \rangle &\longleftrightarrow F(x) \qquad \textit{and} \\ \langle g_0, g_1, g_2, \ldots \rangle &\longleftrightarrow G(x),\end{aligned}$$

then

$$\langle f_0 + g_0,\ f_1 + g_1,\ f_2 + g_2,\ \ldots \rangle \longleftrightarrow F(x) + G(x).$$

The idea behind this rule is that:

$$\begin{aligned}\langle f_0 + g_0,\ f_1 + g_1,\ f_2 + g_2,\ \ldots\rangle \longleftrightarrow\ & \sum_{n=0}^{\infty}(f_n + g_n)x^n \\ &= \left(\sum_{n=0}^{\infty} f_n x^n\right) + \left(\sum_{n=0}^{\infty} g_n x^n\right) \\ &= F(x) + G(x).\end{aligned}$$

12.2.3 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:

$$\begin{aligned}\langle\overbrace{0, 0, \ldots, 0}^{k \text{ zeroes}}, 1, 1, 1, \ldots\rangle \longleftrightarrow\ & x^k + x^{k+1} + x^{k+2} + x^{k+3} + \cdots \\ &= x^k \cdot (1 + x + x^2 + x^3 + \cdots) \\ &= \frac{x^k}{1-x}.\end{aligned}$$

Evidently, adding k leading zeros to the sequence corresponds to multiplying the generating function by x^k. This holds true in general.

Rule 12.2.3 (Right-Shift Rule)**.** *If* $\langle f_0, f_1, f_2, \ldots\rangle \longleftrightarrow F(x)$*, then:*

$$\langle\overbrace{0, 0, \ldots, 0}^{k \text{ zeroes}}, f_0, f_1, f_2, \ldots\rangle \longleftrightarrow x^k \cdot F(x).$$

The idea behind this rule is that:

$$\begin{aligned}\langle\overbrace{0, 0, \ldots, 0}^{k \text{ zeroes}}, f_0, f_1, f_2, \ldots\rangle \longleftrightarrow\ & f_0 x^k + f_1 x^{k+1} + f_2 x^{k+2} + \cdots \\ &= x^k \cdot (f_0 + f_1 x + f_2 x^2 + f_3 x^3 + \cdots) \\ &= x^k \cdot F(x).\end{aligned}$$

12.2.4 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:

$$\begin{aligned}
& & 1 + x + x^2 + x^3 + x^4 + \cdots &= \frac{1}{1-x} \\
&\text{IMPLIES} & \frac{d}{dx}\left(1 + x + x^2 + x^3 + x^4 + \cdots\right) &= \frac{d}{dx}\left(\frac{1}{1-x}\right) \\
&\text{IMPLIES} & 1 + 2x + 3x^2 + 4x^3 + \cdots &= \frac{1}{(1-x)^2} \\
&\text{IMPLIES} & \langle 1, 2, 3, 4, \dots \rangle &\longleftrightarrow \frac{1}{(1-x)^2}. \qquad (12.1)
\end{aligned}$$

We found a generating function for the sequence $\langle 1, 2, 3, 4, \dots \rangle$ of positive integers!

In general, differentiating a generating function has two effects on the corresponding sequence: each term is multiplied by its index and the entire sequence is shifted left one place.

Rule 12.2.4 (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 12.2.3 tells how to cancel out this unwanted left-shift: multiply the generating function by x.

Our procedure, therefore, is to begin with the generating function for $\langle 1, 1, 1, 1, \ldots \rangle$, differentiate, multiply by x, and then differentiate and multiply by x once more. Then

$$\begin{array}{rrcll}
 & \langle 1, 1, 1, 1, \ldots \rangle & \longleftrightarrow & \dfrac{1}{1-x} & \\
\text{Derivative Rule:} & \langle 1, 2, 3, 4, \ldots \rangle & \longleftrightarrow & \dfrac{d}{dx}\,\dfrac{1}{1-x} & = \dfrac{1}{(1-x)^2} \\
\text{Right-shift Rule:} & \langle 0, 1, 2, 3, \ldots \rangle & \longleftrightarrow & x \cdot \dfrac{1}{(1-x)^2} & = \dfrac{x}{(1-x)^2} \\
\text{Derivative Rule:} & \langle 1, 4, 9, 16, \ldots \rangle & \longleftrightarrow & \dfrac{d}{dx}\,\dfrac{x}{(1-x)^2} & = \dfrac{1+x}{(1-x)^3} \\
\text{Right-shift Rule:} & \langle 0, 1, 4, 9, \ldots \rangle & \longleftrightarrow & x \cdot \dfrac{1+x}{(1-x)^3} & = \dfrac{x(1+x)}{(1-x)^3}
\end{array}$$

Thus, the generating function for squares is:

$$\frac{x(1+x)}{(1-x)^3}. \tag{12.2}$$

12.2.5 Products

Rule 12.2.5 (Product Rule)**.** *If*

$$\langle a_0, a_1, a_2, \ldots \rangle \longleftrightarrow A(x), \quad \textit{and} \quad \langle b_0, b_1, b_2, \ldots \rangle \longleftrightarrow B(x),$$

then

$$\langle c_0,\ c_1,\ c_2,\ \ldots \rangle \longleftrightarrow A(x) \cdot B(x),$$

where

$$c_n ::= a_0 b_n + a_1 b_{n-1} + a_2 b_{n-2} + \cdots + a_n b_0.$$

To understand this rule, let

$$C(x) ::= A(x) \cdot B(x) = \sum_{n=0}^{\infty} c_n x^n.$$

We can evaluate the product $A(x) \cdot B(x)$ by using a table to identify all the cross-terms from the product of the sums:

	b_0x^0	b_1x^1	b_2x^2	b_3x^3	$\ldots$
a_0x^0	$a_0b_0x^0$	$a_0b_1x^1$	$a_0b_2x^2$	$a_0b_3x^3$	$\ldots$
a_1x^1	$a_1b_0x^1$	$a_1b_1x^2$	$a_1b_2x^3$	$\ldots$	
a_2x^2	$a_2b_0x^2$	$a_2b_1x^3$	$\ldots$		
a_3x^3	$a_3b_0x^3$	$\ldots$			
$\vdots$	$\ldots$				

Notice that all terms involving the same power of x lie on a 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,

$$a_0b_n + a_1b_{n-1} + a_2b_{n-2} + \cdots + a_nb_0. \tag{12.3}$$

This expression (12.3) 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 $\langle b_0, b_1, b_2, \ldots \rangle$.

12.3 Evaluating Sums

The product rule looks complicated. But it is surprisingly useful. For example, suppose that we set

$$B(x) = \frac{1}{1-x}.$$

Then $b_i = 1$ for $i \geq 0$ and the nth coefficient of $A(x)B(x)$ is

$$a_0 \cdot 1 + a_1 \cdot 1 + a_2 \cdot 1 + \cdots + a_n \cdot 1 = \sum_{i=0}^{n} a_i.$$

In other words, given any sequence $\langle a_0, a_1, a_2, \ldots \rangle$, we can compute

$$s_n = \sum_{i=0}^{n} a_i$$

for all n by simply multiplying the sequence's generating function by $1/(1-x)$. This is the Summation Rule.

Rule 12.3.1 (Summation Rule)**.** *If*

$$\langle a_0, a_1, a_2, \dots \rangle \longleftrightarrow A(x),$$

then

$$\langle s_0, s_1, s_2, \dots \rangle \longleftrightarrow \frac{A(x)}{1-x}$$

where

$$s_n = \sum_{i=0}^{n} a_i \quad \textit{for } n \geq 0.$$

The Summation Rule sounds powerful, and it is! We know from Chapter 9 that computing sums is often not easy. But multiplying by $1/(1-x)$ is about as easy as it gets.

For example, suppose that we want to compute the sum of the first n squares

$$s_n = \sum_{i=0}^{n} i^2$$

and we forgot the method in Chapter 9. All we need to do is compute the generating function for $\langle 0, 1, 4, 9, \dots \rangle$ and multiply by $1/(1-x)$. We already computed the generating function for $\langle 0, 1, 4, 9, \dots \rangle$ in Equation 12.2—it is

$$\frac{x(1+x)}{(1-x)^3}.$$

Hence, the generating function for $\langle s_0, s_1, s_2, \dots \rangle$ is

$$\frac{x(1+x)}{(1-x)^4}.$$

This means that $\sum_{i=0}^{n} i^2$ is the coefficient of x^n in $x(1+x)/(1-x)^4$.

That was pretty easy, but there is one problem—we have no idea how to determine the coefficient of x^n in $x(1+x)/(1-x)^4$! And without that, this whole endeavor (while magical) would be useless. Fortunately, there is a straightforward way to produce the sequence of coefficients from a generating function.

12.4 Extracting Coefficients

12.4.1 Taylor Series

Given a sequence of coefficients $\langle f_0, f_1, f_2, \ldots \rangle$, computing the generating function $F(x)$ is easy since

$$F(x) = f_0 + f_1 x + f_2 x^2 + \cdots .$$

To compute the sequence of coefficients from the generating function, we need to compute the *Taylor Series* for the generating function.

Rule 12.4.1 (Taylor Series)**.** *Let $F(x)$ be the generating function for the sequence*

$$\langle f_0, f_1, f_2, \ldots \rangle.$$

Then

$$f_0 = F(0)$$

and

$$f_n = \frac{F^{(n)}(0)}{n!}$$

for $n \geq 1$, where $F^{(n)}(0)$ is the nth derivative of $F(x)$ evaluated at $x = 0$.

This is because if

$$F(x) = f_0 + f_1 x + f_2 x^2 + \cdots ,$$

then

$$\begin{aligned} F(0) &= f_0 + f_1 \cdot 0 + f_2 \cdot 0^2 + \cdots \\ &= f_0. \end{aligned}$$

Also,

$$\begin{aligned} F'(x) &= \frac{d}{dx}(F(x)) \\ &= f_1 + 2f_2 x + 3f_3 x^2 + 4f_4 x^3 + \cdots \end{aligned}$$

and so

$$F'(0) = f_1,$$

as desired. Taking second derivatives, we find that

$$\begin{aligned} F''(x) &= \frac{d}{dx}(F'(x)) \\ &= 2f_2 + 3 \cdot 2 f_3 x + 4 \cdot 3 f_4 x^2 + \cdots \end{aligned}$$

and so

$$F''(0) = 2f_2,$$

which means that

$$f_2 = \frac{F''(0)}{2}.$$

In general,

$$\begin{aligned} F^{(n)} = n! f_n + (n+1)! f_{n+1} x + \frac{(n+2)!}{2} f_{n+2} x^2 + \cdots \\ + \frac{(n+k)!}{k!} f_{n+k} x^k + \cdots \end{aligned}$$

and so

$$F^{(n)}(0) = n! f_n$$

and

$$f_n = \frac{F^{(n)}(0)}{n!},$$

as claimed.

This means that

$$\left\langle F(0), F'(0), \frac{F''(0)}{2!}, \frac{F'''(0)}{3!}, \ldots, \frac{F^{(n)}(0)}{n!}, \ldots \right\rangle \longleftrightarrow F(x). \tag{12.4}$$

The sequence on the left-hand side of Equation 12.4 gives the well-known Taylor Series expansion for a function

$$F(x) = F(0) + F'(0)x + \frac{F''(0)}{2!}x^2 + \frac{F'''(0)}{3!}x^3 + \cdots + \frac{F^{(n)}(0)}{n!}x^n + \cdots.$$

12.4.2 Examples

Let's try this out on a familiar example:

$$F(x) = \frac{1}{1-x}.$$

Computing derivatives, we find that

$$\begin{aligned}
F'(x) &= \frac{1}{(1-x)^2},\\
F''(x) &= \frac{2}{(1-x)^3},\\
F'''(x) &= \frac{2 \cdot 3}{(1-x)^4},\\
&\vdots\\
F^{(n)} &= \frac{n!}{(1-x)^{n+1}}.
\end{aligned}$$

This means that the coefficient of x^n in $1/(1-x)$ is

$$\frac{F^{(n)}(0)}{n!} = \frac{n!}{n!\,(1-0)^{n+1}} = 1.$$

In other words, we have reconfirmed what we already knew; namely, that

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

Using a similar approach, we can establish some other well-known series:

$$\begin{aligned}
e^x &= 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^n}{n!} + \cdots ,\\
e^{ax} &= 1 + ax + \frac{a^2}{2!}x^2 + \frac{a^3}{3!}x^3 + \cdots + \frac{a^n}{n!}x^n + \cdots ,\\
\ln(1-x) &= -ax - \frac{a^2}{2}x^2 - \frac{a^3}{3}x^3 - \cdots - \frac{a^n}{n}x^n - \cdots .
\end{aligned}$$

But what about the series for

$$F(x) = \frac{x(1+x)}{(1-x)^4}? \tag{12.5}$$

In particular, we need to know the coefficient of x^n in $F(x)$ to determine

$$s_n = \sum_{i=0}^{n} i^2.$$

While it is theoretically possible to compute the nth derivative of $F(x)$, the result is a bloody mess. Maybe these generating functions weren’t such a great idea after all. . . .

12.4.3 Massage Helps

In times of stress, a little massage can often help relieve the tension. The same is true for polynomials with painful derivatives. For example, let's take a closer look at Equation 12.5. If we massage it a little bit, we find that

$$F(x) = \frac{x + x^2}{(1-x)^4} = \frac{x}{(1-x)^4} + \frac{x^2}{(1-x)^4}. \tag{12.6}$$

The goal is to find the coefficient of x^n in $F(x)$. If you stare at Equation 12.6 long enough (or if you combine the Right-Shift Rule with the Addition Rule), you will notice that the coefficient of x^n in $F(x)$ is just the sum of

$$\text{the coefficient of } x^{n-1} \text{ in } \frac{1}{(1-x)^4} \text{ and}$$
$$\text{the coefficient of } x^{n-2} \text{ in } \frac{1}{(1-x)^4}.$$

Maybe there is some hope after all. Let's see if we can produce the coefficients for $1/(1-x)^4$. We'll start by looking at the derivatives:

$$\begin{aligned}
F'(x) &= \frac{4}{(1-x)^5},\\
F''(x) &= \frac{4 \cdot 5}{(1-x)^6},\\
F'''(x) &= \frac{4 \cdot 5 \cdot 6}{(1-x)^7},\\
&\vdots\\
F^{(n)}(x) &= \frac{(n+3)!}{6(1-x)^{n+4}}.
\end{aligned}$$

This means that the nth coefficient of $1/(1-x)^4$ is

$$\frac{F^{(n)}(0)}{n!} = \frac{(n+3)!}{6n!} = \frac{(n+3)(n+2)(n+1)}{6}. \tag{12.7}$$

We are now almost done. Equation 12.7 means that the coefficient of x^{n-1} in $1/(1-x)^4$ is

$$\frac{(n+2)(n+1)n}{6} \tag{12.8}$$

and the coefficient[2] of x^{n-2} is

$$\frac{(n+1)n(n-1)}{6}. \tag{12.9}$$

Adding these values produces the desired sum

$$\begin{aligned}\sum_{i=0}^{n} i^2 &= \frac{(n+2)(n+1)n}{6} + \frac{(n+1)n(n-1)}{6} \\ &= \frac{(2n+1)(n+1)n}{6}.\end{aligned}$$

This matches Equation 9.14 from Chapter 9. Using generating functions to get the result may have seemed to be more complicated, but at least there was no need for guessing or solving a linear system of equations over 4 variables.

You might argue that the massage step was a little tricky. After all, how were you supposed to know that by converting $F(x)$ into the form shown in Equation 12.6, it would be sufficient to compute derivatives of $1/(1-x)^4$, which is easy, instead of derivatives of $x(1+x)/(1-x)^4$, which could be harder than solving a 64-disk Tower of Hanoi problem step-by-step?

The good news is that this sort of massage works for any generating function that is a ratio of polynomials. Even better, you probably already know how to do it from calculus—it's the method of *partial fractions*!

12.4.4 Partial Fractions

The idea behind partial fractions is to express a ratio of polynomials as a sum of a polynomial and terms of the form

$$\frac{cx^a}{(1-\alpha x)^b} \tag{12.10}$$

where a and b are integers and $b > a \geq 0$. That's because it is easy to compute derivatives of $1/(1-\alpha x)^b$ and thus it is easy to compute the coefficients of Equation 12.10. Let's see why.

Lemma 12.4.2. *If $b \in \mathbb{N}^+$, then the nth derivative of $1/(1-\alpha x)^b$ is*

$$\frac{(n+b-1)!\,\alpha^n}{(b-1)!\,(1-\alpha x)^{b+n}}.$$

[2] To be precise, Equation 12.8 holds for $n \geq 1$ and Equation 12.9 holds for $n \geq 2$. But since Equation 12.8 is 0 for $n = 1$ and Equation 12.9 is 0 for $n = 1, 2$, both equations hold for all $n \geq 0$.

Proof. The proof is by induction on n. The induction hypothesis $P(n)$ is the statement of the lemma.

Base case ($n = 1$): The first derivative is

$$\frac{b\alpha}{(1-\alpha x)^{b+1}}.$$

This matches

$$\frac{(1+b-1)!\,\alpha^1}{(b-1)!\,(1-\alpha x)^{b+1}} = \frac{b\alpha}{(1-\alpha x)^{b+1}},$$

and so $P(1)$ is true.

Induction step: We next assume $P(n)$ to prove $P(n+1)$ for $n \geq 1$. $P(n)$ implies that the nth derivative of $1/(1-\alpha x)^b$ is

$$\frac{(n+b-1)!\,\alpha^n}{(b-1)!\,(1-\alpha x)^{b+n}}.$$

Taking one more derivative reveals that the $(n+1)$st derivative is

$$\frac{(n+b-1)!\,(b+n)\alpha^{n+1}}{(b-1)!\,(1-\alpha x)^{b+n+1}} = \frac{(n+b)!\,\alpha^{n+1}}{(b-1)!\,(1-\alpha x)^{b+n+1}},$$

which means that $P(n+1)$ is true. Hence, the induction is complete. ■

Corollary 12.4.3. *If $a, b \in \mathbb{N}$ and $b > a \geq 0$, then for any $n \geq a$, the coefficient of x^n in*

$$\frac{cx^a}{(1-\alpha x)^b}$$

is

$$\frac{c(n-a+b-1)!\,\alpha^{n-a}}{(n-a)!\,(b-1)!}.$$

Proof. By the Taylor Series Rule, the nth coefficient of

$$\frac{1}{(1-\alpha x)^b}$$

is the nth derivative of this expression evaluated at $x = 0$ and then divided by $n!$. By Lemma 12.4.2, this is

$$\frac{(n+b-1)!\,\alpha^n}{n!\,(b-1)!\,(1-0)^{b+n}} = \frac{(n+b-1)!\,\alpha^n}{n!\,(b-1)!}.$$

By the Scaling Rule and the Right-Shift Rule, the coefficient of x^n in

$$\frac{cx^{\alpha}}{(1-\alpha x)^b}$$

is thus

$$\frac{c(n-a+b-1)!\,\alpha^{n-a}}{(n-a)!\,(b-1)!}.$$

as claimed. ■

Massaging a ratio of polynomials into a sum of a polynomial and terms of the form in Equation 12.10 takes a bit of work but is generally straightforward. We will show you the process by means of an example.

Suppose our generating function is the ratio

$$F(x) = \frac{4x^3+2x^2+3x+6}{2x^3-3x^2+1}. \tag{12.11}$$

The first step in massaging $F(x)$ is to get the degree of the numerator to be less than the degree of the denominator. This can be accomplished by dividing the numerator by the denominator and taking the remainder, just as in the Fundamental Theorem of Arithmetic—only now we have polynomials instead of numbers. In this case we have

$$\frac{4x^3+2x^2+3x+6}{2x^3-3x^2+1} = 2 + \frac{8x^2+3x+4}{2x^3-3x^2+1}.$$

The next step is to factor the denominator. This will produce the values of α for Equation 12.10. In this case,

$$\begin{aligned} 2x^3-3x^2+1 &= (2x+1)(x^2-2x+1) \\ &= (2x+1)(x-1)^2 \\ &= (1-x)^2(1+2x). \end{aligned}$$

We next find values c_1, c_2, c_3 so that

$$\frac{8x^2+3x+4}{2x^3-3x^2+1} = \frac{c_1}{1+2x} + \frac{c_2}{(1-x)^2} + \frac{c_3x}{(1-x)^2}. \tag{12.12}$$

This is done by cranking through the algebra:

$$\begin{aligned} \frac{c_1}{1+2x} + \frac{c_2}{(1-x)^2} + \frac{c_3x}{(1-x)^2} &= \frac{c_1(1-x)^2 + c_2(1+2x) + c_3x(1+2x)}{(1+2x)(1-x)^2} \\ &= \frac{c_1 - 2c_1x + c_1x^2 + c_2 + 2c_2x + c_3x + 2c_3x^2}{2x^3-3x^2+1} \\ &= \frac{c_1 + c_2 + (-2c_1 + 2c_2 + c_3)x + (c_1 + 2c_3)x^2}{2x^3-3x^2+1}. \end{aligned}$$

For Equation 12.12 to hold, we need

$$\begin{aligned} 8 &= c_1 + 2c_3, \\ 3 &= -2c_1 + 2c_2 + c_3, \\ 4 &= c_1 + c_2. \end{aligned}$$

Solving these equations, we find that $c_1 = 2$, $c_2 = 2$, and $c_3 = 3$. Hence,

$$\begin{aligned} F(x) &= \frac{4x^3 + 2x^2 + 3x + 6}{2x^3 - 3x^2 + 1} \\ &= 2 + \frac{2}{1 + 2x} + \frac{2}{(1 - x)^2} + \frac{3x}{(1 - x)^2}. \end{aligned}$$

Our massage is done! We can now compute the coefficients of $F(x)$ using Corollary 12.4.3 and the Sum Rule. The result is

$$f_0 = 2 + 2 + 2 = 6$$

and

$$\begin{aligned} f_n &= \frac{2(n - 0 + 1 - 1)!\,(-2)^{n-0}}{(n - 0)!\,(1 - 1)!} \\ &\quad + \frac{2(n - 0 + 2 - 1)!\,(1)^{n-0}}{(n - 0)!\,(2 - 1)!} \\ &\quad + \frac{3(n - 1 + 2 - 1)!\,(1)^{n-1}}{(n - 1)!\,(2 - 1)!} \\ &= (-1)^n 2^{n+1} + 2(n + 1) + 3n \\ &= (-1)^n 2^{n+1} + 5n + 2 \end{aligned}$$

for $n \geq 1$.

Aren’t you glad that you know that? Actually, this method turns out to be useful in solving linear recurrences, as we’ll see in the next section.

12.5 Solving Linear Recurrences

Generating functions can be used to find a solution to any linear recurrence. We’ll show you how this is done by means of a familiar example, the Fibonacci recurrence, so that you can more easily understand the similarities and differences of this approach and the method we showed you in Chapter 10.

12.5.1 Finding the Generating Function

Let's begin by recalling the definition of the Fibonacci numbers:

$$\begin{aligned} f_0 &= 0 \\ f_1 &= 1 \\ f_n &= f_{n-1} + f_{n-2} \qquad \text{for } n \geq 2. \end{aligned}$$

We can expand the final clause into an infinite sequence of equations. Thus, the Fibonacci numbers are defined by:

$$\begin{aligned} f_0 =&0 \\ f_1 =&1 \\ f_2 =&f_1 + f_0 \\ f_3 =&f_2 + f_1 \\ f_4 =&f_3 + f_2 \\ &\vdots \end{aligned}$$

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:

$$F(x) = f_0 + f_1x + f_2x^2 + f_3x^3 + f_4x^4 + \cdots .$$

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:

$$\begin{array}{rcccccl}
\langle 0, & 1, & 0, & 0, & 0, & \ldots \rangle & \longleftrightarrow\ x \\
\langle 0, & f_0, & f_1, & f_2, & f_3, & \ldots \rangle & \longleftrightarrow\ xF(x) \\
+\ \langle 0, & 0, & f_0, & f_1, & f_2, & \ldots \rangle & \longleftrightarrow\ x^2F(x) \\
\hline
\multicolumn{6}{r}{\langle 0, 1 + f_0, f_1 + f_0, f_2 + f_1, f_3 + f_2, \ldots \rangle} & \longleftrightarrow\ x + xF(x) + x^2F(x)
\end{array}$$

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.

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:

$$\begin{array}{rccccc}
F(x) & = f_0 + & f_1 \quad x + & f_2 \quad x^2 + & f_3 \quad x^3 + \cdots \\
\| & \| & \| & \| & \| \\
x + xF(x) + x^2F(x) & = \; 0 & + (1 + f_0)x & + (f_1 + f_0)x^2 & + (f_2 + f_1)x^3 + \cdots
\end{array}$$

Solving for $F(x)$ gives the generating function for the Fibonacci sequence:

$$F(x) = x + xF(x) + x^2F(x)$$

so

$$F(x) = \frac{x}{1 - x - x^2}. \tag{12.13}$$

This is pretty cool. After all, who would have thought that the Fibonacci numbers are precisely the coefficients of such a simple function? Even better, this function is a ratio of polynomials and so we can use the method of partial fractions from Section 12.4.4 to find a closed-form expression for the nth Fibonacci number.

12.5.2 Extracting the Coefficients

Repeated differentiation of Equation 12.13 would be very painful. But it is easy to use the method of partial fractions to compute the coefficients. Since the degree of the numerator in Equation 12.13 is less than the degree of the denominator, the first step is to factor the denominator:

$$1 - x - x^2 = (1 - \alpha_1 x)(1 - \alpha_2 x)$$

where $\alpha_1 = (1 + \sqrt{5})/2$ and $\alpha_2 = (1 - \sqrt{5})/2$. These are the same as the roots of the characteristic equation for the Fibonacci recurrence that we found in Chapter 10. That is not a coincidence.

The next step is to find c_1 and c_2 that satisfy

$$\begin{aligned}
\frac{x}{1 - x - x^2} &= \frac{c_1}{1 - \alpha_1 x} + \frac{c_2}{1 - \alpha_2 x} \\
&= \frac{c_1(1 - \alpha_2 x) + c_2(1 - \alpha_1 x)}{(1 - \alpha_1 x)(1 - \alpha_2 x)} \\
&= \frac{c_1 + c_2 - (c_1\alpha_2 + c_2\alpha_1)x}{1 - x - x^2}.
\end{aligned}$$

Hence,

$$c_1 + c_2 = 0 \qquad \text{and} \qquad -(c_1\alpha_2 + c_2\alpha_1) = 1.$$

Solving these equations, we find that

$$\begin{aligned} c_1 &= \frac{1}{\alpha_1 - \alpha_2} = \frac{1}{\sqrt{5}} \\ c_2 &= \frac{-1}{\alpha_1 - \alpha_2} = \frac{-1}{\sqrt{5}}. \end{aligned}$$

We can now use Corollary 12.4.3 and the Sum Rule to conclude that

$$\begin{aligned} f_n &= \frac{\alpha_1^n}{\sqrt{5}} - \frac{\alpha_2^n}{\sqrt{5}} \\ &= \frac{1}{\sqrt{5}} \left(\left(\frac{1+\sqrt{5}}{2} \right)^n - \left(\frac{1-\sqrt{5}}{2} \right)^n \right). \end{aligned}$$

This is exactly the same formula we derived for the nth Fibonacci number in Chapter 10.

12.5.3 General Linear Recurrences

The method that we just used to solve the Fibonacci recurrence can also be used to solve general linear recurrences of the form

$$f_n = a_1 f_{n-1} + a_2 f_{n-2} + \cdots + a_d f_{n-d} + g_n$$

for $n \geq d$. The generating function for $\langle f_0, f_1, f_2, \ldots \rangle$ is

$$F(x) = \frac{h(x) + G(x)}{1 - a_1 x - a_2 x^2 - \cdots - a_d x^d}$$

where $G(x)$ is the generating function for the sequence

$$\langle \overbrace{0, 0, \ldots, 0}^{d}, g_d, g_{d+1}, g_{d+2}, \ldots \rangle$$

and $h(x)$ is a polynomial of degree at most $d - 1$ that is based on the values of f_0, $f_1, \ldots, f_{d-1}$. In particular,

$$h(x) = \sum_{i=0}^{d-1} h_i x^i$$

where

$$h_i = f_0 - a_1 f_{i-1} - a_2 f_{i-2} - \cdots - a_i f_0$$

for $0 \leq i < d$.

To solve the recurrence, we use the method of partial fractions described in Section 12.4.4 to find a closed-form expression for $F(x)$. This can be easy or hard to do depending on $G(x)$.

12.6 Counting with Generating Functions

Generating functions are particularly useful for solving counting problems. In particular, problems involving choosing items from a set often lead to nice generating functions by letting the coefficient of x^n be the number of ways to choose n items.

12.6.1 Choosing Distinct Items from a Set

The generating function for binomial coefficients follows directly from the Binomial Theorem:

$$\begin{aligned}\left\langle \binom{k}{0}, \binom{k}{1}, \binom{k}{2}, \ldots, \binom{k}{k}, 0, 0, 0, \ldots \right\rangle \longleftrightarrow\ & \binom{k}{0} + \binom{k}{1}x + \binom{k}{2}x^2 + \cdots + \binom{k}{k}x^k \\ &= (1+x)^k\end{aligned}$$

Thus, the coefficient of x^n in $(1+x)^k$ is $\binom{k}{n}$, the number of ways to choose n distinct items[3] 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.

12.6.2 Building Generating Functions that Count

Often we can translate the description of a counting problem directly into a generating function for the solution. For example, we could figure out that $(1+x)^k$ generates the number of ways to select n distinct items from a k-element set without resorting to the Binomial Theorem or even fussing with binomial coefficients! Let's see 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

[3] Watch out for the 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.

n elements from the $\{a_1, a_2\}$ is:

$$\underbrace{(1+x)}_{\text{select from }\{a_1\}} \cdot \underbrace{(1+x)}_{\text{select from }\{a_2\}} = \underbrace{(1+x)^2}_{\text{select from }\{a_1, a_2\}} = 1 + 2x + x^2.$$

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 $\{a_1, a_2, \dots, a_k\}$:

$$\underbrace{(1+x)}_{\text{select from }\{a_1\}} \cdot \underbrace{(1+x)}_{\text{select from }\{a_2\}} \cdots \underbrace{(1+x)}_{\text{select from }\{a_k\}} = \underbrace{(1+x)^k}_{\substack{\text{select from}\\ \{a_1, a_2, \dots, a_k\}}}$$

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 12.6.1 (Convolution Rule)**.** *Let $A(x)$ be the generating function for selecting items from set $\mathcal{A}$, and let $B(x)$ be the generating function for selecting items from set $\mathcal{B}$. If $\mathcal{A}$ and $\mathcal{B}$ are disjoint, then the generating function for selecting items from the union $\mathcal{A} \cup \mathcal{B}$ is the product $A(x) \cdot B(x)$.*

This rule is rather ambiguous: what exactly are the rules governing the selection of items from a set? Remarkably, the Convolution Rule remains valid under *many* interpretations of selection. For example, we could insist that distinct items be selected or we might allow the same item to be picked a limited number of times or any number of times. Informally, the only restrictions are that (1) the order in which items are selected is disregarded and (2) restrictions on the selection of items from sets $\mathcal{A}$ and $\mathcal{B}$ also apply in selecting items from $\mathcal{A} \cup \mathcal{B}$. (Formally, there must be a bijection between n-element selections from $\mathcal{A} \cup \mathcal{B}$ and ordered pairs of selections from $\mathcal{A}$ and $\mathcal{B}$ containing a total of n elements.)

To count the number of ways to select n items from $\mathcal{A} \cup \mathcal{B}$, we observe that we can select n items by choosing j items from $\mathcal{A}$ and $n - j$ items from $\mathcal{B}$, where j is any number from 0 to n. This can be done in $a_j b_{n-j}$ ways. Summing over all the possible values of j gives a total of

$$a_0 b_n + a_1 b_{n-1} + a_2 b_{n-2} + \cdots + a_n b_0$$

ways to select n items from $\mathcal{A} \cup \mathcal{B}$. By the Product Rule, this is precisely the coefficient of x^n in the series for $A(x)B(x)$.

12.6.3 Choosing Items with Repetition

The first counting problem we considered was the number of ways to select a dozen doughnuts when five flavors were available. We can generalize this question as follows: in how many ways can we select n items from a k-element set if we're allowed to pick the same item multiple times? In these terms, the doughnut problem asks how many ways we can select $n = 12$ doughnuts from the set of $k = 5$ flavors

$$\{\text{chocolate, lemon-filled, sugar, glazed, plain}\}$$

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:

$$\langle 1, 1, 1, 1, \ldots \rangle \longleftrightarrow 1 + x + x^2 + x^3 + \cdots = \frac{1}{1-x}.$$

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:

$$\underbrace{\frac{1}{1-x}}_{\text{choose } a_1\text{'s}} \cdot \underbrace{\frac{1}{1-x}}_{\text{choose } a_2\text{'s}} \cdots \underbrace{\frac{1}{1-x}}_{\text{choose } a_k\text{'s}} = \underbrace{\frac{1}{(1-x)^k}}_{\substack{\text{repeatedly choose from} \\ \{a_1, a_2, \ldots, a_k\}}}$$

Therefore, the generating function for choosing items from a k-element set with repetition allowed is $1/(1-x)^k$. Computing derivatives and applying the Taylor Series Rule, we can find that the coefficient of x^n in $1/(1-x)^k$ is

$$\binom{n+k-1}{n}.$$

This is the Bookkeeper Rule from Chapter 11—namely there are $\binom{n+k-1}{n}$ ways to select n items with replication from a set of k items.

12.6.4 Fruit Salad

In this chapter, we have covered a lot of methods and rules for using generating functions. We'll now do an example that demonstrates how the rules and methods can be combined to solve a more challenging problem—making fruit salad.

In how many ways can we make a salad 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 make a salad 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 generating functions can solve the problem in a straightforward way.

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:

$$B(x) = 1 + x^5 + x^{10} + x^{15} + \cdots = \frac{1}{1 - x^5}.$$

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 make a salad with n fruits is just $n+1$. This is consistent with the example we worked out at the start, since there were 7 different salads containing 6 fruits. *Amazing!*

13 Infinite Sets

So you might be wondering how much is there to say about an infinite set other than, well, it has an infinite number of elements. Of course, an infinite set does have an infinite number of elements, but it turns out that not all infinite sets have the same size—some are bigger than others! And, understanding infinity is not as easy as you might think. Some of the toughest questions in mathematics involve infinite sets.

Why should you care? Indeed, isn't computer science only about finite sets? Not exactly. For example, we deal with the set of natural numbers $\mathbb{N}$ all the time and it is an infinite set. In fact, that is why we have induction: to reason about predicates over $\mathbb{N}$. Infinite sets are also important in Part IV of the text when we talk about random variables over potentially infinite sample spaces.

So sit back and open your mind for a few moments while we take a very brief look at *infinity*.

13.1 Injections, Surjections, and Bijections

We know from Theorem 7.2.1 that if there is an injection or surjection between two finite sets, then we can say something about the relative sizes of the two sets. The same is true for infinite sets. In fact, relations are the primary tool for determining the relative size of infinite sets.

Definition 13.1.1. Given any two sets A and B, we say that

A surj B iff there is a surjection from A to B,
A inj B iff there is an injection from A to B,
A bij B iff there is a bijection between A and B, and
A strict B iff there is a surjection from A to B but there is *no* bijection from B to A.

Restating Theorem 7.2.1 with this new terminology, we have:

Theorem 13.1.2. *For any pair of* finite *sets A and B,*

$$\begin{aligned}
|A| \geq |B| \quad &\textit{iff} \quad A \text{ surj } B,\\
|A| \leq |B| \quad &\textit{iff} \quad A \text{ inj } B,\\
|A| = |B| \quad &\textit{iff} \quad A \text{ bij } B,\\
|A| > |B| \quad &\textit{iff} \quad A \text{ strict } B.
\end{aligned}$$

Theorem 13.1.2 suggests a way to generalize size comparisons to infinite sets; namely, we can think of the relation surj as an "at least as big" 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" relation between sets.

Note that we haven't, and won't, define what the size of an infinite set 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. 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:

Theorem 13.1.3. *For any sets, A, B, and C,*

1. A surj B *and* B surj C IMPLIES A surj C.
2. A bij B *and* B bij C IMPLIES A bij C.
3. A bij B IMPLIES B bij A.

Parts 1 and 2 of Theorem 13.1.3 follow immediately from the fact that compositions of surjections are surjections, and likewise for bijections. Part 3 follows from the fact that the inverse of a bijection is a bijection. We'll leave a proof of these facts to the problems.

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

Theorem 13.1.4 (Schröder-Bernstein)**.** *For any pair of sets A and B, if A* surj *B and B* surj *A, then A* bij *B.*

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 $h : A \rightarrow B$. The challenge is to construct h from parts of both f and g. We’ll leave the actual construction to the problems.

13.1.1 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!

Theorem 13.1.5. *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. Since A is infinite, it certainly has at least one element; call it a_0. 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. 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 this 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 $f : A \cup \{b\} \rightarrow A$:

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

■

13.2 Countable Sets

13.2.1 Definitions

A set C is *countable* iff its elements can be listed in order, that is, the distinct elements in C are precisely

$$c_0, c_1, \ldots, c_n, \ldots.$$

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 13.2.1. A set C is *countably infinite* iff $\mathbb{N}$ bij C. A set is *countable* iff it is finite or countably infinite.

Discrete mathematics is often defined as the mathematics of countable sets and so it is probably worth spending a little time understanding what it means to be countable and why countable sets are so special. For example, a small modification of the proof of Theorem 13.1.5 shows that countably infinite sets are the "smallest" infinite sets; namely, if A is any infinite set, then A surj $\mathbb{N}$.

13.2.2 Unions

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—just because it's ok to do something any finite number of times doesn't make it ok to do it an infinite number of times.

For example, suppose that you have two countably infinite sets $A = \{a_0, a_1, a_2, \dots\}$ and $B = \{b_0, b_1, b_2, \dots\}$. You might try to show that $A \cup B$ is countable by making the following "list" for $A \cup B$:

$$a_0, a_1, a_2, \dots, b_0, b_1, b_2, \dots \tag{13.1}$$

But this is not a valid argument because Equation 13.1 is not a list. The key property required for listing the elements in a countable set is that for any element in the set, you can determine its finite index in the list. For example, a_i shows up in position i in Equation 13.1, but there is no index in the supposed "list" for any of the b_i. Hence, Equation 13.1 is not a valid list for the purposes of showing that $A \cup B$ is countable when A is infinite. Equation 13.1 is only useful when A is finite.

