Chapter 4

Induction, Recursion, and Recurrences

4.1 Mathematical Induction

Smallest Counter-Examples

In Section 3.3, we saw one way of proving statements about infinite universes: we considered a "generic" member of the universe and derived the desired statement about that generic member. When our universe is the universe of integers, or is in a one-to-one correspondence with the integers, there is a second technique we can use.

Recall our our proof of Euclid's Division Theorem (Theorem 2.12), which says that for each pair (m, n) of positive integers, there are nonnegative integers q and r such that $m = nq + r$ and $0 \leq r < n$. "Among all pairs (m, n) that make it false, choose the smallest m that makes it false. We cannot have $m < n$ because then the statement would be true with $q = 0$, and we cannot have $m = n$ because then the statement is true with $q = 1$ and $r = 0$. This means $m - n$ is a positive number smaller than k, and so there must exist a q and r such that

$$k - n = qn + r, \text{ with } 0 \leq r < n.$$

Thus $m = (q + 1)n + r$, contradicting the assumption that the statement is false, so the only possibility is that the statement is true."

Focus on the sentences "This means $m - n$ is a positive number smaller than m, and so there must exist a q and r such that

$$m - n = qn + r, \text{ with } 0 \leq r < n.$$

Thus $m = (q+1)n+r, \ldots$." To analyze these sentences, let $p(m, n)$ denote the statement "there are nonnegative integers q and r with $0 \leq r < n$ such that $m = nq + r$" The quoted sentences above are a proof that $p(m - n, n) \Rightarrow p(m, n)$, and this implication is the crux of the proof. Restating the form of the proof: we assumed a counter-example with a smallest m existed, then using the fact that $p(m', n)$ had to be true for every m' smaller than m, we chose $m' = m-n$, and used the implication $p(m-n, n) \Rightarrow p(m, n)$ to conclude the truth of $p(m, n)$. But we had assumed that $p(m, n)$ was false, so this is the assumption we contradicted in the proof by contradiction.

Exercise 4.1-1 In Chapter 1 we learned Gauss's trick for showing that for all positive integers n,

$$1 + 2 + 3 + 4 + ... + n = \frac{n(n+1)}{2} . \tag{4.1}$$

Use the technique of asserting that if there is a counter-example, there is a smallest counter-example and deriving a contradiction to prove that the sum is $n(n + 1)/2$. What implication did you have to prove in the process?

Exercise 4.1-2 For what values of $n \geq 0$ do you think $2^{n+1} \geq n^2 + 2$? Use the technique of asserting there is a smallest counter-example and deriving a contradiction to prove you are right. What implication did you have to prove in the process?

Exercise 4.1-3 For what values of $n \geq 0$ do you think $2^{n+1} \geq n^2 + 3$? Is it possible to use the technique of asserting there is a smallest counter-example and deriving a contradiction to prove you are right? If so, do so and describe the implication did you had to prove in the process. If not, why not?

Exercise 4.1-4 Would it make sense to say that if there is a counter example there is a *largest* counter-example and try to base a proof on this? Why or why not?

In Exercise 4.1-1, suppose the formula for the sum is false. Then there must be a smallest n such that the formula does not hold for the sum of the first n positive integers. Thus for any positive integer i smaller than n,

$$1 + 2 + \cdots + i = \frac{i(i+1)}{2}. \tag{4.2}$$

Because $1 = 1 \cdot 2/2$, Equation 4.1 holds when $n = 1$, and therefore the smallest counterexample is not when $n = 1$. So $n > 1$, and $n - 1$ is one of the positive integers i for which the formula holds. Substituting $n - 1$ for i in Equation 4.2 gives us

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

Adding n to both sides gives

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

Thus n is not a counter-example after all, and therefore there is no counter-example to the formula. Thus the formula holds for all positive integers n. Note that the crucial step was proving that $p(n - 1) \Rightarrow p(n)$, where $p(n)$ is the formula

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

In Exercise 4.1-2, let $p(n)$ be the statement that $2^{n+1} \geq n^2 + 2$. Some experimenting with small values of n leads us to believe this statement is true for all nonnegative integers. Thus we want to prove $p(n)$ is true for all nonnegative integers n. To do so, we assume that the statement that "$p(n)$ is true for all nonnegative integers n" is false. When a "for all" statement is false, there must be some n for which it is false. Therefore, there is some smallest nonnegative integer n so that $2^{n+1} \not\geq n^2 + 2$. Assume now that n has this value. This means that for all nonnegative integers i with $i < n$, $2^{i+1} \geq i^2 + 2$. Since we know from our experimentation that $n \neq 0$, we know $n - 1$ is a nonnegative integer less than n, so using $n - 1$ in place of i, we get

$$2^{(n-1)+1} \geq (n-1)^2 + 2,$$

or

$$\begin{aligned} 2^n &\geq n^2 - 2n + 1 + 2 \\ &= n^2 - 2n + 3. \end{aligned} \tag{4.3}$$

From this we want to draw a contradiction, presumably a contradiction to $2^{n+1} \not\geq n^2 + 2$.

To get the contradiction, we want to convert the left-hand side of Equation 4.3 to 2^{n+1}. For this purpose, we multiply both sides by 2, giving

$$\begin{aligned} 2^{n+1} &= 2 \cdot 2^n \\ &\geq 2n^2 - 4n + 6\ . \end{aligned}$$

You may have gotten this far and wondered "What next?" Since we want to obtain a contradiction, we want to convert the right hand side into something like $n^2 + 2$. More precisely, we will convert the right-hand side into $n^2 + 2$ plus an additional term. If we can show that the additional term is nonnegative, the proof will be complete. Thus we write

$$\begin{aligned} 2^{n+1} &\geq 2n^2 - 4n + 6 \\ &= (n^2 + 2) + (n^2 - 4n + 4) \\ &= n^2 + 2 + (n-2)^2 \\ &\geq n^2 + 2\ , \end{aligned} \tag{4.4}$$

since $(n - 2)^2 \geq 0$. This is a contradiction, so there must not have been a smallest counter-example, and thus there must be no counter-example. Therefore $2^n \geq n^2 + 2$ for all nonnegative integers n.

What implication did we prove above? Let $p(n)$ stand for $2^{n+1} \geq n^2 + 2$. Then in Equations 4.3 and 4.4 we proved that $p(n-1) \Rightarrow p(n)$. Notice that at one point in our proof we had to note that we had considered the case with $n = 0$ already. Although we have given a proof by smallest counterexample, it is natural to ask whether it would make more sense to try to prove the statement directly. Would it make more sense to forget about the contradiction now that we have $p(n-1) \Rightarrow p(n)$ in hand and just observe that $p(0)$ and $p(n-1) \Rightarrow p(n)$ implies $p(1)$,that $p(1)$ and $p(n-1) \Rightarrow p(n)$ implies $p(2)$, and so on so that we have $p(k)$ for every k? We will address this question shortly.

Now let's consider Exercise 4.1-3. Notice that $2^{n+1} \not> n^2+3$ for $n = 0$ and 1, but $2^{n+1} > n^2+3$ for any larger n we look at at. Let us try to prove that $2^{n+1} > n^2 + 3$ for $n \geq 2$. We now let $p'(n)$ be the statement $2^{n+1} > n^2 + 3$. We can easily prove $p'(2)$: since $8 = 2^3 \geq 2^2 + 3 = 7$.

Now suppose that among the integers larger than 2 there is a counter-example m to $p'(n)$. That is, suppose that there is an m such that $m > 2$ and $p'(m)$ is false. Then there is a smallest such m, so that for k between 2 and $m-1$, $p(k)$ is true. If you look back at your proof that $p(n-1) \Rightarrow p(n)$, you will see that, when $n \geq 2$, essentially the same proof applies to p' as well. That is, with very similar computations we can show that $p'(n-1) \Rightarrow p'(n)$, so long as $n \geq 2$. Thus since $p(m-1)$ is true, our implication tells us that $p(m)$ is also true. This is a contradiction to our assumption that $p(m)$ is false. therefore, $p(m)$ is true. Again, we could conclude from $p'(2)$ and $p'(2) \Rightarrow p'(3)$ that $p'(3)$ is true, and similarly for $p'(4)$, and so on. The implication we had to prove was $p'(n-1) \Rightarrow p'(n)$.