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.

Theorem 13.2.2. *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, ..., and the list of B is b_0, b_1, ... Then a valid way to list all the elements of $A \cup B$ is

$$a_0, b_0, a_1, b_1, \dots, a_n, b_n, \dots. \tag{13.2}$$

Of course this list will contain duplicates if A and B have elements in common, but then deleting all but the first occurrence of each element in Equation 13.2 leaves a list of all the distinct elements of A and B. ■

Note that the list in Equation 13.2 does not have the same defect as the purported "list" in Equation 13.1, since every item in $A \cup B$ has a finite index in the list created in Theorem 13.2.2.

$$\begin{array}{c|ccccc}
 & b_0 & b_1 & b_2 & b_3 & \dots \\
\hline
a_0 & c_0 & c_1 & c_4 & c_9 & \\
a_1 & c_3 & c_2 & c_5 & c_{10} & \\
a_2 & c_8 & c_7 & c_6 & c_{11} & \\
a_3 & c_{15} & c_{14} & c_{13} & c_{12} & \\
\vdots & & & & & \ddots
\end{array}$$

Figure 13.1 A listing of the elements of $C = A \times B$ where $A = \{a_0, a_1, a_2, \dots\}$ and $B = \{b_0, b_1, b_2, \dots\}$ are countably infinite sets. For example, $c_5 = (a_1, b_2)$.

13.2.3 Cross Products

Somewhat surprisingly, cross products of countable sets are also countable. At first, you might be tempted to think that “infinity times infinity” (whatever that means) somehow results in a larger infinity, but this is not the case.

Theorem 13.2.3. *The cross product of two countable sets is countable.*

Proof. Let A and B be any pair of countable sets. To show that $C = A \times B$ is also countable, we need to find a listing of the elements

$$\{\, (a, b) \mid a \in A, b \in B \,\}.$$

There are many such listings. One is shown in Figure 13.1 for the case when A and B are both infinite sets. In this listing, (a_i, b_j) is the kth element in the list for C where

$$\begin{aligned}
&a_i \text{ is the } i\text{th element in } A,\\
&b_j \text{ is the } j\text{th element in } B, \text{ and}\\
&k = \max(i, j)^2 + i + \max(i - j, 0).
\end{aligned}$$

The task of finding a listing when one or both of A and B are finite is left to the problems at the end of the chapter.

■

13.2.4 $\mathbb{Q}$ Is Countable

Theorem 13.2.3 also has a surprising Corollary; namely that the set of rational numbers is countable.

Corollary 13.2.4. *The set of rational numbers $\mathbb{Q}$ is countable.*

Proof. Since $\mathbb{Z} \times \mathbb{Z}$ is countable by Theorem 13.2.3, it suffices to find a surjection f from $\mathbb{Z} \times \mathbb{Z}$ to $\mathbb{Q}$. This is easy to to since

$$f(a,b) = \begin{cases} a/b & \text{if } b \neq 0 \\ 0 & \text{if } b = 0 \end{cases}$$

is one such surjection. ■

At this point, you may be thinking that every set is countable. That is *not* the case. In fact, as we will shortly see, there are many infinite sets that are uncountable, including the set of real numbers $\mathbb{R}$.

13.3 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*.

Theorem 13.3.1. *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 $a \in A$ 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

$$A_g ::= \{\, a \in A \mid a \notin g(a) \,\}.$$

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. ■

13.3.1 $\mathbb{R}$ Is Uncountable

To prove that the set of real numbers is uncountable, we will show that there is a surjection from $\mathbb{R}$ to $\mathcal{P}(\mathbb{N})$ and then apply Theorem 13.3.1 to $\mathcal{P}(\mathbb{N})$.

Lemma 13.3.2. $\mathbb{R}$ surj $\mathcal{P}(\mathbb{N})$.

Proof. Let $A \subset \mathbb{N}$ be any subset of the natural numbers. Since $\mathbb{N}$ is countable, this means that A is countable and thus that $A = \{a_0, a_1, a_2, \ldots\}$. For each $i \geq 0$, define $\text{bin}(a_i)$ to be the binary representation of a_i. Let x_A be the real number using only digits 0, 1, 2 as follows:

$$x_A ::= 0.2\,\text{bin}(a_0)2\,\text{bin}(a_1)2\,\text{bin}(a_2)2\ldots \tag{13.3}$$

We can then define a surjection $f : \mathbb{R} \to \mathcal{P}(\mathbb{N})$ as follows:

$$f(x) = \begin{cases} A & \text{if } x = x_A \text{ for some } A \in \mathbb{N}, \\ 0 & \text{otherwise.} \end{cases}$$

Hence $\mathbb{R}$ surj $\mathcal{P}(\mathbb{N})$. ■

Corollary 13.3.3. $\mathbb{R}$ *is uncountable.*

Proof. By contradiction. Assume $\mathbb{R}$ is countable. Then $\mathbb{N}$ surj $\mathbb{R}$. By Lemma 13.3.2, $\mathbb{R}$ surj $\mathcal{P}(\mathbb{N})$. Hence $\mathbb{N}$ surj $\mathcal{P}(\mathbb{N})$. This contradicts Theorem 13.3.1 for the case when $A = \mathbb{N}$. ■

So the set of rational numbers and the set of natural numbers have the same size, but the set of real numbers is strictly larger. In fact, $\mathbb{R}$ bij $\mathcal{P}(N)$, but we won't prove that here.

Is there anything bigger?

13.3.2 Even 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 13.3.1, each of these sets is strictly bigger than all the preceding ones. But that's not all, the union of all the sets in the sequence is strictly bigger than each set in the sequence. In this way, you can keep going, building still bigger infinities.

13.3.3 The Continuum Hypothesis

Georg Cantor was the mathematician 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" infinite set, $\mathbb{N}$, and $\mathcal{P}(\mathbb{N})$. He guessed not:

Cantor's Continuum Hypothesis. *There is no set A such that $\mathcal{P}(\mathbb{N})$ is strictly bigger than A and A is strictly bigger than $\mathbb{N}$.*

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 1930s and Paul Cohen in the 1960s—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.

13.4 Infinities in Computer Science

If the romance of different size infinities and continuum hypotheses doesn't appeal to you, not knowing about them is not going to lower your professional abilities as a computer scientist. These abstract issues about infinite sets rarely come up in mainstream mathematics, and they don't come up at all in computer science, where the focus is generally on countable, and often just finite, sets. In practice, only logicians and set theorists have to worry about collections that are too big to be sets. In fact, at the end of the 19th century, even the general mathematical community doubted the relevance of what they called "Cantor's paradise" of unfamiliar sets of arbitrary infinite size.

That said, it is worth noting that the proof of Theorem 13.3.1 gives the simplest form of what is known as a "diagonal argument." Diagonal arguments are used to prove many fundamental results about the limitations of computation, such as the undecidability of the Halting Problem for programs 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. Ad a well-educated computer scientist will be comfortable dealing with countable sets, finite as well as infinite.

IV Probability

Introduction

Probability is one of the most important disciplines in all of the sciences. It is also one of the least well understood.

Probability is especially important in computer science—it arises in virtually every branch of the field. In algorithm design and game theory, for example, *randomized* algorithms and strategies (those that use a random number generator as a key input for decision making) frequently outperform deterministic algorithms and strategies. In information theory and signal processing, an understanding of randomness is critical for filtering out noise and compressing data. In cryptography and digital rights management, probability is crucial for achieving security. The list of examples is long.

Given the impact that probability has on computer science, it seems strange that probability should be so misunderstood by so many. Perhaps the trouble is that basic human intuition is wrong as often as it is right when it comes to problems involving random events. As a consequence, many students develop a fear of probability. Indeed, we have witnessed many graduate oral exams where a student will solve the most horrendous calculation, only to then be tripped up by the simplest probability question. Indeed, even some faculty will start squirming if you ask them a question that starts "What is the probability that. . . ?"

Our goal in the remaining chapters is to equip you with the tools that will enable you to easily and confidently solve problems involving probability.

We begin in Chapter 14 with the basic definitions and an elementary 4-step process that can be used to determine the probability that a specified event occurs. We illustrate the method on two famous problems where your intuition will probably fail you.

In Chapter 15, we describe conditional probability and the notion of independence. Both notions are important, and sometimes misused, in practice. We will consider the probability of having a disease given that you tested positive, and the probability that a suspect is guilty given that his blood type matches the blood found at the scene of the crime.

We study random variables and distributions in Chapter 17. Random variables provide a more quantitative way to measure random events. For example, instead of determining the probability that it will rain, we may want to determine *how much* or *how long* it is likely to rain. This is closely related to the notion of the expected value of a random variables, which we will consider in Chapter 18.

In Chapter 19, we examine the probability that a random variable deviates significantly from its expected value. This is especially important in practice, where things are generally fine if they are going according to expectation, and you would like to be assured that the probability of deviating from the expectation is very low.

We conclude in Chapter 20 by combining the tools we have acquired to solve problems involving more complex random processes. We will see why you will probably never get very far ahead at the casino, and how two Stanford graduate students became gazillionaires by combining graph theory and probability theory to design a better search engine for the web.

14 Events and Probability Spaces

14.1 Let's Make a Deal

In the September 9, 1990 issue of *Parade* magazine, 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 in 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 became known as the *Monty Hall Problem* and it 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 that we will describe shortly. First, let's make sure we really understand the setup for this problem. This is always a good thing to do when you are dealing with probability.

14.1.1 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. For example, we will assume that:

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?”

14.2 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.

14.2.1 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.

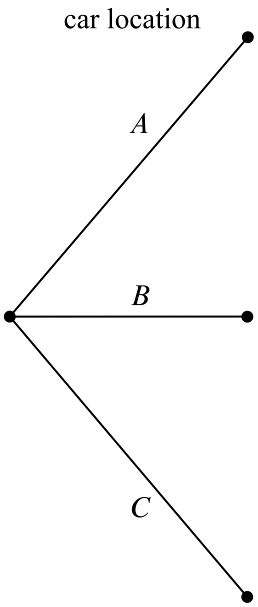


Figure 14.1 The first level in a tree diagram for the Monty Hall Problem. The branches correspond to the door behind which the car is located.

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, as shown in Figure 14.1. 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.

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, as shown in Figure 14.2.

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

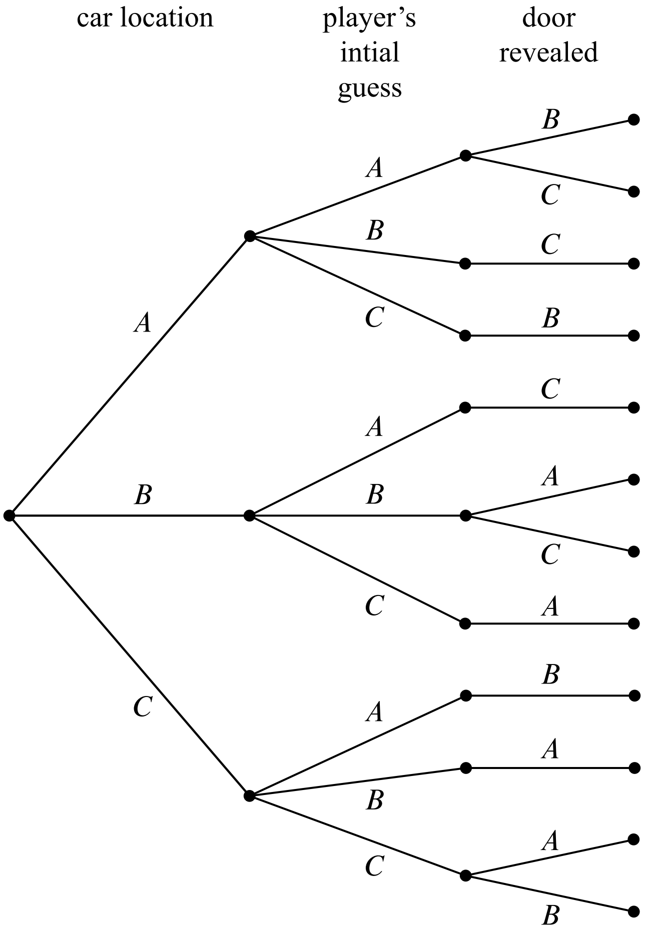


Figure 14.2 The full tree diagram for the Monty Hall Problem. The second level indicates the door initially chosen by the player. The third level indicates the door revealed by Monty Hall.

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 in Figure 14.3 with a triple of doors indicating:

(door concealing prize, door initially chosen, door opened to reveal a goat).

In these terms, the sample space is the set

$$\mathcal{S} = \left\{ \begin{array}{l} (A, A, B),\ (A, A, C),\ (A, B, C),\ (A, C, B),\ (B, A, C),\ (B, B, A), \\ (B, B, C),\ (B, C, A),\ (C, A, B),\ (C, B, A),\ (C, C, A),\ (C, C, B) \end{array} \right\}$$

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.

14.2.2 Step 2: Define Events of Interest

Our objective is to answer questions of the form "What is the probability that ... ?", where, for example, 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". Each of these phrases characterizes a set of outcomes. For example, 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* and it is a subset of the sample space. 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:

$$\{(A, B, C), (A, C, B), (B, A, C), (B, C, A), (C, A, B), (C, B, A)\}.$$

These outcomes are denoted with a check mark in Figure 14.4.

Notice that exactly half of the outcomes are checked, 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.

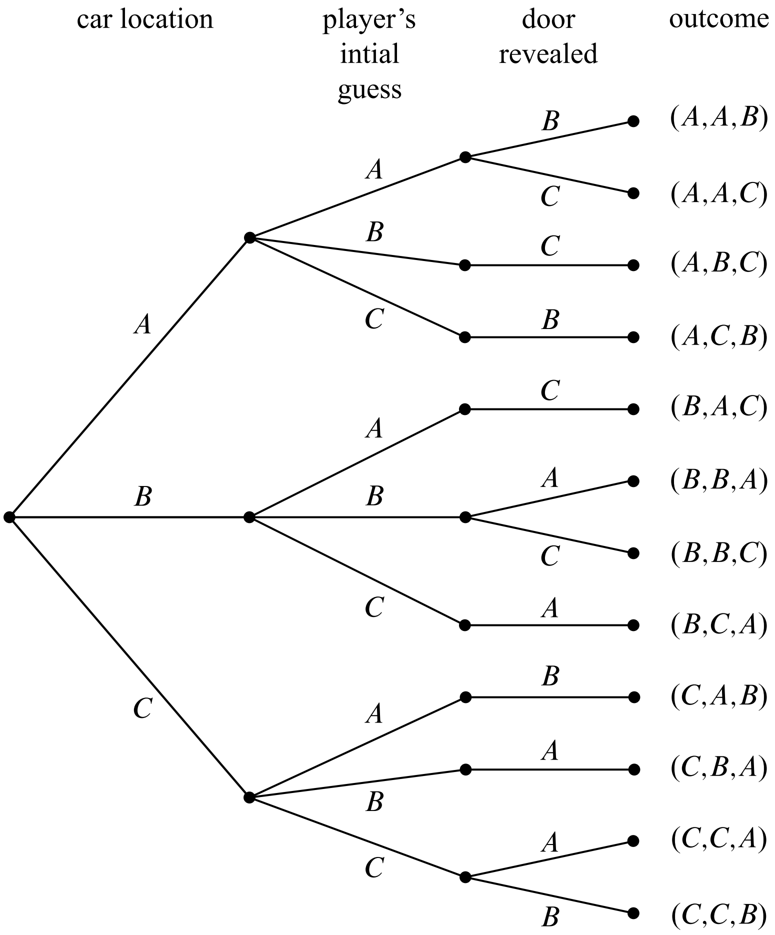


Figure 14.3 The tree diagram for the Monty Hal Problem with the outcomes labeled for each path from root to leaf. For example, outcome (A, A, B) corresponds to the car being behind door A, the player initially choosing door A, and Monty Hall revealing the goat behind door B.

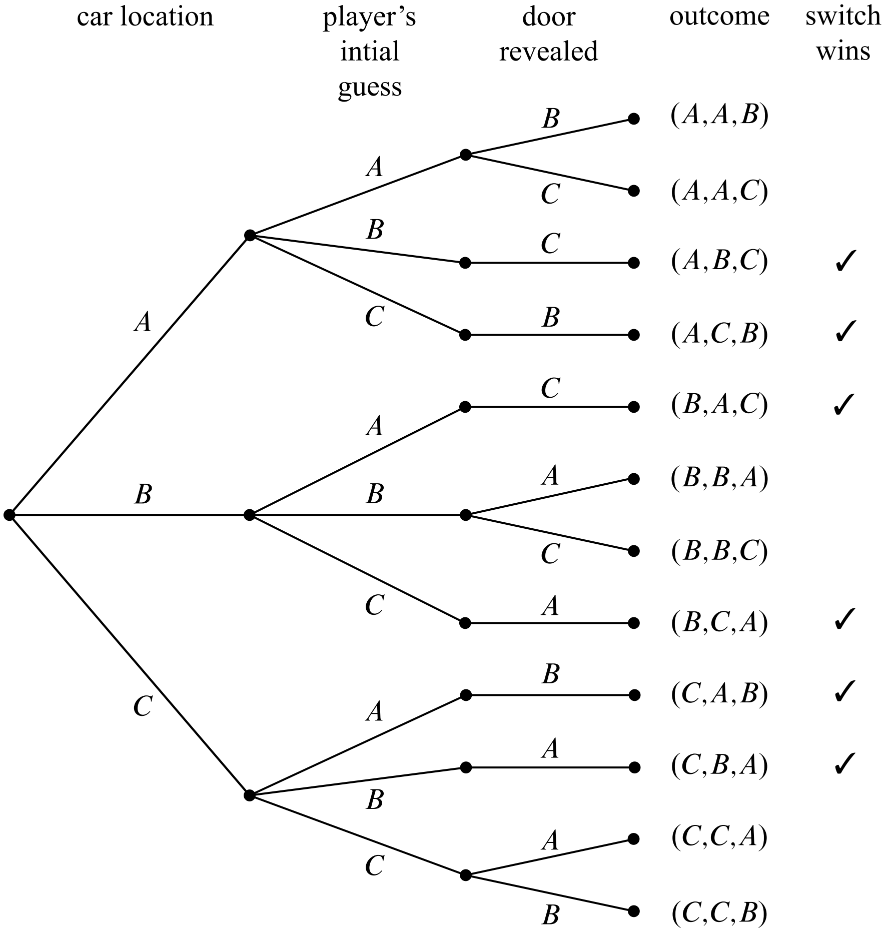


Figure 14.4 The tree diagram for the Monty Hall Problem where the outcomes in the event where the player wins by switching are denoted with a check mark.

14.2.3 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. For example, see Figure 14.5.

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 in Figure 14.5, (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 path 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.

How likely is such a path to arrive at the topmost outcome, (A, A, B)? Well, there is a 1-in-3 chance that a path 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 1 path in 18 should arrive at the (A, A, B) leaf, which is precisely the probability we assign it.

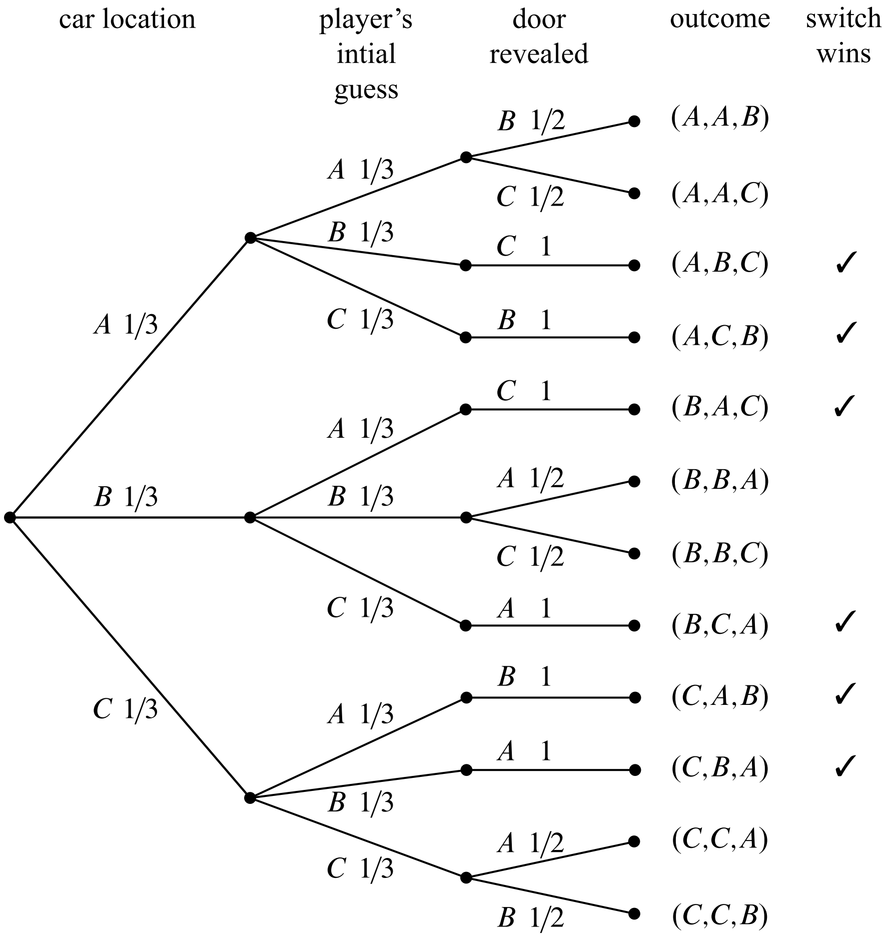


Figure 14.5 The tree diagram for the Monty Hall Problem where edge weights denote the probability of that branch being taken given that we are at the parent of that branch. For example, if the car is behind door A, then there is a 1/3 chance that the player’s initial selection is door B.

We have illustrated all of the outcome probabilities in Figure 14.6.

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:

$$\begin{aligned}
\Pr[(A,A,B)] &= \frac{1}{18},\\
\Pr[(A,A,C)] &= \frac{1}{18},\\
\Pr[(A,B,C)] &= \frac{1}{9},\\
&\text{etc.}
\end{aligned}$$

14.2.4 Step 4: Compute Event Probabilities

We now have a probability for each *outcome*, but we want to determine the probability of an *event*. The probability of an event E is denoted by $\Pr[E]$ and it is the sum of the probabilities of the outcomes in E. For example, the probability of the event that the player wins by switching is:[1]

$$\begin{aligned}
\Pr[\text{switching wins}] &= \Pr[(A,B,C)] + \Pr[(A,C,B)] + \Pr[(B,A,C)] + \\
&\qquad \Pr[(B,C,A)] + \Pr[(C,A,B)] + \Pr[(C,B,A)]\\
&= \frac{1}{9} + \frac{1}{9} + \frac{1}{9} + \frac{1}{9} + \frac{1}{9} + \frac{1}{9}\\
&= \frac{2}{3}.
\end{aligned}$$

It seems Marilyn's answer is correct! A player who switches doors wins the car with probability $2/3$. In contrast, a player who stays with his or her original door wins with probability $1/3$, since staying wins if and only if switching loses.

We're done with the problem! We didn't need any appeals to intuition or ingenious analogies. In fact, no mathematics more difficult than adding and multiplying fractions was required. The only hard part was resisting the temptation to leap to an "intuitively obvious" answer.

14.2.5 An Alternative Interpretation of the Monty Hall Problem

Was Marilyn really right? Our analysis indicates that she was. But a more accurate conclusion is that her answer is correct *provided we accept her interpretation of the*

[1] "Switching wins" is shorthand for the set of outcomes where switching wins; namely, $\{(A,B,C), (A,C,B), (B,A,C), (B,C,A), (C,A,B), (C,B,A)\}$. We will frequently use such shorthand to denote events.

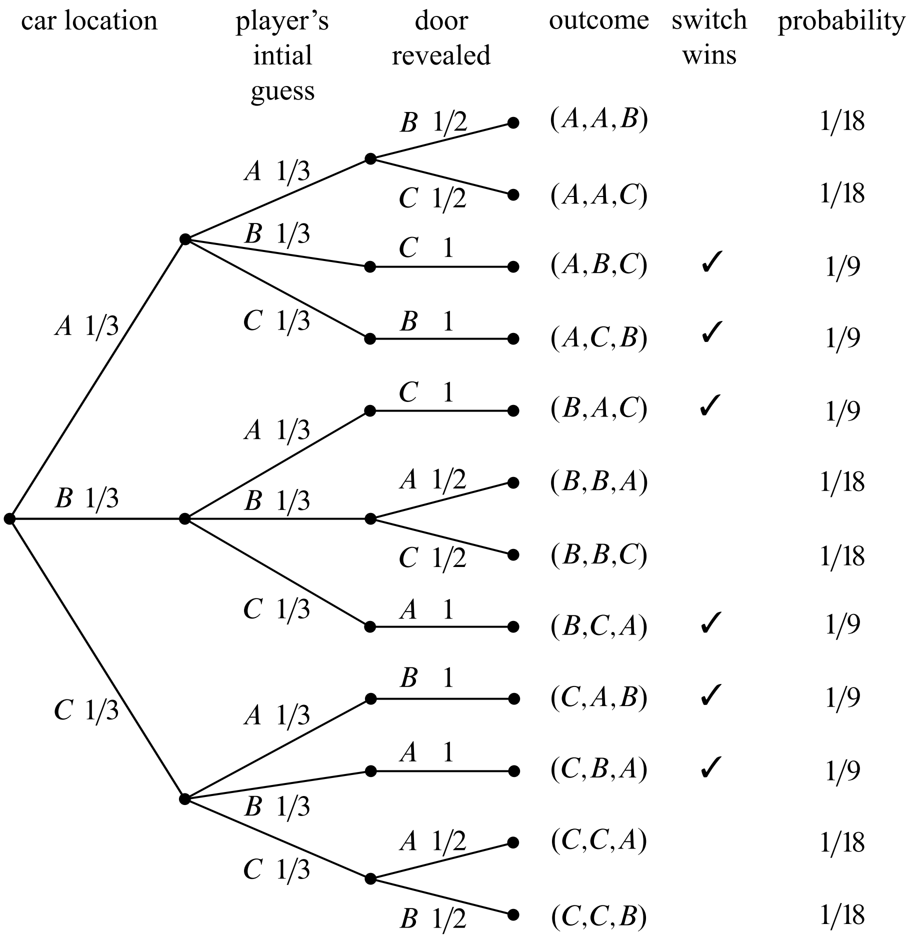


Figure 14.6 The rightmost column shows the outcome probabilities for the Monty Hall Problem. Each outcome probability is simply the product of the probabilities on the branches on the path from the root to the leaf for that outcome.

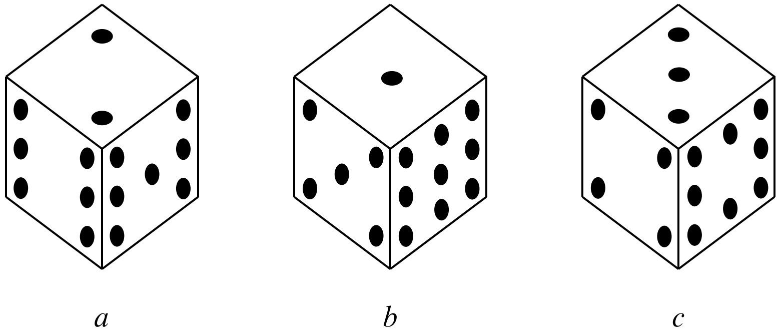


Figure 14.7 The strange dice. The number of pips on each concealed face is the same as the number on the opposite face. For example, when you roll die A, the probabilities of getting a 2, 6, or 7 are each $1/3$.

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!

14.3 Strange Dice

The four-step method is surprisingly powerful. Let's get some more practice with it. Imagine, if you will, the following scenario.

It's a typical Saturday night. You're at your favorite pub, contemplating the true meaning of infinite cardinalities, when a burly-looking biker plops down on the stool next to you. Just as you are about to get your mind around $\mathcal{P}(\mathcal{P}(\mathbb{R}))$, biker dude slaps three strange-looking dice on the bar and challenges you to a \$100 wager.

The rules are simple. Each player selects one die and rolls it once. The player with the lower value pays the other player \$100.

Naturally, you are skeptical. A quick inspection reveals that these are not ordinary dice. They each have six sides, but the numbers on the dice are different, as shown in Figure 14.7.

Biker dude notices your hesitation and so he offers to let you pick a die first, and then he will choose his die from the two that are left. That seals the deal since you figure that you now have an advantage.

But which of the dice should you choose? Die B is appealing because it has a 9, which is a sure winner if it comes up. Then again, die A has two fairly large numbers and die B has an 8 and no really small values.

In the end, you choose die B because it has a 9, and then biker dude selects die A. Let’s see what the probability is that you will win.[2] Not surprisingly, we will use the four-step method to compute this probability.

14.3.1 Die A versus Die B

Step 1: Find the sample space.

The sample space for this experiment is worked out in the tree diagram shown in Figure 14.8.[3]

For this experiment, the sample space is a set of nine outcomes:

$$\mathcal{S} = \{\,(2,1),\ (2,5),\ (2,9),\ (6,1),\ (6,5),\ (6,9),\ (7,1),\ (7,5),\ (7,9)\,\}.$$

Step 2: Define events of interest.

We are interested in the event that the number on die A is greater than the number on die B. This event is a set of five outcomes:

$$\{\,(2,1),\ (6,1),\ (6,5),\ (7,1),\ (7,5)\,\}.$$

These outcomes are marked A in the tree diagram in Figure 14.8.

Step 3: Determine outcome probabilities.

To find outcome probabilities, we first assign probabilities to edges in the tree diagram. Each number on each die comes up with probability $1/3$, regardless of the value of the other die. Therefore, we assign all edges probability $1/3$. The probability of an outcome is the product of the probabilities on the corresponding root-to-leaf path, which means that every outcome has probability $1/9$. These probabilities are recorded on the right side of the tree diagram in Figure 14.8.

Step 4: Compute event probabilities.

The probability of an event is the sum of the probabilities of the outcomes in that event. In this case, all the outcome probabilities are the same. In general, when the probability of every outcome is the same, we say that the sample space is *uniform*. Computing event probabilities for uniform sample spaces is particularly easy since

[2] Of course, you probably should have done this before picking die B in the first place.

[3] Actually, the whole probability space is worked out in this one picture. But pretend that each component sort of fades in—nyyrrroom!—as you read about the corresponding step below.

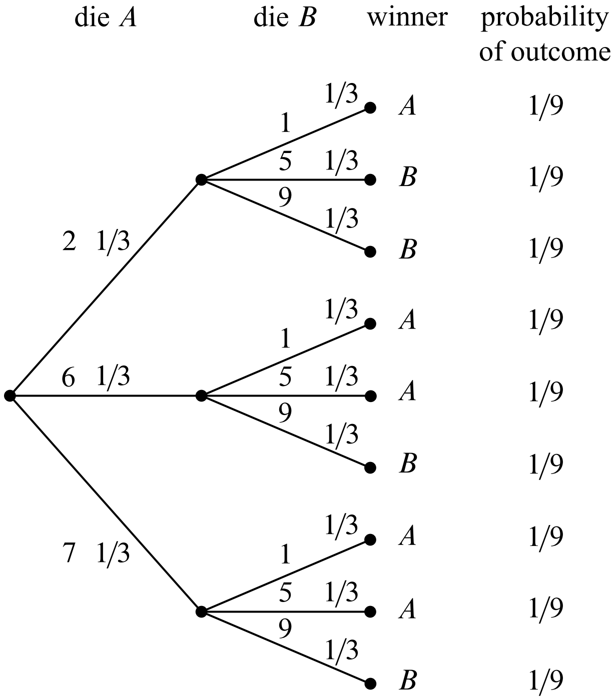


Figure 14.8 The tree diagram for one roll of die A versus die B. Die A wins with probability $5/9$.

you just have to compute the number of outcomes in the event. In particular, for any event E in a uniform sample space $\mathcal{S}$,

$$\Pr[E] = \frac{|E|}{|\mathcal{S}|}. \tag{14.1}$$

In this case, E is the event that die A beats die B, so $|E| = 5$, $|\mathcal{S}| = 9$, and

$$\Pr[E] = 5/9.$$

This is bad news for you. Die A beats die B more than half the time and, not surprisingly, you just lost \$100.

Biker dude consoles you on your "bad luck" and, given that he's a sensitive guy beneath all that leather, he offers to go double or nothing.[4] Given that your wallet only has \$25 in it, this sounds like a good plan. Plus, you figure that choosing die A will give *you* the advantage.

So you choose A, and then biker dude chooses C. Can you guess who is more likely to win? (Hint: it is generally not a good idea to gamble with someone you don't know in a bar, especially when you are gambling with strange dice.)

14.3.2 Die A versus Die C

We can construct the three diagram and outcome probabilities as before. The result is shown in Figure 14.9 and there is bad news again. Die C will beat die A with probability $5/9$, and you lose once again.

You now owe the biker dude \$200 and he asks for his money. You reply that you need to go to the bathroom.

Being a sensitive guy, biker dude nods understandingly and offers yet another wager. This time, he'll let you have die C. He'll even let you raise the wager to \$200 so you can win your money back.

This is too good a deal to pass up. You know that die C is likely to beat die A and that die A is likely to beat die B, and so die C is *surely* the best. Whether biker dude picks A or B, the odds are *surely* in your favor this time. Biker dude must really be a nice guy.

So you pick C, and then biker dude picks B. Let's use the tree method to figure out the probability that you win.

[4] *Double or nothing* is slang for doing another wager after you have lost the first. If you lose again, you will owe biker dude *double* what you owed him before. If you win, you will now be even and you will owe him *nothing*.

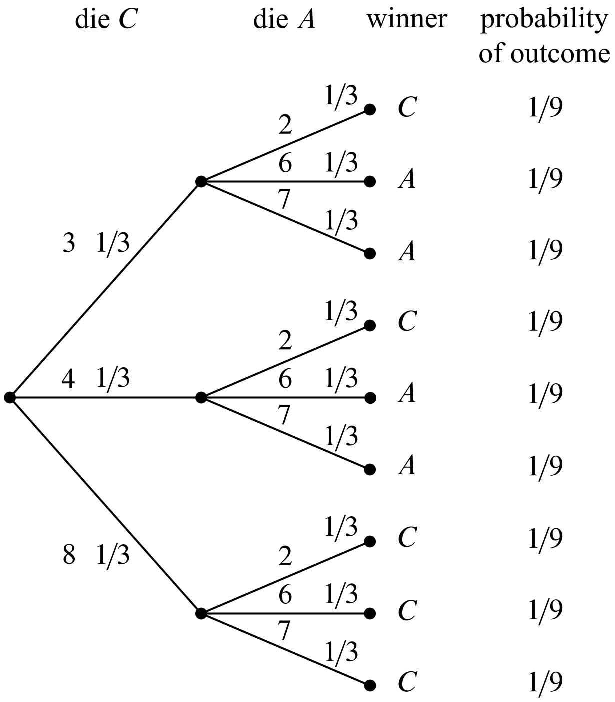


Figure 14.9 The tree diagram for one roll of die C versus die A. Die C wins with probability $5/9$.

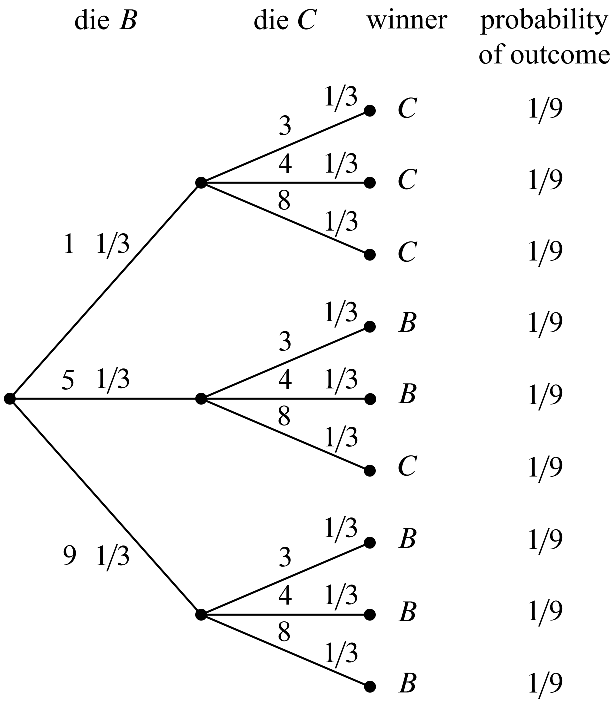


Figure 14.10 The tree diagram for one roll of die B versus die C. Die B wins with probability $5/9$.

14.3.3 Die B versus Die C

The tree diagram and outcome probabilities for B versus C are shown in Figure 14.10. But surely there is a mistake! The data in Figure 14.10 shows that *die B* wins with probability $5/9$. How is it possible that

C beats A with probability $5/9$,

A beats B with probability $5/9$, and

B beats C with probability $5/9$?

The problem is not with the math, but with your intuition. It *seems* that the “likely-to-beat” relation should be transitive. But it is not, and whatever die you pick, biker dude can pick one of the others and be likely to win. So picking first is a big disadvantage and you now owe biker dude $400.

Just when you think matters can’t get worse, biker dude offers you one final wager for $1,000. This time, you demand to choose second. Biker dude agrees,

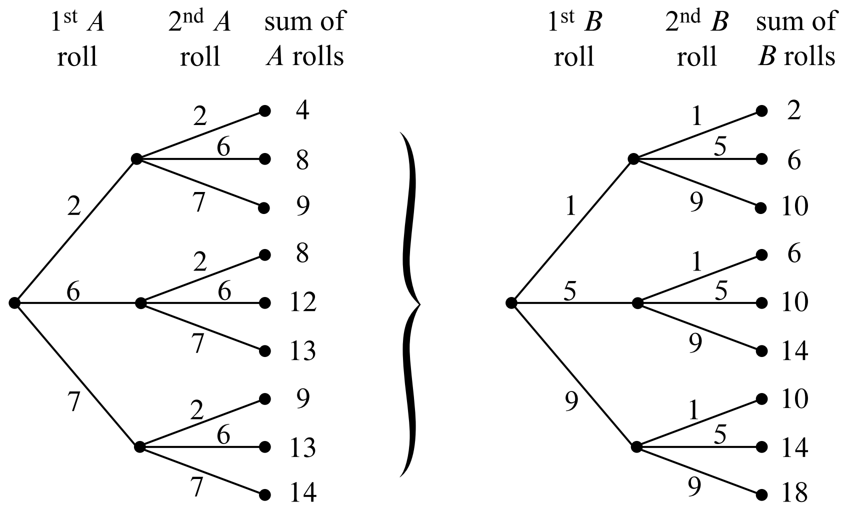


Figure 14.11 Parts of the tree diagram for die B versus die A where each die is rolled twice. The first two levels are shown in (a). The last two levels consist of nine copies of the tree in (b).

but with the condition that instead of rolling each die once, you each roll your die twice and your score is the sum of your rolls.