For Exercise 4.1-4 if we have a counter-example to a statement $p(n)$ about an integer n, this means that there is an m such that $p(m)$ is false. To find a smallest counter example we would need to examine $p(0)$, $p(1)$, ..., perhaps all the way up to $p(m)$ in order to find a smallest counter-example, that is a smallest number k such that $p(k)$ is false. Since this involves only a finite number of cases, it makes sense to assert that there is a smallest counter-example. But, in answer to Exercise 4.1-4, it does not make sense to assert that there is a largest counter example, because there are infinitely many cases n that we would have to check in hopes if finding a largest one, and thus we might never find one. Even if we found one, we wouldn't be able to figure out that we had a largest counter-example just by checking larger and larger values of n, because we would never run out of values of n to check. Sometimes there is a largest counter-example, as in Exercise 4.1-3. To prove this, though, we didn't check all cases. Instead, based on our intuition, we guessed that the largest counter example was $n = 1$. Then we proved that we were right by showing that among numbers greater than or equal to two, there is no smallest counter-example. Sometimes there is no largest counter example n to a statement $p(n)$; for example $n^2 < n$ is false for all all integers n, and therefore there is no largest counter-example.

The Principle of Mathematical Induction

It may seem clear that repeatedly using the implication $p(n-1) \Rightarrow p(n)$ will prove $p(n)$ for all n (or all $n \geq 2$). That observation is the central idea of the Principle of Mathematical Induction, which we are about to introduce. In a theoretical discussion of how one constructs the integers from first principles, the principle of mathematical induction (or the equivalent principle that every set of nonnegative integers has a smallest element, thus letting us use the "smallest counter-example" technique) is one of the first principles we assume. The principle of mathematical induction is usually described in two forms. The one we have talked about so far is called the "weak form." It applies to statements about integers n.

The Weak Principle of Mathematical Induction. If the statement $p(b)$ is true, and the statement $p(n-1) \Rightarrow p(n)$ is true for all $n > b$, then $p(n)$ is true for all integers $n \geq b$.

Suppose, for example, we wish to give a direct inductive proof that $2^{n+1} > n^2 + 3$ for $n \geq 2$. We would proceed as follows. (The material in square brackets is not part of the proof; it is a running commentary on what is going on in the proof.)

> We shall prove by induction that $2^{n+1} > n^2 + 3$ for $n \geq 2$. First, $2^{2+1} = 2^3 = 8$, while $2^2 + 3 = 7$. [We just proved $p(2)$. We will now proceed to prove $p(n-1) \Rightarrow p(n)$.] Suppose now that $n > 2$ and that $2^n > (n-1)^2 + 3$. [We just made the hypothesis of $p(n-1)$ in order to use Rule 8 of our rules of inference.]

Now multiply both sides of this inequality by 2, giving us

$$\begin{aligned} 2^{n+1} &> 2(n^2 - 2n + 1) + 6 \\ &= n^2 + 3 + n^2 - 4n + 4 + 1 \\ &= n^2 + 3 + (n-2)^2 + 1\ . \end{aligned}$$

Since $(n-2)^2 + 1$ is positive for $n > 2$, this proves $2^{n+1} > n^2 + 3$. [We just showed that from the hypothesis of $p(n-1)$ we can derive $p(n)$. Now we can apply Rule 8 to assert that $p(n-1) \Rightarrow p(n)$.] Therefore

$$2^n > (n-1)^2 + 3 \quad \Rightarrow \quad 2^{n+1} > n^2 + 3.$$

Therefore by the principle of mathematical induction, $2^{n+1} > n^2 + 3$ for $n \geq 2$.

In the proof we just gave, the sentence "First, $2^{2+1} = 2^3 = 8$, while $2^2 + 3 = 7$" is called the *base case*. It consisted of proving that $p(b)$ is true, where in this case b is 2 and $p(n)$ is $2^{n+1} > n^2 + 3$. The sentence "Suppose now that $n > 2$ and that $2^n > (n-1)^2 + 3$." is called the *inductive hypothesis*. This is the assumption that $p(n-1)$ is true. In inductive proofs, we always make such a hypothesis[1] in order to prove the implication $p(n-1) \Rightarrow p(n)$. The proof of the implication is called the *inductive step* of the proof. The final sentence of the proof is called the *inductive conclusion*.

Exercise 4.1-5 Use mathematical induction to show that

$$1 + 3 + 5 + \cdots + (2k-1) = k^2$$

for each positive integer k.

Exercise 4.1-6 For what values of n is $2^n > n^2$? Use mathematical induction to show that your answer is correct.

For Exercise 4.1-5, we note that the formula holds when $k = 1$. Assume inductively that the formula holds when $k = n-1$, so that $1 + 3 + \cdots + (2n-3) = (n-1)^2$. Adding $2n-1$ to both sides of this equation gives

$$\begin{aligned} 1 + 3 + \cdots + (2n-3) + (2n-1) &= n^2 - 2n + 1 + 2n - 1 \\ &= n^2. \end{aligned} \tag{4.5}$$

Thus the formula holds when $k = n$, and so by the principle of mathematical induction, the formula holds for all positive integers k.

Notice that in our discussion of Exercise 4.1-5 we nowhere mentioned a statement $p(n)$. In fact, $p(n)$ is the statement we get by substituting n for k in the formula, and in Equation 4.5 we were proving $p(n-1) \Rightarrow p(n)$. Next notice that we did not explicitly say we were going to give a proof by induction; instead we told the reader when we were making the inductive hypothesis by saying "Assume inductively that ...." This convention makes the prose flow nicely but still tells the reader that he or she is reading a proof by induction. Notice also how the notation in

[1]At times, it might be more convenient to assume that $p(n)$ is true and use this assumption to prove that $p(n+1)$ is true. This proves the implication $p(n) \Rightarrow p(n+1)$, which lets us reason in the same way.

the statement of the exercise helped us write the proof. If we state what we are trying to prove in terms of a variable other than n, say k, then we can assume that our desired statement holds when this variable (k) is $n-1$ and then prove that the statement holds when $k = n$. Without this notational device, we have to either mention our statement $p(n)$ explicitly, or avoid any discussion of substituting values into the formula we are trying to prove. Our proof above that $2^{n+1} > n^2 + 3$ demonstrates this last approach to writing an inductive proof in plain English. This is usually the "slickest" way of writing an inductive proof, but it is often the hardest to master. We will use this approach first for the next exercise.

For Exercise 4.1-6 we note that $2 = 2^1 > 1^2 = 1$, but then the inequality fails for $n = 2, 3, 4$. However, $32 > 25$. Now we assume inductively that for $n > 5$ we have $2^{n-1} > (n-1)^2$. Multiplying by 2 gives us

$$\begin{aligned} 2^n > 2(n^2 - 2n + 1) &= n^2 + n^2 - 4n + 2 \\ &> n^2 + n^2 - n \cdot n \\ &= n^2 \ , \end{aligned}$$

since $n > 5$ implies that $-4n > -n \cdot n$. (We also used the fact that $n^2+n^2-4n+2 > n^2+n^2-4n$.) Thus by the principle of mathematical induction, $2^n > n^2$ for all $n \geq 5$.

Alternatively, we could write the following. Let $p(n)$ denote the inequality $2^n > n^2$. Then $p(5)$ is true because $32 > 25$. Assume that $n > 5$ and $p(n-1)$ is true. This gives us $2^{n-1} > (n-1)^2$. Multiplying by 2 gives

$$\begin{aligned} 2^n &> 2(n^2 - 2n + 1) \\ &= n^2 + n^2 - 4n + 2 \\ &> n^2 + n^2 - n \cdot n \\ &= n^2 \ , \end{aligned}$$

since $n > 5$ implies that $-4n > -n \cdot n$. Therefore $p(n-1) \Rightarrow p(n)$. Thus by the principle of mathematical induction, $2^n > n^2$ for all $n \geq 5$.

Notice how the "slick" method simply assumes that the reader knows we are doing a proof by induction from our "Assume inductively. . . ," and mentally supplies the appropriate $p(n)$ and observes that we have proved $p(n-1) \Rightarrow p(n)$ at the right moment.

Here is a slight variation of the technique of changing variables. To prove that $2^n > n^2$ when $n \geq 5$, we observe that the inequality holds when $n = 5$ since $32 > 25$. Assume inductively that the inequality holds when $n = k$, so that $2^k > k^2$. Now when $k \geq 5$, multiplying both sides of this inequality by 2 yields

$$\begin{aligned} 2^{k+1} > 2k^2 &= k^2 + k^2 \\ &\geq k^2 + 5k \\ &> k^2 + 2k + 1 \\ &= (k+1)^2 \ , \end{aligned}$$

since $k \geq 5$ implies that $k^2 \geq 5k$ and $5k = 2k+3k > 2k+1$. Thus by the principle of mathematical induction, $2^n > n^2$ for all $n \geq 5$.

This last variation of the proof illustrates two ideas. First, there is no need to save the name n for the variable we use in applying mathematical induction. We used k as our "inductive variable" in this case. Second, as suggested in a footnote earlier, there is no need to restrict ourselves to proving the implication $p(n-1) \Rightarrow p(n)$. In this case, we proved the implication $p(k) \Rightarrow p(k+1)$. Clearly these two implications are equivalent as n ranges over all integers larger than b and as k ranges over all integers larger than or equal to b.

Strong Induction

In our proof of Euclid's division theorem we had a statement of the form $p(m,n)$ and, assuming that it was false, we chose a smallest m such that $p(m,n)$ is false for some n. This meant we could assume that $p(m',n)$ is true for **all** $m' < m$, and we needed this assumption, because we ended up showing that $p(m-n,n) \Rightarrow p(m,n)$ in order to get our contradiction. This situation differs from the examples we used to introduce mathematical induction, for in those we used an implication of the form $p(n-1) \Rightarrow p(n)$. The essence of our method in proving Euclid's division theorem is that we have a statement $q(k)$ we want to prove. We suppose it is false, so that there must be a smallest k for which $q(k)$ is false. This means we may assume $q(k')$ is true for **all** k' in the universe of q with $k' < k$. We then use this assumption to derive a proof of $q(k)$, thus generating our contradiction.

Again, we can avoid the step of generating a contradiction in the following way. Suppose first we have a proof of $q(0)$. Suppose also that we have a proof that

$$q(0) \wedge q(1) \wedge q(2) \wedge \ldots \wedge q(k-1) \Rightarrow q(k)$$

for all k larger than 0. Then from $q(0)$ we can prove $q(1)$, from $q(0) \wedge q(1)$ we can prove $q(2)$, from $q(0) \wedge q(1) \wedge q(2)$ we can prove $q(3)$ and so on, giving us a proof of $q(n)$ for any n we desire. This is another form of the mathematical induction principle. We use it when, as in Euclid's division theorem, we can get an implication of the form $q(k') \Rightarrow q(k)$ for *some* $k' < k$ or when we can get an implication of the form $q(0) \wedge q(1) \wedge q(2) \wedge \ldots \wedge q(k-1) \Rightarrow q(k)$. (As is the case in Euclid's division theorem, we often don't really know what the k' is, so in these cases the first kind of situation is really just a special case of the second. Thus, we do not treat the first of the two implications separately.) We have described the method of proof known as the Strong Principle of Mathematical Induction.

The Strong Principle of Mathematical Induction. If the statement $p(b)$ is true, and the statement $p(b) \wedge p(b+1) \wedge \ldots \wedge p(n-1) \Rightarrow p(n)$ is true for all $n > b$, then $p(n)$ is true for all integers $n \geq b$.

Exercise 4.1-7 Prove that every positive integer is either a power of a prime number or the product of powers of prime numbers.

In Exercise 4.1-7 we can observe that 1 is a power of a prime number; for example $1 = 2^0$. Suppose now we know that every number less than n is a power of a prime number or a product of powers of prime numbers. Then if n is not a prime number, it is a product of two smaller numbers, each of which is, by our supposition, a power of a prime number or a product of powers of prime numbers. Therefore n is a power of a prime number or a product of powers of prime numbers. Thus, by the strong principle of mathematical induction, every positive integer is a power of a prime number or a product of powers of prime numbers.

Note that there was no explicit mention of an implication of the form

$$p(b) \wedge p(b+1) \wedge \ldots \wedge p(n-1) \Rightarrow p(n)\ .$$

This is common with inductive proofs. Note also that we did not explicitly identify the base case or the inductive hypothesis in our proof. This is common too. Readers of inductive proofs are expected to recognize when the base case is being given and when an implication of the form $p(n-1) \Rightarrow p(n)$ or $p(b) \wedge p(b+1) \wedge \cdots \wedge p(n-1) \Rightarrow p(n)$ is being proved.

Mathematical induction is used frequently in discrete math and computer science. Many quantities that we are interested in measuring, such as running time, space, or output of a program, typically are restricted to positive integers, and thus mathematical induction is a natural way to prove facts about these quantities. We will use it frequently throughout this book. We typically will not distinguish between strong and weak induction, we just think of them both as induction. (In Problems 14 and 15 at the end of the section you will be asked to derive each version of the principle from the other.)

Induction in general

To summarize what we have said so far, a typical proof by mathematical induction showing that a statement $p(n)$ is true for all integers $n \geq b$ consists of three steps.

1. First we show that $p(b)$ is true. This is called "establishing a *base case*."

2. Then we show either that for all $n > b$, $p(n-1) \Rightarrow p(n)$, or that for all $n > b$,

   $$p(b) \wedge p(b+1) \wedge \ldots \wedge p(n-1) \Rightarrow p(n).$$

   For this purpose, we make either the *inductive hypothesis* of $p(n-1)$ or the *inductive hypothesis* $p(b) \wedge p(b+1) \wedge \ldots \wedge p(n-1)$. Then we derive $p(n)$ to complete the proof of the implication we desire, either $p(n-1) \Rightarrow p(n)$ or $p(b) \wedge p(b+1) \wedge \ldots \wedge p(n-1) \Rightarrow p(n)$.

   Instead we could

2.′ show either that for all $n \geq b$, $p(n) \Rightarrow p(n+1)$ or

   $$p(b) \wedge p(b+1) \wedge \cdots \wedge p(n) \Rightarrow p(n+1).$$

   For this purpose, we make either the *inductive hypothesis* of $p(n)$ or the *inductive hypothesis* $p(b) \wedge p(b+1) \wedge \ldots \wedge p(n)$. Then we derive $p(n=1)$ to complete the proof of the implication we desire, either $p(n) \Rightarrow p(n=1)$ or $p(b) \wedge p(b+1) \wedge \ldots \wedge p(n) \Rightarrow p(n=1)$.

3. Finally, we conclude on the basis of the principle of mathematical induction that $p(n)$ is true for all integers n greater than or equal to b.

The second step is the core of an inductive proof. This is usually where we need the most insight into what we are trying to prove. In light of our discussion of Exercise 4.1-6, it should be clear that step $2'$ is simply a variation on the theme of writing an inductive proof.

It is important to realize that induction arises in some circumstances that do not fit the "pat" typical description we gave above. These circumstances seem to arise often in computer science However, inductive proofs always involve three things. First we always need a base case or cases.

Second, we need to show an implication that demonstrates that $p(n)$ is true given that $p(n')$ is true for some set of $n' < n$, or possibly we may need to show a set of such implications. Finally, we reach a conclusion on the basis of the first two steps.

For example, consider the problem of proving the following statement:

$$\sum_{i=0}^{n} \left\lfloor \frac{i}{2} \right\rfloor = \begin{cases} \frac{n^2}{4} & \text{if } n \text{ is even} \\ \frac{n^2-1}{4} & \text{if } n \text{ is odd} \end{cases} \tag{4.6}$$

In order to prove this, one must show that $p(0)$ is true, $p(1)$ is true, $p(n-2) \Rightarrow p(n)$ if n is odd, and that $p(n-2) \Rightarrow p(n)$, if n is even. Putting all these together, we see that our formulas hold for all $n \geq 0$. We can view this as either two proofs by induction, one for even and one for odd numbers, or one proof in which we have two base cases and two methods of deriving results from previous ones. This second view is more profitable, because it expands our view of what induction means, and makes it easier to find inductive proofs. In particular we could find situations where we have just one implication to prove but several base cases to check to cover all cases, or just one base case, but several different implications to prove to cover all cases.