Believing that you finally have a winning wager, you agree.[5] Biker dude chooses die B and, of course, you grab die A. That's because you know that die A will beat die B with probability $5/9$ on one roll and so *surely* two rolls of die A are likely to beat two rolls of die B, right?

Wrong!

14.3.4 Rolling Twice

If each player rolls twice, the tree diagram will have four levels and $3^4 = 81$ outcomes. This means that it will take a while to write down the entire tree diagram. We can, however, easily write down the first two levels (as we have done in Figure 14.11(a)) and then notice that the remaining two levels consist of nine identical copies of the tree in Figure 14.11(b).

The probability of each outcome is $(1/3)^4 = 1/81$ and so, once again, we have a uniform probability space. By Equation 14.1, this means that the probability that A wins is the number of outcomes where A beats B divided by 81.

To compute the number of outcomes where A beats B, we observe that the sum

[5] Did we mention that playing strange gambling games with strangers in a bar is a bad idea?

of the two rolls of die A is equally likely to be any element of the following multiset:

$$\mathcal{S}_A = \{4, 8, 8, 9, 9, 12, 13, 13, 14\}.$$

The sum of two rolls of die B is equally likely to be any element of the following multiset:

$$\mathcal{S}_B = \{2, 6, 6, 10, 10, 10, 14, 14, 18\}.$$

We can treat each outcome as a pair $(x, y) \in \mathcal{S}_A \times \mathcal{S}_B$, where A wins iff $x > y$. If $x = 4$, there is only one y (namely $y = 2$) for which $x > y$. If $x = 8$, there are three values of y for which $x > y$. Continuing the count in this way, the number of pairs for which $x > y$ is

$$1 + 3 + 3 + 3 + 3 + 6 + 6 + 6 + 6 = 37.$$

A similar count shows that there are 42 pairs for which $x > y$, and there are two pairs $((14, 14), (14, 14))$ which result in ties. This means that A *loses* to B with probability $42/81 > 1/2$ and ties with probability $2/81$. Die A wins with probability only $37/81$.

How can it be that A is more likely than B to win with 1 roll, but B is more likely to win with 2 rolls?!? Well, why not? The only reason we'd think otherwise is our (faulty) intuition. In fact, the die strength reverses no matter which two die we picked. So for 1 roll,

$$A \succ B \succ C \succ A,$$

but for two rolls,

$$A \prec B \prec C \prec A,$$

where we have used the symbols $\succ$ and $\prec$ to denote which die is more likely to result in the larger value. This is surprising even to us, but at least we don't owe biker dude \$1400.

14.3.5 Even Stranger Dice

Now that we know that strange things can happen with strange dice, it is natural, at least for mathematicians, to ask how strange things can get. It turns out that things can get very strange. In fact, mathematicians[6] recently made the following discovery:

Theorem 14.3.1. *For any $n \geq 2$, there is a set of n dice $D_1, D_2, \ldots, D_n$ such that for any n-node tournament graph*[7] *G, there is a number of rolls k such that if each*

[6] Reference Ron Graham paper.

[7] Recall that a tournament graph is a directed graph for which there is precisely one directed edge between any two distinct nodes. In other words, for every pair of distinct nodes u and v, either u beats v or v beats u, but not both.

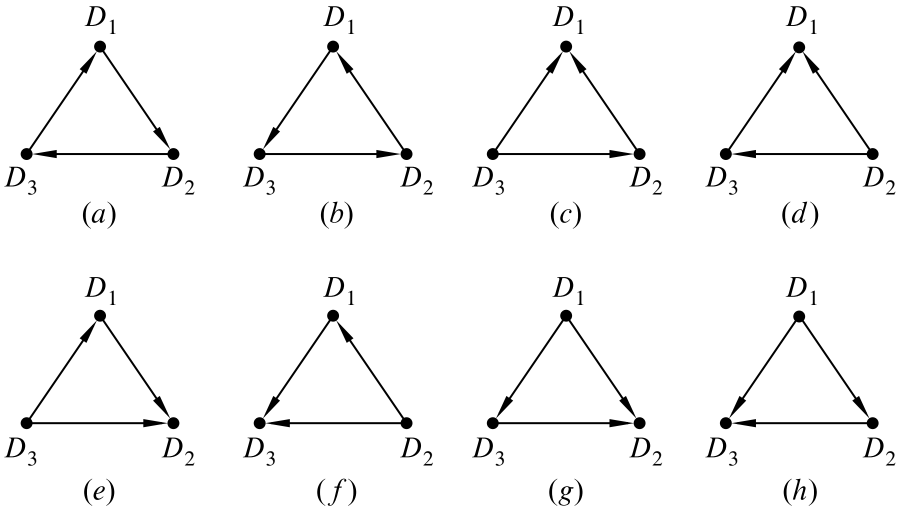


Figure 14.12 All possible relative strengths for three dice D_1, D_2, and D_3. The edge $D_i \rightarrow D_j$ denotes that the sum of rolls for D_i is likely to be greater than the sum of rolls for D_j.

die is rolled k times, then for all $i \neq j$, the sum of the k rolls for D_i will exceed the sum for D_j with probability greater than $1/2$ iff $D_i \rightarrow D_j$ is in G.

It will probably take a few attempts at reading Theorem 14.3.1 to understand what it is saying. The idea is that for some sets of dice, by rolling them different numbers of times, the dice have varying strengths relative to each other. (This is what we observed for the dice in Figure 14.7.) Theorem 14.3.1 says that there is a set of (very) strange dice where *every* possible collection of relative strengths can be observed by varying the number of rolls. For example, the eight possible relative strengths for $n = 3$ dice are shown in Figure 14.12.

Our analysis for the dice in Figure 14.7 showed that for 1 roll, we have the relative strengths shown in Figure 14.12(a), and for two rolls, we have the (reverse) relative strengths shown in Figure 14.12(b). Can you figure out what other relative strengths are possible for the dice in Figure 14.7 by using more rolls? This might be worth doing if you are prone to gambling with strangers in bars.

14.4 Set Theory and Probability

The study of probability is very closely tied to set theory. That is because any set can be a sample space and any subset can be an event. This means that most of the rules and identities that we have developed for sets extend very naturally to probability. We'll cover several examples in this section, but first let's review some definitions that should already be familiar.

14.4.1 Probability Spaces

Definition 14.4.1. A countable[8] *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 14.4.2. 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
- $\sum_{w \in \mathcal{S}} \Pr[w] = 1$.

A 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:

$$\Pr[E] ::= \sum_{w \in E} \Pr[w].$$

14.4.2 Probability Rules from Set Theory

An immediate consequence of the definition of event probability is that for *disjoint* events E and F,

$$\Pr[E \cup F] = \Pr[E] + \Pr[F].$$

This generalizes to a countable number of events, as follows.

Rule 14.4.3 (Sum Rule)**.** *If* $\{E_0, E_1, \ldots\}$ *is collection of disjoint events, then*

$$\Pr\left[\bigcup_{n \in \mathbb{N}} E_n\right] = \sum_{n \in \mathbb{N}} \Pr[E_n].$$

[8] Yes, sample spaces can be infinite. We'll see some examples shortly. If you did not read Chapter 13, don't worry—*countable* means that you can list the elements of the sample space as w_1, w_2, $w_3, \ldots$.

The Sum Rule lets us analyze a complicated event by breaking it down into simpler cases. For example, if the probability that a randomly chosen MIT student is native to the United States is 60%, to Canada is 5%, and to Mexico is 5%, then the probability that a random MIT student is native to North America is 70%.

Another consequence of the Sum Rule is that $\Pr[A] + \Pr[\overline{A}] = 1$, 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 14.4.4 (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 finite sets. In particular:

$$\begin{aligned}
&\Pr[B - A] = \Pr[B] - \Pr[A \cap B], && \text{(Difference Rule)}\\
&\Pr[A \cup B] = \Pr[A] + \Pr[B] - \Pr[A \cap B], && \text{(Inclusion-Exclusion)}\\
&\Pr[A \cup B] \leq \Pr[A] + \Pr[B], && \text{(Boole's Inequality}\\
&\text{If } A \subseteq B, \text{ then } \Pr[A] \leq \Pr[B]. && \text{(Monotonicity)}
\end{aligned}$$

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. Monotonicity follows from the definition of event probability and the fact that outcome 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 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. If $\sum_{i=1}^{n} \Pr[E_i]$ is small, then the Union Bound can give an adequate upper bound on this vital probability.

14.4.3 Uniform Probability Spaces

Definition 14.4.5. A finite probability space $\mathcal{S}$, Pr is said to be *uniform* if $\Pr[w]$ is the same for every outcome $w \in \mathcal{S}$.

As we saw in the strange dice problem, uniform sample spaces are particularly easy to work with. That's because for any event $E \subseteq \mathcal{S}$,

$$\Pr[E] = \frac{|E|}{|\mathcal{S}|}. \tag{14.2}$$

This means that once we know the cardinality of E and $\mathcal{S}$, we can immediately obtain $\Pr[E]$. That's great news because we developed lots of tools for computing the cardinality of a set in Part III.

For example, suppose that you select five cards at random from a standard deck of 52 cards. What is the probability of having a full house? Normally, this question would take some effort to answer. But from the analysis in Section 11.7.2, we know that

$$|\mathcal{S}| = \binom{13}{5}$$

and

$$|E| = 13 \cdot \binom{4}{3} \cdot 12 \cdot \binom{4}{2}$$

where E is the event that we have a full house. Since every five-card hand is equally likely, we can apply Equation 14.2 to find that

$$\begin{aligned}
\Pr[E] &= \frac{13 \cdot 12 \cdot \binom{4}{3} \cdot \binom{4}{2}}{\binom{13}{5}} \\
&= \frac{13 \cdot 12 \cdot 4 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2}{52 \cdot 51 \cdot 50 \cdot 49 \cdot 48} \\
&= \frac{18}{12495} \\
&\approx \frac{1}{694}.
\end{aligned}$$

14.5 Infinite Probability Spaces

General probability theory deals with uncountable sets like $\mathbb{R}$, but in computer science, it is usually sufficient to restrict our attention to countable probability spaces.

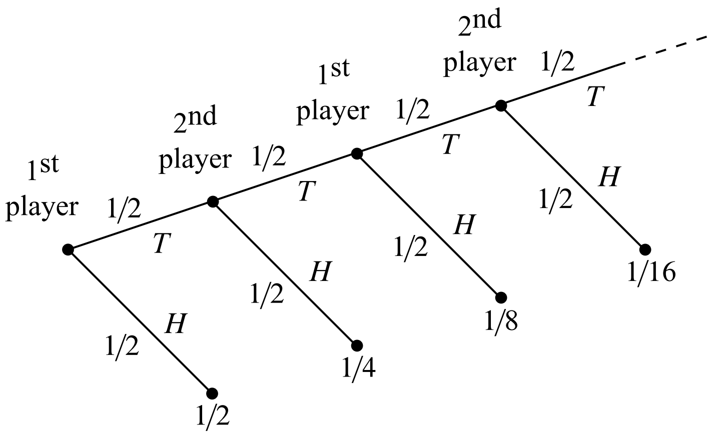


Figure 14.13 The tree diagram for the game where players take turns flipping a fair coin. The first player to flip heads wins.

It's also a lot easier—infinite sample spaces are hard enough to work with without having to deal with uncountable spaces.

Infinite probability spaces are fairly common. For example, 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 in Figure 14.13.

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:

$$\Pr[\text{second player wins}] = \frac{1}{4} + \frac{1}{16} + \frac{1}{64} + \frac{1}{256} + \cdots = \frac{1}{3}.$$

In this case, the sample space is the infinite set

$$\mathcal{S} ::= \{\, \mathrm{T}^n\mathrm{H} \mid n \in \mathbb{N} \,\},$$

where T^n stands for a length n string of T's. The probability function is

$$\Pr[\mathrm{T}^n\mathrm{H}] ::= \frac{1}{2^{n+1}}.$$

To verify that this is a probability space, we just have to check that all the probabilities are nonnegative and that they sum to 1. Nonnegativity is obvious, and applying the formula for the sum of a geometric series, we find that

$$\sum_{n \in \mathbb{N}} \Pr[\mathrm{T}^n\mathrm{H}] = \sum_{n \in \mathbb{N}} \frac{1}{2^{n+1}} = 1.$$

Notice that this model does not have an outcome corresponding to the possibility that both players keep flipping tails forever.[9] That's because the probability of flipping forever would be

$$\lim_{n \to \infty} \frac{1}{2^{n+1}} = 0,$$

and outcomes with probability zero will have no impact on our calculations.

[9] In the diagram, flipping forever corresponds to following the infinite path in the tree without ever reaching a leaf or outcome. Some texts deal with this case by adding a special "infinite" sample point w_{forever} to the sample space, but we will follow the more traditional approach of excluding such sample points, as long as they collectively have probability 0.

15 Conditional Probability

15.1 Definition

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 in Figure 15.1 and asking how likely that point is to fall into region A or B.

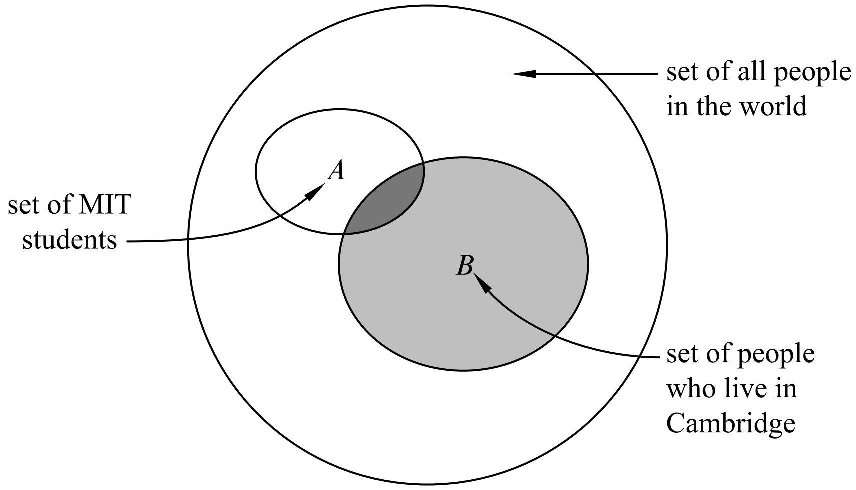


Figure 15.1 Selecting a random person. A is the event that the person is an MIT student. B is the even that the person lives in Cambridge.

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 about 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 $A \cap B$ (the darkly shaded region in Figure 15.1) divided by the probability that the person is in set B (the lightly shaded region). This motivates the definition of conditional probability:

Definition 15.1.1.

$$\Pr[A \mid B] ::= \frac{\Pr[A \cap B]}{\Pr[B]}$$

If $\Pr[B] = 0$, then the conditional probability $\Pr[A \mid B]$ is undefined.

Pure probability is often counterintuitive, but conditional probability is even 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 that you rely on formal reasoning and not intuition. The four-step method will also be very helpful as we will see in the next examples.

15.2 Using the Four-Step Method to Determine Conditional Probability

15.2.1 The “Halting Problem”

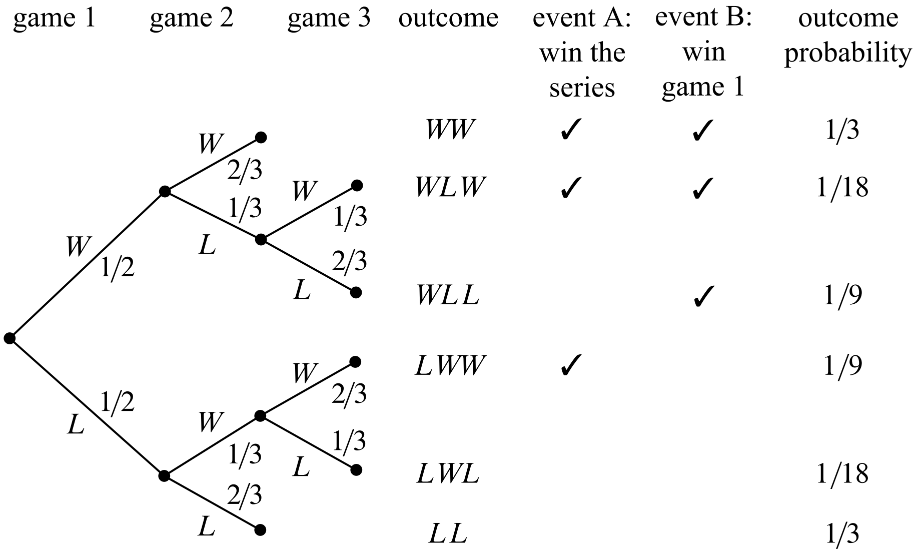
The *Halting Problem* was the first example of a property that could not be tested by any program. It was introduced by Alan Turing in his seminal 1936 paper. The problem is to determine whether a Turing machine halts on a given . . . yadda yadda

yadda . . . more importantly, 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 probability 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 in Figure 15.2.



outcome	event A: win the series	event B: win game 1	outcome probability
WW	✓	✓	1/3
WLW	✓	✓	1/18
WLL		✓	1/9
LWW	✓		1/9
LWL			1/18
LL			1/3

Figure 15.2 The tree diagram for computing the probability that the "Halting Problem" wins two out of three games given that they won the first game.

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 check marks in the tree diagram in Figure 15.2.

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:

$$\begin{aligned}
\Pr\left[A \mid B\right] &= \frac{\Pr[A \cap B]}{\Pr[B]} \\
&= \frac{\Pr[\{WW, WLW\}]}{\Pr[\{WW, WLW, WLL\}]} \\
&= \frac{1/3 + 1/18}{1/3 + 1/18 + 1/9} \\
&= \frac{7}{9}.
\end{aligned}$$

We’re done! If the Halting Problem wins the first game, then they win the whole tournament with probability $7/9$.

15.2.2 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:

$$\Pr[WW] = \frac{1}{2} \cdot \frac{2}{3} = \frac{1}{3}.$$

Why is this correct?

The answer goes back to Definition 15.1.1 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:*

$$\Pr[E_1 \cap E_2] = \Pr[E_1] \cdot \Pr\left[E_2 \mid E_1\right].$$

Multiplying edge probabilities in a tree diagram amounts to evaluating the right side of this equation. For example:

$$\begin{aligned}&\Pr[\text{win first game} \cap \text{win second game}]\\&\qquad = \Pr[\text{win first game}] \cdot \Pr\left[\text{win second game} \mid \text{win first game}\right]\\&\qquad = \frac{1}{2} \cdot \frac{2}{3}.\end{aligned}$$

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.

Rule (Product Rule for n Events)**.**

$$\begin{aligned}\Pr[E_1 \cap E_2 \cap \ldots \cap E_n] = {}&\Pr[E_1] \cdot \Pr\left[E_2 \mid E_1\right] \cdot \Pr\left[E_3 \mid E_1 \cap E_2\right] \cdots\\&\cdot \Pr\left[E_n \mid E_1 \cap E_2 \cap \ldots \cap E_{n-1}\right]\end{aligned}$$

provided that

$$\Pr[E_1 \cap E_2 \cap \cdots \cap E_{n-1}] \neq 0.$$

This rule follows from the definition of conditional probability and induction on n.

15.2.3 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 in Figure 15.3.

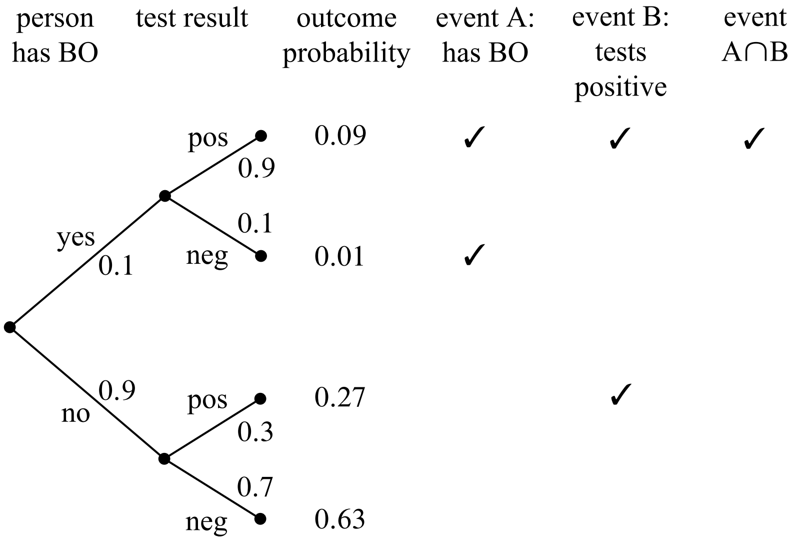


Figure 15.3 The tree diagram for the BO problem.

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 Figure 15.3.

Step 4: Compute Event Probabilities

From Definition 15.1.1, we have

$$\Pr[A \mid B] = \frac{\Pr[A \cap B]}{\Pr[B]} = \frac{0.09}{0.09 + 0.27} = \frac{1}{4}.$$

So, 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 have the condition 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 have the condition and test positive (probability 0.09), or the person could be healthy and 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. So a better strategy by this measure 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!

15.3 *A Posteriori* Probabilities

If you think about it too much, the medical testing problem we just considered could start to trouble you. The concern would be that by the time you take the test, you either have the BO condition or you don’t—you just don’t know which it is. So you may wonder if a statement like “If you tested positive, then you have the condition with probability 25%” makes sense.

In fact, such a statement does make sense. It means that 25% of the people who test positive actually have the condition. It is true that any particular person has it or they don’t, but a *randomly selected* person among those who test positive will have the condition with probability 25%.

Anyway, if the medical testing example bothers you, you will definitely be worried by the following examples, which go even further down this path.

15.3.1 The “Halting Problem,” in Reverse

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 $\Pr\left[B \mid A\right]$ 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 $\Pr\left[B \mid A\right]$. We can compute this using the definition of conditional probability and the tree diagram in Figure 15.2:

$$\Pr\left[B \mid A\right] = \frac{\Pr[B \cap A]}{\Pr[A]} = \frac{1/3 + 1/18}{1/3 + 1/18 + 1/9} = \frac{7}{9}.$$

This answer is suspicious! In the preceding section, we showed that $\Pr\left[A \mid B\right]$ was also $7/9$. Could it be true that $\Pr\left[A \mid B\right] = \Pr\left[B \mid A\right]$ 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 $\Pr\left[A \mid B\right] = \Pr\left[B \mid A\right]$. 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; namely if

$$\Pr[B] = \Pr[A] \qquad \text{or} \qquad \Pr[A \cap B] = 0.$$

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) since both probabilities are $1/2$.

In general, such pairs of probabilities are related by Bayes' Rule:

Theorem 15.3.1 (Bayes' Rule)**.** *If* $\Pr[A]$ *and* $\Pr[B]$ *are nonzero, then:*

$$\Pr\left[B \mid A\right] = \frac{\Pr\left[A \mid B\right] \cdot \Pr[B]}{\Pr[A]} \tag{15.1}$$

Proof. When $\Pr[A]$ and $\Pr[B]$ are nonzero, we have

$$\Pr\left[A \mid B\right] \cdot \Pr[B] = \Pr[A \cap B] = \Pr\left[B \mid A\right] \cdot \Pr[A]$$

by definition of conditional probability. Dividing by $\Pr[A]$ gives (15.1). ■

Next, let's look at a problem that even bothers us.

15.3.2 A Coin Problem

Suppose that 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 shown in Figure 15.4. We do not know the probability p that you were handed the fair coin initially—you were just given one coin or the other. Let A be the event that you were handed the fair coin, and let B be the event that you flipped 100 straight heads. 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 Figure 15.4. Plugging the results into the definition of conditional probability gives:

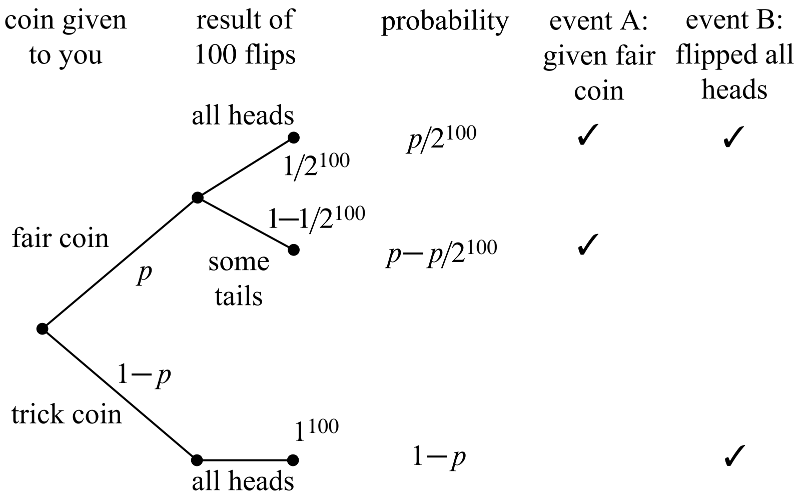


Figure 15.4 The tree diagram for the coin-flipping problem.

$$\begin{aligned}\Pr[A \mid B] &= \frac{\Pr[A \cap B]}{\Pr[B]}\\ &= \frac{p/2^{100}}{1-p+p/2^{100}}\\ &= \frac{p}{2^{100}(1-p)+p}.\end{aligned}$$

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!

Of course, it is extremely unlikely that you would flip 100 straight heads, but in this case, that is a given from the assumption of the conditional probability. And so if you really did see 100 straight heads, it would be very tempting to also assume that p is not close to 1 and hence that you are very likely to have flipped the trick coin.

We will encounter a very similar issue when we look at methods for estimation by sampling in Section 17.5.5.

15.4 Conditional Identities

15.4.1 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 15.4.1 (Law of Total Probability, single event)**.** *If* $\Pr[E]$ *and* $\Pr[\overline{E}]$ *are nonzero, then*

$$\Pr[A] = \Pr\left[A \mid E\right] \cdot \Pr[E] + \Pr\left[A \mid \overline{E}\right] \cdot \Pr[\overline{E}].$$

For example, suppose we conduct the following experiment. First, we flip a fair 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 probability $\Pr\left[A \mid E\right] = 1/6$. On the other hand, if we flip tails, then we get a sum of 2 on two dice with probability $\Pr\left[A \mid \overline{E}\right] = 1/36$. Therefore, the probability that the whole process yields a 2 is

$$\Pr[A] = \frac{1}{2} \cdot \frac{1}{6} + \frac{1}{2} \cdot \frac{1}{36} = \frac{7}{72}.$$

There is also a form of the rule to handle more than two cases.

Rule 15.4.2 (Law of Total Probability)**.** *If $E_1, \dots, E_n$ are disjoint events whose union is the whole sample space, then:*

$$\Pr[A] = \sum_{i=1}^{n} \Pr\left[A \mid E_i\right] \cdot \Pr[E_i].$$

15.4.2 Conditioning on a Single Event

The probability rules that we derived in Chapter 14 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:

$$\Pr\left[A \cup B \mid C\right] = \Pr\left[A \mid C\right] + \Pr\left[B \mid C\right] - \Pr\left[A \cap B \mid C\right].$$

This follows from the fact that if $\Pr[C] \neq 0$, then

$$\begin{aligned}
\Pr\left[A \cup B \mid C\right] &= \frac{\Pr[(A \cup B) \cap C]}{\Pr[C]} \\
&= \frac{\Pr[(A \cap C) \cup (B \cap C)]}{\Pr[C]} \\
&= \frac{\Pr[A \cap C] + \Pr[B \cap C] - \Pr[A \cap B \cap C]}{\Pr[C]} \\
&= \Pr\left[A \mid C\right] + \Pr\left[B \mid C\right] - \Pr\left[A \cap B \mid C\right].
\end{aligned}$$

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.

$$\Pr\left[A \mid B \cup C\right] = \Pr\left[A \mid B\right] + \Pr\left[A \mid C\right] - \Pr\left[A \mid B \cap C\right]. \tag{15.2}$$

A counterexample is shown in Figure 15.5. In this case, $\Pr\left[A \mid B\right] = 1/2$, $\Pr\left[A \mid C\right] = 1/2$, $\Pr\left[A \mid B \cap C\right] = 1$, and $\Pr\left[A \mid B \cup C\right] = 1/3$. However, since $1/3 \neq 1/2 + 1/2 - 1$, Equation 15.2 does not hold.

So you're convinced that this equation is false in general, right? Let's see if you *really* believe that.

15.4.3 Discrimination Lawsuit

Several years ago there was a sex discrimination lawsuit against a famous university. A female math professor was denied tenure, allegedly because she was

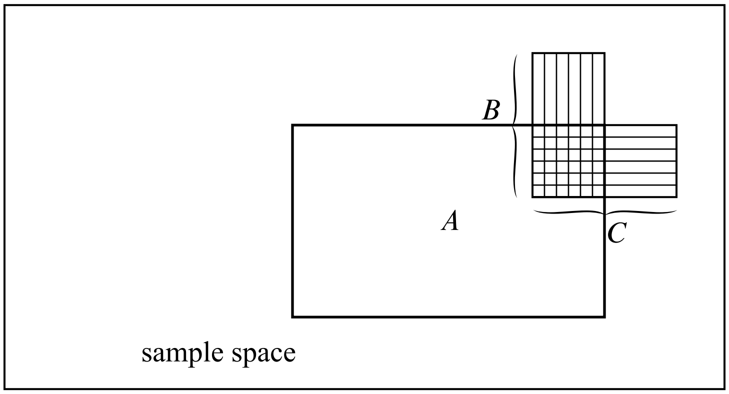


Figure 15.5 A counterexample to Equation 15.2. Event A is the gray rectangle, event B is the rectangle with vertical stripes, and event C is the rectangle with horizontal stripes. $B \cap C$ lies entirely within A while $B - C$ and $C - B$ are entirely outside of A.

a woman. She argued that in every one of the university's 22 departments, the percentage of male applicants accepted was greater than the percentage of female applicants accepted. This sounds very suspicious!

However, the university's lawyers argued that across the university as a whole, the percentage of male 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. To simplify matters, 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.

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	≈ 70%
	51 males accepted, 101 applied	≈ 51%

Table 15.1 A scenario where females are less likely to be admitted than males in each department, but more likely to be admitted overall.

In these terms, the plaintiff is making the following argument:

$$\begin{aligned}\Pr\left[A \mid F_{EE}\right] &< \Pr\left[A \mid M_{EE}\right] \quad \text{and} \\ \Pr\left[A \mid F_{CS}\right] &< \Pr\left[A \mid M_{CS}\right].\end{aligned}$$

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; namely that

$$\Pr\left[A \mid F_{EE} \cup F_{CS}\right] > \Pr\left[A \mid M_{EE} \cup M_{CS}\right].$$

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:

$$\begin{aligned} & \Pr\left[A \mid F_{EE}\right] + \Pr\left[A \mid F_{CS}\right] < \Pr\left[A \mid M_{EE}\right] + \Pr\left[A \mid M_{CS}\right] \\ \Rightarrow \quad & \Pr\left[A \mid F_{EE} \cup F_{CS}\right] < \Pr\left[A \mid M_{EE} \cup M_{CS}\right].\end{aligned}$$

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 (15.2). This argument is bogus! Maybe the two parties do not hold contradictory positions after all!

In fact, Table 15.1 shows a set of application statistics for which the assertions of both the plaintiff and the university hold. In this case, a higher percentage of males were accepted in both departments, but overall a higher percentage of females were accepted! Bizarre!

16 Independence

16.1 Definitions

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 16.1.1. Events A and B are independent if $\Pr[B] = 0$ or if

$$\Pr\left[A \mid B\right] = \Pr[A]. \tag{16.1}$$

In other words, A and B are independent if knowing that B happens does not alter the probability that A happens, as is the case with flipping two coins on opposite sides of a room.

16.1.1 Potential Pitfall

Students sometimes get the idea that disjoint events are independent. The *opposite* is true: if $A \cap B = \emptyset$, then knowing that A happens means you know that B does not happen. So disjoint events are *never* independent—unless one of them has probability zero.

16.1.2 Alternative Formulation

Sometimes it is useful to express independence in an alternate form:

Theorem 16.1.2. *A and B are independent if and only if*

$$\Pr[A \cap B] = \Pr[A] \cdot \Pr[B]. \tag{16.2}$$

Proof. There are two cases to consider depending on whether or not $\Pr[B] = 0$.

Case 1 ($\Pr[B] = 0$)**:** If $\Pr[B] = 0$, A and B are independent by Definition 16.1.1. In addition, Equation 16.2 holds since both sides are 0. Hence, the theorem is true in this case.

Case 2 ($\Pr[B] > 0$)**:** By Definition 15.1.1,

$$\Pr[A \cap B] = \Pr\left[A \mid B\right]\Pr[B].$$

So Equation 16.2 holds if

$$\Pr\left[A \mid B\right] = \Pr[A],$$

which, by Definition 16.1.1, is true iff A and B are independent. Hence, the theorem is true in this case as well. ■

16.2 Independence Is an Assumption

Generally, independence is something that you *assume* in modeling a phenomenon. For example, consider 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:

$$\Pr[A \cap B] = \Pr[A] \cdot \Pr[B] = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}.$$

In this example, the assumption of independence is reasonable. The result of one coin toss should have negligible impact on the outcome of the other coin toss. And if we were to repeat the experiment many times, we would be likely to have $A \cap B$ about 1/4 of the time.

There are, of course, many examples of events where assuming independence is *not* justified, For example, let C be the event that tomorrow is cloudy and R be the event that tomorrow is rainy. Perhaps $\Pr[C] = 1/5$ and $\Pr[R] = 1/10$ in Boston. If these events were independent, then we could conclude that the probability of a rainy, cloudy day was quite small:

$$\Pr[R \cap C] = \Pr[R] \cdot \Pr[C] = \frac{1}{5} \cdot \frac{1}{10} = \frac{1}{50}.$$

Unfortunately, these events are definitely not independent; in particular, every rainy day is cloudy. Thus, the probability of a rainy, cloudy day is actually $1/10$.

Deciding when to *assume* that events are independent is a tricky business. In practice, there are strong motivations to assume independence since many useful formulas (such as Equation 16.2) only hold if the events are independent. But you need to be careful lest you end up deriving false conclusions. We'll see several famous examples where (false) assumptions of independence led to trouble over the next several chapters. This problem gets even trickier when there are more than two events in play.

16.3 Mutual Independence

16.3.1 Definition

We have defined what it means for two events to be independent. What if there are more than two events? For example, how can we say that the flips of n coins are all independent of one another?

Events $E_1, \dots, E_n$ are said to be *mutually independent* if and only if the probability of any event E_i is unaffected by knowledge of the other events. More formally:

Definition 16.3.1. A set of events $E_1, E_2, \dots, E_n$, is *mutually independent* if $\forall i \in [1, n]$ and $\forall S \subseteq [1, n] - \{i\}$, either

$$\Pr\left[\bigcap_{j \in S} E_j\right] = 0 \quad \text{or} \quad \Pr[E_i] = \Pr\left[E_i \;\middle|\; \bigcap_{j \in S} E_j\right].$$

In other words, no matter which other events are known to occur, the probability that E_i occurs is unchanged for any i.

For example, if we toss 100 fair coins at different times, we might reasonably assume that the tosses are mutually independent since the probability that the ith coin is heads should be $1/2$, no matter which other coin tosses came out heads.

16.3.2 Alternative Formulation

Just as Theorem 16.1.2 provided an alternative definition of independence for two events, there is an alternative definition for mutual independence.

Theorem 16.3.2. *A set of events $E_1, E_2, \dots, E_n$ is mutually independent iff $\forall S \subseteq [1, n]$,*

$$\Pr\left[\bigcap_{j \in S} E_j\right] = \prod_{j \in S} \Pr[E_j].$$

The proof of Theorem 16.3.2 uses induction and reasoning similar to the proof of Theorem 16.1.2. We will not include the details here.

Theorem 16.3.2 says that $E_1, E_2, \dots, E_n$ are mutually independent if and only if all of the following equations hold for all distinct i, j, k, and l:

$$\begin{aligned}
\Pr[E_i \cap E_j] &= \Pr[E_i] \cdot \Pr[E_j] \\
\Pr[E_i \cap E_j \cap E_k] &= \Pr[E_i] \cdot \Pr[E_j] \cdot \Pr[E_k] \\
\Pr[E_i \cap E_j \cap E_k \cap E_l] &= \Pr[E_i] \cdot \Pr[E_j] \cdot \Pr[E_k] \cdot \Pr[E_l] \\
&\vdots \\
\Pr[E_1 \cap \cdots \cap E_n] &= \Pr[E_1] \cdots \Pr[E_n].
\end{aligned}$$

For example, if we toss n fair coins, the tosses are mutually independent iff for all $m \in [1, n]$ and every subset of m coins, the probability that every coin in the subset comes up heads is 2^{-m}.

16.3.3 DNA Testing

Assumptions about independence are routinely made in practice. Frequently, such assumptions are quite reasonable. Sometimes, however, the reasonableness of an independence assumption is not so clear, and the consequences of a faulty assumption can be severe.

For example, consider the following 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.

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?

Presumably, this dialogue was as confusing to you as it was for the jury. Essentially, the jury was told that genetic markers in blood found at the crime scene matched Simpson's. Furthermore, they were told that 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.

After the trial, 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?

16.4 Pairwise Independence

The definition of mutual independence seems awfully complicated—there are so many subsets of events to consider! Here's an example that illustrates the subtlety of independence when more than two events are involved. 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 in Theorem 16.3.2:

$$\begin{aligned}\Pr[A_1 \cap A_2] &= \Pr[HHH] + \Pr[TTT]\\ &= \frac{1}{8} + \frac{1}{8}\\ &= \frac{1}{4}\\ &= \frac{1}{2} \cdot \frac{1}{2}\\ &= \Pr[A_1]\Pr[A_2].\end{aligned}$$

By symmetry, $\Pr[A_1 \cap A_3] = \Pr[A_1] \cdot \Pr[A_3]$ and $\Pr[A_2 \cap A_3] = \Pr[A_2] \cdot \Pr[A_3]$ must hold also. Finally, we must check one last condition:

$$\begin{aligned}\Pr[A_1 \cap A_2 \cap A_3] &= \Pr[HHH] + \Pr[TTT]\\ &= \frac{1}{8} + \frac{1}{8}\\ &= \frac{1}{4}\\ &\neq \Pr[A_1]\Pr[A_2]\Pr[A_3] = \frac{1}{8}.\end{aligned}$$

The three events A_1, A_2, and A_3 are not mutually independent even though any two of them are independent! This not-quite mutual independence seems weird at first, but it happens. It even generalizes:

Definition 16.4.1. A set A_1, A_2, …, 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.

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:

$$\begin{aligned}\Pr[A \cap B \cap C \cap D \cap E] &\le \Pr[A \cap E] \\ &= \Pr[A] \cdot \Pr[E] \\ &= \frac{1}{100} \cdot \frac{1}{170} \\ &= \frac{1}{17{,}000}.\end{aligned}$$

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.

On the other hand, the 1 in 17,000 bound that we get by assuming pairwise independence is a lot better than the bound that we would have if there were no independence at all. For example, if the markers are dependent, then it is possible that

everyone with marker E has marker A,

everyone with marker A has marker B,

everyone with marker B has marker C, and

everyone with marker C has marker D.

In such a scenario, the probability of a match is

$$\Pr[E] = 1/170.$$

So a stronger independence assumption leads to a smaller bound on the probability of a match. The trick is to figure out what independence assumption is reasonable. Assuming that the markers are *mutually* independent may well *not* be reasonable unless you have examined hundreds of millions of blood samples. Otherwise, how would you know that marker D does not show up more frequently whenever the other four markers are simultaneously present?

We will conclude our discussion of independence with a useful, and somewhat famous, example known as the Birthday Paradox.

16.5 The Birthday Paradox

Suppose that there are 100 students in a class. What is the probability that some birthday is shared by two people? Comparing 100 students to the 365 possible birthdays, you might guess the probability lies somewhere around $1/3$—but you'd be wrong: the probability that there will be two people in the class with matching birthdays is actually $0.999999692\ldots$. In other words, the probability that all 100 birthdays are different is less than 1 in 3,000,000.

Why is this probability so small? The answer involves a phenomenon known as the *Birthday Paradox* (or the *Birthday Principle*), which is surprisingly important in computer science, as we'll see later.

Before delving into the analysis, we'll need to make some modeling assumptions:

- For each student, all possible birthdays are equally likely. The idea underlying this assumption is that each student's birthday is determined by a random process involving parents, fate, and, um, some issues that we discussed earlier in the context of graph theory. The assumption is not completely accurate, however; a disproportionate number of babies are born in August and September, for example.

- Birthdays are mutually independent. This isn't perfectly accurate either. For example, if there are twins in the class, then their birthdays are surely not independent.

We'll stick with these assumptions, despite their limitations. Part of the reason is to simplify the analysis. But the bigger reason is that our conclusions will apply to many situations in computer science where twins, leap days, and romantic holidays are not considerations. After all, whether or not two items collide in a hash table really has nothing to do with human reproductive preferences. Also, in pursuit of generality, let's switch from specific numbers to variables. Let m be the number of people in the room, and let N be the number of days in a year.

We can solve this problem using the standard four-step method. However, a tree diagram will be of little value because the sample space is so enormous. This time we'll have to proceed without the visual aid!

Step 1: Find the Sample Space

Let's number the people in the room from 1 to m. An outcome of the experiment is a sequence $(b_1, \dots, b_m)$ where b_i is the birthday of the ith person. The sample space is the set of all such sequences:

$$\mathcal{S} = \{\, (b_1, \dots, b_m) \mid b_i \in \{1, \dots N\} \,\}.$$

Step 2: Define Events of Interest

Our goal is to determine the probability of the event A in which some pair of people have the same birthday. This event is a little awkward to study directly, however. So we'll use a common trick, which is to analyze the *complementary* event $\overline{A}$, in which all m people have different birthdays:

$$\overline{A} = \{\, (b_1, \dots, b_m) \in \mathcal{S} \mid \text{all } b_i \text{ are distinct} \,\}.$$

If we can compute $\Pr[\overline{A}]$, then we can compute what really want, $\Pr[A]$, using the identity

$$\Pr[A] + \Pr[\overline{A}] = 1.$$

Step 3: Assign Outcome Probabilities

We need to compute the probability that m people have a particular combination of birthdays $(b_1, \dots, b_m)$. There are N possible birthdays and all of them are equally likely for each student. Therefore, the probability that the ith person was born on day b_i is $1/N$. Since we're assuming that birthdays are mutually independent, we can multiply probabilities. Therefore, the probability that the first person was born on day b_1, the second on b_2, and so forth is $(1/N)^m$. This is the probability of every outcome in the sample space, which means that the sample space is uniform. That's good news, because, as we have seen, it means that the analysis will be simpler.

Step 4: Compute Event Probabilities

We're interested in the probability of the event $\overline{A}$ in which everyone has a different birthday:

$$\overline{A} = \{\, (b_1, \dots, b_n) \mid \text{all } b_i \text{ are distinct} \,\}.$$

This is a gigantic set. In fact, there are N choices for b_i, $N-1$ choices for b_2, and so forth. Therefore, by the Generalized Product Rule,

$$|\overline{A}| = \frac{N!}{(N-m)!} = N(N-1)(N-2)\cdots(N-m+1).$$

Since the sample space is uniform, we can conclude that

$$\Pr[\overline{A}] = \frac{|\overline{A}|}{N^m} = \frac{N!}{N^m(N-m)!}. \tag{16.3}$$

We're done!

Or are we? While correct, it would certainly be nicer to have a closed-form expression for Equation 16.3. That means finding an approximation for $N!$ and $(N-m)!$. But this is what we learned how to do in Section 9.6. In fact, since N and $N-m$ are each at least 100, we know from Corollary 9.6.2 that

$$\sqrt{2\pi N}\left(\frac{N}{e}\right)^N \quad \text{and} \quad \sqrt{2\pi(N-m)}\left(\frac{N-m}{e}\right)^{N-m}$$

are excellent approximations (accurate to within .09%) of $N!$ and $(N-m)!$, respectively. Plugging these values into Equation 16.3 means that (to within .2%)[1]

$$\begin{aligned}
\Pr[\overline{A}] &= \frac{\sqrt{2\pi N}\left(\frac{N}{e}\right)^N}{N^m\sqrt{2\pi(N-m)}\left(\frac{N-m}{e}\right)^{N-m}} \\
&= \sqrt{\frac{N}{N-m}}\frac{e^{N\ln(N)-N}}{e^{m\ln(N)}e^{(N-m)\ln(N-m)-(N-m)}} \\
&= \sqrt{\frac{N}{N-m}}e^{(N-m)\ln(N)-(N-m)\ln(N-m)-m} \\
&= \sqrt{\frac{N}{N-m}}e^{(N-m)\ln\left(\frac{N}{N-m}\right)-m} \\
&= e^{\left(N-m+\frac{1}{2}\right)\ln\left(\frac{N}{N-m}\right)-m}.
\end{aligned} \tag{16.4}$$

[1] If there are two terms that can be off by .09%, then the ratio can be off by at most a factor of $(1.0009)^2 < 1.002$.

We can now evaluate Equation 16.4 for $m = 100$ and $N = 365$ to find that the probability that all 100 birthdays are different is[2]

$$3.07\ldots \cdot 10^{-7}.$$

We can also plug in other values of m to find the number of people so that the probability of a matching birthday will be about $1/2$. In particular, for $m = 23$ and $N = 365$, Equation 16.4 reveals that the probability that all the birthdays differ is $0.49\ldots$. So if you are in a room with 23 other people, the probability that some pair of people share a birthday will be a little better than $1/2$. It is because 23 seems like such a small number of people for a match that the phenomenon is called the *Birthday Paradox*.

16.5.1 Applications to Hashing

Hashing is frequently used in computer science to map large strings of data into short strings of data. In a typical scenario, you have a set of m items and you would like to assign each item to a number from 1 to N where no pair of items is assigned to the same number and N is as small as possible. For example, the items might be messages, addresses, or variables. The numbers might represent storage locations, devices, indices, or digital signatures.

If two items are assigned to the same number, then a *collision* is said to occur. Collisions are generally bad. For example, collisions can correspond to two variables being stored in the same place or two messages being assigned the same digital signature. Just imagine if you were doing electronic banking and your digital signature for a \$10 check were the same as your signature for a \$10 million dollar check. In fact, finding collisions is a common technique in breaking cryptographic codes.[3]

In practice, the assignment of a number to an item is done using a hash function

$$h : S \to [1, N],$$

where S is the set of items and $m = |S|$. Typically, the values of $h(S)$ are assigned randomly and are assumed to be equally likely in $[1, N]$ and mutually independent.

For efficiency purposes, it is generally desirable to make N as small as necessary to accommodate the hashing of m items without collisions. Ideally, N would be only a little larger than m. Unfortunately, this is not possible for random hash functions. To see why, let's take a closer look at Equation 16.4.

[2] The possible .2% error is so small that it is lost in the ... after 3.07.

[3] Such techniques are often referred to as *birthday attacks* because of the association of such attacks with the Birthday Paradox.

By Theorem 9.6.1 and the derivation of Equation 16.4, we know that the probability that there are no collisions for a random hash function is

$$\sim e^{\left(N-m+\frac{1}{2}\right)\ln\left(\frac{N}{N-m}\right)-m}. \tag{16.5}$$

For any m, we now need to find a value of N for which this expression is at least 1/2. That will tell us how big the hash table needs to be in order to have at least a 50% chance of avoiding collisions. This means that we need to find a value of N for which

$$\left(N-m+\frac{1}{2}\right)\ln\left(\frac{N}{N-m}\right)-m \sim \ln\left(\frac{1}{2}\right). \tag{16.6}$$

To simplify Equation 16.6, we need to get rid of the $\ln\left(\frac{N}{N-m}\right)$ term. We can do this by using the Taylor Series expansion for

$$\ln(1-x) = -x - \frac{x^2}{2} - \frac{x^3}{3} - \cdots$$

to find that[4]

$$\begin{aligned}
\ln\left(\frac{N}{N-m}\right) &= -\ln\left(\frac{N-m}{N}\right)\\
&= -\ln\left(1-\frac{m}{N}\right)\\
&= -\left(-\frac{m}{N} - \frac{m^2}{2N^2} - \frac{m^3}{3N^3} - \cdots\right)\\
&= \frac{m}{N} + \frac{m^2}{2N^2} + \frac{m^3}{3N^3} + \cdots .
\end{aligned}$$

[4] This may not look like a simplification, but stick with us here.

Hence,

$$\begin{aligned}
\left(N - m + \frac{1}{2}\right) \ln\left(\frac{N}{N-m}\right) - m &= \left(N - m + \frac{1}{2}\right)\left(\frac{m}{N} + \frac{m^2}{2N^2} + \frac{m^3}{3N^3} + \cdots\right) - m \\
&= \left(m + \frac{m^2}{2N} + \frac{m^3}{3N^2} + \cdots\right) \\
&\qquad - \left(\frac{m^2}{N} + \frac{m^3}{2N^2} + \frac{m^4}{3N^3} + \cdots\right) \\
&\qquad + \frac{1}{2}\left(\frac{m}{N} + \frac{m^2}{2N^2} + \frac{m^3}{3N^3} + \cdots\right) - m \\
&= -\left(\frac{m^2}{2N} + \frac{m^3}{6N^2} + \frac{m^4}{12N^3} + \cdots\right) \\
&\qquad + \frac{1}{2}\left(\frac{m}{N} + \frac{m^2}{2N^2} + \frac{m^3}{3N^3} + \cdots\right).
\end{aligned} \tag{16.7}$$

If N grows faster than m^2, then the value in Equation 16.7 tends to 0 and Equation 16.6 cannot be satisfied. If N grows more slowly than m^2, then the value in Equation 16.7 diverges to negative infinity, and, once again, Equation 16.6 cannot be satisfied. This suggests that we should focus on the case where $N = \Theta(m^2)$, when Equation 16.7 simplifies to

$$\sim \frac{-m^2}{2N}$$

and Equation 16.6 becomes

$$\frac{-m^2}{2N} \sim \ln\left(\frac{1}{2}\right). \tag{16.8}$$

Equation 16.8 is satisfied when

$$N \sim \frac{m^2}{2\ln(2)}. \tag{16.9}$$

In other words, N needs to grow quadratically with m in order to avoid collisions. This unfortunate fact is known as the *Birthday Principle* and it limits the efficiency of hashing in practice—either N is quadratic in the number of items being hashed or you need to be able to deal with collisions.

17 Random Variables and Distributions

Thus far, we have focused on probabilities of events. For example, we computed the probability that you win the Monty Hall game, or that you have a rare medical condition given that you tested positive. But, in many cases we would like to more more. For example, *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 gambling with strange dice all night? To answer such questions, we need to work with random variables.

17.1 Definitions and Examples

Definition 17.1.1. 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[1], 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. 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

$$\mathcal{S} = \{HHH, HHT, HTH, HTT, THH, THT, TTH, TTT\}$$

and C is a function that maps each outcome in the sample space to a number as

[1] Going forward, when we talk about flipping independent coins, we will assume that they are *mutually* independent.

follows:

$$\begin{aligned} C(HHH) &= 3 & \qquad C(THH) &= 2 \\ C(HHT) &= 2 & \qquad C(THT) &= 1 \\ C(HTH) &= 2 & \qquad C(TTH) &= 1 \\ C(HTT) &= 1 & \qquad C(TTT) &= 0. \end{aligned}$$

Similarly, M is a function mapping each outcome another way:

$$\begin{aligned} M(HHH) &= 1 & \qquad M(THH) &= 0 \\ M(HHT) &= 0 & \qquad M(THT) &= 0 \\ M(HTH) &= 0 & \qquad M(TTH) &= 0 \\ M(HTT) &= 0 & \qquad M(TTT) &= 1. \end{aligned}$$

So C and M are random variables.

17.1.1 Indicator Random Variables

An *indicator random variable* is a random variable that maps every outcome to either 0 or 1. Indicator random variables 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 random variable 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 $I_E(w) = 1$ for outcomes $w \in E$ and $I_E(w) = 0$ for outcomes $w \notin E$. Thus, $M = I_E$ where E is the event that all three coins match.

17.1.2 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,

$$\begin{aligned}\Pr[C = 2] &= \Pr[THH] + \Pr[HTH] + \Pr[HHT]\\ &= \frac{1}{8} + \frac{1}{8} + \frac{1}{8}\\ &= \frac{3}{8}.\end{aligned}$$

As another example:

$$\begin{aligned}\Pr[M = 1] &= \Pr[TTT] + \Pr[HHH]\\ &= \frac{1}{8} + \frac{1}{8}\\ &= \frac{1}{4}.\end{aligned}$$

17.1.3 Functions of Random Variables

Random variables can be combined to form other random variables. For example, suppose that you roll two unbiased, independent 6-sided dice. Let D_i be the random variable denoting the outcome of the ith die for $i = 1, 2$. For example,

$$\Pr[D_1 = 3] = 1/6.$$

Then let $T = D_1 + D_2$. T is also a random variable and it denotes the sum of the two dice. For example,

$$\Pr[T = 2] = 1/36$$

and

$$\Pr[T = 7] = 1/6.$$

Random variables can be combined in complicated ways, as we will see in Chapter 19. For example,

$$Y = e^T$$

is also a random variable. In this case,

$$\Pr[Y = e^2] = 1/36$$

and

$$\Pr[Y = e^7] = 1/6.$$

17.1.4 Conditional Probability

Mixing conditional probabilities and events involving random variables creates no new difficulties. For example, $\Pr\left[C \geq 2 \mid M = 0\right]$ 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:

$$\begin{aligned}
\Pr\left[C \geq 2 \mid M = 0\right] &= \frac{\Pr[C \geq 2 \cap M = 0]}{\Pr[M = 0]} \\
&= \frac{\Pr[\{THH, HTH, HHT\}]}{\Pr[\{THH, HTH, HHT, HTT, THT, TTH\}]} \\
&= \frac{3/8}{6/8} \\
&= \frac{1}{2}.
\end{aligned}$$

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*, and so they are *sets* of outcomes.

17.1.5 Independence

The notion of independence carries over from events to random variables as well. Random variables R_1 and R_2 are *independent* iff for all x_1 in the codomain of R_1, and x_2 in the codomain of R_2 for which $\Pr[R_2 = X_2] > 0$, we have:

$$\Pr\left[R_1 = x_1 \mid R_2 = x_2\right] = \Pr[R_1 = x_1].$$

As with events, we can formulate independence for random variables in an equivalent and perhaps more useful way: random variables R_1 and R_2 are independent if for all x_1 and x_2

$$\Pr[R_1 = x_1 \cap R_2 = x_2] = \Pr[R_1 = x_1] \cdot \Pr[R_2 = x_2].$$

For example, are C and M independent? Intuitively, the answer should be "no". The number of heads, C, completely determines whether all three coins match; that is, whether $M = 1$. But, to verify this intuition, we must find some $x_1, x_2 \in \mathbb{R}$ such that:

$$\Pr[C = x_1 \cap M = x_2] \neq \Pr[C = x_1] \cdot \Pr[M = x_2].$$

One appropriate choice of values is $x_1 = 2$ and $x_2 = 1$. In this case, we have:

$$\Pr[C = 2 \cap M = 1] = 0$$

and

$$\Pr[M = 1] \cdot \Pr[C = 2] = \frac{1}{4} \cdot \frac{3}{8} \neq 0.$$

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 F be the indicator variable for the event that the first flip is a Head, so

$$\text{“}F = 1\text{”} = \{HHH, HTH, HHT, HTT\}.$$

Then F is independent of M, since

$$\Pr[M = 1] = 1/4 = \Pr\left[M = 1 \mid F = 1\right] = \Pr\left[M = 1 \mid F = 0\right]$$

and

$$\Pr[M = 0] = 3/4 = \Pr\left[M = 0 \mid F = 1\right] = \Pr\left[M = 0 \mid F = 0\right].$$

This example is an instance of a simple lemma:

Lemma 17.1.2. *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 17.1.3. Random variables $R_1, R_2, \ldots, R_n$ are *mutually independent* iff

$$\begin{aligned}&\Pr[R_1 = x_1 \cap R_2 = x_2 \cap \cdots \cap R_n = x_n]\\ &\quad = \Pr[R_1 = x_1] \cdot \Pr[R_2 = x_2] \cdots \Pr[R_n = x_n].\end{aligned}$$

for all $x_1, x_2, \ldots, x_n$.

A consequence of Definition 17.1.3 is 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:

$$\begin{aligned}&\Pr[R_1 = 7 \cap R_7 = 9.1 \cap R_{23} = \pi]\\ &\qquad = \Pr[R_1 = 7] \cdot \Pr[R_7 = 9.1] \cdot \Pr[R_{23} = \pi].\end{aligned}$$

The proof is based on summing over all possible values for all of the other random variables.

17.2 Distribution Functions

A random variable maps outcomes to values. Often, 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 having any given value. Hence, random variables on different probability spaces may wind up having the same *probability density function*.

Definition 17.2.1. 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 is 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, suppose that you roll two unbiased, independent, 6-sided dice. Let T be the random variable that equals the sum 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 for T is shown in Figure 17.1: 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 concept to a PDF is the *cumulative distribution function (cdf)* for a random variable whose codomain is the 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 in Figure 17.2: The height of the ith bar in the cumulative distribution function is equal to the *sum* of the heights of the leftmost i bars in the probability

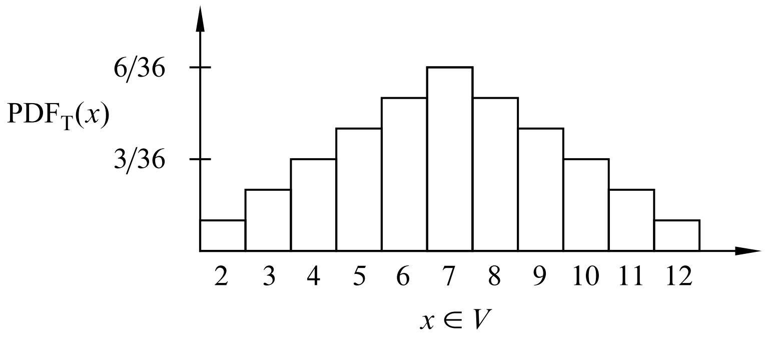


Figure 17.1 The probability density function for the sum of two 6-sided dice.

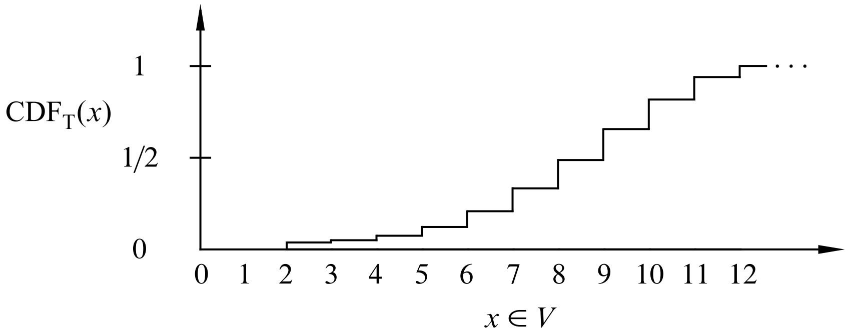


Figure 17.2 The cumulative distribution function for the sum of two 6-sided dice.

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 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.

One of the really interesting things about density functions and distribution functions is that many random variables turn out to have the *same* pdf and cdf. In other words, even though R and S are different random variables on different probability spaces, it is often the case that

$$\text{PDF}_R = \text{PDF}_s.$$

In fact, some pdfs are so common that they are given special names. For example, the three most important distributions in computer science are the *Bernoulli distribution*, the *uniform distribution*, and the *binomial distribution*. We look more closely at these common distributions in the next several sections.

17.3 Bernoulli Distributions

The Bernoulli distribution is the simplest and most common distribution function. That's because it is the distribution function for an indicator random variable. Specifically, the *Bernoulli distribution* has a probability density function of the form $f_p : \{0, 1\} \to [0, 1]$ where

$$\begin{aligned} f_p(0) &= p, \quad \text{and} \\ f_p(1) &= 1 - p, \end{aligned}$$

for some $p \in [0, 1]$. The corresponding cumulative distribution function is $F_p : \mathbb{R} \to [0, 1]$ where:

$$F_p(x) = \begin{cases} 0 & \text{if } x < 0 \\ p & \text{if } 0 \le x < 1 \\ 1 & \text{if } 1 \le x. \end{cases}$$

17.4 Uniform Distributions

17.4.1 Definition

A random variable that takes on each possible value with the same probability is said to be *uniform*. If the sample space is $\{1, 2, \ldots, n\}$, then the *uniform distribution* has a pdf of the form

$$f_n : \{1, 2, \ldots, n\} \to [0, 1]$$

where

$$f_n(k) = \frac{1}{n}$$

for some $n \in \mathbb{N}^+$. The cumulative distribution function is then $F_n : \mathbb{R} \to [0, 1]$ where

$$F_n(x) = \begin{cases} 0 & \text{if } x < 1 \\ k/n & \text{if } k \le x < k + 1 \text{ for } 1 \le k < n \\ 1 & \text{if } n \le x. \end{cases}$$

Uniform distributions arise frequently in practice. For example, the number rolled on a fair die is uniform on the set $\{1, 2, \ldots, 6\}$. If $p = 1/2$, then an indicator random variable is uniform on the set $\{0, 1\}$.

17.4.2 The Numbers Game

Enough definitions—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, we'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 one envelope and see the number 12. Since 12 is a small number, you might guess that that the number in the other envelope is larger. But perhaps we've been 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. We're picking the numbers and we're choosing them in a way that we think will defeat your guessing strategy. We'll only use randomization to choose the numbers if that serves our purpose, which is to make you lose!

Intuition Behind the Winning Strategy

Amazingly, there is a strategy that wins more than 50% of the time, regardless of what numbers we put in the envelopes!

Suppose that you somehow knew a number x that was in between the numbers in the envelopes. Now you peek in one envelope and see a number. 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 the numbers in the envelopes, then you are certain to win the game.

The only flaw with this brilliant strategy is that you do *not* know such an 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 we 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 we figured out that you were unlikely to pick some number—say $50\frac{1}{2}$—then we'd always put 50 and 51 in the envelopes. 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 T. If $T > x$, then you guess that you're looking at the larger number. If $T < x$, 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 ($T = L$) or the higher number ($T = H$). This gives a total of six possible outcomes, as show in Figure 17.3.

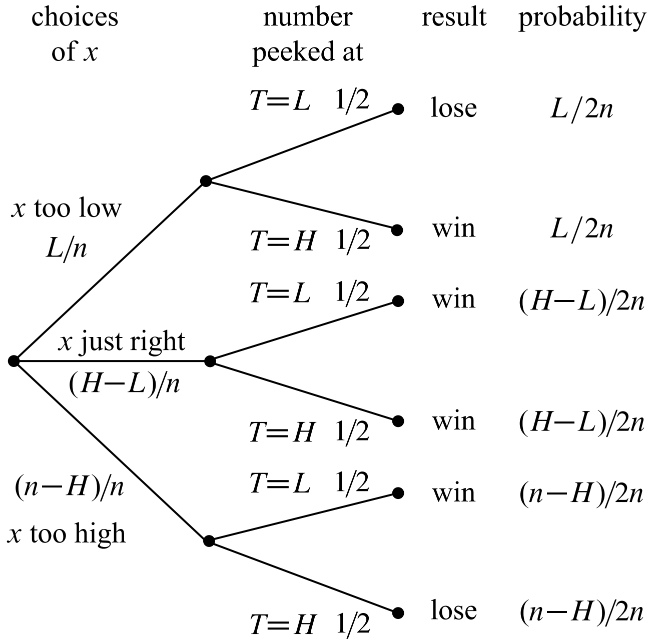


Figure 17.3 The tree diagram for the numbers game.

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!

17.4.3 Randomized Algorithms

The best strategy to win the numbers game is an example of a *randomized algorithm*—it uses random numbers to influence decisions. Protocols and algorithms that make use of random numbers are very important in computer science. There are many problems for which the best known solutions are based on a random number generator.

For example, the most commonly-used protocol for deciding when to send a broadcast on a shared bus or Ethernet is a randomized algorithm known as *exponential backoff*. One of the most commonly-used sorting algorithms used in practice, called *quicksort*, uses random numbers. You'll see many more examples if you take an algorithms course. In each case, randomness is used to improve the probability that the algorithm runs quickly or otherwise performs well.

17.5 Binomial Distributions

17.5.1 Definitions

The third commonly-used distribution in computer science is the *binomial distribution*. 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. If the coin is

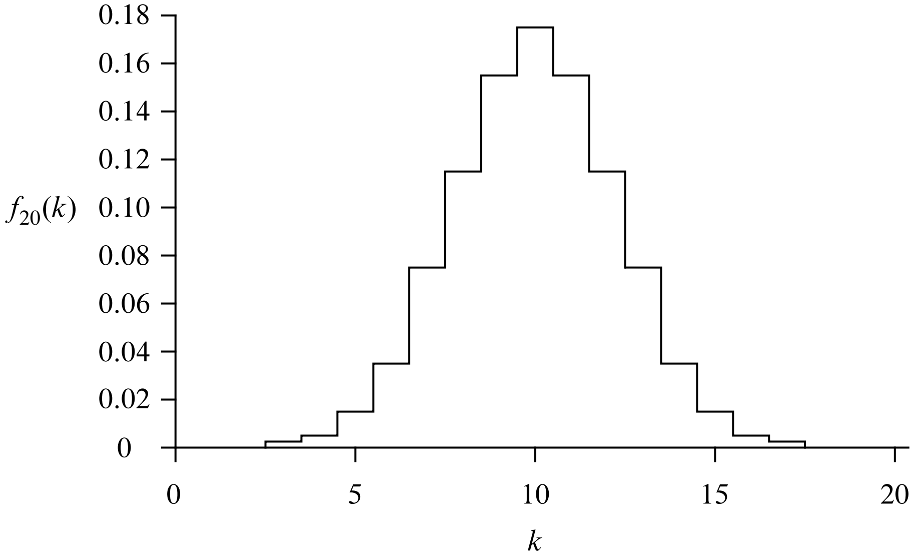


Figure 17.4 The pdf for the unbiased binomial distribution for $n = 20$, $f_{20}(k)$.

fair, then the number of heads has an *unbiased binomial distribution*, specified by the pdf

$$f_n : \{1, 2, \ldots, n\} \to [0, 1]$$

where

$$f_n(k) = \binom{n}{k} 2^{-n}$$

for some $n \in \mathbb{N}^+$. This is because there are $\binom{n}{k}$ sequences of n coin tosses with exactly k heads, and each such sequence has probability 2^{-n}.

A plot of $f_{20}(k)$ is shown in Figure 17.4. The most likely outcome is $k = 10$ heads, and the probability falls off rapidly for larger and smaller values of k. The falloff regions to the left and right of the main hump are called the *tails of the distribution*. We'll talk a lot more about these tails shortly.

The cumulative distribution function for the unbiased binomial distribution is $F_n : \mathbb{R} \to [0, 1]$ where

$$F_n(x) = \begin{cases} 0 & \text{if } x < 1 \\ \sum_{i=0}^{k} \binom{n}{i} 2^{-n} & \text{if } k \leq x < k + 1 \text{ for } 1 \leq k < n \\ 1 & \text{if } n \leq x. \end{cases}$$

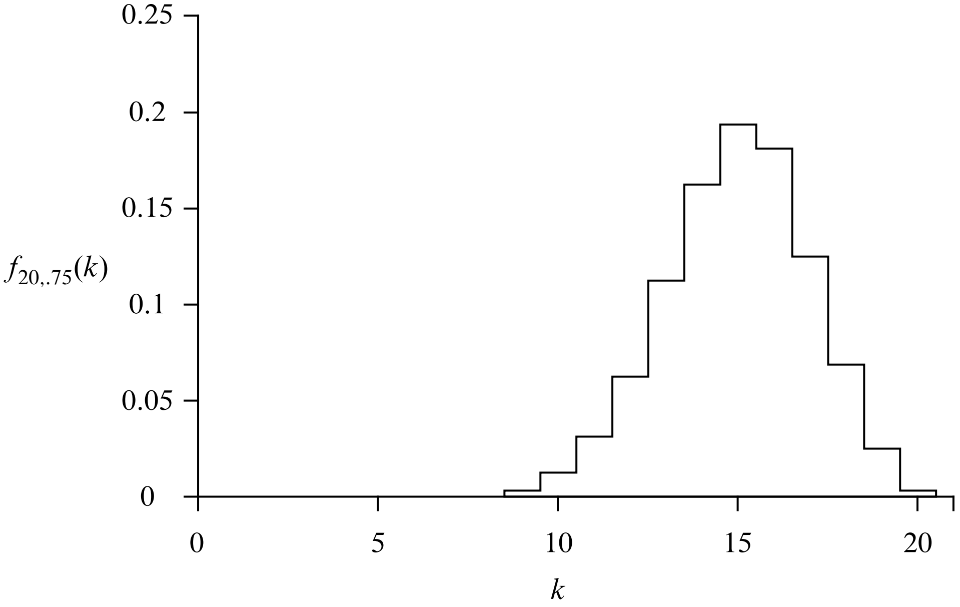


Figure 17.5 The pdf for the general binomial distribution $f_{n,p}(k)$ for $n = 20$ and $p = .75$.

The General Binomial Distribution

If the coins are biased so that each coin is heads with probability p, then the number of heads has a *general binomial density function* specified by the pdf

$$f_{n,p} : \{1, 2, \ldots, n\} \to [0, 1]$$

where

$$f_{n,p}(k) = \binom{n}{k} p^k (1-p)^{n-k}.$$

for some $n \in \mathbb{N}^+$ and $p \in [0, 1]$. This is because there are $\binom{n}{k}$ sequences with k heads and $n - k$ tails, but now the probability of each such sequence is $p^k(1-p)^{n-k}$.

For example, the plot in Figure 17.5 shows the probability density function $f_{n,p}(k)$ corresponding to flipping $n = 20$ independent coins that are heads with probability $p = 0.75$. The graph shows that we are most likely to get $k = 15$ heads, as you might expect. Once again, the probability falls off quickly for larger and smaller values of k.

The cumulative distribution function for the general binomial distribution is $F_{n,p} : \mathbb{R} \to [0, 1]$ where

$$F_{n,p}(x) = \begin{cases} 0 & \text{if } x < 1 \\ \sum_{i=0}^{k} \binom{n}{i} p^i (1-p)^{n-i} & \text{if } k \leq x < k+1 \text{ for } 1 \leq k < n \\ 1 & \text{if } n \leq x. \end{cases} \tag{17.1}$$

17.5.2 Approximating the Probability Density Function

Computing the general binomial density function is daunting when k and n are large. Fortunately, there is an approximate closed-form formula for this function based on an approximation for the binomial coefficient. In the formula below, k is replaced by αn where α is a number between 0 and 1.

Lemma 17.5.1.

$$\binom{n}{\alpha n} \sim \frac{2^{nH(\alpha)}}{\sqrt{2\pi\alpha(1-\alpha)n}} \tag{17.2}$$

and

$$\binom{n}{\alpha n} < \frac{2^{nH(\alpha)}}{\sqrt{2\pi\alpha(1-\alpha)n}} \tag{17.3}$$

where $H(\alpha)$ is the entropy function[2]

$$H(\alpha) ::= \alpha \log\left(\frac{1}{\alpha}\right) + (1-\alpha)\log\left(\frac{1}{1-\alpha}\right).$$

Moreover, if $\alpha n > 10$ and $(1-\alpha)n > 10$, then the left and right sides of Equation 17.2 differ by at most 2%. If $\alpha n > 100$ and $(1-\alpha)n > 100$, then the difference is at most 0.2%.

The graph of H is shown in Figure 17.6.

Lemma (17.5.1) provides an excellent approximation for binomial coefficients. We'll skip its derivation, which consists of plugging in Theorem 9.6.1 for the factorials in the binomial coefficient and then simplifying.

Now let's plug Equation 17.2 into the general binomial density function. The probability of flipping αn heads in n tosses of a coin that comes up heads with

[2] $\log(x)$ means $\log_2(x)$.

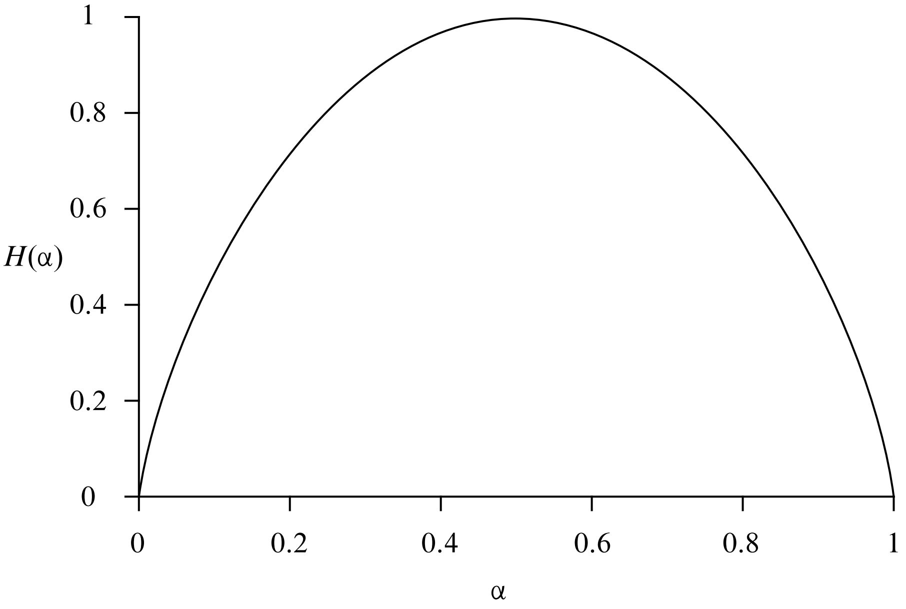


Figure 17.6 The Entropy Function

probability p is:

$$f_{n,p}(\alpha n) \sim \frac{2^{nH(\alpha)}\, p^{\alpha n}(1-p)^{(1-\alpha)n}}{\sqrt{2\pi\alpha(1-\alpha)n}}$$

$$= \frac{2^{n\left(\alpha\log\left(\frac{p}{\alpha}\right)+(1-\alpha)\log\left(\frac{1-p}{1-\alpha}\right)\right)}}{\sqrt{2\pi\alpha(1-\alpha)n}}, \tag{17.4}$$

where the margin of error in the approximation is the same as in Lemma 17.5.1. From Equation 17.3, we also find that

$$f_{n,p}(\alpha n) < \frac{2^{n\left(\alpha\log\left(\frac{p}{\alpha}\right)+(1-\alpha)\log\left(\frac{1-p}{1-\alpha}\right)\right)}}{\sqrt{2\pi\alpha(1-\alpha)n}}. \tag{17.5}$$

The formula in Equations 17.4 and 17.5 is as ugly as a bowling shoe, but it's useful because it's easy to evaluate. For example, suppose we flip a fair coin n times. What is the probability of getting *exactly* pn heads? Plugging $\alpha = p$ into Equation 17.4 gives:

$$f_{n,p}(pn) \sim \frac{1}{\sqrt{2\pi p(1-p)n}}.$$

Thus, for example, if we flip a fair coin (where $p = 1/2$) $n = 100$ times, the probability of getting exactly 50 heads is within 2% of 0.079, which is about 8%.

17.5.3 Approximating the Cumulative Distribution Function

In many fields, including computer science, probability analyses come down to getting small bounds on the tails of the binomial distribution. In a typical application, you want to bound the tails in order to show that there is very small probability that too many *bad* things happen. For example, we might like to know that it is very unlikely that too many bits are corrupted in a message, or that too many servers or communication links become overloaded, or that a randomized algorithm runs for too long.

So it is usually good news that the binomial distribution has small tails. To get a feel for their size, consider the probability of flipping at most 25 heads in 100 independent tosses of a fair coin.

The probability of getting at most αn heads is given by the binomial cumulative distribution function

$$F_{n,p}(\alpha n) = \sum_{i=0}^{\alpha n} \binom{n}{i} p^i (1-p)^{n-i}. \tag{17.6}$$

We can bound this sum by bounding the ratio of successive terms.