Logically speaking, we could rework the example above so that if fits the pattern of strong induction. For example, when we prove a second base case, then we have just proved that the first base case implies it, because a true statement implies a true statement. Writing a description of mathematical induction that covers all kinds of base cases and implications one might want to consider in practice would simply give students one more unnecessary thing to memorize, so we shall not do so. However, in the mathematics literature and especially in the computer science literature, inductive proofs are written with multiple base cases and multiple implications with no effort to reduce them to one of the standard forms of mathematical induction. So long as it is possible to "cover" all the cases under consideration with such a proof, it can be rewritten as a standard inductive proof. Since readers of such proofs are expected to know this is possible, and since it adds unnecessary verbiage to a proof to do so, this is almost always left out.

Important Concepts, Formulas, and Theorems

1. *Weak Principle of Mathematical Induction.* The weak principle of mathematical induction states that

   If the statement $p(b)$ is true, and the statement $p(n-1) \Rightarrow p(n)$ is true for all $n > b$, then $p(n)$ is true for all integers $n \geq b$.

2. *Strong Principle of Mathematical Induction.* The strong principle of mathematical induction states that

   If the statement $p(b)$ is true, and the statement $p(b) \wedge p(b+1) \wedge \ldots \wedge p(n-1) \Rightarrow p(n)$ is true for all $n > b$, then $p(n)$ is true for all integers $n \geq b$.

3. *Base Case.* Every proof by mathematical induction, strong or weak, begins with a *base case* which establishes the result being proved for at least one value of the variable on which we are inducting. This base case should prove the result for the smallest value of the variable for which we are asserting the result. In a proof with multiple base cases, the base cases should cover all values of the variable which are not covered by the inductive step of the proof.

4. *Inductive Hypothesis.* Every proof by induction includes an inductive hypothesis in which we assume the result $p(n)$ we are trying to prove is true when $n = k - 1$ or when $n < k$ (or in which we assume an equivalent statement).

5. *Inductive Step.* Every proof by induction includes an inductive step in which we prove the implication that $p(k-1) \Rightarrow p(k)$ or the implication that $p(b) \wedge p(b+1) \wedge \cdots \wedge p(k-1) \Rightarrow p(k)$, or some equivalent implication.

6. *Inductive Conclusion.* A proof by mathematical induction should include, at least implicitly, a concluding statement of the form "Thus by the principle of mathematical induction . . . ," which asserts that by the principle of mathematical induction the result $p(n)$ which we are trying to prove is true for all values of n including and beyond the base case(s).

Problems

1. This exercise explores ways to prove that $\frac{2}{3} + \frac{2}{9} + \cdots + \frac{2}{3^n} = 1 - \left(\frac{1}{3}\right)^n$ for all positive integers n.

   (a) First, try proving the formula by contradiction. Thus you assume that there is some integer n that makes the formula false. Then there must be some smallest n that makes the formula false. Can this smallest n be 1? What do we know about $\frac{2}{3} + \frac{2}{9} + \cdots + \frac{2}{3^i}$ when i is a positive integer smaller than this smallest n? Is $n - 1$ a positive integer for this smallest n? What do we know about $\frac{2}{3} + \frac{2}{9} + \cdots + \frac{2}{3^{n-1}}$ for this smallest n? Write this as an equation and add $\frac{2}{3^n}$ to both sides and simplify the right side. What does this say about our assumption that the formula is false? What can you conclude about the truth of the formula? If $p(k)$ is the statement $\frac{2}{3} + \frac{2}{9} + \cdots + \frac{2}{3^k} = 1 - \left(\frac{1}{3}\right)^k$, what implication did we prove in the process of deriving our contradiction?

   (b) What is the base step in a proof by mathematical induction that $\frac{2}{3} + \frac{2}{9} + \cdots + \frac{2}{3^n} = 1 - \left(\frac{1}{3}\right)^n$ for all positive integers n? What would you assume as an inductive hypothesis? What would you prove in the inductive step of a proof of this formula by induction? Prove it. What does the principle of mathematical induction allow you to conclude? If $p(k)$ is the statement $\frac{2}{3} + \frac{2}{9} + \cdots + \frac{2}{3^k} = 1 - \left(\frac{1}{3}\right)^k$, what implication did we prove in the process of doing our proof by induction?

2. Use contradiction to prove that $1 \cdot 2 + 2 \cdot 3 + \cdots + n(n+1) = \frac{n(n+1)(n+2)}{3}$.

3. Use induction to prove that $1 \cdot 2 + 2 \cdot 3 + \cdots + n(n+1) = \frac{n(n+1)(n+2)}{3}$.

4. Prove that $1^3 + 2^3 + 3^3 + \cdots + n^3 = \frac{n^2(n+1)^2}{4}$.

5. Write a careful proof of Euclid's division theorem using strong induction.

6. Prove that $\sum_{i=j}^{n} \binom{i}{j} = \binom{n+1}{j+1}$. As well as the inductive proof that we are expecting, there is a nice "story" proof of this formula. It is well worth trying to figure it out.

7. Prove that every number greater than 7 is a sum of a nonnegative integer multiple of 3 and a nonnegative integer multiple of 5.

8. The usual definition of exponents in an advanced mathematics course (or an intermediate computer science course) is that $a^0 = 1$ and $a^{n+1} = a^n \cdot a$. Explain why this defines a^n for all nonnegative integers n. Prove the rule of exponents $a^{m+n} = a^m a^n$ from this definition.

9. Our arguments in favor of the sum principle were quite intuitive. In fact the sum principle for n sets follows from the sum principle for two sets. Use induction to prove the sum principle for a union of n sets from the sum principle for a union of two sets.

10. We have proved that every positive integer is a power of a prime number or a product of powers of prime numbers. Show that this factorization is unique in the following sense: If you have two factorizations of a positive integer, both factorizations use exactly the same primes, and each prime occurs to the same power in both factorizations. For this purpose, it is helpful to know that if a prime divides a product of integers, then it divides one of the integers in the product. (Another way to say this is that if a prime is a factor of a product of integers, then it is a factor of one of the integers in the product.)

11. Prove that $1^4 + 2^4 + \cdots + n^4 = O(n^5 - n^4)$.

12. Find the error in the following "proof" that all positive integers n are equal. Let $p(n)$ be the statement that all numbers in an n-element set of positive integers are equal. Then $p(1)$ is true. Now assume $p(n-1)$ is true, and let N be the set of the first n integers. Let N' be the set of the first $n-1$ integers, and let N'' be the set of the last $n-1$ integers. Then by $p(n-1)$ all members of N' are equal and all members of N'' are equal. Thus the first $n-1$ elements of N are equal and the last $n-1$ elements of N are equal, and so all elements of N are equal. Thus all positive integers are equal.

13. Prove by induction that the number of subsets of an n-element set is 2^n.

14. Prove that the Strong Principal of Mathematical Induction implies the Weak Principal of Mathematical Induction.

15. Prove that the Weak Principal of Mathematical Induction implies the Strong Principal of Mathematical Induction.

16. Prove (4.6).

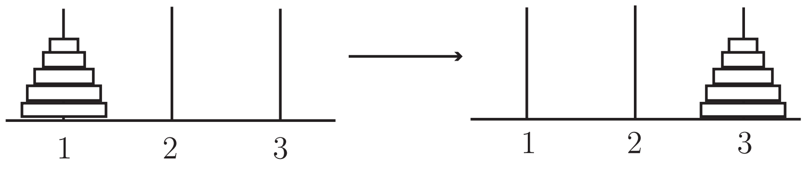
4.2 Recursion, Recurrences and Induction

Recursion

Exercise 4.2-1 Describe the uses you have made of recursion in writing programs. Include as many as you can.