In particular, for $i \le \alpha n$,

$$\begin{aligned}
\frac{\binom{n}{i-1} p^{i-1}(1-p)^{n-(i-1)}}{\binom{n}{i} p^i (1-p)^{n-i}} &= \frac{\dfrac{n!p^{i-1}(1-p)^{n-i+1}}{(i-1)!(n-i+1)!}}{\dfrac{n!p^i(1-p)^{n-i}}{i!(n-i)!}} \\
&= \frac{i(1-p)}{(n-i+1)p} \\
&\le \frac{\alpha n(1-p)}{(n-\alpha n+1)p} \\
&\le \frac{\alpha(1-p)}{(1-\alpha)p}.
\end{aligned}$$

This means that for $\alpha < p$,

$$\begin{aligned} F_{n,p}(\alpha n) &< f_{n,p}(\alpha n) \sum_{i=0}^{\infty} \left[\frac{\alpha(1-p)}{(1-\alpha)p} \right]^i \\ &= \frac{f_{n,p}(\alpha n)}{1 - \dfrac{\alpha(1-p)}{(1-\alpha)p}} \\ &= \left(\frac{1-\alpha}{1-\alpha/p} \right) f_{n,p}(\alpha n). \end{aligned} \tag{17.7}$$

In other words, the probability of at most αn heads is at most

$$\frac{1-\alpha}{1-\alpha/p}$$

times the probability of exactly αn heads. For our scenario, where $p = 1/2$ and $\alpha = 1/4$,

$$\frac{1-\alpha}{1-\alpha/p} = \frac{3/4}{1/2} = \frac{3}{2}.$$

Plugging $n = 100$, $\alpha = 1/4$, and $p = 1/2$ into Equation 17.5, we find that the probability of at most 25 heads in 100 coin flips is

$$F_{100,1/2}(25) < \frac{3}{2} \cdot \frac{2^{100\left(\frac{1}{4}\log(2) + \frac{3}{4}\log\left(\frac{2}{3}\right)\right)}}{\sqrt{75\pi/2}} \le 3 \cdot 10^{-7}.$$

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 $F_{n,p}(\alpha n)$ in Equation 17.7 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 fact, this is precisely what we will do in the next example.

17.5.4 Noisy Channels

Suppose you are sending packets of data across a communication channel and that each packet is lost with probability $p = .01$. Also suppose that packet losses are independent. You need to figure out how much redundancy (or error correction) to build into your communication protocol. Since redundancy is expensive overheard, you would like to use as little as possible. On the other hand, you never want to be caught short. Would it be safe for you to assume that in any batch of 10,000 packets, only 200 (or 2%) are lost? Let's find out.

The noisy channel is analogous to flipping $n = 10{,}000$ independent coins, each with probability $p = .01$ of coming up heads, and asking for the probability that there are at least αn heads where $\alpha = .02$. Since $\alpha > p$, we cannot use Equation 17.7. So we need to recast the problem by looking at the numbers of tails. In this case, the probability of tails is $p = .99$ and we are asking for the probability of at most αn tails where $\alpha = .98$.

Now we can use Equations 17.5 and 17.7 to find that the probability of losing 2% or more of the 10,000 packets is at most

$$\left(\frac{1-.98}{1-.98/.99}\right)\frac{2^{10000\left(.98\log\left(\frac{.99}{.98}\right)+.02\log\left(\frac{.01}{.02}\right)\right)}}{\sqrt{2\pi(.98)(1-.98)10000}} < 2^{-60}.$$

This is good news. It says that planning on at most 2% packet loss in a batch of 10,000 packets should be very safe, at least for the next few millennia.

17.5.5 Estimation by Sampling

Sampling is a very common technique for estimating the fraction of elements in a set that have a certain property. For example, suppose that you would like to know how many Americans plan to vote for the Republican candidate in the next presidential election. It is infeasible to ask every American how they intend to vote, so pollsters will typically contact n Americans selected at random and then compute the fraction of *those* Americans that will vote Republican. This value is then used as the *estimate* of the number of all Americans that will vote Republican. For example, if 45% of the n contacted voters report that they will vote Republican, the pollster reports that 45% of all Americans will vote Republican. In addition, the pollster will usually also provide some sort of qualifying statement such as

> "There is a 95% probability that the poll is accurate to within ±4 percentage points."

The qualifying statement is often the source of confusion and misinterpretation. For example, many people interpret the qualifying statement to mean that there is a 95% chance that between 41% and 49% of Americans intend to vote Republican. But this is wrong! The fraction of Americans that intend to vote Republican is a fixed (and unknown) value p that is *not* a random variable. Since p is not a random variable, we cannot say anything about the probability that $.41 \le p \le .49$.

To obtain a correct interpretation of the qualifying statement and the results of the poll, it is helpful to introduce some notation.

Define R_i to be the indicator random variable for the ith contacted American in the sample. In particular, set $R_i = 1$ if the ith contacted American intends to vote Republican and $R_i = 0$ otherwise. For the purposes of the analysis, we will assume that the ith contacted American is selected uniformly at random (with replacement) from the set of all Americans.[3] We will also assume that every contacted person responds honestly about whether or not they intend to vote Republican and that there are only two options—each American intends to vote Republican or they don't. Thus,

$$\Pr[R_i = 1] = p \tag{17.8}$$

where p is the (unknown) fraction of Americans that intend to vote Republican.

We next define

$$T = R_1 + R_2 + \cdots + R_n$$

to be the number of contacted Americans who intend to vote Republican. Then T/n is a random variable that is the estimate of the fraction of Americans that intend to vote Republican.

We are now ready to provide the correct interpretation of the qualifying statement. The poll results mean that

$$\Pr\big[\,|T/n - p| \le .04\big] \ge .95. \tag{17.9}$$

In other words, there is a 95% chance that the sample group will produce an estimate that is within ±4 percentage points of the correct value for the overall population. So either we were "unlucky" in selecting the people to poll or the results of the poll will be correct to within ±4 points.

How Many People Do We Need to Contact?

There remains an important question: how many people n do we need to contact to make sure that Equation 17.9 is true? In general, we would like n to be as small as possible in order to minimize the cost of the poll.

Surprisingly, the answer depends only on the desired *accuracy* and *confidence* of the poll and not on the number of items in the set being sampled. In this case, the desired accuracy is .04, the desired confidence is .95, and the set being sampled is the set of Americans. It's a good thing that n won't depend on the size of the set being sampled—there are over 300 million Americans!

[3] This means that someone could be contacted multiple times.

The task of finding an n that satisfies Equation 17.9 is made tractable by observing that T has a general binomial distribution with parameters n and p and then applying Equations 17.5 and 17.7. Let's see how this works.

Since we will be using bounds on the tails of the binomial distribution, we first do the standard conversion

$$\Pr\big[\,|T/n - p| \le .04\big] \;=\; 1 - \Pr\big[\,|T/n - p| > .04\big].$$

We then proceed to upper bound

$$\begin{aligned}\Pr\big[\,|T/n - p| > .04\big] &= \Pr[T < (p - .04)n] + \Pr[T > (p + .04)n] \\ &= F_{n,p}((p - 0.4)n) + F_{n,1-p}((1 - p - .04)n). \qquad (17.10)\end{aligned}$$

We don't know the true value of p, but it turns out that the expression on the righthand side of Equation 17.10 is maximized when $p = 1/2$ and so

$$\begin{aligned}\Pr\big[\,|T/n - p| > .04\big] &\le 2F_{n,1/2}(.46n) \\ &< 2\left(\frac{1 - .46}{1 - (.46/.5)}\right) f_{n,1/2}(.46n) \\ &< 13.5 \cdot \frac{2^{n(.46\log(\frac{.5}{.46}) + .54\log(\frac{.5}{.54}))}}{\sqrt{2\pi \cdot 0.46 \cdot 0.54 \cdot n}} \\ &< \frac{10.81 \cdot 2^{-.00462n}}{\sqrt{n}}. \qquad (17.11)\end{aligned}$$

The second line comes from Equation 17.7 using $\alpha = .46$. The third line comes from Equation 17.5.

Equation 17.11 provides bounds on the confidence of the poll for different values of n. For example, if $n = 665$, the bound in Equation 17.11 evaluates to .04978 . . . . Hence, if the pollster contacts 665 Americans, the poll will be accurate to within ±4 percentage points with at least 95% probability.

Since the bound in Equation 17.11 is exponential in n, the confidence increases greatly as n increases. For example, if $n = 6{,}650$ Americans are contacted, the poll will be accurate to within ±4 points with probability at least $1 - 10^{-10}$. Of course, most pollsters are not willing to pay the added cost of polling 10 times as many people when they already have a confidence level of 95% from polling 665 people.

18 Expectation

18.1 Definitions and Examples

The *expectation* or *expected value* of a random variable is a single number that tells you a lot about the behavior of the variable. Roughly, the expectation is the average value of the random variable where each value is weighted according to its probability. Formally, the expected value (also known as the *average* or *mean*) of a random variable is defined as follows.

Definition 18.1.1. If R is a random variable defined on a sample space $\mathcal{S}$, then the expectation of R is

$$\text{Ex}[R] ::= \sum_{w \in \mathcal{S}} R(w) \Pr[w]. \tag{18.1}$$

For example, suppose $\mathcal{S}$ is the set of students in a class, and we select a student uniformly at random. Let R be the selected student's exam score. Then $\text{Ex}[R]$ 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.

Let's work through some examples.

18.1.1 The Expected Value of a Uniform Random Variable

Let R be the value that comes up with you roll a fair 6-sided die. The the expected value of R is

$$\text{Ex}[R] = 1 \cdot \frac{1}{6} + 2 \cdot \frac{1}{6} + 3 \cdot \frac{1}{6} + 4 \cdot \frac{1}{6} + 5 \cdot \frac{1}{6} + 6 \cdot \frac{1}{6} = \frac{7}{2}.$$

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!

Also note that the mean of a random variable is not the same as the *median*. The median is the midpoint of a distribution.

Definition 18.1.2. The *median*[1] of a random variable R is the value $x \in \text{range}(R)$

[1] Some texts define the median to be the value of $x \in \text{range}(R)$ for which $\Pr[R \le x] < 1/2$ and $\Pr[R > x] \le 1/2$. The difference in definitions is not important.

such that

$$\begin{aligned}
\Pr[R \leq x] &\leq \frac{1}{2} \qquad \text{and} \\
\Pr[R > x] &< \frac{1}{2}.
\end{aligned}$$

In this text, we will not devote much attention to the median. Rather, we will focus on the expected value, which is much more interesting and useful.

Rolling a 6-sided die provides an example of a uniform random variable. In general, if R_n is a random variable with a uniform distribution on $\{1, 2, \ldots, n\}$, then

$$\text{Ex}[R_n] = \sum_{i=1}^{n} i \cdot \frac{1}{n} = \frac{n(n+1)}{2n} = \frac{n+1}{2}.$$

18.1.2 The Expected Value of an Indicator Random Variable

The expected value of an indicator random variable for an event is just the probability of that event.

Lemma 18.1.3. *If I_A is the indicator random variable for event A, then*

$$\text{Ex}[I_A] = \Pr[A].$$

Proof.

$$\begin{aligned}
\text{Ex}[I_A] &= 1 \cdot \Pr[I_A = 1] + 0 \cdot \Pr[I_A = 0] \\
&= \Pr[I_A = 1] \\
&= \Pr[A]. \qquad \text{(def of } I_A\text{)}
\end{aligned}$$

■

For example, if A is the event that a coin with bias p comes up heads, then $\text{Ex}[I_A] = \Pr[I_A = 1] = p$.

18.1.3 Alternate Definitions

There are several equivalent ways to define expectation.

Theorem 18.1.4. *If R is a random variable defined on a sample space $\mathcal{S}$ then*

$$\text{Ex}[R] = \sum_{x \in \text{range}(R)} x \cdot \Pr[R = x]. \tag{18.2}$$

The proof of Theorem 18.1.4, like many of the elementary proofs about expectation in this chapter, follows by judicious regrouping of terms in the Equation 18.1:

Proof.

$$
\begin{aligned}
\operatorname{Ex}[R] &= \sum_{\omega \in \mathcal{S}} R(\omega)\Pr[\omega] && \text{(Def 18.1.1 of expectation)}\\
&= \sum_{x \in \operatorname{range}(R)} \sum_{\omega \in [R=x]} R(\omega)\Pr[\omega] \\
&= \sum_{x \in \operatorname{range}(R)} \sum_{\omega \in [R=x]} x\Pr[\omega] && \text{(def of the event } [R=x])\\
&= \sum_{x \in \operatorname{range}(R)} x \left(\sum_{\omega \in [R=x]} \Pr[\omega] \right) && \text{(distributing } x \text{ over the inner sum)}\\
&= \sum_{x \in \operatorname{range}(R)} x \cdot \Pr[R=x]. && \text{(def of } \Pr[R=x])
\end{aligned}
$$

The first equality follows because the events $[R = x]$ for $x \in \operatorname{range}(R)$ partition the sample space $\mathcal{S}$, so summing over the outcomes in $[R = x]$ for $x \in \operatorname{range}(R)$ is the same as summing over $\mathcal{S}$. ■

In general, Equation 18.2 is more useful than Equation 18.1 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. It is especially useful when the range of the random variable is $\mathbb{N}$, as we will see from the following corollary.

Corollary 18.1.5. *If the range of a random variable R is $\mathbb{N}$, then*

$$\operatorname{Ex}[R] = \sum_{i=1}^{\infty} i\Pr[R=i] = \sum_{i=0}^{\infty} \Pr[R>i].$$

Proof. The first equality follows directly from Theorem 18.1.4 and the fact that $\operatorname{range}(R) = \mathbb{N}$. The second equality is derived by adding the following equations:

$$
\begin{array}{rcccccl}
\Pr[R>0] = & \Pr[R=1] & + & \Pr[R=2] & + & \Pr[R=3] & + \cdots\\
\Pr[R>1] = & & & \Pr[R=2] & + & \Pr[R=3] & + \cdots\\
\Pr[R>2] = & & & & & \Pr[R=3] & + \cdots\\
& & & \vdots & & &\\
\hline
\end{array}
$$

$$
\begin{aligned}
\sum_{i=0}^{\infty} \Pr[R>i] &= 1 \cdot \Pr[R=1] + 2 \cdot \Pr[R=2] + 3 \cdot \Pr[R=3] + \cdots\\
&= \sum_{i=1}^{\infty} i\Pr[R=i].
\end{aligned}
$$

■

18.1.4 Mean Time to Failure

The mean time to failure is a critical parameter in the design of most any system. For example, suppose that 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 $\text{Ex}[C]$. C is a random variable with values in $\mathbb{N}$ and so we can use Corollary 18.1.5 to determine that

$$\text{Ex}[C] = \sum_{i=0}^{\infty} \Pr[C > i]. \tag{18.3}$$

$\Pr[C > i]$ 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 Equation 18.3 gives:

$$\begin{aligned}
\text{Ex}[C] &= \sum_{i=0}^{\infty} (1-p)^i \\
&= \frac{1}{1-(1-p)} && \text{(sum of geometric series)} \\
&= \frac{1}{p}. && (18.4)
\end{aligned}$$

For example, if there is a 1% chance that the program crashes at the end of each hour, then the expected time until the program crashes is $1/0.01 = 100$ hours.

The general principle here is well-worth remembering:

> *If a system fails at each time step with probability p, then the expected number of steps up to (and including) the first failure is $1/p$.*

Making Babies

As a related example, suppose a couple really wants to have a baby girl. For simplicity, assume that there is a 50% chance that each child they have is a girl, and that 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?

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.

18.1.5 Dealing with Infinity

The analysis of the mean time to failure was easy enough. But if you think about it further, you might start to wonder about the case when the computer program *never* fails. For example, what if the program runs forever? How do we handle outcomes with an infinite value?

These are good questions and we wonder about them too. Indeed, mathematicians have gone to a lot of work to reason about sample spaces with an infinite number of outcomes or outcomes with infinite value.

To keep matters simple in this text, we will follow the common convention of ignoring the contribution of outcomes that have probability zero when computing expected values. This means that we can safely ignore the "never-fail" outcome, because it has probability

$$\lim_{n \to \infty} (1 - p)^n = 0.$$

In general, when we are computing expectations for infinite sample spaces, we will generally focus our attention on a subset of outcomes that occur with collective probability one. For the most part, this will allow us to ignore the "infinite" outcomes because they will typically happen with probability zero.[2]

This assumption does *not* mean that the expected value of a random variable is always finite, however. Indeed, there are many examples where the expected value is infinite. And where infinity raises its ugly head, trouble is sure to follow. Let's see an example.

18.1.6 Pitfall: Computing Expectations by Sampling

Suppose that you are trying to estimate a parameter such as the average delay across a communication channel. So you set up an experiment to measure how long it takes to send a test packet from one end to the other and you run the experiment 100 times.

You record the latency, rounded to the nearest millisecond, for each of the hundred experiments, and then compute the average of the 100 measurements. Suppose that this average is 8.3 ms.

Because you are careful, you repeat the entire process twice more and get averages of 7.8 ms and 7.9 ms. You conclude that the average latency across the channel

[2] If this still bothers you, you might consider taking a course on measure theory.

is

$$\frac{7.8 + 7.9 + 8.3}{3} = 8\,\text{ms}.$$

You might be right but you might also be horribly wrong. In fact, the expected latency might well be *infinite*. Here’s how.

Let D be a random variable that denotes the time it takes for the packet to cross the channel. Suppose that

$$\Pr[D = i] = \begin{cases} 0 & \text{for } i = 0 \\ \frac{1}{i} - \frac{1}{i+1} & \text{for } i \in \mathbb{N}^+. \end{cases} \tag{18.5}$$

It is easy to check that

$$\sum_{i=0}^{\infty} \Pr[D = i] = \left(1 - \frac{1}{2}\right) + \left(\frac{1}{2} - \frac{1}{3}\right) + \left(\frac{1}{3} - \frac{1}{4}\right) + \cdots = 1$$

and so D is, in fact, a random variable.

From Equation 18.5, we might expect that D is likely to be small. Indeed, $D = 1$ with probability $1/2$, $D = 2$ with probability $1/6$, and so forth. So if we took 100 samples of D, about 50 would be 1 ms, about 16 would be 2 ms, and very few would be large. In summary, it might well be the case that the average of the 100 measurements would be under 10 ms, just as in our example.

This sort of reasoning and the calculation of expected values by averaging experimental values is very common in practice. It can easily lead to incorrect conclusions, however. For example, using Corollary 18.1.5, we can quickly (and accurately) determine that

$$\begin{aligned}
\text{Ex}[D] &= \sum_{i=1}^{\infty} i \Pr[D = i] \\
&= \sum_{i=1}^{\infty} i \left(\frac{1}{i} - \frac{1}{i+1}\right) \\
&= \sum_{i=1}^{\infty} i \left(\frac{1}{i(i+1)}\right) \\
&= \sum_{i=1}^{\infty} \left(\frac{1}{i+1}\right) \\
&= \infty.
\end{aligned}$$

Uh-oh! The expected time to cross the communication channel is *infinite*! This result is a far cry from the 10 ms that we calculated. What went wrong?

It is true that most of the time, the value of D will be small. But sometimes D will be very large and this happens with sufficient probability that the expected value of D is unbounded. In fact, if you keep repeating the experiment, you are likely to see some outcomes and averages that are much larger than 10 ms. In practice, such "outliers" are sometimes discarded, which masks the true behavior of D.

In general, the best way to compute an expected value in practice is to first use the experimental data to figure out the distribution as best you can, and then to use Theorem 18.1.4 or Corollary 18.1.5 to compute its expectation. This method will help you identify cases where the expectation is infinite, and will generally be more accurate than a simple averaging of the data.

18.1.7 Conditional Expectation

Just like event probabilities, expectations can be conditioned on some event. Given a random variable R, the expected value of R conditioned on an event A is the (probability-weighted) average value of R over outcomes in A. More formally:

Definition 18.1.6. The *conditional expectation* $\text{Ex}[R \mid A]$ of a random variable R given event A is:

$$\text{Ex}[R \mid A] ::= \sum_{r \in \text{range}(R)} r \cdot \Pr\left[R = r \mid A\right]. \tag{18.6}$$

For example, we can compute the expected value of a roll of a fair die, *given*, for example, that the number rolled is at least 4. We do this by letting R be the outcome of a roll of the die. Then by equation (18.6),

$$\text{Ex}[R \mid R \geq 4] = \sum_{i=1}^{6} i \cdot \Pr\left[R = i \mid R \geq 4\right] = 1 \cdot 0 + 2 \cdot 0 + 3 \cdot 0 + 4 \cdot \tfrac{1}{3} + 5 \cdot \tfrac{1}{3} + 6 \cdot \tfrac{1}{3} = 5.$$

As another example, consider the channel latency problem from Section 18.1.6. The expected latency for this problem was infinite. But what if we look at the expected latency conditioned on the latency not exceeding n. Then

$$\begin{aligned}
\text{Ex}[D] &= \sum_{i=1}^{\infty} i \Pr\big[D = i \mid D \le n\big] \\
&= \sum_{i=1}^{\infty} i \frac{\Pr[D = i \wedge D \le n]}{\Pr[D \le n]} \\
&= \sum_{i=1}^{n} \frac{i \Pr[D = i]}{\Pr[D \le n]} \\
&= \frac{1}{\Pr[D \le n]} \sum_{i=1}^{n} i \left(\frac{1}{i(i+1)}\right) \\
&= \frac{1}{\Pr[D \le n]} \sum_{i=1}^{n} \frac{1}{i+1} \\
&= \frac{1}{\Pr[D \le n]} (H_{n+1} - 1),
\end{aligned}$$

where H_{n+1} is the $(n+1)$st Harmonic number

$$H_{n+1} = \ln(n+1) + \gamma + \frac{1}{2n} + \frac{1}{12n^2} + \frac{\epsilon(n)}{120n^4}$$

and $0 \le \epsilon(n) \le 1$. The second equality follows from the definition of conditional expectation, the third equality follows from the fact that $\Pr[D = i \wedge D \le n] = 0$ for $i > n$, and the fourth equality follows from the definition of D in Equation 18.5.

To compute $\Pr[D \le n]$, we observe that

$$\begin{aligned}
\Pr[D \le n] &= 1 - \Pr[D > n] \\
&= 1 - \sum_{i=n+1}^{\infty} \left(\frac{1}{i} - \frac{1}{i+1}\right) \\
&= 1 - \left[\left(\frac{1}{n+1} - \frac{1}{n+2}\right) + \left(\frac{1}{n+2} - \frac{1}{n+3}\right)\right. \\
&\qquad\qquad \left. + \left(\frac{1}{n+3} - \frac{1}{n+4}\right) + \cdots\right] \\
&= 1 - \frac{1}{n+1} \\
&= \frac{n}{n+1}.
\end{aligned}$$

Hence,

$$\text{Ex}[D] = \frac{n+1}{n}(H_{n+1} - 1). \tag{18.7}$$

For $n = 1000$, this is about 6.5. This explains why the expected value of D appears to be finite when you try to evaluate it experimentally. If you compute 100 samples of D, it is likely that all of them will be at most 1000 ms. If you condition on not having any outcomes greater than 1000 ms, then the conditional expected value will be about 6.5 ms, which would be a commonly observed result in practice. Yet we know that $\text{Ex}[D]$ is infinite. For this reason, expectations computed in practice are often really just conditional expectations where the condition is that rare “outlier” sample points are eliminated from the analysis.

18.1.8 The Law of Total Expectation

Another useful feature of conditional expectation is that it lets us divide complicated expectation calculations into simpler cases. We can then 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. Then

$$\begin{aligned}
\text{Ex}[H] &= \text{Ex}[H \mid M] \Pr[M] + \text{Ex}[H \mid F] \Pr[F] \\
&= (5 + 11/12) \cdot 0.498 + (5 + 5/12) \cdot 0.502 \\
&= 5.665
\end{aligned}$$

which is a little less than 5’ 8”.

This method is justified by the Law of *Total Expectation*.

Theorem 18.1.7 (Law of Total Expectation)**.** *Let R be a random variable on a sample space $\mathcal{S}$ and suppose that A_1, A_2, ..., is a partition of $\mathcal{S}$. Then*

$$\text{Ex}[R] = \sum_i \text{Ex}[R \mid A_i] \Pr[A_i].$$

Proof.

$$\begin{aligned}
\text{Ex}[R] &= \sum_{r \in \text{range}(R)} r \cdot \Pr[R = r] && \text{(Equation 18.2)} \\
&= \sum_r r \cdot \sum_i \Pr\big[R = r \mid A_i\big] \Pr[A_i] && \text{(Law of Total Probability)} \\
&= \sum_r \sum_i r \cdot \Pr\big[R = r \mid A_i\big] \Pr[A_i] && \text{(distribute constant } r\text{)} \\
&= \sum_i \sum_r r \cdot \Pr\big[R = r \mid A_i\big] \Pr[A_i] && \text{(exchange order of summation)} \\
&= \sum_i \Pr[A_i] \sum_r r \cdot \Pr\big[R = r \mid A_i\big] && \text{(factor constant } \Pr[A_i]\text{)} \\
&= \sum_i \Pr[A_i] \, \text{Ex}[R \mid A_i]. && \text{(Def 18.1.6 of cond. expectation)}
\end{aligned}$$

■

As a more interesting application of the Law of Total Expectation, let’s take another look at the mean time to failure of a system that fails with probability p at each step. We’ll define A to be the event that the system fails on the first step and $\overline{A}$ to be the complementary event (namely, that the system does not fail on the first step). Then the mean time to failure $\text{Ex}[C]$ is

$$\text{Ex}[C] = \text{Ex}[C \mid A] \Pr[A] + \text{Ex}[C \mid \overline{A}] \Pr[\overline{A}]. \tag{18.8}$$

Since A is the condition that the system crashes on the first step, we know that

$$\text{Ex}[C \mid A] = 1. \tag{18.9}$$

Since $\overline{A}$ is the condition that the system does *not* crash on the first step, conditioning on $\overline{A}$ is equivalent to taking a first step without failure and then starting over without conditioning. Hence,

$$\text{Ex}[C \mid \overline{A}] = 1 + \text{Ex}[C]. \tag{18.10}$$

Plugging Equations 18.9 and 18.10 into Equation 18.8, we find that

$$\begin{aligned}
\text{Ex}[C] &= 1 \cdot p + (1 + \text{Ex}[C])(1 - p) \\
&= p + 1 - p + (1 - p) \, \text{Ex}[C] \\
&= 1 + (1 - p) \, \text{Ex}[C].
\end{aligned}$$

Rearranging terms, we find that

$$1 = \text{Ex}[C] - (1-p)\,\text{Ex}[C] = p\,\text{Ex}[C],$$

and thus that

$$\text{Ex}[C] = \frac{1}{p},$$

as expected.

We will use this sort of analysis extensively in Chapter 20 when we examine the expected behavior of random walks.

18.1.9 Expectations of Functions

Expectations can also be defined for functions of random variables.

Definition 18.1.8. Let $R : \mathcal{S} \to V$ be a random variable and $f : V \to \mathbb{R}$ be a total function on the range of R. Then

$$\text{Ex}[f(R)] = \sum_{w \in \mathcal{S}} f(R(w))\,\Pr[w]. \tag{18.11}$$

Equivalently,

$$\text{Ex}[f(R)] = \sum_{r \in \text{range}(R)} f(r)\,\Pr[R = r]. \tag{18.12}$$

For example, suppose that R is the value obtained by rolling a fair 6-sided die. Then

$$\text{Ex}\left[\frac{1}{R}\right] = \frac{1}{1}\cdot\frac{1}{6} + \frac{1}{2}\cdot\frac{1}{6} + \frac{1}{3}\cdot\frac{1}{6} + \frac{1}{4}\cdot\frac{1}{6} + \frac{1}{5}\cdot\frac{1}{6} + \frac{1}{6}\cdot\frac{1}{6} = \frac{49}{120}.$$

18.2 Expected Returns in Gambling Games

Some of the most interesting examples of expectation can be explained in terms of gambling games. For straightforward games where you win \$$A$ with probability p and you lose \$$B$ with probability $1 - p$, it is easy to compute your *expected return* or *winnings*. It is simply

$$pA - (1-p)B.$$

For example, if you are flipping a fair coin and you win \$1 for heads and you lose \$1 for tails, then your expected winnings are

$$\frac{1}{2}\cdot 1 - \left(1 - \frac{1}{2}\right)\cdot 1 = 0.$$

In such cases, the game is said to be *fair* since your expected return is zero.

Some gambling games are more complicated and thus more interesting. For example, consider the following game where the winners split a pot. This sort of game is representative of many poker games, betting pools, and lotteries.

18.2.1 Splitting the Pot

After your last encounter with biker dude, one thing lead to another and you have dropped out of school and become a Hell's Angel. It's late on a Friday night and, feeling nostalgic for the old days, you drop by your old hangout, where you encounter two of your former TAs, Eric and Nick. Eric and Nick propose that you join them in a simple wager. Each player will put $2 on the bar and secretly write "heads" or "tails" on their napkin. Then one player will flip a fair coin. The $6 on the bar will then be divided equally among the players who correctly predicted the outcome of the coin toss.

After your life-altering encounter with strange dice, you are more than a little skeptical. So Eric and Nick agree to let you be the one to flip the coin. This certainly seems fair. How can you lose?

But you have learned your lesson and so before agreeing, you go through the four-step method and write out the tree diagram to compute your expected return. The tree diagram is shown in Figure 18.1.

The "payoff" values in Figure 18.1 are computed by dividing the $6 pot[3] among those players who guessed correctly and then subtracting the $2 that you put into the pot at the beginning. For example, if all three players guessed correctly, then you payoff is $0, since you just get back your $2 wager. If you and Nick guess correctly and Eric guessed wrong, then your payoff is

$$\frac{6}{2} - 2 = 1.$$

In the case that everyone is wrong, you all agree to split the pot and so, again, your payoff is zero.

To compute your expected return, you use Equation 18.1 in the definition of expected value. This yields

$$\begin{aligned}
\text{Ex}[\text{payoff}] = {} & 0 \cdot \frac{1}{8} + 1 \cdot \frac{1}{8} + 1 \cdot \frac{1}{8} + 4 \cdot \frac{1}{8} \\
& + (-2) \cdot \frac{1}{8} + (-2) \cdot \frac{1}{8} + (-2) \cdot \frac{1}{8} + 0 \cdot \frac{1}{8} \\
= {} & 0.
\end{aligned}$$

[3] The money invested in a wager is commonly referred to as the *pot*.

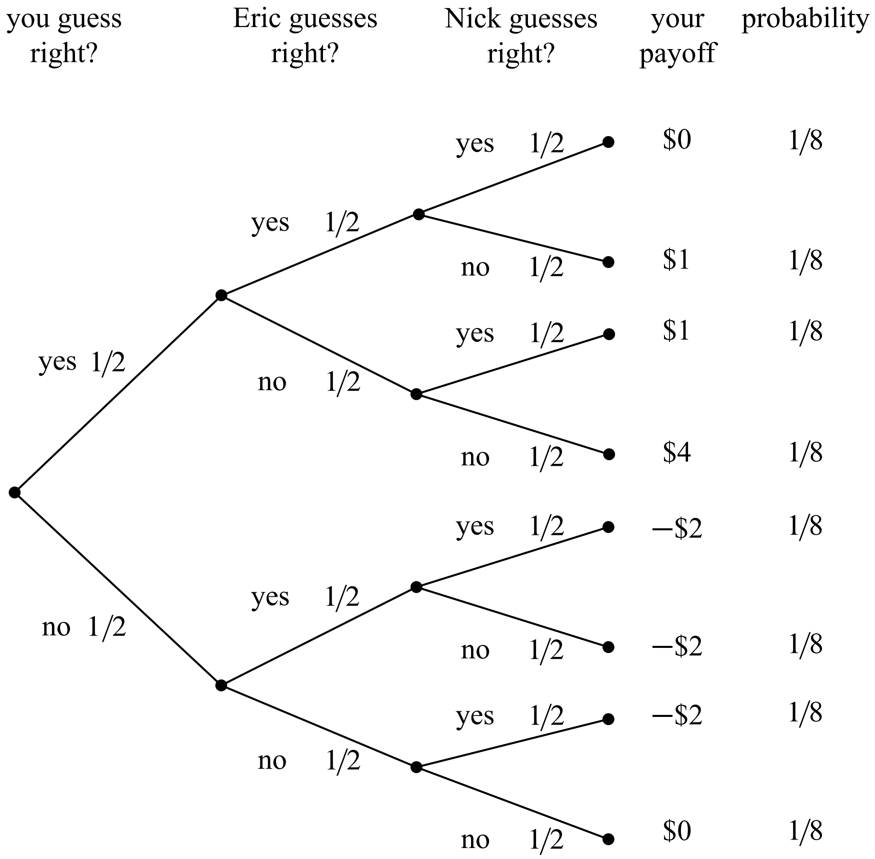


Figure 18.1 The tree diagram for the game where three players each wager $2 and then guess the outcome of a fair coin toss. The winners split the pot.

This confirms that the game is fair. So, for old time’s sake, you break your solemn vow to never ever engage in strange gambling games.

18.2.2 The Impact of Collusion

Needless to say, things are not turning out well for you. The more times you play the game, the more money you seem to be losing. After 1000 wagers, you have lost over \$500. As Nick and Eric are consoling you on your “bad luck,” you do a back-of-the-napkin calculation using the bounds on the tails of the binomial distribution from Section 17.5 that suggests that the probability of losing \$500 in 1000 wagers is less than the probability of a Vietnamese Monk waltzing in and handing you one of those golden disks. How can this be?

It is possible that you are truly very very unlucky. But it is more likely that something is wrong with the tree diagram in Figure 18.1 and that “something” just might have something to do with the possibility that Nick and Eric are colluding against you.

To be sure, Nick and Eric can only guess the outcome of the coin toss with probability $1/2$, but what if Nick and Eric always guess differently? In other words, what if Nick always guesses “tails” when Eric guesses “heads,” and vice-versa? This would result in a slightly different tree diagram, as shown in Figure 18.2.

The payoffs for each outcome are the same in Figures 18.1 and 18.2, but the probabilities of the outcomes are different. For example, it is no longer possible for all three players to guess correctly, since Nick and Eric are always guessing differently. More importantly, the outcome where your payoff is \$4 is also no longer possible. Since Nick and Eric are always guessing differently, one of them will always get a share of the pot. As you might imagine, this is not good for you!

When we use Equation 18.1 to compute your expected return in the collusion scenario, we find that

$$\begin{aligned}
\text{Ex}[\text{payoff}] &= 0 \cdot 0 + 1 \cdot \frac{1}{4} + 1 \cdot \frac{1}{4} + 4 \cdot 0 \\
&\qquad + (-2) \cdot 0 + (-2) \cdot \frac{1}{4} + (-2) \cdot \frac{1}{4} + 0 \cdot 0 \\
&= -\frac{1}{2}.
\end{aligned}$$

This is very bad indeed. By colluding, Nick and Eric have made it so that you expect to lose \$.50 every time you play. No wonder you lost \$500 over the course of 1000 wagers.

Maybe it would be a good idea to go back to school—your Hell’s Angels buds may not be too happy that you just lost their \$500.

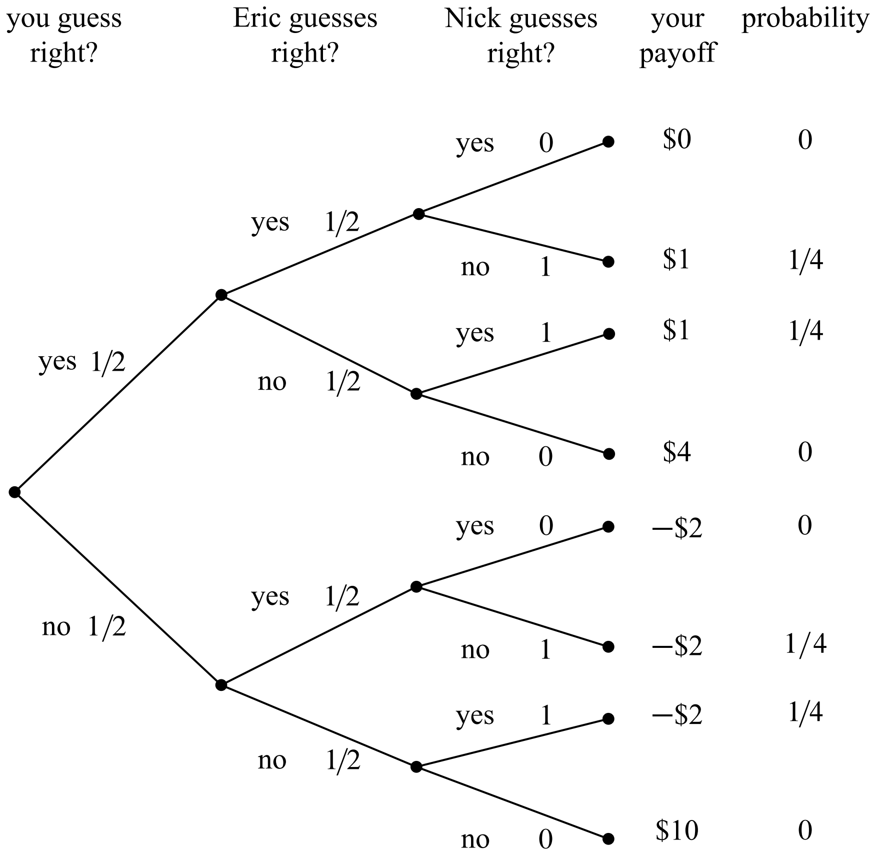


Figure 18.2 The revised tree diagram reflecting the scenario where Nick always guesses the opposite of Eric.

18.2.3 How to Win the Lottery

Similar opportunities to "collude" arise in many betting games. For example, consider the typical weekly football betting pool, where each participant wagers \$10 and the participants that pick the most games correctly split a large pot. The pool seems fair if you think of it as in Figure 18.1. But, in fact, if two or more players collude by guessing differently, they can get an "unfair" advantage at your expense!

In some cases, the collusion is inadvertent and you can profit from it. For example, many years ago, a former MIT Professor of Mathematics named Herman Chernoff figured out a way to make money by playing the state lottery. This was surprising since state lotteries typically have very poor expected returns. That's because the state usually takes a large share of the wagers before distributing the rest of the pot among the winners. Hence, anyone who buys a lottery ticket is expected to *lose* money. So how did Chernoff find a way to make money? It turned out to be easy!

In a typical state lottery,

- all players pay \$1 to play and select 4 numbers from 1 to 36,
- the state draws 4 numbers from 1 to 36 uniformly at random,
- the states divides 1/2 of the money collected among the people who guessed correctly and spends the other half redecorating the governor's residence.

This is a lot like the game you played with Nick and Eric, except that there are more players and more choices. Chernoff discovered that a small set of numbers was selected by a large fraction of the population. Apparently many people think the same way; they pick the same numbers not on purpose as in the previous game with Nick and Eric, but based on Manny's batting average or today's date.

It was as if the players were colluding to lose! If any one of them guessed correctly, then they'd have to split the pot with many other players. By selecting numbers uniformly at random, Chernoff was unlikely to get one of these favored sequences. So if he won, he'd likely get the whole pot! By analyzing actual state lottery data, he determined that he could win an average of 7 cents on the dollar. In other words, his expected return was not $-\$.50$ as you might think, but $+\$.07$.[4]

Inadvertent collusion often arises in betting pools and is a phenomenon that you can take advantage of. For example, suppose you enter a Super Bowl betting pool where the goal is to get closest to the total number of points scored in the game. Also suppose that the average Super Bowl has a total of 30 point scored and that

[4] Most lotteries now offer randomized tickets to help smooth out the distribution of selected sequences.

everyone knows this. Then most people will guess around 30 points. Where should you guess? Well, you should guess just outside of this range because you get to cover a lot more ground and you don't share the pot if you win. Of course, if you are in a pool with math students and they all know this strategy, then maybe you should guess 30 points after all.

18.3 Expectations of Sums

18.3.1 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 18.3.1. *For any random variables R_1 and R_2,*

$$\text{Ex}[R_1 + R_2] = \text{Ex}[R_1] + \text{Ex}[R_2].$$

Proof. Let $T ::= R_1 + R_2$. The proof follows straightforwardly by rearranging terms in Equation (18.1):

$$\begin{aligned}
\text{Ex}[T] &= \sum_{\omega \in \mathcal{S}} T(\omega) \cdot \Pr[\omega] && \text{(Definition 18.1.1)} \\
&= \sum_{\omega \in \mathcal{S}} (R_1(\omega) + R_2(\omega)) \cdot \Pr[\omega] && \text{(definition of } T\text{)} \\
&= \sum_{\omega \in \mathcal{S}} R_1(\omega) \Pr[\omega] + \sum_{\omega \in \mathcal{S}} R_2(\omega) \Pr[\omega] && \text{(rearranging terms)} \\
&= \text{Ex}[R_1] + \text{Ex}[R_2]. && \text{(Definition 18.1.1)}
\end{aligned}$$

■

A small extension of this proof, which we leave to the reader, implies

Theorem 18.3.2. *For random variables R_1, R_2 and constants $a_1, a_2 \in \mathbb{R}$,*

$$\text{Ex}[a_1 R_1 + a_2 R_2] = a_1 \text{Ex}[R_1] + a_2 \text{Ex}[R_2].$$

In other words, expectation is a linear function. A routine induction extends the result to more than two variables:

Corollary 18.3.3 (Linearity of Expectation)**.** *For any random variables $R_1, \ldots, R_k$ and constants $a_1, \ldots, a_k \in \mathbb{R}$,*

$$\text{Ex}[\sum_{i=1}^{k} a_i R_i] = \sum_{i=1}^{k} a_i \text{Ex}[R_i].$$

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 known to be independent.

As an example, let's compute 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:

$$\text{Ex}[R_1 + R_2] = \text{Ex}[R_1] + \text{Ex}[R_2] = 3.5 + 3.5 = 7.$$

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 remains 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!

18.3.2 Sums of Indicator Random Variables

Linearity of expectation is especially useful when you have a sum of indicator random variables. As an example, suppose 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[5] 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 the ith man gets his own hat, and $G_i = 0$ otherwise. The number of men that get their own hat is then the sum of these indicator random variables:

$$G = G_1 + G_2 + \cdots + G_n. \tag{18.13}$$

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!

[5] We are going to use this trick a lot so it is important to understand it.

Since G_i is an indicator random variable, we know from Lemma 18.1.3 that

$$\text{Ex}[G_i] = \Pr[G_i = 1] = 1/n. \tag{18.14}$$

By Linearity of Expectation and Equation 18.13, this means that

$$\begin{aligned}
\text{Ex}[G] &= \text{Ex}[G_1 + G_2 + \cdots + G_n] \\
&= \text{Ex}[G_1] + \text{Ex}[G_2] + \cdots + \text{Ex}[G_n] \\
&= \overbrace{\frac{1}{n} + \frac{1}{n} + \cdots + \frac{1}{n}}^{n} \\
&= 1.
\end{aligned}$$

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!

More generally, Linearity of Expectation provides a very good method for computing the expected number of events that will happen.

Theorem 18.3.4. *Given any collection of n events $A_1, A_2, \ldots, A_n \subseteq \mathcal{S}$, the expected number of events that will occur is*

$$\sum_{i=1}^{n} \Pr[A_i].$$

For example, A_i could be the event that the ith man gets the right hat back. But in general, it could be any subset of the sample space, and we are asking for the expected number of events that will contain a random sample point.

Proof. Define R_i to be the indicator random variable for A_i, where $R_i(w) = 1$ if $w \in A_i$ and $R_i(w) = 0$ if $w \notin A_i$. Let $R = R_1 + R_2 + \cdots + R_n$. Then

$$\begin{aligned}
\text{Ex}[R] &= \sum_{i=1}^{n} \text{Ex}[R_i] && \text{(by Linearity of Expectation)} \\
&= \sum_{i=1}^{n} \Pr[R_i = 1] && \text{(by Lemma 18.1.3)} \\
&= \sum_{i=1}^{n} \sum_{w \in A_i} \Pr[w] && \text{(definition of indicator variable)} \\
&= \sum_{i=1}^{n} \Pr[A_i]. && \blacksquare
\end{aligned}$$

So whenever you are asked for the expected number of events that occur, all you have to do is sum the probabilities that each event occurs. Independence is not needed.

18.3.3 Expectation of a Binomial Distribution

Suppose that we independently flip n biased coins, each with probability p of coming up heads. What is the expected number of heads?

Let J be the random variable denoting the number of heads. Then J has a binomial distribution with parameters n, p, and

$$\Pr[J = k] = \binom{n}{k} k^p (n-k)^{1-p}.$$

Applying Equation 18.2, this means that

$$\begin{aligned}\text{Ex}[J] &= \sum_{k=0}^{n} k \Pr[J = k] \\ &= \sum_{k=0}^{n} k \binom{n}{k} k^p (n-k)^{1-p}. \qquad (18.15)\end{aligned}$$

Ouch! This is one nasty looking sum. Let's try another approach.

Since we have just learned about linearity of expectation for sums of indicator random variables, maybe Theorem 18.3.4 will be helpful. But how do we express J as a sum of indicator random variables? It turns out to be easy. Let J_i be the indicator random variable for the ith coin. In particular, define

$$J_i = \begin{cases} 1 & \text{if the } i\text{th coin is heads} \\ 0 & \text{if the } i\text{th coin is tails.} \end{cases}$$

Then the number of heads is simply

$$J = J_1 + J_2 + \cdots + J_n.$$

By Theorem 18.3.4,

$$\begin{aligned}\text{Ex}[J] &= \sum_{i=1}^{n} \Pr[J_i] \\ &= np. \qquad (18.16)\end{aligned}$$

That really was easy. If we flip n mutually independent coins, we expect to get pn heads. Hence the expected value of a binomial distribution with parameters n and p is simply pn.

But what if the coins are not mutually independent? It doesn't matter—the answer is still pn because Linearity of Expectation and Theorem 18.3.4 do not assume any independence.

If you are not yet convinced that Linearity of Expectation and Theorem 18.3.4 are powerful tools, consider this: without even trying, we have used them to prove a very complicated identity, namely[6]

$$\sum_{k=0}^{n} k \binom{n}{k} k^p (n-k)^{1-p} = pn.$$

If you are still not convinced, then take a look at the next problem.

18.3.4 The Coupon Collector Problem

Every time we purchase a kid's meal at Taco Bell, we are graciously presented with a miniature "Racin' Rocket" car together with a launching device which enables us to project our new vehicle across any tabletop or smooth floor at high velocity. Truly, our delight knows no bounds.

There are n different types of Racin' Rocket cars (blue, green, red, gray, etc.). The type of car awarded to us 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 we 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 cars, and we receive this sequence:

$$\text{blue} \quad \text{green} \quad \text{green} \quad \text{red} \quad \text{blue} \quad \text{orange} \quad \text{blue} \quad \text{orange} \quad \text{gray}.$$

Let's partition the sequence into 5 segments:

$$\underbrace{\text{blue}}_{X_0} \quad \underbrace{\text{green}}_{X_1} \quad \underbrace{\text{green} \quad \text{red}}_{X_2} \quad \underbrace{\text{blue} \quad \text{orange}}_{X_3} \quad \underbrace{\text{blue} \quad \text{orange} \quad \text{gray}}_{X_4}.$$

[6] This follows by combining Equations 18.15 and 18.16.

The rule is that a segment ends whenever we get a new kind of car. For example, the middle segment ends when we 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 we're collecting n Racin' Rockets. Let X_k be the length of the kth segment. The total number of kid's meals we 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, we have k different types of car, and the segment ends when we acquire a new type. When we own k types, each kid's meal contains a type that we 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 we get a new kind of car is $n/(n-k)$ by the "mean time to failure" formula in Equation 18.4. This means that

$$\text{Ex}[X_k] = \frac{n}{n-k}.$$

Linearity of expectation, together with this observation, solves the coupon collector problem:

$$\begin{aligned}
\text{Ex}[T] &= \text{Ex}[X_0 + X_1 + \cdots + X_{n-1}] \\
&= \text{Ex}[X_0] + \text{Ex}[X_1] + \cdots + \text{Ex}[X_{n-1}] \\
&= \frac{n}{n-0} + \frac{n}{n-1} + \cdots + \frac{n}{3} + \frac{n}{2} + \frac{n}{1} \\
&= n\left(\frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{3} + \frac{1}{2} + \frac{1}{1}\right) \\
&= n\left(\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n-1} + \frac{1}{n}\right) \\
&= nH_n \qquad (18.17) \\
&\sim n \ln n. \qquad (18.18)
\end{aligned}$$

Wow! It's those Harmonic Numbers again!

We can use Equation 18.18 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:

$$365H_{365} = 2364.6\ldots.$$

18.3.5 Infinite Sums

Linearity of expectation also works for an infinite number of random variables provided that the variables satisfy some stringent absolute convergence criteria.

Theorem 18.3.5 (Linearity of Expectation)**.** *Let R_0, R_1, …, be random variables such that*

$$\sum_{i=0}^{\infty} \text{Ex}[|R_i|]$$

converges. Then

$$\text{Ex}\left[\sum_{i=0}^{\infty} R_i\right] = \sum_{i=0}^{\infty} \text{Ex}[R_i].$$

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:

$$\begin{aligned}
\sum_{i=0}^{\infty} \text{Ex}[R_i] &= \sum_{i=0}^{\infty} \sum_{s \in \mathcal{S}} R_i(s) \cdot \Pr[s] && \text{(Def. 18.1.1)} \\
&= \sum_{s \in \mathcal{S}} \sum_{i=0}^{\infty} R_i(s) \cdot \Pr[s] && \text{(exchanging order of summation)} \\
&= \sum_{s \in \mathcal{S}} \left[\sum_{i=0}^{\infty} R_i(s)\right] \cdot \Pr[s] && \text{(factoring out } \Pr[s]) \\
&= \sum_{s \in \mathcal{S}} T(s) \cdot \Pr[s] && \text{(Def. of } T) \\
&= \text{Ex}[T] && \text{(Def. 18.1.1)} \\
&= \text{Ex}[\sum_{i=0}^{\infty} R_i]. && \text{(Def. of } T). \quad \blacksquare
\end{aligned}$$

18.4 Expectations of Products

While the expectation of a sum is the sum of the expectations, the same is usually not true for products. For example, suppose that we roll a fair 6-sided die and denote the outcome with the random variable R. Does $\text{Ex}[R \cdot R] = \text{Ex}[R] \cdot \text{Ex}[R]$?

We know that $\text{Ex}[R] = 3\frac{1}{2}$ and thus $\text{Ex}[R]^2 = 12\frac{1}{4}$. Let's compute $\text{Ex}[R^2]$ to see if we get the same result.

$$\begin{aligned}
\text{Ex}[R^2] &= \sum_{w \in \mathcal{S}} R^2(w) \Pr[w] \\
&= \sum_{i=1}^{6} i^2 \cdot \Pr[R_i = i] \\
&= \frac{1^2}{6} + \frac{2^2}{6} + \frac{3^2}{6} + \frac{4^2}{6} + \frac{5^2}{6} + \frac{6^2}{6} \\
&= 15\ 1/6 \\
&\neq 12\ 1/4.
\end{aligned}$$

Hence,

$$\text{Ex}[R \cdot R] \neq \text{Ex}[R] \cdot \text{Ex}[R]$$

and so the expectation of a product is not always equal to the product of the expectations.

There is a special case when such a relationship *does* hold however; namely, when the random variables in the product are *independent*.

Theorem 18.4.1. *For any two* independent *random variables R_1, R_2,*

$$\text{Ex}[R_1 \cdot R_2] = \text{Ex}[R_1] \cdot \text{Ex}[R_2].$$

Proof. The event $[R_1 \cdot R_2 = r]$ can be split up into events of the form $[R_1 =$

r_1 and $R_2 = r_2]$ where $r_1 \cdot r_2 = r$. So

$$\begin{aligned}
&\text{Ex}[R_1 \cdot R_2] \\
&= \sum_{r \in \text{range}(R_1 \cdot R_2)} r \cdot \Pr[R_1 \cdot R_2 = r] && \text{(Theorem 18.1.4)} \\
&= \sum_{r_1 \in \text{range}(R_1)} \sum_{r_2 \in \text{range}(R_2)} r_1 r_2 \cdot \Pr[R_1 = r_1 \text{ and } R_2 = r_2] \\
&= \sum_{r_1 \in \text{range}(R_1)} \sum_{r_2 \in \text{range}(R_2)} r_1 r_2 \cdot \Pr[R_1 = r_1] \cdot \Pr[R_2 = r_2] && \text{(independence of } R_1, R_2\text{)} \\
&= \sum_{r_1 \in \text{range}(R_1)} r_1 \Pr[R_1 = r_1] \left(\sum_{r_2 \in \text{range}(R_2)} r_2 \Pr[R_2 = r_2] \right) && \text{(factor out } r_1 \Pr[R_1 = r_1]\text{)} \\
&= \sum_{r_1 \in \text{range}(R_1)} r_1 \Pr[R_1 = r_1] \cdot \text{Ex}[R_2] && \text{(Theorem 18.1.4)} \\
&= \text{Ex}[R_2] \left(\sum_{r_1 \in \text{range}(R_1)} r_1 \Pr[R_1 = r_1] \right) && \text{(factor out Ex}[R_2]\text{)} \\
&= \text{Ex}[R_2] \cdot \text{Ex}[R_1]. && \text{(Theorem 18.1.4)}
\end{aligned}$$

■

For example, let R_1 and R_2 be random variables denoting the result of rolling two independent and fair 6-sided dice. Then

$$\text{Ex}[R_1 \cdot R_2] = \text{Ex}[R_1]\,\text{Ex}[R_2] = 3\frac{1}{2} \cdot 3\frac{1}{2} = 12\frac{1}{4}.$$

Theorem 18.4.1 extends by induction to a collection of mutually independent random variables.

Corollary 18.4.2. *If random variables $R_1, R_2, \ldots, R_k$ are mutually independent, then*

$$\text{Ex}\left[\prod_{i=1}^{k} R_i\right] = \prod_{i=1}^{k} \text{Ex}[R_i].$$

18.5 Expectations of Quotients

If S and T are random variables, we know from Linearity of Expectation that

$$\text{Ex}[S + T] = \text{Ex}[S] + \text{Ex}[T].$$

If S and T are independent, we know from Theorem 18.4.1 that

$$\text{Ex}[ST] = \text{Ex}[S] \, \text{Ex}[T].$$

Is it also true that

$$\text{Ex}[S/T] = \text{Ex}[S]/\, \text{Ex}[T]? \tag{18.19}$$

Of course, we have to worry about the situation when $\text{Ex}[T] = 0$, but what if we assume that T is always positive? As we will soon see, Equation 18.19 is usually not true, but let's see if we can prove it anyway.

False Claim 18.5.1. *If S and T are independent random variables with $T > 0$, then*

$$\text{Ex}[S/T] = \text{Ex}[S]/\, \text{Ex}[T]. \tag{18.20}$$

Bogus proof.

$$\begin{aligned}
\text{Ex}[\frac{S}{T}] &= \text{Ex}[S \cdot \frac{1}{T}] \\
&= \text{Ex}[S] \cdot \text{Ex}\left[\frac{1}{T}\right] && \text{(independence of } S \text{ and } T\text{)} && (18.21) \\
&= \text{Ex}[S] \cdot \frac{1}{\text{Ex}[T]}. && && (18.22) \\
&= \frac{\text{Ex}[S]}{\text{Ex}[T]}. && \blacksquare
\end{aligned}$$

Note that line 18.21 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 are independent. It is a fact that needs proof, which we will leave to the reader, but it is not the bug. The bug is in line (18.22), which assumes

False Claim 18.5.2.

$$\text{Ex}[\frac{1}{T}] = \frac{1}{\text{Ex}[T]}.$$

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

Table 18.1 Sample program lengths for benchmark problems using RISC and CISC compilers.

Here is a counterexample. Define T so that

$$\Pr[T = 1] = \frac{1}{2} \quad \text{and} \quad \Pr[T = 2] = \frac{1}{2}.$$

Then

$$\text{Ex}[T] = 1 \cdot \frac{1}{2} + 2 \cdot \frac{1}{2} = \frac{3}{2}$$

and

$$\frac{1}{\text{Ex}[T]} = \frac{2}{3}$$

and

$$\text{Ex}\left[\frac{1}{T}\right] = \frac{1}{1} \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2} = \frac{3}{4} \neq \frac{1}{\text{Ex}[1/T]}.$$

This means that Claim 18.5.1 is also false since we could define $S = 1$ with probability 1. In fact, both Claims 18.5.1 and 18.5.2 are untrue for most all choices of S and T. Unfortunately, the fact that they are false does not keep them from being widely used in practice! Let's see an example.

18.5.1 A RISC Paradox

The data in Table 18.1 is representative of data in 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.

Each row in Table 18.1 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 conclusion is that CISC programs are 20% longer on average.

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

Table 18.2 The same data as in Table 18.1, but with the opposite ratio in the last column.

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, as shown in Table 18.2.

From Table 18.2, 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?

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 $\text{Ex}[R]$ of a RISC program to the average length $\text{Ex}[C]$ 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 $\text{Ex}[C/R] = 1.2$. This much is correct. The authors then conclude that CISC programs are 20% longer on average; that is, they conclude that $\text{Ex}[C] = 1.2 \text{ Ex}[R]$. Therein lies the problem. The authors have implicitly used False Claim 18.5.1 to assume that $\text{Ex}[C/R] = \text{Ex}[C]/\text{Ex}[R]$. By using the same false logic, the critics can arrive at the opposite conclusion; namely, that RISC programs are 10% longer on average.

The Proper Quotient

We can compute Ex[R] and Ex[C] as follows:

$$\begin{aligned}
\text{Ex}[R] &= \sum_{i \in \text{Range(R)}} i \cdot \Pr[R = i] \\
&= \frac{150}{4} + \frac{120}{4} + \frac{150}{4} + \frac{2800}{4} \\
&= 805,
\end{aligned}$$

$$\begin{aligned}
\text{Ex}[C] &= \sum_{i \in \text{Range(C)}} i \cdot \Pr[C = i] \\
&= \frac{120}{4} + \frac{180}{4} + \frac{300}{4} + \frac{1400}{4} \\
&= 500
\end{aligned}$$

Now since $\text{Ex}[R]/\text{Ex}[C] = 1.61$, 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.

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!

19 Deviations

In some cases, a random variable is likely to be very close to its expected value. For example, if we flip 100 fair, mutually-independent coins, it is very likely that we will get about 50 heads. In fact, we proved in Section 17.5 that the probability of getting fewer than 25 or more than 75 heads are each less than $3 \cdot 10^{-7}$. In such cases, the mean provides a lot of information about the random variable.

In other cases, a random variable is likely to be *far* from its expected value. For example, suppose we flipped 100 fair coins that are glued together so that they all come out "heads" or they call all come out "tails." In this case, the expected value of the number of heads is still 50, but the actual number of heads is guaranteed to be far from this value—it will be 0 or 100, each with probability $1/2$.

Mathematicians have developed a variety of measures and methods to help us understand how a random variable performs in comparison to its mean. The simplest and most widely used measure is called the *variance* of the random variable. The variance is a single value associated with the random variable that is large for random variables that are likely to deviate significantly from the mean and that is small otherwise.

19.1 Variance

19.1.1 Definition and Examples

Consider the following two gambling games:

Game A: You win \$2 with probability $2/3$ and lose \$1 with probability $1/3$.

Game B: You win \$1002 with probability $2/3$ and lose \$2001 with probability $1/3$.

Which game would you rather play? Which game is better financially? We have the same probability, 2/3, of winning each game, but that does not tell the whole story. What about the expected return for each game? Let random variables A and B be the payoffs for the two games. For example, A is 2 with probability 2/3 and -1 with probability 1/3. We can compute the expected payoff for each game as follows:

$$\begin{aligned}\text{Ex}[A] &= 2 \cdot \frac{2}{3} + (-1) \cdot \frac{1}{3} = 1,\\ \text{Ex}[B] &= 1002 \cdot \frac{2}{3} + (-2001) \cdot \frac{1}{3} = 1.\end{aligned}$$

The expected payoff is the same for both games, but they are obviously very different! The stakes are a lot higher for Game B and so it is likely to deviate much farther from its mean than is Game A. This fact is captured by the notion of *variance*.

Definition 19.1.1. The *variance* Var$[R]$ of a random variable R is

$$\text{Var}[R] ::= \text{Ex}[(R - \text{Ex}[R])^2].$$

In words, the variance of a random variable R is the expectation of the square of the amount by which R differs from its expectation.

Yikes! That’s a mouthful. Try saying that 10 times in a row!

Let’s look at this definition more carefully. We’ll start with $R - \text{Ex}[R]$. That’s the amount by which R differs from its expectation and it is obviously an important measure. Next, we square this value. More on why we do that in a moment. Finally, we take the the expected value of the square. If the square is likely to be large, then the variance will be large. If it is likely to be small, then the variance will be small. That’s just the kind of statistic we are looking for. Let’s see how it works out for our two gambling games.

We’ll start with Game A:

$$\begin{aligned}A - \text{Ex}[A] &= \begin{cases} 1 & \text{with probability } \frac{2}{3} \\ -2 & \text{with probability } \frac{1}{3} \end{cases}\\ (A - \text{Ex}[A])^2 &= \begin{cases} 1 & \text{with probability } \frac{2}{3} \\ 4 & \text{with probability } \frac{1}{3} \end{cases}\\ \text{Ex}[(A - \text{Ex}[A])^2] &= 1 \cdot \frac{2}{3} + 4 \cdot \frac{1}{3}\\ \text{Var}[A] &= 2. \qquad (19.1)\end{aligned}$$

For Game B, we have

$$\begin{aligned}
B - \text{Ex}[B] &= \begin{cases} 1001 & \text{with probability } \frac{2}{3} \\ -2002 & \text{with probability } \frac{1}{3} \end{cases} \\
(B - \text{Ex}[B])^2 &= \begin{cases} 1{,}002{,}001 & \text{with probability } \frac{2}{3} \\ 4{,}008{,}004 & \text{with probability } \frac{1}{3} \end{cases} \\
\text{Ex}[(B - \text{Ex}[B])^2] &= 1{,}002{,}001 \cdot \frac{2}{3} + 4{,}008{,}004 \cdot \frac{1}{3} \\
\text{Var}[B] &= 2{,}004{,}002.
\end{aligned}$$

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!

Why Bother Squaring?

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 $\text{Ex}[R - \text{Ex}[R]]$?

The problem with this definition is that the positive and negative deviations from the mean exactly cancel. By linearity of expectation, we have:

$$\text{Ex}\left[R - \text{Ex}[R]\right] = \text{Ex}[R] - \text{Ex}\left[\text{Ex}[R]\right].$$

Since $\text{Ex}[R]$ is a constant, its expected value is itself. Therefore

$$\text{Ex}\left[R - \text{Ex}[R]\right] = \text{Ex}[R] - \text{Ex}[R] = 0.$$

By this definition, every random variable would have zero variance, which would not be very useful! Because of the square in the conventional definition, both positive and negative deviations from the mean increase the variance, and they do not cancel.

Of course, we could also prevent positive and negative deviations from canceling by taking an absolute value. In other words, we could compute $\text{Ex}[\,|R - \text{Ex}[R]|\,]$. But this measure doesn’t have the many useful properties that variance has, and so mathematicians went with squaring.

19.1.2 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 these reasons, people often describe the deviation of a random variable using *standard deviation* instead of variance.

Definition 19.1.2. The *standard deviation* σ_R of a random variable R is the square root of the variance:

$$\sigma_R ::= \sqrt{\text{Var}[R]} = \sqrt{\text{Ex}[(R - \text{Ex}[R])^2]}.$$

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 roughly canceling the square on the inside.

For example, the standard deviations for A and B are

$$\begin{aligned} \sigma_A &= \sqrt{\text{Var}[A]} = \sqrt{2} \approx 1.41, \\ \sigma_B &= \sqrt{\text{Var}[B]} = \sqrt{2,004,002} \approx 1416. \end{aligned}$$

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.

19.1.3 An Alternative Formulation

Applying linearity of expectation to the formula for variance yields a convenient alternative formula.

Lemma 19.1.3. *For any random variable R,*

$$\text{Var}[R] = \text{Ex}[R^2] - \text{Ex}^2[R].$$

Here we use the notation $\text{Ex}^2[R]$ as shorthand for $(\text{Ex}[R])^2$. Remember that $\text{Ex}[R^2]$ is generally not equal to $\text{Ex}^2[R]$. 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 of Lemma 19.1.3. Let $\mu = \text{Ex}[R]$. Then

$$\begin{aligned}
\text{Var}[R] &= \text{Ex}[(R - \text{Ex}[R])^2] && \text{(Definition 19.1.1 of variance)}\\
&= \text{Ex}[(R - \mu)^2] && \text{(definition of } \mu\text{)}\\
&= \text{Ex}[R^2 - 2\mu R + \mu^2] \\
&= \text{Ex}[R^2] - 2\mu\,\text{Ex}[R] + \mu^2 && \text{(linearity of expectation)}\\
&= \text{Ex}[R^2] - 2\mu^2 + \mu^2 && \text{(definition of } \mu\text{)}\\
&= \text{Ex}[R^2] - \mu^2 \\
&= \text{Ex}[R^2] - \text{Ex}^2[R]. && \text{(definition of } \mu\text{)} \quad \blacksquare
\end{aligned}$$

For example, let's take another look at Game A from Section 19.1 where you win \$2 with probability $2/3$ and lose \$1 with probability $1/3$. Then

$$\text{Ex}[A] = 2 \cdot \frac{2}{3} + (-1) \cdot \frac{1}{3} = 1$$

and

$$\text{Ex}[A^2] = 4 \cdot \frac{2}{3} + 1 \cdot \frac{1}{3} = 3.$$

By Lemma 19.1.3, this means that

$$\text{Var}[A] = \text{Ex}[A^2] - \text{Ex}^2[A] = 3 - 1^2 = 2,$$

confirming the result in Equation 19.1.

The alternate formulation of variance given in Lemma 19.1.3 has a cute implication:

Corollary 19.1.4. *If R is a random variable, then* $\text{Ex}[R^2] \geq \text{Ex}^2[R]$.

Proof. We defined $\text{Var}[R]$ as an average of a squared expression, so $\text{Var}[R]$ is nonnegative. Then we proved that $\text{Var}[R] = \text{Ex}[R^2] - \text{Ex}^2[R]$. This implies that $\text{Ex}[R^2] - \text{Ex}^2[R]$ is nonnegative. Therefore, $\text{Ex}[R^2] \geq \text{Ex}^2[R]$. ■

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:

$$\text{Ex}[R^2] = \text{Ex}^2[R] \quad \text{iff} \quad \text{Ex}[R^2] - \text{Ex}^2[R] = 0 \quad \text{iff} \quad \text{Var}[R] = 0.$$

This happens precisely when

$$\Pr\big[R = \text{Ex}[R]\big] = 1;$$

namely, when R is a constant.[1]

[1] Technically, R could deviate from its mean on some sample points with probability 0, but we are ignoring events of probability 0 when computing expectations and variances.

19.1.4 Indicator Random Variables

Computing the variance of an indicator random variable is straightforward given Lemma 19.1.3.

Lemma 19.1.5. *Let B be an indicator random variable for which* $\Pr[B = 1] = p$. *Then*

$$\text{Var}[B] = p - p^2 = p(1 - p). \tag{19.2}$$

Proof. By Lemma 18.1.3, $\text{Ex}[B] = p$. But since B only takes values 0 and 1, $B^2 = B$. So

$$\text{Var}[B] = \text{Ex}[B^2] - \text{Ex}^2[B] = p - p^2,$$

as claimed. ■

For example, let R be the number of heads when you flip a single fair coin. Then

$$\text{Var}[R] = \frac{1}{2} - \left(\frac{1}{2}\right)^2 = \frac{1}{4} \tag{19.3}$$

and

$$\sigma_R = \sqrt{\frac{1}{4}} = \frac{1}{2}.$$

19.1.5 Mean Time to Failure

As another example, consider the mean time to failure problem, described in Section 18.1.4. If the system crashes at each step with probability p, then we already know that the mean time to failure is $1/p$. In other words, if C is the number of steps up to and including the step when the first crash occurs, then

$$\text{Ex}[C] = \frac{1}{p}.$$

What about the variance of C? To use Lemma 19.1.3, we need to compute $\text{Ex}[C^2]$. As in Section 18.1.4, we can do this by summing over all the sample points or we can use the Law of Total Expectation. The latter approach is simpler, so we'll do that. The analysis breaks into two cases: the system crashes in the first step or it doesn't. Hence,

$$\begin{aligned}
\text{Ex}[C^2] &= 1^2 \cdot p + \text{Ex}[(C + 1)^2](1 - p) \\
&= p + \text{Ex}[C^2](1 - p) + 2\,\text{Ex}[C](1 - p) + (1 - p) \\
&= 1 + \text{Ex}[C^2](1 - p) + 2\left(\frac{1 - p}{p}\right).
\end{aligned}$$

Simplifying, we find that

$$p\,\text{Ex}[C^2] = \frac{2-p}{p}$$

and that

$$\text{Ex}[C^2] = \frac{2-p}{p^2}.$$

Using Lemma 19.1.3, we conclude that

$$\begin{aligned}
\text{Var}[C] &= \text{Ex}[C^2] - \text{Ex}^2[C] \\
&= \frac{2-p}{p^2} - \frac{1}{p^2} \\
&= \frac{1-p}{p^2}.
\end{aligned}$$

19.1.6 Uniform Random Variables

Computing the variance of a uniform random variable is also straightforward given Lemma 19.1.3. For example, we can compute the variance of the outcome of a fair die R as follows:

$$\begin{aligned}
\text{Ex}[R^2] &= \frac{1}{6}(1^2 + 2^2 + 3^2 + 4^2 + 5^2 + 6^2) = \frac{91}{6}, \\
\text{Ex}^2[R] &= \left(3\frac{1}{2}\right)^2 = \frac{49}{4}, \\
\text{Var}[R] &= \text{Ex}[R^2] - \text{Ex}^2[R] = \frac{91}{6} - \frac{49}{4} = \frac{35}{12}.
\end{aligned}$$

For a general uniform random variable R on $\{1, 2, 3, \ldots n\}$, the variance can be computed as follows:

$$\begin{aligned}
\operatorname{Ex}[R] &= \frac{1}{n}(1+2+\cdots+n)\\
&= \frac{1}{n}\cdot\frac{n(n+1)}{2}\\
&= \frac{n+1}{2}.\\[1ex]
\operatorname{Ex}[R^2] &= \frac{1}{n}(1^2+2^2+\cdots+n^2)\\
&= \frac{1}{n}\cdot\frac{(2n+1)n(n+1)}{6}\\
&= \frac{(2n+1)(n+1)}{6}.\\[1ex]
\operatorname{Var}[R] &= \operatorname{Ex}[R^2]-\operatorname{Ex}^2[R]\\
&= \frac{(2n+1)(n+1)}{6}-\left(\frac{n+1}{2}\right)^2\\
&= \frac{n^2-1}{12}.
\end{aligned}$$

19.1.7 Dealing with Constants

It helps to know how to calculate the variance of $aR+b$:

Theorem 19.1.6. *Let R be a random variable, and let a and b be constants. Then*

$$\operatorname{Var}[aR+b] = a^2\operatorname{Var}[R]. \tag{19.4}$$

Proof. Beginning with Lemma 19.1.3 and repeatedly applying linearity of expectation, we have:

$$\begin{aligned}
\operatorname{Var}[aR] &= \operatorname{Ex}[(aR+b)^2]-\operatorname{Ex}^2[aR+b]\\
&= \operatorname{Ex}[a^2R^2+2abR+b^2]-(a\operatorname{Ex}[R]+b)^2\\
&= a^2\operatorname{Ex}[R^2]+2ab\operatorname{Ex}[R]+b^2-a^2\operatorname{Ex}^2[R]-2ab\operatorname{Ex}[R]-b^2\\
&= a^2\operatorname{Ex}[R^2]-a^2\operatorname{Ex}^2[R]\\
&= a^2\left(\operatorname{Ex}[R^2]-\operatorname{Ex}^2[R]\right)\\
&= a^2\operatorname{Var}[R] \qquad \text{(by Lemma 19.1.3).}
\end{aligned}$$

■

Corollary 19.1.7.

$$\sigma_{aR+b} = |a|\,\sigma_R.$$

19.1.8 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* random variables. In fact, *mutual* independence is not necessary: *pairwise* independence will do.

Theorem 19.1.8. *If R_1 and R_2 are independent random variables, then*

$$\text{Var}[R_1 + R_2] = \text{Var}[R_1] + \text{Var}[R_2]. \tag{19.5}$$

Proof. As with the proof of Theorem 19.1.6, this proof uses repeated applications of Lemma 19.1.3 and Linearity of Expectation.

$$\begin{aligned}
\text{Var}[R_1 + R_2] &= \text{Ex}[(R_1 + R_2)^2] - \text{Ex}^2[R_1 + R_2] \\
&= \text{Ex}[R_1^2 + 2R_1R_2 + R^2] - (\text{Ex}[R_1] + \text{Ex}[R_2])^2 \\
&= \text{Ex}[R_1^2] + 2\,\text{Ex}[R_1R_2] + \text{Ex}[R_2^2] \\
&\qquad - \text{Ex}^2[R_1] - 2\,\text{Ex}[R_1]\,\text{Ex}[R_2] - \text{Ex}^2[R_2] \\
&= \text{Var}[R_1] + \text{Var}[R_2] + 2(\text{Ex}[R_1R_2] - \text{Ex}[R_1]\,\text{Ex}[R_2]) \\
&= \text{Var}[R_1] + \text{Var}[R_2].
\end{aligned}$$

The last step follows because

$$\text{Ex}[R_1R_2] = \text{Ex}[R_1]\,\text{Ex}[R_2]$$

when R_1 and R_2 are independent. ■

Note that Theorem 19.1.8 does not necessarily hold if R_1 and R_2 are dependent since then it would generally not be true that

$$\text{Ex}[R_1R_2] = \text{Ex}[R_1]\,\text{Ex}[R_2] \tag{19.6}$$

in the last step of the proof. For example, suppose that $R_1 = R_2 = R$. Then Equation 19.6 holds only if R is essentially constant.

The proof of Theorem 19.1.8 carries over straightforwardly to the sum of any finite number of variables.

Theorem 19.1.9 (Pairwise Independent Additivity of Variance)**.** *If $R_1, R_2, \ldots, R_n$ are* pairwise *independent random variables, then*

$$\text{Var}[R_1 + R_2 + \cdots + R_n] = \text{Var}[R_1] + \text{Var}[R_2] + \cdots + \text{Var}[R_n]. \tag{19.7}$$

Unfortunately, there is no product rule for computing variances, even if the random variables are mutually independent. However, we can use Theorem 19.1.9 to quickly compute the variance of a random variable with a general binomial distribution.

19.1.9 Binomial Distributions

Lemma 19.1.10 (Variance of the Binomial Distribution)**.** *If J has a binomial distribution with parameters n and p, then*

$$\text{Var}[J] = np(1-p). \tag{19.8}$$

Proof. From the definition of the binomial distribution, we can think of J as being the number of "heads" when you flip n mutually independent coins, each of which is "heads" with probability p. Thus J can be expressed as the sum of n mutually independent indicator variables J_i where

$$\Pr[J_i = 1] = p$$

for $1 \le i \le n$. From Lemma 19.1.5, we know that

$$\text{Var}[J_i] = p(1-p).$$

By Theorem 19.1.9, this means that

$$\text{Var}[J] = \sum_{i=1}^{n} \text{Var}[J_i] = np(1-p). \qquad \blacksquare$$

For example, suppose we flip n mutually independent[2] fair coins. Let R be the number of heads. Then Theorem 19.1.9 tells us that

$$\text{Var}[R] = n\left(\frac{1}{2}\right)\left(1 - \frac{1}{2}\right) = \frac{n}{4}.$$

Hence,

$$\sigma_R = \frac{\sqrt{n}}{2}.$$

This value is small compared with

$$\text{Ex}[R] = \frac{n}{2},$$

which should not be surprising since we already knew from Section 17.5 that R is unlikely to stray very far from its mean.

[2] Actually, we only need to assume pairwise independence for this to be true using Theorem 19.1.9.

19.2 Markov's Theorem

The variance of a random variable gives us a rough idea of the amount by which a random variable is likely to deviate from its mean. But it does not directly give us specific bounds on the probability that the deviation exceeds a specified threshold. To obtain such specific bounds, we'll need to work a little harder.

In this section, we derive a famous result known as Markov's Theorem that gives an upper bound on the probability that a random variable exceeds a specified threshold. In the next section, we give a similar but stronger result known as Chebyshev's Theorem. The difference between these results is that Markov's Theorem depends only on the mean of the random variable, whereas Chebyshev's Theorem makes use of the mean *and* the variance. Basically, the more you know about a random variable, the better bounds you can derive on the probability that it deviates from its mean.