Exercise 4.2-2 Recall that in the Towers of Hanoi problem we have three pegs numbered 1, 2 and 3, and on one peg we have a stack of n disks, each smaller in diameter than the one below it as in Figure 4.1. An allowable move consists of removing a disk

Figure 4.1: The Towers of Hanoi



from one peg and sliding it onto another peg so that it is not above another disk of smaller size. We are to determine how many allowable moves are needed to move the disks from one peg to another. Describe the strategy you have used or would use in a recursive program to solve this problem.

For the Tower of Hanoi problem, to solve the problem with no disks you do nothing. To solve the problem of moving all disks to peg 2, we do the following

1. (Recursively) solve the problem of moving $n-1$ disks from peg 1 to peg 3,
2. move disk n to peg 2,
3. (Recursively) solve the problem of moving $n-1$ disks on peg 3 to peg 2.

Thus if $M(n)$ is the number of moves needed to move n disks from peg i to peg j, we have

$$M(n) = 2M(n-1) + 1.$$

This is an example of a **recurrence equation** or **recurrence**. A recurrence equation for a function defined on the set of integers greater than or equal to some number b is one that tells us how to compute the nth value of a function from the $(n-1)$st value or some or all the values preceding n. To completely specify a function on the basis of a recurrence, we have to give enough information about the function to get started. This information is called the *initial condition (or the initial conditions)* (which we also call the *base case*) for the recurrence. In this case we have said that $M(0) = 0$. Using this, we get from the recurrence that $M(1) = 1$, $M(2) = 3$, $M(3) = 7$, $M(4) = 15$, $M(5) = 31$, and are led to guess that $M(n) = 2^n - 1$.

Formally, we write our recurrence and initial condition together as

$$M(n) = \begin{cases} 0 & \text{if } n = 0 \\ 2M(n-1) + 1 & \text{otherwise} \end{cases} \tag{4.7}$$

Now we give an inductive proof that our guess is correct. The base case is trivial, as we have defined $M(0) = 0$, and $0 = 2^0 - 1$. For the inductive step, we assume that $n > 0$ and $M(n-1) = 2^{n-1} - 1$. From the recurrence, $M(n) = 2M(n-1) + 1$. But, by the inductive hypothesis, $M(n-1) = 2^{n-1} - 1$, so we get that:

$$\begin{aligned} M(n) &= 2M(n-1) + 1 && (4.8) \\ &= 2(2^{n-1} - 1) + 1 && (4.9) \\ &= 2^n - 1. && (4.10) \end{aligned}$$

thus by the principle of mathematical induction, $m(n) = 2^n - 1$ for all nonnegative integers n.

The ease with which we solved this recurrence and proved our solution correct is no accident. Recursion, recurrences and induction are all intimately related. The relationship between recursion and recurrences is reasonably transparent, as recurrences give a natural way of analyzing recursive algorithms. Recursion and recurrences are abstractions that allow you to specify the solution to an instance of a problem of size n as some function of solutions to smaller instances. Induction also falls naturally into this paradigm. Here, you are deriving a statement $p(n)$ from statements $p(n')$ for $n' < n$. Thus we really have three variations on the same theme.

We also observe, more concretely, that the mathematical correctness of solutions to recurrences is naturally proved via induction. In fact, the correctness of recurrences in describing the number of steps needed to solve a recursive problem is also naturally proved by induction. The recurrence or recursive structure of the problem makes it straightforward to set up the induction proof.

First order linear recurrences

Exercise 4.2-3 The empty set ($\emptyset$) is a set with no elements. How many subsets does it have? How many subsets does the one-element set $\{1\}$ have? How many subsets does the two-element $\{1, 2\}$ set have? How many of these contain 2? How many subsets does $\{1, 2, 3\}$ have? How many contain 3? Give a recurrence for the number $S(n)$ of subsets of an n-element set, and prove by induction that your recurrence is correct.

Exercise 4.2-4 When someone is paying off a loan with initial amount A and monthly payment M at an interest rate of p percent, the total amount $T(n)$ of the loan after n months is computed by adding $p/12$ percent to the amount due after $n-1$ months and then subtracting the monthly payment M. Convert this description into a recurrence for the amount owed after n months.

Exercise 4.2-5 Given the recurrence

$$T(n) = rT(n-1) + a,$$

where r and a are constants, find a recurrence that expresses $T(n)$ in terms of $T(n-2)$ instead of $T(n-1)$. Now find a recurrence that expresses $T(n)$ in terms of $T(n-3)$ instead of $T(n-2)$ or $T(n-1)$. Now find a recurrence that expresses $T(n)$ in terms of $T(n-4)$ rather than $T(n-1)$, $T(n-2)$, or $T(n-3)$. Based on your work so far, find a general formula for the solution to the recurrence

$$T(n) = rT(n-1) + a,$$

with $T(0) = b$, and where r and a are constants.

If we construct small examples for Exercise 4.2-3, we see that $\emptyset$ has only 1 subset, $\{1\}$ has 2 subsets, $\{1,2\}$ has 4 subsets, and $\{1,2,3\}$ has 8 subsets. This gives us a good guess as to what the general formula is, but in order to prove it we will need to think recursively. Consider the subsets of $\{1,2,3\}$:

$$\begin{array}{cccc} \emptyset & \{1\} & \{2\} & \{1,2\} \\ \{3\} & \{1,3\} & \{2,3\} & \{1,2,3\} \end{array}$$

The first four subsets do not contain three, and the second four do. Further, the first four subsets are exactly the subsets of $\{1,2\}$, while the second four are the four subsets of $\{1,2\}$ with 3 added into each one. This suggests that the recurrence for the number of subsets of an n-element set (which we may assume is $\{1,2,\ldots,n\}$) is

$$S(n) = \begin{cases} 2S(n-1) & \text{if } n \geq 1 \\ 1 & \text{if } n = 0 \end{cases}. \tag{4.11}$$

To prove this recurrence is correct, we note is that the subsets of an n-element set can be partitioned by whether they contain element n or not. The subsets of $\{1,2,\ldots,n\}$ containing element n can be constructed by adjoining the element n to the subsets without element n. So the number of subsets with element n is the same as the number of subsets without element n. The number of subsets without element n is just the number of subsets of an $n-1$-element set. Thus the number of subsets of $\{1,2,\ldots,n\}$ is twice the number of subsets of $\{1,2,\ldots,n-1\}$. this proves that $S(n) = 2S(n-1)$ if $n > 0$. We already observed that $\emptyset$ has no subsets, so we have proved the correctness of Recurrence 4.11.

For Exercise 4.2-4 we can algebraically describe what the problem said in words by

$$T(n) = (1 + .01p/12) \cdot T(n-1) - M,$$

with $T(0) = A$. Note that we add $.01p/12$ times the principal to the amount due each month, because $p/12$ percent of a number is $.01p/12$ times the number.

Iterating a recurrence

Turning to Exercise 4.2-5, we can substitute the right hand side of the equation $T(n-1) = rT(n-2) + a$ for $T(n-1)$ in our recurrence, and then substitute the similar equations for $T(n-2)$ and $T(n-3)$ to write

$$\begin{aligned} T(n) &= r(rT(n-2)+a)+a \\ &= r^2T(n-2)+ra+a \\ &= r^2(rT(n-3)+a)+ra+a \\ &= r^3T(n-3)+r^2a+ra+a \\ &= r^3(rT(n-4)+a)+r^2a+ra+a \\ &= r^4T(n-4)+r^3a+r^2a+ra+a \end{aligned}$$

From this, we can guess that

$$T(n) = r^nT(0) + a\sum_{i=0}^{n-1} r^i$$

$$= r^n b + a\sum_{i=0}^{n-1} r^i. \tag{4.12}$$

The method we used to guess the solution is called *iterating the recurrence* because we repeatedly use the recurrence with smaller and smaller values in place of n. We could instead have written

$$\begin{aligned}
T(0) &= b \\
T(1) &= rT(0) + a \\
&= rb + a \\
T(2) &= rT(1) + a \\
&= r(rb + a) + a \\
&= r^2b + ra + a \\
T(3) &= rT(2) + a \\
&= r^3b + r^2a + ra + a
\end{aligned}$$

This leads us to the same guess, so why have we introduced two methods? Having different approaches to solving a problem often yields insights we would not get with just one approach. For example, when we study recursion trees, we will see how to visualize the process of iterating certain kinds of recurrences in order to simplify the algebra involved in solving them.