19.2.1 A Motivating Example

The idea behind Markov's Theorem can be explained with a simple example involving *intelligence quotients*, or IQs. This quantity was devised so that the average IQ measurement would be 100. 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 at least 300, then the average IQ 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 since no IQ over 200 has ever been recorded.

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 over 150 have certainly been recorded, although a much smaller fraction than 2/3 of the population actually has an IQ that high.

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.

Markov's Theorem characterizes the bounds that can be achieved with this kind of analysis

19.2.2 The Theorem

Theorem 19.2.1 (Markov's Theorem)**.** *If R is a nonnegative random variable, then for all $x > 0$,*

$$\Pr[R \geq x] \leq \frac{\text{Ex}[R]}{x}.$$

Proof. For any $x > 0$

$$\begin{aligned}
\text{Ex}[R] &= \sum_{y \in \text{range}(R)} y \Pr[R = y] \\
&\geq \sum_{\substack{y \geq x, \\ y \in \text{range}(R)}} y \Pr[R = y] && \text{(because } R \geq 0\text{)} \\
&\geq \sum_{\substack{y \geq x, \\ y \in \text{range}(R)}} x \Pr[R = y] \\
&= x \sum_{\substack{y \geq x, \\ y \in \text{range}(R)}} \Pr[R = y] \\
&= x \Pr[R \geq x]. && (19.9)
\end{aligned}$$

Hence,

$$\Pr[R \geq x] \leq \frac{\text{Ex}[R]}{x}. \qquad ■$$

Corollary 19.2.2. *If R is a nonnegative random variable, then for all $c \geq 1$,*

$$\Pr\big[R \geq c \cdot \text{Ex}[R]\big] \leq \frac{1}{c}. \tag{19.10}$$

Proof. Set $x = c\,\text{Ex}[R]$ in Theorem 19.2.1. ■

As an example, suppose we flip 100 fair coins and use Markov's Theorem to compute the probability of getting all heads:

$$\Pr[\text{heads} \geq 100] \leq \frac{\text{Ex}[\text{heads}]}{100} = \frac{50}{100} = \frac{1}{2}.$$

If the coins are mutually independent, then the actual probability of getting all heads is a minuscule 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!

The Chinese Appetizer Problem

Suppose that n people are seated at a circular table and that each person has an appetizer in front of them on a rotating Chinese banquet tray. Just as everyone is about to dig in, some joker spins the tray so that each person receives a random appetizer. We are interested in the number of people R that get their same appetizer as before, assuming that the n appetizers are all different.

Each person gets their original appetizer with probability $1/n$. Hence, by Linearity of Expectation,

$$\text{Ex}[R] = n \cdot \frac{1}{n} = 1.$$

What is the probability that all n people get their original appetizer back? Markov's Theorem tells us that

$$\Pr[R = n] = \Pr[R \geq n] \leq \frac{\text{Ex}[R]}{n} = \frac{1}{n}.$$

In fact, this bound is tight sine everyone gets their original appetizers back if and only if the rotating tray returns to its original configuration, which happens with probability $1/n$.

The Chinese Appetizer problem is similar to the Hat Check problem that we studied in Section 18.3.2, except that no distribution was specified in the Hat Check problem—we were told only that each person gets their correct hat back with probability $1/n$. If the hats are scrambled according to uniformly random permutations, then the probability that everyone gets the right hat back is $1/n!$, which is much less than the $1/n$ upper bound given by Markov's Theorem. So, in this case, the bound given by Markov's Theorem is not close to the actual probability.

What is the probability that at least two people get their right hats back? Markov's Theorem tells us that

$$\Pr[R \geq 2] \leq \frac{\text{Ex}[R]}{2} = \frac{1}{2}.$$

In this case, Markov's Theorem is not too far off from the right answer if the hats are distributed according to a random permutation[3] but it is not very close to the correct answer $1/n$ for the case when the hats are distributed as in the Chinese Appetizer problem.

Why *R* Must be Nonnegative

Remember that Markov's Theorem applies only to nonnegative random variables! Indeed, the theorem is false if this restriction is removed. For example, let R be -10

[3] Proving this requires some effort.

with probability 1/2 and 10 with probability 1/2. Then

$$\text{Ex}[R] = -10 \cdot \frac{1}{2} + 10 \cdot \frac{1}{2} = 0.$$

Suppose that we now tried to compute $\Pr[R \geq 5]$ using Markov's Theorem:

$$\Pr[R \geq 5] \leq \frac{\text{Ex}[R]}{5} = \frac{0}{5} = 0.$$

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 or more. For example, given any random variable, R with expectation 0 and values ≥ -10, we can conclude that $\Pr[R \geq 5] \leq 2/3$. To prove this fact, we define $T ::= R + 10$. Then T is a nonnegative random variable with expectation $\text{Ex}[R + 10] = \text{Ex}[R] + 10 = 10$, 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.

19.2.3 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. That's because if R is the IQ of a random MIT student, then

$$\Pr[R > 200] \leq \frac{\text{Ex}[R]}{200} = \frac{150}{200} = \frac{3}{4}.$$

But let's also suppose that no MIT student has an IQ less than 100 (which may be true). This means that if we let $T ::= R - 100$, then T is nonnegative and $\text{Ex}[T] = 50$, so we can apply Markov's Theorem to T and conclude:

$$\Pr[R > 200] = \Pr[T > 100] \leq \frac{\text{Ex}[T]}{100} = \frac{50}{100} = \frac{1}{2}.$$

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 on R, even when l is negative.

Theorem 19.2.3. *Let R be a random variable for which $R \geq l$ for some $l \in \mathbb{R}$. Then for all $x \geq l$,*

$$\Pr[R \geq x] \leq \frac{\text{Ex}[R] - l}{x - l}.$$

Proof. Define

$$T ::= R - l.$$

Then T is a nonnegative random variable with mean

$$\text{Ex}[T] = \text{Ex}[R - l] = \text{Ex}[R] - l.$$

Hence, Markov's Theorem implies that

$$\begin{aligned}\Pr[T \geq x - l] &\leq \frac{\text{Ex}[T]}{x - l} \\ &= \frac{\text{Ex}[R] - l}{x - l}.\end{aligned}$$

The result then follows from the fact that

$$\begin{aligned}\Pr[R \geq x] &= \Pr[R - l \geq x - l] \\ &= \Pr[T \geq x - l].\end{aligned}$$ ■

19.2.4 Deviations Below the Mean

Markov's Theorem says that a random variable is unlikely to greatly exceed the mean. Correspondingly, there is a variation of Markov's Theorem that says a random variable is unlikely to be much smaller than its mean.

Theorem 19.2.4. *Let $u \in \mathbb{R}$ and let R be a random variable such that $R \leq u$. Then for all $x < u$,*

$$\Pr[R \leq x] \leq \frac{u - \text{Ex}[R]}{u - x}.$$

Proof. The proof is similar to that of Theorem 19.2.3. Define

$$S ::= u - R.$$

Then S is a nonnegative random variable with mean

$$\text{Ex}[S] = \text{Ex}[u - R] = u - \text{Ex}[R].$$

Hence, Markov's Theorem implies that

$$\Pr[S \geq u - x] \leq \frac{\text{Ex}[S]}{u - x} = \frac{u - \text{Ex}[R]}{u - x}.$$

The result then follows from the fact that

$$\Pr[R \leq x] = \Pr[u - S \leq x] = \Pr[S \geq u - x].$$ ■

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 19.2.4 gives an upper bound. Let R be the score of a random student. Since 100 is the highest possible score, we can set $u = 100$ to meet the condition in the theorem that $R \leq u$. Applying Theorem 19.2.4, we find:

$$\Pr[R \leq 50] \leq \frac{100 - 75}{100 - 50} = \frac{1}{2}.$$

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 19.2.4 often gives weak results. In fact, based on the data given, the *entire* class could have scored *above* 50.

19.2.5 Using Markov's Theorem to Analyze Non-Random Events

In the previous example, 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 19.2.4 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 19.2.4 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 $n/2$.

19.3 Chebyshev's Theorem

As we have just seen, Markov's Theorem can be extended by applying it to functions of a random variable R such as $R - l$ and $u - R$. Even stronger results can be obtained by applying Markov's Theorem to powers of R.

Lemma 19.3.1. *For any random variable R, $\alpha \in \mathbb{R}^+$, and $x > 0$,*

$$\Pr[|R| \geq x] \leq \frac{\operatorname{Ex}[|R|^\alpha]}{x^\alpha}.$$

Proof. The event $|R| \geq x$ is the same as the event $|R|^\alpha \geq x^\alpha$. Since $|R|^\alpha$ is nonnegative, the result follows immediately from Markov's Theorem. ■

Similarly,

$$\Pr[|R - \operatorname{Ex}[R]| \geq x] \leq \frac{\operatorname{Ex}[(R - \operatorname{Ex}[R])^\alpha]}{x^\alpha}. \tag{19.11}$$

The restatement of Equation 19.11 for $\alpha = 2$ is known as *Chebyshev's Theorem*.

Theorem 19.3.2 (Chebyshev)**.** *Let R be a random variable and $x \in \mathbb{R}^+$. Then*

$$\Pr[|R - \operatorname{Ex}[R]| \geq x] \leq \frac{\operatorname{Var}[R]}{x^2}.$$

Proof. Define

$$T ::= R - \operatorname{Ex}[R].$$

Then

$$\begin{aligned}
\Pr\big[\,|R - \operatorname{Ex}[R]| \geq x\big] &= \Pr[|T| \geq x] \\
&= \Pr[T^2 \geq x^2] \\
&\leq \frac{\operatorname{Ex}[T^2]}{x^2} && \text{(by Markov's Theorem)} \\
&= \frac{\operatorname{Ex}[(R - \operatorname{Ex}[R])^2]}{x^2} \\
&= \frac{\operatorname{Var}[R]}{x^2}. && \text{(by Definition 19.1.1)}
\end{aligned}$$
■

Corollary 19.3.3. *Let R be a random variable, and let c be a positive real number.*

$$\Pr[|R - \operatorname{Ex}[R]| \geq c\sigma_R] \leq \frac{1}{c^2}.$$

Proof. Substituting $x = c\sigma_R$ in Chebyshev's Theorem gives:

$$\Pr[|R - \text{Ex}[R]| \geq c\sigma_R] \leq \frac{\text{Var}[R]}{(c\sigma_R)^2} = \frac{\sigma_R^2}{(c\sigma_R)^2} = \frac{1}{c^2}. \qquad \blacksquare$$

As an 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 $\text{Ex}[R] = 100$, $\sigma_R = 10$, and R is nonnegative. We want to compute $\Pr[R \geq 300]$.

We have already seen that Markov's Theorem 19.2.1 gives a coarse bound, namely,

$$\Pr[R \geq 300] \leq \frac{1}{3}.$$

Now we apply Corollary 19.3.3 to the same problem:

$$\Pr[R \geq 300] \leq \Pr\left[\,|R - 100| \geq 20\sigma_R\right] \leq \frac{1}{400}. \tag{19.12}$$

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 standard deviation of R, than we could get knowing only the expectation.

More generally, Corollary 19.3.3 tells us that a random variable is never likely to stray by more than a few standard deviations from its mean. For example, plugging $c = 3$ into Corollary 19.3.3, we find that the probability that a random variable strays from the mean by more than 3σ is at most $1/9$.

This fact has a nice pictorial characterization for pdf's with a "bell-curve" shape; namely, the width of the bell is $O(\sigma)$, as shown in Figure 19.1.

19.3.1 Bounds on One-Sided Errors

Corollary 19.3.3 gives bounds on the probability of deviating from the mean in *either* direction. If you only care about deviations in one direction, as was the case in the IQ example, then slightly better bounds can be obtained.

Theorem 19.3.4. *For any random variable R and any $c > 0$,*

$$\Pr[R - \text{Ex}[R] \geq c\sigma_R] \leq \frac{1}{c^2 + 1}$$

and

$$\Pr[R - \text{Ex}[R] \leq -c\sigma_R] \leq \frac{1}{c^2 + 1}.$$

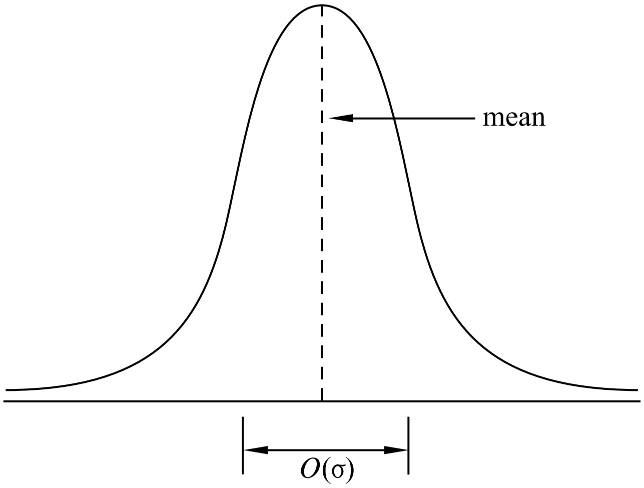


Figure 19.1 If the pdf of a random variable is "bell-shaped," then the width of the bell is $O(\sigma)$.

The proof of Theorem 19.3.4 is trickier than the proof of Chebyshev's Theorem and we will not give the details here. Nor will we prove the fact that the bounds in Theorem 19.3.4 are the best bounds that you can obtain if you know only the mean and standard deviation of the random variable R.

Returning to the IQ example, Theorem 19.3.4 tells us that

$$\Pr[R \geq 300] \leq \Pr[R - 100 \geq 20\sigma_R] \leq \frac{1}{401},$$

which is a *very slight* improvement over Equation 19.12.

As another example, suppose we give an exam. What fraction of the class can score more than 2 standard deviations from the average? If R is the score of a random student, then

$$\Pr[|R - \text{Ex}[R]| \geq 2\sigma_R] \leq \frac{1}{4}.$$

For one-sided error, the fraction that could be 2 standard deviations or more above the average is at most

$$\frac{1}{2^2 + 1} = \frac{1}{5}.$$

This results holds no matter what the test scores are, and is again a deterministic fact derived using probabilistic tools.

19.4 Bounds for Sums of Random Variables

If all you know about a random variable is its mean and variance, then Chebyshev's Theorem is the best you can do when it comes to bounding the probability that the random variable deviates from its mean. In some cases, however, we know more—for example, that the random variable has a binomial distribution—and then it is possible to prove much stronger bounds. Instead of polynomially small bounds such as $1/c^2$, we can sometimes even obtain exponentially small bounds such as $1/e^c$. As we will soon discover, this is the case whenever the random variable T is the sum of n mutually independent random variables $T_1, T_2, \ldots, T_n$ where $0 \leq T_i \leq 1$. A random variable with a binomial distribution is just one of many examples of such a T. Here is another.

19.4.1 A Motivating Example

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 in the real world.

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? 100? We'll answer that question with a new mathematical tool.

19.4.2 The Chernoff Bound

The Chernoff[4] 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 of the sum.* The Markov and Chebyshev bounds lead to the same kind of conclusion but typically provide much weaker bounds. In particular, the Markov and Chebyshev bounds are polynomial, while the Chernoff bound is exponential.

Here is the theorem. The proof will come later in Section 19.4.3.

Theorem 19.4.1 (Chernoff Bound)**.** *Let $T_1, \dots T_n$ be mutually independent random variables such that $0 \le T_i \le 1$ for all i. Let $T = T_1 + \cdots + T_n$. Then for all $c \ge 1$,*

$$\Pr[T \ge c \operatorname{Ex}[T]] \le e^{-k \operatorname{Ex}[T]} \tag{19.13}$$

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 there 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!

[4] Yes, this is the same Chernoff who figured out how to beat the state lottery. So you might want to pay attention—this guy knows a thing or two.

More Examples

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 is the probability 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 mutually independent and take on values in the range $[0, 1]$. Both conditions hold here. In fact, this example is similar to many applications of the Chernoff bound in that every T_i is *either* 0 or 1, since they’re indicators.

The goal is to bound the probability that the number of heads exceeds its expectation by 20% or more; that is, to bound $\Pr[T \geq c \operatorname{Ex}[T]]$ 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:

$$\begin{aligned} \Pr\big[T \geq 1.2 \operatorname{Ex}[T]\big] &\leq e^{-k \operatorname{Ex}[T]} \\ &= e^{-(0.0187\ldots)\cdot 500} \\ &< 0.0000834. \end{aligned}$$

So the probability of getting 20% or more extra heads on 1000 coins is less than 1 in 10,000.[5]

The bound becomes much stronger as the number of coins increases, because the expected number of heads appears in the exponent of the upper bound. For example, the probability of getting at least 20% extra heads on a million coins is at most

$$e^{-(0.0187\ldots)\cdot 500000} < e^{-9392}$$

which is pretty darn small.

Alternatively, the bound also becomes stronger for larger deviations. For example, suppose we’re interested in the odds of getting 30% or more extra heads in 1000 tosses, rather than 20%. In that case, $c = 1.3$ instead of 1.2. Consequently, the parameter k rises from 0.0187 to about 0.0410, which may seem insignificant.

[5] Since we are analyzing a binomial distribution here, we can get somewhat better bounds using the methods from Section 17.5, but it is much easier to use the Chernoff bounds, and they provide results that are nearly as good.

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 $5,000. 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 is the probability that will happen?

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[6] that the players' picks and the winning number are random, independent and uniform, then the indicators T_i are independent, as required by the Chernoff bound.

Since 2000 winners would be twice the expected number, we choose $c = 2$, compute $k = c\ln(c) - c + 1 = 0.386\ldots$, and plug these values into the Chernoff bound:

$$\begin{aligned}
\Pr[T \geq 2000] &= \Pr\big[T \geq 2\operatorname{Ex}[T]\big] \\
&\leq e^{-k\operatorname{Ex}[T]} \\
&= e^{-(0.386\ldots)\cdot 1000} \\
&< e^{-386}.
\end{aligned}$$

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:

$$\begin{aligned}
\Pr\big[T \geq 1.1\operatorname{Ex}[T]\big] &\leq e^{-k\operatorname{Ex}[T]} \\
&= e^{-(0.00484)\cdot 1000} \\
&< 0.01.
\end{aligned}$$

So the Pick-4 lottery may be exciting for the players, but the lottery operator has little doubt about the outcome!

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;

[6] As we noted in Chapter 18, human choices are often not uniform and they can be highly dependent. For example, lots of people will pick an important date. So the lottery folks should not get too much comfort from the analysis that follows, unless they assign random 4-digit numbers to each player.

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 $T = \sum_{i=1}^{n} T_i$ is the total length of tasks assigned to the server, where $n = 24{,}000$. We need an upper bound on $\Pr[T \geq 600]$; 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 task 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:

$$\begin{aligned}\text{Ex}[T] &= \frac{24{,}000 \text{ tasks} \cdot 1/4 \text{ second per task}}{m \text{ machines}} \\ &= 6000/m \text{ seconds.} \qquad (19.14)\end{aligned}$$

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:

$$\begin{aligned}\Pr\left[T \geq 600\right] &= \Pr\left[T \geq \frac{m}{10}\,\text{Ex}[T]\right] \\ &= \Pr\left[T \geq c\,\text{Ex}[T]\right] \\ &\leq e^{-(c\ln(c)-c+1)\cdot 6000/m},\end{aligned}$$

where $c = m/10$. The first equality follows from Equation 19.14.

The probability that *some* server is overloaded is at most m times the probability that the first server is overloaded by the Sum Rule in Section 14.4.2. So

$$\begin{aligned}\Pr[\text{some server is overloaded}] &\leq \sum_{i=1}^{m} \Pr[\text{server } i \text{ is overloaded}] \\ &= m\Pr[\text{the first server is overloaded}] \\ &\leq me^{-(c\ln(c)-c+1)\cdot 6000/m},\end{aligned}$$

where $c = m/10$. Some values of this upper bound are tabulated below:

$$\begin{aligned} m &= 11: \quad 0.784\ldots \\ m &= 12: \quad 0.000999\ldots \\ m &= 13: \quad 0.0000000760\ldots \end{aligned}$$

These values suggest that a system with $m = 11$ machines might suffer immediate overload, $m = 12$ machines could fail in a few days, but $m = 13$ should be fine for a century or two!

19.4.3 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![7]

Here is the theorem again, for reference:

Theorem 19.4.2 (Chernoff Bound)**.** *Let $T_1, \dots, T_n$ be mutually independent random variables such that $0 \le T_i \le 1$ for all i. Let $T = T_1 + \cdots + T_n$. Then for all $c \ge 1$,*

$$\Pr[T \ge c \operatorname{Ex}[T]] \le e^{-k \operatorname{Ex}[T]} \tag{19.13}$$

where $k = c \ln(c) - c + 1$.

Proof. For clarity, we'll go through the proof "top down"; that is, we'll use facts that are proved immediately afterward.

The key step is to exponentiate both sides of the inequality $T \ge c \operatorname{Ex}[T]$ and then apply the Markov bound:

$$\begin{aligned}
\Pr[T \ge c \operatorname{Ex}[T]] &= \Pr[c^T \ge c^{c \operatorname{Ex}[T]}] \\
&\le \frac{\operatorname{Ex}[c^T]}{c^{c \operatorname{Ex}[T]}} && \text{(by Markov)} \\
&\le \frac{e^{(c-1) \operatorname{Ex}[T]}}{c^{c \operatorname{Ex}[T]}} \\
&= e^{-(c \ln(c) - c + 1) \operatorname{Ex}[T]}.
\end{aligned}$$

In the third step, the numerator is rewritten using the inequality

$$\operatorname{Ex}[c^T] \le e^{(c-1) \operatorname{Ex}[T]}$$

which is proved below in Lemma 19.4.3. The final step is simplification, using the fact that c^c is equal to $e^{c \ln(c)}$. ■

[7] See "A Conversation with Herman Chernoff," *Statistical Science* 1996, Vol. 11, No. 4, pp 335–350.

Algebra aside, there is a brilliant idea in this proof: in this context, exponentiating somehow supercharges the Markov bound. This is not true in general! One unfortunate side-effect is that we have to bound some nasty expectations involving exponentials in order to complete the proof. This is done in the two lemmas below, where variables take on values as in Theorem 19.4.1.

Lemma 19.4.3.

$$\text{Ex}[c^T] \le e^{(c-1)\,\text{Ex}[T]}.$$

Proof.

$$\begin{aligned}
\text{Ex}[c^T] &= \text{Ex}[c^{T_1+\cdots+T_n}] \\
&= \text{Ex}[c^{T_1} \cdots c^{T_n}] \\
&= \text{Ex}[c^{T_1}] \cdots \text{Ex}[c^{T_n}] \\
&\le e^{(c-1)\,\text{Ex}[T_1]} \cdots e^{(c-1)\,\text{Ex}[T_n]} \\
&= e^{(c-1)(\text{Ex}[T_1]+\cdots+\text{Ex}[T_n])} \\
&= e^{(c-1)\,\text{Ex}[T_1+\cdots+T_n]} \\
&= e^{(c-1)\,\text{Ex}[T]}.
\end{aligned}$$

The first step uses the definition of T, and the second is just algebra. 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 inequality

$$\text{Ex}[c^{T_i}] \le e^{(c-1)\,\text{Ex}[T_i]},$$

which is proved in Lemma 19.4.4. The last steps are simplifications using algebra and linearity of expectation. ■

Lemma 19.4.4.

$$\text{Ex}[c^{T_i}] \le e^{(c-1)\,\text{Ex}[T_i]}$$

Proof. All summations below range over values v taken by the random variable T_i, which are all required to be in the interval [0, 1].

$$\begin{aligned}
\operatorname{Ex}[c^{T_i}] &= \sum_v c^v \Pr[T_i = v] \\
&\le \sum_v (1 + (c-1)v) \Pr[T_i = v] \\
&= \sum_v \Pr[T_i = v] + (c-1)v \Pr[T_i = v] \\
&= \sum_v \Pr[T_i = v] + \sum (c-1)v \Pr[T_i = v] \\
&= 1 + (c-1) \sum_v v \Pr[T_i = v] \\
&= 1 + (c-1) \operatorname{Ex}[T_i] \\
&\le e^{(c-1)\operatorname{Ex}[T_i]}.
\end{aligned}$$

The first step uses the definition of expectation. The second step relies on the inequality $c^v \le 1 + (c-1)v$, which holds for all v in [0, 1] and $c \ge 1$. 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 $\operatorname{Ex}[T_i]$. The final step uses the standard inequality $1 + z \le e^z$, which holds for all $z > 0$. ■

19.5 Mutually Independent Events

Suppose that we have a collection of mutually independent events $A_1, A_2, \ldots, A_n$, and we want to know how many of the events are likely to occur.

Let T_i be the indicator random variable for A_i and define

$$p_i = \Pr[T_i = 1] = \Pr\left[A_i\right]$$

for $1 \le i \le n$. Define

$$T = T_1 + T_2 + \cdots + T_n$$

to be the number of events that occur.

We know from Linearity of Expectation that

$$\begin{aligned}\text{Ex}[T] &= \text{Ex}[T_1] + \text{Ex}[T_2] + \cdots + \text{Ex}[T_n] \\ &= \sum_{i=1}^{n} p_i.\end{aligned}$$

This is true even if the events are *not* independent.

By Theorem 19.1.9, we also know that

$$\begin{aligned}\text{Var}[T] &= \text{Var}[T_1] + \text{Var}[T_2] + \cdots + \text{Var}[T_n] \\ &= \sum_{i=1}^{n} p_i(1 - p_i),\end{aligned}$$

and thus that

$$\sigma_T = \sqrt{\sum_{i=1}^{n} p_i(1 - p_i)}.$$

This is true even if the events are only pairwise independent.

Markov's Theorem tells us that for any $c > 1$,

$$\Pr[T \geq c\,\text{Ex}[T]] \leq \frac{1}{c}.$$

Chebyshev's Theorem gives us the stronger result that

$$\Pr[|T - \text{Ex}[T]| \geq c\sigma_T] \leq \frac{1}{c^2}.$$

The Chernoff Bound gives us an even stronger result; namely, that for any $c > 0$,

$$\Pr[T - \text{Ex}[T] \geq c\,\text{Ex}[T]] \leq e^{-(c\ln(c) - c + 1)\,\text{Ex}[T]}.$$

In this case, the probability of exceeding the mean by $c\,\text{Ex}[T]$ decreases as an exponentially small function of the deviation.

By considering the random variable $n - T$, we can also use the Chernoff Bound to prove that the probability that T is much lower than $\text{Ex}[T]$ is also exponentially small.

19.5.1 Murphy's Law

Suppose we want to know the probability that at least 1 event occurs. If $\text{Ex}[T] < 1$, then Markov's Theorem tells us that

$$\Pr[T \geq 1] \leq \text{Ex}[T].$$

On the other hand, if $\text{Ex}[T] \geq 1$, then we can obtain a lower bound on $\Pr[T \geq 1]$ using a result that we call Murphy's Law[8].

Theorem 19.5.1 (Murphy's Law)**.** *Let $A_1, A_2, \ldots, A_n$ be mutually independent events. Let T_i be the indicator random variable for A_i and define*

$$T ::= T_1 + T + 2 + \cdots + T_n$$

to be the number of events that occur. Then

$$\Pr[T = 0] \leq e^{-\text{Ex}[T]}.$$

Proof.

$$\begin{aligned}
\Pr[T = 0] &= \Pr[\overline{A}_1 \wedge \overline{A}_2 \wedge \cdots \wedge \overline{A}_n] \\
&= \prod_{i=1}^{n} \Pr[\overline{A}_i] && \text{(by independence of } A_i\text{)} \\
&= \prod_{i=1}^{n} (1 - \Pr[A_i]) \\
&\leq \prod_{i=1}^{n} e^{-\Pr[A_i]} && \text{(since } \forall x. 1 - x \leq e^{-x}\text{)} \\
&= e^{-\sum_{i=1}^{n} \Pr[A_i]} \\
&= e^{-\sum_{i=1}^{n} \text{Ex}[T_i]} && \text{(since } T_i \text{ is an indicator for } A_i\text{)} \\
&= e^{-\text{Ex}[T]} && \text{(Linearity of Expectation)} \quad \blacksquare
\end{aligned}$$

For example, given any set of mutually independent events, if you expect 10 of them to happen, then at least one of them will happen with probability at least $1 - e^{-10}$. The probability that none of them happen is at most $e^{-10} < 1/22000$.

So if there are a lot of independent things that can go wrong and their probabilities sum to a number much greater than 1, then Theorem 19.5.1 proves that some of them surely will go wrong.

[8] This is in reference and deference to the famous saying that "If something can go wrong, it will go wrong."

This result can help to explain “coincidences,” “miracles,” and crazy events that seem to have been very unlikely to happen. Such events do happen, in part, because there are so many possible unlikely events that the sum of their probabilities is greater than one. For example, someone *does* win the lottery.

In fact, if there are 100,000 random tickets in Pick-4, Theorem 19.5.1 says that the probability that there is no winner is less than $e^{-10} < 1/22000$. More generally, there are literally millions of one-in-a-million possible events and so some of them will surely occur.

19.5.2 Another Magic Trick

Theorem 19.5.1 is surprisingly powerful. In fact, it is so powerful that it can enable us to read your mind. Here's how.

You choose a secret number n from 1 to 9. Then we randomly shuffle an ordinary deck of 52 cards and display the cards one at a time. You watch as we reveal the cards and when we reveal the nth card, that card becomes your *secret card.* If the card is an Ace, a 10, or a face card, then you assign that card a *value* of 1. Otherwise, you assign that card a value that is its number. For example, the $J\heartsuit$ gets assigned a value $v_1 = 1$ and the $4\diamondsuit$ gets assigned a value $v_1 = 4$. You do all of this in your mind so that we can't tell when the nth card shows up.

We keep revealing the cards, and when the $(n + v_1)$th card shows up, that card becomes your *new* secret card. You compute its value v_2 using the same scheme as for v_1. For example, if your new secret card is the $10\clubsuit$, then $v_2 = 1$. The $(n + v_1 + v_2)$th card will then become your next secret card, and so forth.

We proceed in this fashion until all 52 cards have been revealed, whereupon we read your mind by predicting your last secret card! How is this possible?

For the purposes of illustration, suppose that your secret number was $n = 3$ and the deck consisted of the 11 cards:

$$3\diamondsuit \quad 5\spadesuit \quad 2\diamondsuit \quad 3\clubsuit \quad 10\clubsuit \quad Q\diamondsuit \quad 3\heartsuit \quad 7\spadesuit \quad 6\clubsuit \quad 4\diamondsuit \quad 2\heartsuit.$$

Then your secret cards would be

$$2\diamondsuit,\ 10\clubsuit,\ Q\diamondsuit,\ 3\heartsuit,\ 4\diamondsuit$$

since $v_1 = 2$, $v_2 = 1$, $v_3 = 1$, $v_4 = 3$, and $v_5 = 4$. In this example, your last secret card is the $4\diamondsuit$.

To make the trick work, we follow the same rules as you, except that we start with $n = 1$. With the 11-card deck shown above, our secret cards would be

$$3\diamondsuit,\ 3\clubsuit,\ 3\heartsuit,\ 4\diamondsuit.$$

We have the same last secret card as you do! That is *not* a coincidence. In fact, this is how we predict your last card—we just guess that it is the same as our last card. And, we will be right with probability greater than 90%.

To see why the trick is likely to work, you need to notice that if we ever share a secret card, then we will surely have the same *last* secret card. That's because we will perform exactly the same steps as the cards are revealed.

Each time we get a new secret card, there is always a chance that it was one of your secret cards. For any given step, the chance of a match is small but we get a lot of chances. In fact, the number of chances will typically outweigh the inverse of the probability of a match on any given step and so, at least informally, Murphy's Law suggests that we are likely to eventually get a match, whereupon we can read your mind.

The details of the proof are complicated and we will not present them here. One of the main complications is that when you are revealing cards from a deck without replacement, the probability of getting a match on a given step is conditional based on the cards that have already been revealed.

19.5.3 The Subprime Mortgage Disaster

Throughout the last few chapters, we have seen many examples where powerful conclusions can be drawn about a collection of events if the events are independent. Of course, such conclusions are totally invalid if the events have dependencies. Unforeseen dependencies can result in disaster in practice. For example, misguided assumptions about the independence of loans (combined with a large amount of greed) triggered the global financial meltdown in 2008–2009.

In what follows, we'll explain some of what went wrong. You may notice that we have changed the names of the key participants. That is not to protect the innocent, since innocents are few and far between in this sordid tale. Rather, we changed the names to protect ourselves.[9] In fact, just to be on the safe side, we'll forget about what really happened here on Earth and instead tell you a fairy tale that took place in a land far, far away.

The central players in our story are the major Wall Street firms, of which Golden Scoundrels (commonly referred to as "Golden") is the biggest and most aggressive. Firms such as Golden ostensibly exist to make markets; they purport to create an open and orderly market in which buyers and sellers can be brought together and through which capitalism can flourish. It all sounds good, but the fees that can be had from facilitating transactions in a truly open and orderly market are often just not enough to satisfy the ever-increasing need to make more. So the employees at

[9] For a much more detailed accounting of these events (and one that does name names), you may enjoy reading *The Big Short* by Michael Lewis.

such firms are always trying to figure out a way to create new opportunities to make even more money.

One day, they came up with a whopper. Suppose they bought a collection of 1000 (say) subprime mortgage loans from all around the country and packaged them up into a single entity called a *bond*. A *mortgage loan* is a loan to a homeowner using the house as collateral; if the homeowner stops paying on the loan (in which case the loan is said to be in *default*), then the owner of the loan takes ownership of the house. A mortgage loan is classified as *subprime* if the homeowner does not have a very good credit history. Subprime loans are considered to be more risky than *prime* loans since they are more likely to default. Defaults are bad for everyone; the homeowner loses the home and the loan owner gets stuck trying to sell the house, which can take years and often results in very high losses.

Of course, a bond consisting of 1000 subprime loans doesn't sound very appealing to investors, so to dress it up, Golden sells the bond in *tranches*. The idea behind the tranches is to provide a way to assign losses from defaults. In a typical scenario, there would be 10 tranches and they are prioritized from 1 to 10. The defaults are assessed against the lowest tranches first. For example, suppose that there were 150 defaults in the collection of 1000 loans (an impossibly high number of defaults according to Golden). Then the lowest tranche would absorb the first 100 defaults (effectively wiping them out since all 100 of "their" loans would be in default) and the second-lowest tranche would be assigned the next 50 defaults, (wiping out half of their investment). The remaining 8 tranches would be doing great—none of "their" loans would be in default.

Because they are taking on more risk, the lower tranches would get more of the interest payments. The top tranche would get the lowest rate of return and would also be the safest. The lowest tranche would get the most interest, but also be the most exposed.

But how much should you pay for a tranche? Suppose the probability that any given loan defaults in a year is 1%. In other words, suppose you expect 10 of the 1000 loans to default in each year. If the defaults are independent, then we can use the Chernoff bound to conclude that the chances of more than 100 defaults (10%) in the 1000-loan collection is exceedingly tiny. This means that every tranche but the lowest is essentially risk-free. That is excellent news for Golden since they can buy 1000 cheap[10] subprime loans and then sell the top 9 tranches at premium rates, thereby making a large and instant profit on 900 of the 1000 loans. It is like turning a bunch of junk into a bunch of gold with a little junk left over.

There remains the problem of the lowest tranche, which is expected to have 10 defaults in a pool of 100 loans for a default rate of 10%. This isn't so good

[10] They are *subprime* loans after all.

so the first thing to do is to give the tranche a better sounding name than "lowest tranche." "Mezzanine" tranche sounds much less ominous and so that is what they used.

By the Chernoff bound, the default rate in the Mezzanine tranche is very unlikely to be much greater than 10%, and so the risk of owning this tranche can be addressed in part by increasing the interest payments for the tranche by 10%. But Golden had an even better idea (whopper number two)—rather than pay the extra 10%, why not collect together a bunch of mezzanine tranches from a bunch of bonds and then package them together into a "super bond" and then create tranches in the super-bond? The technical name for such a super bond is a *collateralized debt obligation* or CDO. This way, 90% of the mezzanine tranches instantly became essentially "risk-free," or so Golden claimed as they were marketing them.

The only problem now is getting the pension funds and other big investors to buy the CDOs at the same price as if they were AAA-rated "risk-free" bonds. This was a little tricky because 1) it was virtually impossible for the buyer to figure out exactly what loans they were effectively buying since they were buying a tranche of a collection of tranches, and 2) if you could ever figure out what it was, you would discover that it was the junk of the junk when it comes to loans.

The solution was to enlist the help of the big bond-rating agencies: Substandard and Prevaricators (S&P) and Mopey's. If Golden could get AAA ratings[11] on their tranches, then the pension funds and other big investors would buy them at premium rates.

It turned out to be easier than you might think (or hope) to convince S&P and Mopey's to give high ratings to the CDO tranches. After all, the ratings agencies are trying to make money too and they make money by rating bonds. And Golden was only going to pay them if their bonds and CDOs got good ratings. And, since defaults were assumed to be essentially independent, there was a good argument as to why all but the mezzanine tranche of a bond or CDO would be essentially risk-free.[12]

So the stage is set for Golden to make a bundle of money. Cheap junk loans come in the back door and exit as expensive AAA-rated bonds and CDOs out the front door. The remaining challenge is to ramp up the new money-making machine. That

[11] AAA ratings are the best you can get and are supposed to imply that there is virtually no chance of default.

[12] The logic gets a little fuzzy when you keep slicing and dicing the tranches—after a few iterations, you should be able to conclude that the mezzanine tranche of a CDO is sure to have 100% defaults, but it required effort to see what was going on under the covers and effort costs money, and so the ratings agencies considered the risk of the mezzanine tranche of one CDO to be the same as the mezzanine tranche of any other, even though they could have wildly different probabilities of sustaining large numbers of defaults.

means creating more (preferably, many, many more) junk loans to fuel the machine.

This is where Joe enters the scene. Joe is a migrant laborer earning $15,000 per year. Joe's credit history is not great (since he has never had a loan or credit card) but it is also not bad (since he has never missed a payment on a credit card and never defaulted on a loan). In short, Joe is a perfect candidate for a subprime mortgage loan on a $750,000 home.

When Loans-Я-Us approaches Joe for a home loan,[13] Joe dutifully explains that while he would love to own a $750,000 home, he doesn't have enough money to pay for food, let alone the interest payments on the mortgage. "No problem!" replies Loans-Я-Us. It is Joe's lucky day. The interest rates are super-low for the first 2 years and Joe can take out a second loan to cover them during that period. "What happens after 2 years?" Joe wants to know. "No problem!" replies Loans-Я-Us. Joe can refinance—his home will surely be worth more in 2 years. Indeed, Joe can even make money while he enjoys the comforts of his new home. If all goes well, he can even ease off on the laborer work, and maybe even by a second home. Joe is sold. In fact, millions of Joes are sold and, before long, the subprime loan business is booming.