Geometric series

You may recognize that sum $\sum_{i=0}^{n-1} r^i$ in Equation 4.12. It is called a *finite geometric series with common ratio r*. The sum $\sum_{i=0}^{n-1} ar^i$ is called a *finite geometric series with common ratio r and initial value a*. Recall from algebra the factorizations

$$\begin{aligned}
(1-x)(1+x) &= 1-x^2 \\
(1-x)(1+x+x^2) &= 1-x^3 \\
(1-x)(1+x+x^2+x^3) &= 1-x^4
\end{aligned}$$

These factorizations are easy to verify, and they suggest that $(1-r)(1+r+r^2+\cdots+r^{n-1}) = 1-r^n$, or

$$\sum_{i=0}^{n-1} r^i = \frac{1-r^n}{1-r}. \tag{4.13}$$

In fact this formula is true, and lets us rewrite the formula we got for $T(n)$ in a very nice form.

Theorem 4.1 *If $T(n) = rT(n-1) + a$, $T(0) = b$, and $r \neq 1$ then*

$$T(n) = r^n b + a\frac{1-r^n}{1-r} \tag{4.14}$$

for all nonnegative integers n.

Proof: We will prove our formula by induction. Notice that the formula gives $T(0) = r^0b + a\frac{1-r^0}{1-r}$ which is b, so the formula is true when $n = 0$. Now assume that

$$T(n-1) = r^{n-1}b + a\frac{1-r^{n-1}}{1-r}.$$

Then we have

$$\begin{aligned} T(n) &= rT(n-1) + a \\ &= r\left(r^{n-1}b + a\frac{1-r^{n-1}}{1-r}\right) + a \\ &= r^n b + \frac{ar - ar^n}{1-r} + a \\ &= r^n b + \frac{ar - ar^n + a - ar}{1-r} \\ &= r^n b + a\frac{1-r^n}{1-r}\ . \end{aligned}$$

Therefore by the principle of mathematical induction, our formula holds for all integers n greater than 0. ■

We did not prove Equation 4.13. However it is easy to use Theorem 4.1 to prove it.

Corollary 4.2 *The formula for the sum of a geometric series with $r \neq 1$ is*

$$\sum_{i=0}^{n-1} r^i = \frac{1-r^n}{1-r}. \tag{4.15}$$

Proof: Define $T(n) = \sum_{i=0}^{n-1} r^i$. Then $T(n) = rT(n-1) + 1$, and since $T(0)$ is a sum with no terms, $T(0) = 0$. Applying Theorem 4.1 with $b = 0$ and $a = 1$ gives us $T(n) = \frac{1-r^n}{1-r}$. ■

Often, when we see a geometric series, we will only be concerned with expressing the sum in big-O notation. In this case, we can show that the sum of a geometric series is at most the largest term times a constant factor, where the constant factor depends on r, but not on n.

Lemma 4.3 *Let r be a quantity whose value is independent of n and not equal to 1. Let $t(n)$ be the largest term of the geometric series*

$$\sum_{i=0}^{n-1} r^i.$$

Then the value of the geometric series is $O(t(n))$.

Proof: It is straightforward to see that we may limit ourselves to proving the lemma for $r > 0$. We consider two cases, depending on whether $r > 1$ or $r < 1$. If $r > 1$, then

$$\begin{aligned} \sum_{i=0}^{n-1} r^i &= \frac{r^n - 1}{r-1} \\ &\leq \frac{r^n}{r-1} \\ &= r^{n-1}\frac{r}{r-1} \\ &= O(r^{n-1}). \end{aligned}$$

On the other hand, if $r < 1$, then the largest term is $r^0 = 1$, and the sum has value

$$\frac{1-r^n}{1-r} < \frac{1}{1-r}.$$

Thus the sum is $O(1)$, and since $t(n) = 1$, the sum is $O(t(n))$. ■

In fact, when r is nonnegative, an even stronger statement is true. Recall that we said that, for two functions f and g from the real numbers to the real numbers that $f = \Theta(g)$ if $f = O(g)$ and $g = O(f)$.

Theorem 4.4 *Let r be a nonnegative quantity whose value is independent of n and not equal to 1. Let $t(n)$ be the largest term of the geometric series*

$$\sum_{i=0}^{n-1} r^i.$$

Then the value of the geometric series is $\Theta(t(n))$.

Proof: By Lemma 4.3, we need only show that $t(n) = O(\frac{r^n-1}{r-1})$. Since all r^i are nonnegative, the sum $\sum_{i=0}^{n-1} r^i$ is at least as large as any of its summands. But $t(n)$ is one of these summands, so $t(n) = O(\frac{r^n-1}{r-1})$. ■

Note from the proof that $t(n)$ and the constant in the big-O upper bound depend on r. We will use this lemma in subsequent sections.

First order linear recurrences

A recurrence of the form $T(n) = f(n)T(n-1) + g(n)$ is called a *first order linear recurrence.* When $f(n)$ is a constant, say r, the general solution is almost as easy to write down as in the case we already figured out. Iterating the recurrence gives us

$$\begin{aligned}
T(n) &= rT(n-1) + g(n) \\
&= r\Big(rT(n-2) + g(n-1)\Big) + g(n) \\
&= r^2T(n-2) + rg(n-1) + g(n) \\
&= r^2\Big(rT(n-3) + g(n-2)\Big) + rg(n-1) + g(n) \\
&= r^3T(n-3) + r^2g(n-2) + rg(n-1) + g(n) \\
&= r^3\Big(rT(n-4) + g(n-3)\Big) + r^2g(n-2) + rg(n-1) + g(n) \\
&= r^4T(n-4) + r^3g(n-3) + r^2g(n-2) + rg(n-1) + g(n) \\
&\;\;\vdots \\
&= r^nT(0) + \sum_{i=0}^{n-1} r^i g(n-i)
\end{aligned}$$

This suggests our next theorem.

Theorem 4.5 *For any positive constants a and r, and any function g defined on the nonnegative integers, the solution to the first order linear recurrence*

$$T(n) = \begin{cases} rT(n-1) + g(n) & \text{if } n > 0 \\ a & \text{if } n = 0 \end{cases}$$

is

$$T(n) = r^n a + \sum_{i=1}^{n} r^{n-i} g(i). \tag{4.16}$$

Proof: Let's prove this by induction.

Since the sum $\sum_{i=1}^{n} r^{n-i}g(i)$ in Equation 4.16 has no terms when $n = 0$, the formula gives $T(0) = 0$ and so is valid when $n = 0$. We now assume that n is positive and $T(n-1) = r^{n-1}a + \sum_{i=1}^{n-1} r^{(n-1)-i}g(i)$. Using the definition of the recurrence and the inductive hypothesis we get that

$$\begin{aligned} T(n) &= rT(n-1) + g(n) \\ &= r\left(r^{n-1}a + \sum_{i=1}^{n-1} r^{(n-1)-i}g(i)\right) + g(n) \\ &= r^n a + \sum_{i=1}^{n-1} r^{(n-1)+1-i}g(i) + g(n) \\ &= r^n a + \sum_{i=1}^{n-1} r^{n-i}g(i) + g(n) \\ &= r^n a + \sum_{i=1}^{n} r^{n-i}g(i). \end{aligned}$$

Therefore by the principle of mathematical induction, the solution to

$$T(n) = \begin{cases} rT(n-1) + g(n) & \text{if } n > 0 \\ a & \text{if } n = 0 \end{cases}$$

is given by Equation 4.16 for all nonnegative integers n. ■

The formula in Theorem 4.5 is a little less easy to use than that in Theorem 4.1 because it gives us a sum to compute. Fortunately, for a number of commonly occurring functions g the sum $\sum_{i=1}^{n} r^{n-i}g(i)$ is reasonable to compute.

Exercise 4.2-6 Solve the recurrence $T(n) = 4T(n-1) + 2^n$ with $T(0) = 6$.

Exercise 4.2-7 Solve the recurrence $T(n) = 3T(n-1) + n$ with $T(0) = 10$.

For Exercise 4.2-6, using Equation 4.16, we can write

$$\begin{aligned} T(n) &= 6 \cdot 4^n + \sum_{i=1}^{n} 4^{n-i} \cdot 2^i \\ &= 6 \cdot 4^n + 4^n \sum_{i=1}^{n} 4^{-i} \cdot 2^i \end{aligned}$$

$$\begin{aligned}
&= 6 \cdot 4^n + 4^n \sum_{i=1}^{n} \Big(\frac{1}{2}\Big)^i \\
&= 6 \cdot 4^n + 4^n \cdot \frac{1}{2} \cdot \sum_{i=0}^{n-1} \Big(\frac{1}{2}\Big)^i \\
&= 6 \cdot 4^n + (1 - (\frac{1}{2})^n) \cdot 4^n \\
&= 7 \cdot 4^n - 2^n
\end{aligned}$$

For Exercise 4.2-7 we begin in the same way and face a bit of a surprise. Using Equation 4.16, we write

$$\begin{aligned}
T(n) &= 10 \cdot 3^n + \sum_{i=1}^{n} 3^{n-i} \cdot i \\
&= 10 \cdot 3^n + 3^n \sum_{i=1}^{n} i3^{-i} \\
&= 10 \cdot 3^n + 3^n \sum_{i=1}^{n} i \left(\frac{1}{3}\right)^i . \qquad (4.17)
\end{aligned}$$

Now we are faced with a sum that you may not recognize, a sum that has the form

$$\sum_{i=1}^{n} ix^i = x \sum_{i=1}^{n} ix^{i-1},$$

with $x = 1/3$. However by writing it in in this form, we can use calculus to recognize it as x times a derivative. In particular, using the fact that $0x^0 = 0$, we can write

$$\sum_{i=1}^{n} ix^i = x \sum_{i=0}^{n} ix^{i-1} = x \frac{d}{dx} \sum_{i=0}^{n} x^i = x \frac{d}{dx} \left(\frac{1 - x^{n+1}}{1 - x}\right).$$

But using the formula for the derivative of a quotient from calculus, we may write

$$x \frac{d}{dx} \left(\frac{1 - x^{n+1}}{1 - x}\right) = x \frac{(1-x)(-(n+1)x^n) - (a - x^{n+1})(-1)}{(1-x)^2} = \frac{nx^{n+2} - (n+1)x^{n+1} + x}{(1-x)^2}.$$

Connecting our first and last equations, we get

$$\sum_{i=1}^{n} ix^i = \frac{nx^{n+2} - (n+1)x^{n+1} + x}{(1-x)^2}. \qquad (4.18)$$

Substituting in $x = 1/3$ and simplifying gives us

$$\sum_{i=1}^{n} i \left(\frac{1}{3}\right)^i = -\frac{3}{2}(n+1)\left(\frac{1}{3}\right)^{n+1} - \frac{3}{4}\left(\frac{1}{3}\right)^{n+1} + \frac{3}{4}.$$

Substituting this into Equation 4.17 gives us

$$\begin{aligned}
T(n) &= 10 \cdot 3^n + 3^n \left(-\frac{3}{2}(n+1)\left(\frac{1}{3}\right)^{n+1} - \frac{3}{4}(1/3)^{n+1} + \frac{3}{4}\right) \\
&= 10 \cdot 3^n - \frac{n+1}{2} - \frac{1}{4} + \frac{3^{n+1}}{4} \\
&= \frac{43}{4} 3^n - \frac{n+1}{2} - \frac{1}{4}.
\end{aligned}$$

The sum that arises in this exercise occurs so often that we give its formula as a theorem.

Theorem 4.6 *For any real number $x \neq 1$,*

$$\sum_{i=1}^{n} ix^i = \frac{nx^{n+2} - (n+1)x^{n+1} + x}{(1-x)^2}. \tag{4.19}$$

Proof: Given before the statement of the theorem.■

Important Concepts, Formulas, and Theorems

1. *Recurrence Equation or Recurrence.* A recurrence equation is one that tells us how to compute the nth term of a sequence from the $(n-1)$st term or some or all the preceding terms.

2. *Initial Condition.* To completely specify a function on the basis of a recurrence, we have to give enough information about the function to get started. This information is called the *initial condition (or the initial conditions)* for the recurrence.

3. *First Order Linear Recurrence.* A recurrence $T(n) = f(n)T(n-1) + g(n)$ is called a *first order linear recurrence.*

4. *Constant Coefficient Recurrence.* A recurrence in which $T(n)$ is expressed in terms of a sum of constant multiples of $T(k)$ for certain values $k < n$ (and perhaps another function of n) is called a *constant coefficient recurrence.*

5. *Solution to a First Order Constant Coefficient Linear Recurrence.* If $T(n) = rT(n-1) + a$, $T(0) = b$, and $r \neq 1$ then
$$T(n) = r^n b + a\frac{1 - r^n}{1 - r}$$
for all nonnegative integers n.

6. *Finite Geometric Series.* A finite geometric series with common ratio r is a sum of the form $\sum_{i=0}^{n-1} r^i$. The formula for the sum of a geometric series with $r \neq 1$ is
$$\sum_{i=0}^{n-1} r^i = \frac{1 - r^n}{1 - r}.$$

7. *Big-Theta Bounds on the Sum of a Geometric Series.* Let r be a nonnegative quantity whose value is independent of n and not equal to 1. Let $t(n)$ be the largest term of the geometric series
$$\sum_{i=0}^{n-1} r^i.$$
Then the value of the geometric series is $\Theta(t(n))$.

8. *Solution to a First Order Linear Recurrence.* For any positive constants a and r, and any function g defined on the nonnegative integers, the solution to the first order linear recurrence
$$T(n) = \begin{cases} rT(n-1) + g(n) & \text{if } n > 0 \\ a & \text{if } n = 0 \end{cases}$$
is
$$T(n) = r^n a + \sum_{i=1}^{n} r^{n-i} g(i).$$

9. *Iterating a Recurrence.* We say we are *iterating* a recurrence when we guess its solution by using the equation that expresses $T(n)$ in terms of $T(k)$ for k smaller than n to re-express $T(n)$ in terms of $T(k)$ for k smaller than $n-1$, then for k smaller than $n-2$, and so on until we can guess the formula for the sum.

10. *An Important Sum.* For any real number $x \neq 1$,
$$\sum_{i=1}^{n} ix^i = \frac{nx^{n+2} - (n+1)x^{n+1} + x}{(1-x)^2}.$$

Problems

1. Prove Equation 4.15 directly by induction.

2. Prove Equation 4.18 directly by induction.

3. Solve the recurrence $M(n) = 2M(n-1) + 2$, with a base case of $M(1) = 1$. How does it differ from the solution to Recurrence 4.7?

4. Solve the recurrence $M(n) = 3M(n-1) + 1$, with a base case of $M(1) = 1$. How does it differ from the solution to Recurrence 4.7.

5. Solve the recurrence $M(n) = M(n-1) + 2$, with a base case of $M(1) = 1$. How does it differ from the solution to Recurrence 4.7.

6. There are m functions from a one-element set to the set $\{1, 2, \ldots, m\}$. How many functions are there from a two-element set to $\{1, 2, \ldots, m\}$? From a three-element set? Give a recurrence for the number $T(n)$ of functions from an n-element set to $\{1, 2, \ldots, m\}$. Solve the recurrence.

7. Solve the recurrence that you derived in Exercise 4.2-4.

8. At the end of each year, a state fish hatchery puts 2000 fish into a lake. The number of fish in the lake at the beginning of the year doubles due to reproduction by the end of the year. Give a recurrence for the number of fish in the lake after n years and solve the recurrence.

9. Consider the recurrence $T(n) = 3T(n-1) + 1$ with the initial condition that $T(0) = 2$. We know that we could write the solution down from Theorem 4.1. Instead of using the theorem, try to guess the solution from the first four values of $T(n)$ and then try to guess the solution by iterating the recurrence four times.

10. What sort of big-Θ bound can we give on the value of a geometric series $1+r+r^2+\cdots+r^n$ with common ratio $r=1$?