It turns out that there were a few folks out there who really did their math homework when they were in college. They were running hedge funds and, as the money-making machine was cranking away, they realized that a disaster was looming. They knew that loan defaults are not independent—in fact, they are very dependent. Once home values stop rising, or a recession hits, or it comes time for Joe to refinance, defaults will occur at much higher rates than projected and the CDOs and many tranches of the underlying bonds will become worthless. And there is so much money invested in these bonds and CDOs that the economy could be ruined.

Unfortunately, the folks who figured out what was going to happen didn't alert anyone. They didn't go to the newspapers. They didn't call the See no Evil Commission. They didn't even call 911. Instead, they worked with Golden to find a new way to make even more money—betting against the CDO market.

If you think a stock is going to decline, you can profit from the decline by borrowing the stock and selling it. After the stock declines in value, you buy it back and return it to the person that lent it to you. Your profit is the decline in price. This process is called *shorting* the stock.

So the hedge funds wanted to short the CDOs. Unfortunately, there was no established way to borrow a tranche of a CDO. Always looking for a new way to make money, the investment houses came up with an even bigger whopper than the

[13] Yes, we know it is supposed to go the other way around—Joe is supposed to approach the loan company—but these are extraordinary times.

CDO—they invented the *credit default swap*.

The idea behind the credit default swap is to provide a kind of insurance against the event that a bond or CDO suffers a certain number of defaults. Since the hedge funds believe that the CDOs were going to have lots of defaults, they want to buy the insurance. The trick is to find someone dumb enough to sell the insurance. That’s where the world’s largest insurance company, Awful Insurance Group (AIG), enters the fray. AIG sells insurance on just about anything and they, too, are looking for new ways to make money, so why not sell insurance on CDO defaults?

Golden has a new business! They buy the CDO insurance from AIG for an astonishingly low price (about $2 annually for every $1000 of CDO value) and sell it to the hedge funds for a much higher price (about $20 annually for every $1000 of CDO value). If a CDO sustains defaults, then AIG needs to pay the value of the CDO ($1000 in this hypothetical example) to the hedge funds who own the insurance. Until that time, the hedge funds are paying the annual fee for the insurance, 90% of which is pocketed by Golden. This is a great business; Golden pockets 90% of the money and AIG takes all the risk. The only risk that Golden has is if AIG goes down, but AIG is “too big to fail. . . . ”

Golden’s new credit default swap business is even better than the CDO business. The only trouble now is that there are only so many Joes out there who can take out subprime loans. This means that there is a hard limit on how many billions Golden can make. This challenge led to whopper number four.

If the hedge funds want to buy insurance and AIG wants to sell it, who really cares if there is only one insurance policy per loan or CDO? Indeed, why not just sell lots of credit default swaps on the same set of junk CDOs? This way, the profits could be unlimited! And so it went. “Synthetic” CDOs were created and soon the “insurance” quickly turned into a very high-stakes (and very stupid, at least for AIG) bet. The odds were weighted heavily in favor of the folks who did their math homework (the hedge funds); the hedge funds had figured out that the failure of the CDOs was a virtual certainty, whereas AIG believed that failure was virtually impossible.

Of course, we all know how the story ends. The holders of the CDOs and subprime debt and the sellers of insurance got wiped out, losing hundreds of billions of dollars. Since many of these folks were deemed by the Government as “too big to fail,” they were bailed out using nearly a trillion dollars of taxpayer money. The executives who presided over the disaster were given huge bonuses because, well, that’s how it works for executives in the land far, far away. The story also ends well for the hedge funds that bought the insurance—they made many, many billions of dollars.

So everyone involved in the disaster ends up very rich. Everyone except Joe, of course. Joe got kicked out of his home and lost his job in the recession.

Too bad for Joe that it isn’t just a fairy tale.

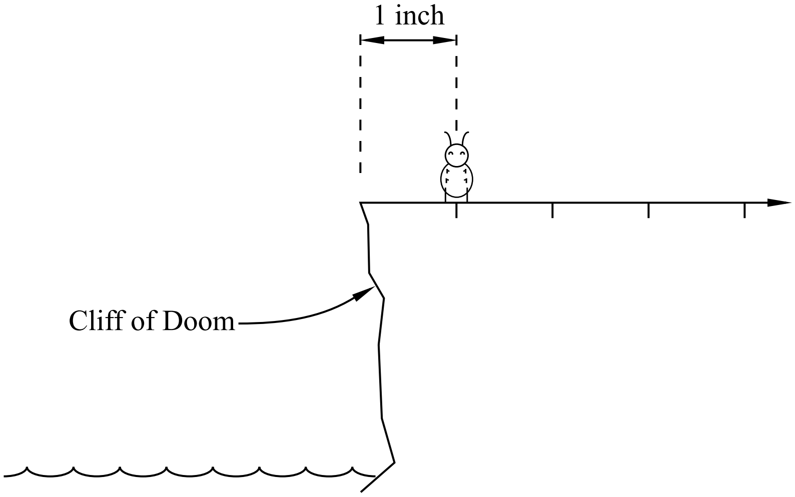
20 Random Walks

Random Walks are used to model situations in which an object moves in a sequence of steps in randomly chosen directions. Many phenomena can be modeled as a random walk and we will see several examples in this chapter. Among other things, we'll see why it is rare that you leave the casino with more money than you entered with and we'll see how the Google search engine uses random walks through the graph of the world-wide web links to determine the relative importance of websites.

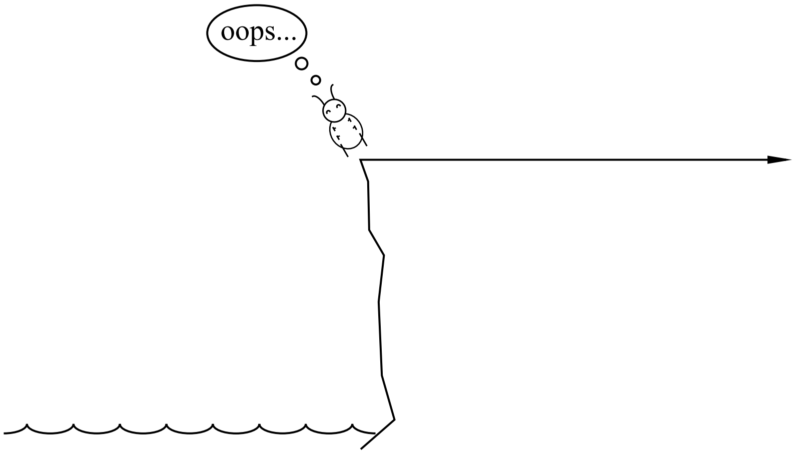
20.1 Unbiased Random Walks

20.1.1 A Bug's Life

There is a small flea named Stencil. To his right, there is an endless flat plateau. One inch to his left is the Cliff of Doom, which drops to a raging sea filled with flea-eating monsters.



Each second, Stencil hops 1 inch to the right or 1 inch to the left with equal probability, independent of the direction of all previous hops. If he ever lands on the very edge of the cliff, then he teeters over and falls into the sea. So, for example, if Stencil's first hop is to the left, he's fishbait. On the other hand, if his first few hops are to the right, then he may bounce around happily on the plateau for quite



some time.

Our job is to analyze the life of Stencil. Does he have any chance of avoiding a fatal plunge? If not, how long will he hop around before he takes the plunge?

Stencil's movement is an example of a *random walk*. A typical *one-dimensional* random walk involves some value that randomly wavers up and down over time. The walk is said to be *unbiased* if the value is equally likely to move up or down. If the walk ends when a certain value is reached, then that value is called a *boundary condition* or *absorbing barrier*. For example, the Cliff of Doom is a boundary condition in the example above.

Many natural phenomena are nicely modeled by random walks. However, for some reason, they are traditionally discussed in the context of some social vice. For example, the value is often regarded as the position of a drunkard who randomly staggers left, staggers right, or just wobbles in place during each time step. Or the value is the wealth of a gambler who is continually winning and losing bets. So discussing random walks in terms of fleas is actually sort of elevating the discourse.

20.1.2 A Simpler Problem

Let's begin with a simpler problem. Suppose that Stencil is on a small island; now, not only is the Cliff of Doom 1 inch to his left, but also there is another boundary condition, the Pit of Disaster, 2 inches to his right! For example, see Figure 20.1

In the figure, we've worked out a tree diagram for Stencil's possible fates. In

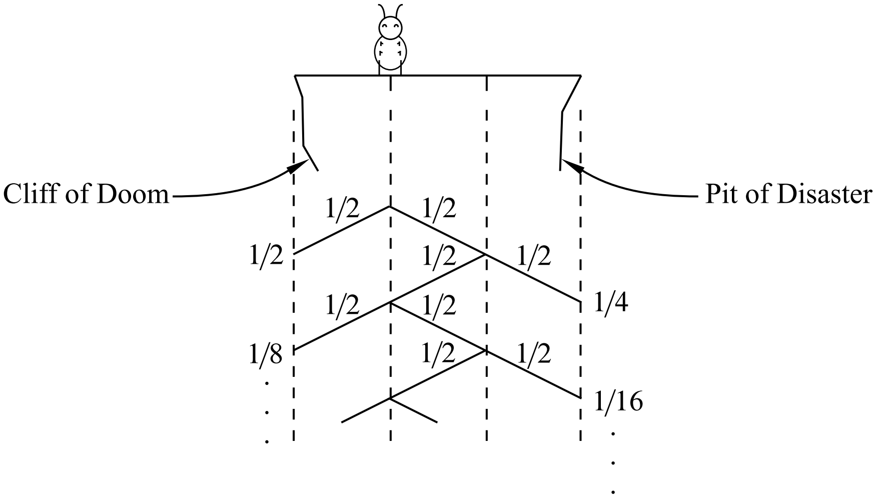


Figure 20.1 An unbiased, one-dimensional random walk with absorbing barriers at positions 0 and 3. The walk begins at position 1. The tree diagram shows the probabilities of hitting each barrier.

particular, he falls off the Cliff of Doom on the left side with probability:

$$\begin{aligned}\frac{1}{2} + \frac{1}{8} + \frac{1}{32} + \dots &= \frac{1}{2}\left(1 + \frac{1}{4} + \frac{1}{16} + \dots\right)\\ &= \frac{1}{2} \cdot \frac{1}{1 - 1/4}\\ &= \frac{2}{3}.\end{aligned}$$

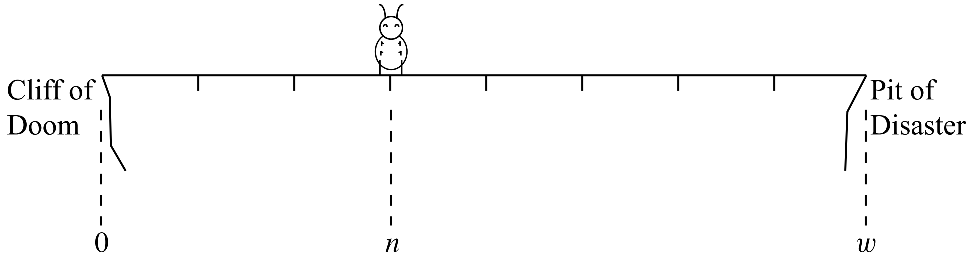
Similarly, he falls into the Pit of Disaster on the right side with probability:

$$\frac{1}{4} + \frac{1}{16} + \frac{1}{64} + \dots = \frac{1}{3}.$$

There is a remaining possibility: Stencil *could* hop back and forth in the middle of the island forever. However, we’ve already identified two disjoint events with probabilities $2/3$ and $1/3$, so this happy alternative must have probability 0.

20.1.3 A Big Island

Putting Stencil on such a tiny island was sort of cruel. Sure, he’s probably carrying bubonic plague, but there’s no reason to pick on the little fella. So suppose that we instead place him n inches from the left side of an island w inches across: In



other words, Stencil starts at position n and his random walk ends if he ever reaches positions 0 or w.

Now he has three possible fates: he could fall off the Cliff of Doom, fall into the Pit of Disaster, or hop around on the island forever. We could compute the probabilities of these three events with a horrific summation, but fortunately there's a far easier method: we can use a linear recurrence.

Let R_n be the probability that Stencil falls to the right into the Pit of Disaster, given that he starts at position n. In a couple special cases, the value of R_n is easy to determine. If he starts at position w, he falls into the Pit of Disaster immediately, so $R_w = 1$. On the other hand, if he starts at position 0, then he falls from the Cliff of Doom immediately, so $R_0 = 0$.

Now suppose that our frolicking friend starts somewhere in the middle of the island; that is, $0 < n < w$. Then we can break the analysis of his fate into two cases based on the direction of his first hop:

- If his first hop is to the left, then he lands at position $n - 1$ and eventually falls into the Pit of Disaster with probability R_{n-1}.
- On the other hand, if his first hop is to the right, then he lands at position $n+1$ and eventually falls into the Pit of Disaster with probability R_{n+1}.

Therefore, by the Total Probability Theorem, we have:

$$R_n = \frac{1}{2}R_{n-1} + \frac{1}{2}R_{n+1}.$$

Solving the Recurrence

Let's assemble all our observations about R_n, the probability that Stencil falls into the Pit of Disaster if he starts at position n:

$$\begin{aligned} R_0 &= 1 \\ R_w &= 0 \\ R_n &= \frac{1}{2}R_{n-1} + \frac{1}{2}R_{n+1} \qquad (0 < n < w). \end{aligned}$$

This is just a linear recurrence—and we know how to solve those! Uh, right? Remember Chapter 10 or Chapter 12?

There is one unusual complication: in a normal recurrence, R_n is written a function of preceding terms. In this recurrence equation, however, R_n is a function of both a preceding term (R_{n-1}) and a *following* term (R_{n+1}). This is no big deal, however, since we can just rearrange the terms in the recurrence equation:

$$R_{n+1} = 2R_n - R_{n-1}.$$

Now we're back on familiar territory.

Let's solve the recurrence. The characteristic equation is:

$$x^2 - 2x + 1 = 0.$$

This equation has a double root at $x = 1$. There is no inhomogeneous part, so the general solution has the form:

$$R_n = a \cdot 1^n + b \cdot n1^n = a + bn.$$

Substituting in the boundary conditions $R_0 = 0$ and $R_w = 1$ gives two linear equations:

$$\begin{aligned} 0 &= a, \\ 1 &= a + bw. \end{aligned}$$

The solution to this system is $a = 0$, $b = 1/w$. Therefore, the solution to the recurrence is:

$$R_n = n/w.$$

20.1.4 Death Is Certain

Our analysis shows that if we place Stencil n inches from the left side of an island w inches across, then he falls off the right side with probability n/w. For example, if Stencil is $n = 4$ inches from the left side of an island $w = 12$ inches across, then he falls off the right side with probability $n/w = 4/12 = 1/3$.

We can compute the probability that he falls off the *left* side by exploiting the symmetry of the problem: the probability that he falls off the *left* side starting at position n is the same as the probability that he falls of the *right* side starting at position $w - n$, which is $(w - n)/n$.

This is bad news. The probability that Stencil eventually falls off one side or the other is:

$$\frac{n}{w} + \frac{w-n}{w} = 1.$$

There's no hope! The probability that Stencil hops around on the island forever is zero.

And there's even worse news. Let's go back to the original problem where Stencil is 1 inch from the left edge of an *infinite* plateau. In this case, the probability that he eventually falls into the sea is:

$$\lim_{w \to \infty} \frac{w-1}{w} = 1.$$

So even if there were no Pit of Disaster, Stencil still falls off the Cliff of Doom with probability 1. And since

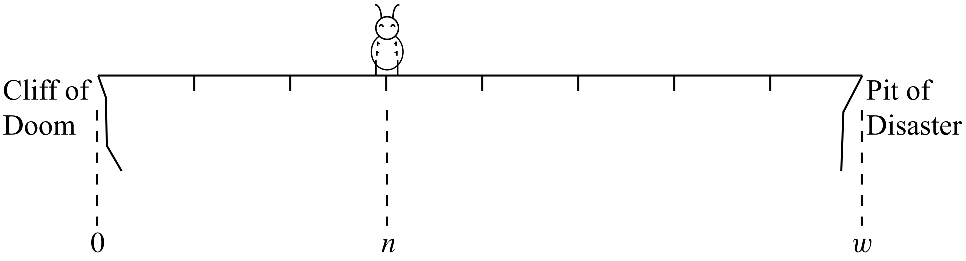
$$\lim_{w \to \infty} \frac{w-n}{w} = 1$$

for any finite n, this is true no matter where Stencil starts. Our little friend is doomed!

Hey, you know how in the movies they often make it look like the hero dies, but then he comes back in the end and everything turns out okay? Well, we're not sayin' anything, just pointing that out.

20.1.5 Life Expectancy

On the bright side, Stencil may get to hop around for a while before he goes over an edge. Let's use the same setup as before, where he starts out n inches from the left side of an island w inches across: What is the expected number of hops he takes



before falling off an edge?

Let X_n be Stencil's expected lifespan, measured in hops. If he starts at either edge of the island, then he dies immediately:

$$X_0 = 0,$$
$$X_w = 0.$$

If he starts somewhere in the middle of the island ($0 < n < w$), then we can again break down the analysis into two cases based on his first hop:

- If his first hop is to the left, then he lands at position $n - 1$ and can expect to live for another X_{n-1} steps.
- If his first hop is to the right, then he lands at position $n + 1$ and can expect to live for another X_{n+1} steps.

Thus, by the Law of Total Expectation and Linearity of Expectation, Stencil’s expected lifespan is:

$$X_n = 1 + \frac{1}{2}X_{n-1} + \frac{1}{2}X_{n+1}.$$

The leading 1 accounts for his first hop.

Solving the Recurrence

Once again, Stencil’s fate hinges on a recurrence equation:

$$\begin{aligned} X_0 &= 0 \\ X_w &= 0 \\ X_n &= 1 + \tfrac{1}{2}X_{n-1} + \tfrac{1}{2}X_{n+1} \qquad (0 < n < w). \end{aligned}$$

We can rewrite the last line as:

$$X_{n+1} = 2X_n - X_{n-1} - 2. \tag{20.1}$$

As before, the characteristic equation is:

$$x^2 - 2x + 1 = 0.$$

There is a double-root at 1, so the homogeneous solution has the form:

$$X_n = a + bn.$$

But this time, there’s an inhomogeneous term, so we also need to find a particular solution. Since this term is a constant, we should try a particular solution of the form $X_n = c$ and then try $X_n = c + dn$ and then $X_n = c + dn + en^2$ and so forth. As it turns out, the first two possibilities don’t work, but the third does. Substituting $X_n = c + dn + en^2$ into Equation 20.1 gives

$$c + d(n+1) + e(n+1)^2 = 2(c + dn + en^2) - \bigl(c + d(n-1) + e(n-1)^2\bigr) - 2,$$

which simplifies to $e = -1$. Since all the c and d terms cancel, $X_n = c + dn - n^2$ is a particular solution for all c and d. For simplicity, let’s take $c = d = 0$. Thus, our particular solution is $X_n = -n^2$.

Adding the homogeneous and particular solutions gives the general form of the solution:

$$X_n = a + bn - n^2.$$

Substituting in the boundary conditions $X_0 = 0$ and $X_w = 0$ gives two linear equations:

$$\begin{aligned} 0 &= a, \\ 0 &= a + bw - w^2. \end{aligned}$$

The solution to this system is $a = 0$ and $b = w$. Therefore, the solution to the recurrence equation is:

$$X_n = wn - n^2 = n(w - n).$$

Interpreting the Solution

Stencil's expected lifespan is $X_n = n(w - n)$, which is the *product* of the distances to the two edges. Thus, for example, if he's 4 inches from the left edge and 8 inches from the right cliff, then his expected lifespan is $4 \cdot 8 = 32$.

Let's return to the original problem where Stencil has the Cliff of Doom 1 inch to his left and an infinite plateau to this right. (Also, cue the "hero returns" theme music.) In this case, his expected lifespan is:

$$\lim_{w \to \infty} 1(w - 1) = \infty$$

Yes, Stencil is certain to eventually fall off the Cliff of Doom—but his expected lifespan is infinite! This sounds almost like a contradiction, but both answers are correct!

Here's an informal explanation. It turns out that the probability p_k that Stencil falls from the Cliff of Doom on the kth step is $\Theta(1/k^{3/2})$. You can verify by the integration bound that $\sum_{i=1}^{\infty} 1/k^{3/2}$ converges.

On the other hand, the expected time until Stencil falls over the edge is

$$\begin{aligned} \sum_{i=1}^{\infty} k p_k &\geq c \sum_{k=1}^{\infty} \frac{k}{k^{3/2}} \\ &= c \sum_{k=1}^{\infty} \frac{1}{\sqrt{k}} \\ &= \infty, \end{aligned}$$

where c is a constant that comes from the Θ notation. So our answers are compatible.

20.1.6 Application to Fair Gambling Games

We took the high road for a while, but let's now discuss random walks in a more conventional setting—gambling.

A gambler goes to Las Vegas with $\$n$ in her pocket. Her plan is to make only \$1 bets and somehow she has found a casino that will offer her truly even odds[1]; namely, she will win or lose \$1 on each bet with probability $1/2$. She'll play until she is broke or she has won $\$m$. In the latter case, she will go home with

$$w = n + m$$

dollars. What's the probability that she goes home a winner?

This is identical to the flea problem that we just analyzed. Going broke is analogous to falling off the Cliff of Doom. Going home a winner is analogous to falling into the Pit of Disaster, just a lot more fun.

Our analysis of Stencil's life tells us everything we want to know about the gambler's prospects:

- The gambler goes home broke with probability
$$\frac{n}{w} = \frac{m}{n+m},$$
- the gambler goes home a winner with probability
$$\frac{w-n}{w} = \frac{n}{n+m},$$
- the gambler goes home with probability
$$\frac{n}{n+m} + \frac{m}{n+m} = 1,$$
- and the number of bets before the gambler goes home is expected to be
$$n(w-n) = nm.$$

If the gambler gets greedy and keeps playing until she goes broke, then

- the gambler eventually goes broke with probability 1, and
- the number of bets before the gambler goes broke is expected to be infinite.

The bottom line here is clear: when gambling, quit while you are ahead—if you play until you go broke, you will certainly go broke.

And that's the good news! Matters get much worse for the more typical scenario where the odds are against you. Let's see why.

[1] Don't worry, we'll get to the more realistic scenario when she is more likely to lose than win in a moment, but let's just fantasize about the fair scenario for a bit.

20.2 Gambler’s Ruin

So far, we have considered *unbiased* random walks, where the probability of moving up or down (or left or right) is $1/2$. Unfortunately, things are never quite this simple (or fair) in real casinos.

For example, suppose the gambler goes to Las Vegas and makes \$1 bets on red or black in roulette. In this case, she will win \$1 with probability

$$\frac{18}{38} \approx 0.473$$

and she will lose \$1 with probability

$$\frac{20}{38} \approx 0.527.$$

That’s because the casinos add those bothersome green 0 and 00 to give the house a slight advantage.

At first glance (or after a few drinks), $18/38$ seems awfully close to $1/2$ and so our intuition tells us that the game is “almost fair.” So we might expect the analysis we just did for the fair game to be “almost right” for the real game. For example, if the gambler starts with \$100 and quits when she gets ahead by \$100 in the fair game, then she goes home a winner with probability

$$\frac{100}{200} = .5.$$

And, if she wants to improve her chances of going home a winner, she could bring more money. If she brings \$1000 and quits when she gets ahead by \$100 in the fair game, then she goes home a winner with probability

$$\frac{1000}{1100} \approx .91.$$

So, given that the real game is “almost fair,” we might expect the probabilities of going home a winner in these two scenarios to be “almost 50% and 91%,” respectively.

Unfortunately for the gambler, all this “almost” reasoning will almost surely lead to disaster. Here are the grim facts for the real game where the gambler wins \$1 with probability $18/38$.

n = starting wealth	probability she reaches $n + \$100$ before \$0
\$100	1 in 3764$\underline{9}$.619496…
\$1000	1 in 37648.619496…
$\$1,000,000,000$	1 in 37648.619496…

Except on the very low end, the amount of money she brings makes almost no difference![2] She is almost certain to go broke before winning \$100. Let's see why.

20.2.1 Finding a Recurrence

We can approach the gambling problem the same way we studied the life of Stencil. Suppose that the gambler starts with n dollars. She wins each bet with probability p and plays until she either goes bankrupt or has $w = n + m$ dollars in her pocket. (To be clear, w is the total amount of money she wants to end up with, not the amount by which she wants to increase her wealth, which is m.) Our objective is to compute R_n, the probability that she goes home a winner.

As usual, we begin by identifying some boundary conditions. If she starts with no money, then she's bankrupt immediately so $R_0 = 0$. On the other hand, if she starts with w dollars, then she's an instant winner, so $R_w = 1$.

Now we divide the analysis of the general situation into two cases based on the outcome of her first bet:

- She wins her first bet with probability p. She then has $n + 1$ dollars and probability R_{n+1} of reaching her goal of w dollars.
- She loses her first bet with probability $1 - p$. This leaves her with $n - 1$ dollars and probability R_{n-1} of reaching her goal.

Plugging these facts into the Total Probability Theorem gives the equation:

$$R_n = pR_{n+1} + (1-p)R_{n-1}. \tag{20.2}$$

20.2.2 Solving the Recurrence

Rearranging the terms in Equation 20.2 gives us a recurrence for R_n, the probability that the gambler reaches her goal of w dollars if she starts with n:

$$\begin{aligned} R_0 &= 0 \\ R_w &= 1 \\ pR_{n+1} - R_n + (1-p)R_{n-1} &= 0 \qquad\qquad (0 < n < w). \end{aligned}$$

The characteristic equation is:

$$px^2 - x + (1-p) = 0.$$

[2] The fact that only one digit changes from the first case to the second is a peripheral bit of bizarreness that we'll leave in your hands.

The quadratic formula gives the roots:

$$\begin{aligned} x &= \frac{1 \pm \sqrt{1-4p(1-p)}}{2p} \\ &= \frac{1 \pm \sqrt{(1-2p)^2}}{2p} \\ &= \frac{1 \pm (1-2p)}{2p} \\ &= \frac{1-p}{p} \text{ or } 1. \end{aligned}$$

There’s an important point lurking here. If the gambler is equally likely to win or lose each bet, then $p = 1/2$, and the characteristic equation has a double root at $x = 1$. This is the situation we considered in the flea problem. The double root led to a general solution of the form:

$$R_n = a + bn$$

Now suppose that the gambler is *not* equally likely to win or lose each bet; that is, $p \neq 1/2$. Then the two roots of the characteristic equation are different, which means that the solution has a completely different form:

$$R_n = a \cdot \left(\frac{1-p}{p}\right)^n + b \cdot 1^n$$

In mathematical terms, this is where the fair game and the “almost fair” game take off in completely different directions: in one case we get a linear solution and in the other we get an exponential solution! This is going to be bad news for anyone playing the “almost fair” game.

Anyway, substituting the boundary conditions into the general form of the solution gives a system of linear equations:

$$\begin{aligned} 0 &= a + b \\ 1 &= a \cdot \left(\frac{1-p}{p}\right)^w + b. \end{aligned}$$

Solving this system, gives:

$$a = \frac{1}{\left(\frac{1-p}{p}\right)^w - 1}, \qquad b = -\frac{1}{\left(\frac{1-p}{p}\right)^w - 1}.$$

Substituting these values back into the general solution gives:

$$\begin{aligned} R_n &= \left(\frac{1}{\left(\frac{1-p}{p}\right)^w - 1} \right) \cdot \left(\frac{1-p}{p} \right)^n - \frac{1}{\left(\frac{1-p}{p}\right)^w - 1} \\ &= \frac{\left(\frac{1-p}{p}\right)^n - 1}{\left(\frac{1-p}{p}\right)^w - 1}. \end{aligned}$$

(Suddenly, Stencil's life doesn't seem so bad, huh?)

20.2.3 Bad News!

We have an answer! But it's not good news. If the gambler starts with n dollars and wins each bet with probability p, then the probability she reaches w dollars before going broke is:

$$\frac{\left(\frac{1-p}{p}\right)^n - 1}{\left(\frac{1-p}{p}\right)^w - 1}.$$

Let's try to make sense of this expression. If the game is biased against her, as with roulette, then $1-p$ (the probability she loses) is greater than p (the probability she wins). If n, her starting wealth, is also reasonably large, then both exponentiated fractions are big numbers and the -1's don't make much difference. Thus, her probability of reaching w dollars is very close to:

$$\left(\frac{1-p}{p} \right)^{n-w} = \left(\frac{1-p}{p} \right)^m.$$

In particular, if she is hoping to come out $m = \$100$ ahead in roulette, then $p = 18/38$ and her probability of success is:

$$\left(\frac{10}{9} \right)^{-100} = 1 \text{ in } 37648.619496.$$

This explains the strange number we arrived at earlier! In fact, this number does not change no matter how large n gets, so even if the gambler starts with a trillion dollars, she is sill not likely to ever get ahead by even \$100.

20.2.4 But Why?

Why does the gambler's starting wealth have so little impact on her probability of coming out ahead? Intuitively, there are two forces at work. First, the gambler's wealth has random upward and downward *swings* due to runs of good and bad luck. Second, her wealth has a steady, downward *drift* because she has a small expected loss on every bet. The situation is illustrated in Figure 20.2.

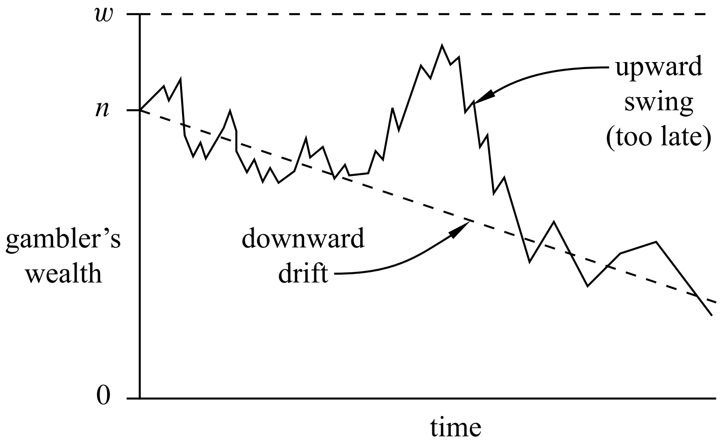


Figure 20.2 In a biased random walk, the downward drift usually dominates swings of good luck.

For example, in roulette, the gambler wins a dollar with probability $18/38$ and loses a dollar with probability $20/38$. Therefore, her expected return on each bet is

$$1 \cdot \frac{18}{38} + (-1) \cdot \frac{20}{38} = \frac{-2}{38} = -\frac{1}{19}.$$

Thus, her expected wealth drifts downward by a little over 5 cents per bet.

One might think that if the gambler starts with a billion dollars, then she will play for a long time, so at some point she should have a lucky, upward swing that puts her \$100 ahead. The problem is that her capital is steadily drifting downward. And after her capital drifts down a few hundred dollars, she needs a huge upward swing to save herself. And such a huge swing is extremely improbable. So if she does not have a lucky, upward swing early on, she's doomed forever. As a rule of thumb, *drift* dominates *swings* over the long term.

20.2.5 Expected Playing Time

Even though casino gamblers are destined to lose, some of them enjoy the process. So let's figure out how long their enjoyment is expected to last.

Let X_n be the expected number of bets before going home (broke or a winner).

Reasoning as in Section 20.1.5, we can set up a recurrence for X_n:

$$\begin{aligned} X_0 &= 0, \\ X_w &= 0, \\ X_n &= 1 + (1-p)X_{n-1} + pX_{n+1}. \end{aligned} \tag{20.3}$$

This is the same as the recurrence for R_n in Equation 20.2 except for the inhomogeneous part.

To find the particular solution, we try $X_n = c$ (which doesn't work) and then $X_n = c + dn$ (which does work as long as $p \neq 1/2$). Plugging $X_n = c + dn$ into Equation 20.3 yields:

$$\begin{aligned} c + dn &= 1 + (1-p)(c + d(n-1)) + p(c + d(n+1)) \\ &= 1 + c + dn - (1-p)d + pd \end{aligned}$$

and thus that

$$d = \frac{1}{1-2p}.$$

Since c is arbitrary, we will set $c = 0$ and our particular solution is

$$X_n = \frac{n}{1-2p}.$$

The characteristic equation for Equation 20.3 is

$$px^2 - x + (1-p) = 0.$$

We have already determined that the roots for this equation are

$$\frac{1-p}{p} \quad \text{and} \quad 1.$$

Hence, the general solution to the recurrence is

$$X_n = a\left(\frac{1-p}{p}\right)^n + b + \frac{n}{1-2p}.$$

Plugging in the boundary conditions, we find that

$$\begin{aligned} 0 &= a + b, \\ 0 &= a\left(\frac{1-p}{p}\right)^w + b + \frac{w}{1-2p}. \end{aligned}$$

Hence

$$a = \frac{-\left(\dfrac{w}{1-2p}\right)}{\left(\dfrac{1-p}{p}\right)^w - 1} \qquad \text{and} \qquad b = \frac{\left(\dfrac{w}{1-2p}\right)}{\left(\dfrac{1-p}{p}\right)^w - 1}.$$

The final solution to the recurrence is then

$$\begin{aligned} X_n &= \frac{-\left(\dfrac{w}{1-2p}\right)\left(\dfrac{1-p}{p}\right)^n}{\left(\dfrac{1-p}{p}\right)^w - 1} + \frac{\left(\dfrac{w}{1-2p}\right)}{\left(\dfrac{1-p}{p}\right)^w - 1} + \frac{n}{1-2p} \\ &= \frac{n}{1-2p} - \left(\frac{w}{1-2p}\right)\left[\frac{\left(\dfrac{1-p}{p}\right)^n - 1}{\left(\dfrac{1-p}{p}\right)^w - 1}\right]. \end{aligned}$$

Yikes! The gambler won't have any fun at all if she is thinking about this equation. Let's see if we can make it simpler in the case when $m = w - n$ is large.

Since $p < 1/2$, $(1-p)/p > 1$ and for large m,

$$\lim_{m\to\infty}\left(\frac{w}{1-2p}\right)\left[\frac{\left(\dfrac{1-p}{p}\right)^n - 1}{\left(\dfrac{1-p}{p}\right)^w - 1}\right] = \lim_{m\to\infty}\left(\frac{w}{1-2p}\right)\left(\frac{1-p}{p}\right)^{-m} = 0.$$

This means that as m gets large,

$$X_n \sim \frac{n}{1-2p},$$

which is much simpler. It says that if the gambler starts with $\$n$, she will expect to make about $n/(1-2p)$ bets before she goes home broke. This seems to make sense since she expects to lose

$$1 \cdot (1-p) + (-1)p = 1 - 2p$$

dollars on every bet and she started with n dollars.[3]

[3] Be careful, it is tempting to use such a direct and simple argument instead of all those nasty recurrences, but such an argument is not correct. There are examples where the expected duration of a process is not close to the starting point divided by the expected decrease at each step.

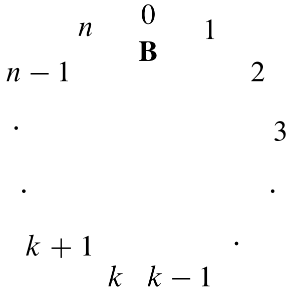


Figure 20.3 $n + 1$ people sitting in a circle. The B indicates the person with the broccoli—in this case, person 0.

20.3 Walking in Circles

So far, we have considered random walks on a line. Now we'll look at a problem where the random walk is on a circle. Going from a line to a circle may not seem like such a big change, but as we have seen so often with probability, small changes can have large consequences that are often beyond the grasp of our intuition.

20.3.1 Pass the Broccoli

Suppose there are $n + 1$ people, numbered $0, 1, \ldots, n$, sitting in a circle as shown in Figure 20.3. The B in Figure 20.3 indicates that person 0 has a big stalk of nutritious broccoli, which provides 250% of the US recommended daily allowance of vitamin C and is also a good source of vitamin A and iron. (Typical for a random walk problem, this game originally involved a pitcher of beer instead of a broccoli. We're taking the high road again.)

Person 0 passes the broccoli either to the person on his left or the person on his right with equal probability. Then, that person also passes the broccoli left or right at random and so on. After a while, everyone in an arc of the circle has touched the broccoli and everyone outside that arc has not. Eventually, the arc grows until all but one person has touched the broccoli. That final person is declared the winner because they have avoided the brocolli for the longest time.

Suppose that you are allowed to position yourself anywhere in the circle. Where should you stand in order to maximize the probability that you win? You shouldn't be person 0; you can't win in that position. The answer is "intuitively obvious": you should sit as far as possible from person 0, which would be position $n/2$ or $(n + 1)/2$ depending on whether n is even or odd.

20.3.2 There Is No Escape

Let's try to verify this intuition. Suppose that you sit at position $k \neq 0$. At some point, the broccoli is going to end up in the hands of one of your neighbors. This has to happen eventually; the game can't end until at least one of them touches it. Let's say that person $k - 1$ gets the broccoli first. Now let's cut the circle between yourself and your other neighbor, person $k + 1$:

$$\begin{array}{ccccccccccc} k & (k-1) & \dots & 3 & 2 & 1 & 0 & n & (n-1) & \dots & (k+1). \\ & \mathbf{B} & & & & & & & & & \end{array}$$

There are two possibilities. If the broccoli reaches you before it reaches person $k + 1$, then you lose. But if the broccoli reaches person $k + 1$ before it reaches you, then every other person has touched the broccoli and you win. So we need to compute the probability that the broccoli hops $n - 1$ people to the right before it takes 1 hop to the left. This will be the probability that you win.

But this is just the flea problem all over again. From the analysis in Section 20.1.3, we know that the probability of moving $n - 1$ steps rightward before moving one step leftward is simply $1/n$. This means that wherever you sit (aside from position 0, of course), your probability of getting the broccoli last is $1/n$.

So our intuition was completely wrong (again)! It doesn't matter where you sit. Being close to the broccoli or far away at the start makes no difference; there is no escape—you still get the broccoli last with probability $1/n$.

Enough with the bad news: Stencil's doomed, you go home broke from the casino, and you can't escape the broccoli. Let's see how to use probability to *make* some money.

6.042J / 18.062J Mathematics for Computer Science
Fall 2010