11. Solve the recurrence $T(n) = 2T(n-1) + n2^n$ with the initial condition that $T(0) = 1$.

12. Solve the recurrence $T(n) = 2T(n-1) + n^3 2^n$ with the initial condition that $T(0) = 2$.

13. Solve the recurrence $T(n) = 2T(n-1) + 3^n$ with $T(0) = 1$.

14. Solve the recurrence $T(n) = rT(n-1) + r^n$ with $T(0) = 1$.

15. Solve the recurrence $T(n) = rT(n-1) + r^{2n}$ with $T(0) = 1$

16. Solve the recurrence $T(n) = rT(n-1) + s^n$ with $T(0) = 1$.

17. Solve the recurrence $T(n) = rT(n-1) + n$ with $T(0) = 1$.

18. The Fibonacci numbers are defined by the recurrence

$$T(n) = \begin{cases} T(n-1) + T(n-2) & \text{if } n > 0 \\ 1 & \text{if } n = 0 \text{ or } n = 1 \end{cases}$$

(a) Write down the first ten Fibonacci numbers.

(b) Show that $(\frac{1+\sqrt{5}}{2})^n$ and $(\frac{1-\sqrt{5}}{2})^n$ are solutions to the equation $F(n) = F(n-1) + F(n-2)$.

(c) Why is

$$c_1(\frac{1+\sqrt{5}}{2})^n + c_2(\frac{1-\sqrt{5}}{2})^n$$

a solution to the equation $F(n) = F(n-1) + F(n-2)$ for any real numbers c_1 and c_2?

(d) Find constants c_1 and c_2 such that the Fibonacci numbers are given by

$$F(n) = c_1(\frac{1+\sqrt{5}}{2})^n + c_2(\frac{1-\sqrt{5}}{2})^n$$

4.3 Growth Rates of Solutions to Recurrences

Divide and Conquer Algorithms

One of the most basic and powerful algorithmic techniques is *divide and conquer*. Consider, for example, the binary search algorithm, which we will describe in the context of guessing a number between 1 and 100. Suppose someone picks a number between 1 and 100, and allows you to ask questions of the form "Is the number greater than k?" where k is an integer you choose. Your goal is to ask as few questions as possible to figure out the number. Your first question should be "Is the number greater than 50?" Why is this? Well, after asking if the number is bigger than 50, you have learned either that the number is between one and 50, or that the number is between 51 and 100. In either case have reduced your problem to one in which the range is only half as big. Thus you have *divided* the problem up into a problem that is only half as big, and you can now (recursively) *conquer* this remaining problem. (If you ask any other question, the size of one of the possible ranges of values you could end up with would be more than half the size of the original problem.) If you continue in this fashion, always cutting the problem size in half, you will reduce the problem size down to one fairly quickly, and then you will know what the number is. Of course it would be easier to cut the problem size exactly in half each time if we started with a number in the range from one to 128, but the question doesn't sound quite so plausible then. Thus to analyze the problem we will assume someone asks you to figure out a number between 0 and n, where n is a power of 2.

Exercise 4.3-1 Let $T(n)$ be number of questions in binary search on the range of numbers between 1 and n. Assuming that n is a power of 2, give a recurrence for $T(n)$.

For Exercise 4.3-1 we get:

$$T(n) = \begin{cases} T(n/2) + 1 & \text{if } n \geq 2 \\ 1 & \text{if } n = 1 \end{cases} \tag{4.20}$$

That is, the number of guesses to carry out binary search on n items is equal to 1 step (the guess) plus the time to solve binary search on the remaining $n/2$ items.

What we are really interested in is how much time it takes to use binary search in a computer program that looks for an item in an ordered list. While the number of questions gives us a feel for the amount of time, processing each question may take several steps in our computer program. The exact amount of time these steps take might depend on some factors we have little control over, such as where portions of the list are stored. Also, we may have to deal with lists whose length is not a power of two. Thus a more realistic description of the time needed would be

$$T(n) \leq \begin{cases} T(\lceil n/2 \rceil) + C_1 & \text{if } n \geq 2 \\ C_2 & \text{if } n = 1, \end{cases} \tag{4.21}$$

where C_1 and C_2 are constants.

Note that $\lceil x \rceil$ stands for the smallest integer larger than or equal to x, while $\lfloor x \rfloor$ stands for the largest integer less than or equal to x. It turns out that the solution to (4.20) and (4.21) are roughly the same, in a sense that will hopefully become clear later. (This is almost always the case.) For now, let us not worry about floors and ceilings and the distinction between things that take 1 unit of time and things that take no more than some constant amount of time.

Let's turn to another example of a divide and conquer algorithm, *mergesort*. In this algorithm, you wish to sort a list of n items. Let us assume that the data is stored in an array A in positions 1 through n. Mergesort can be described as follows:

```
MergeSort(A,low,high)
    if (low == high)
        return
    else
        mid = (low + high)/2
        MergeSort(A,low,mid)
        MergeSort(A,mid+1,high)
        Merge the sorted lists from the previous two steps
```

More details on mergesort can be found in almost any algorithms textbook. Suffice to say that the base case (low = high) takes one step, while the other case executes 1 step, makes two recursive calls on problems of size $n/2$, and then executes the Merge instruction, which can be done in n steps.

Thus we obtain the following recurrence for the running time of mergesort:

$$T(n) = \begin{cases} 2T(n/2) + n & \text{if } n > 1 \\ 1 & \text{if } n = 1 \end{cases} \tag{4.22}$$

Recurrences such as this one can be understood via the idea of a recursion tree, which we introduce below. This concept allows us to analyze recurrences that arise in divide-and-conquer algorithms, and those that arise in other recursive situations, such as the Towers of Hanoi, as well. A recursion tree for a recurrence is a visual and conceptual representation of the process of iterating the recurrence.

Recursion Trees

We will introduce the idea of a recursion tree via several examples. It is helpful to have an "algorithmic" interpretation of a recurrence. For example, (ignoring for a moment the base case) we can interpret the recurrence

$$T(n) = 2T(n/2) + n \tag{4.23}$$

as "in order to solve a problem of size n we must solve 2 problems of size $n/2$ and do n units of additional work." Similarly we can interpret

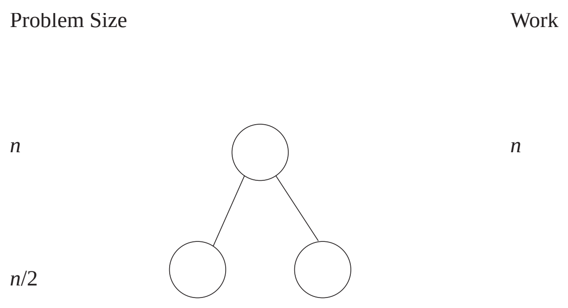
$$T(n) = T(n/4) + n^2$$

as "in order to solve a problem of size n we must solve one problem of size $n/4$ and do n^2 units of additional work."

We can also interpret the recurrence

$$T(n) = 3T(n-1) + n$$

Figure 4.2: The initial stage of drawing a recursion tree diagram.



as "in order to solve a problem of size n, we must solve 3 subproblems of size $n-1$ and do n additional units of work.

In Figure 4.2 we draw the beginning of the recursion tree diagram for (4.23). For now, assume n is a power of 2. A recursion tree diagram has three parts, a left, a middle, and a right. On the left, we keep track of the problem size, in the middle we draw the tree, and on right we keep track of the work done. We draw the diagram in levels, each level of the diagram representing a level of recursion. Equivalently, each level of the diagram represents a level of iteration of the recurrence. So to begin the recursion tree for (4.23), we show, in level 0 on the left, that we have problem of size n. Then by drawing a root vertex with two edges leaving it, we show in the middle that we are splitting our problem into 2 problems. We note on the right that we do n units of work in addition to whatever is done on the two new problems we created. In the next level, we draw two vertices in the middle representing the two problems into which we split our main problem and show on the left that each of these problems has size $n/2$.

You can see how the recurrence is reflected in levels 0 and 1 of the recursion tree. The top vertex of the tree represents $T(n)$, and on the next level we have two problems of size $n/2$, representing the recursive term $2T(n/2)$ of our recurrence. Then after we solve these two problems we return to level 0 of the tree and do n additional units of work for the nonrecursive term of the recurrence.

Now we continue to draw the tree in the same manner. Filling in the rest of level one and adding a few more a few more levels, we get Figure 4.3.

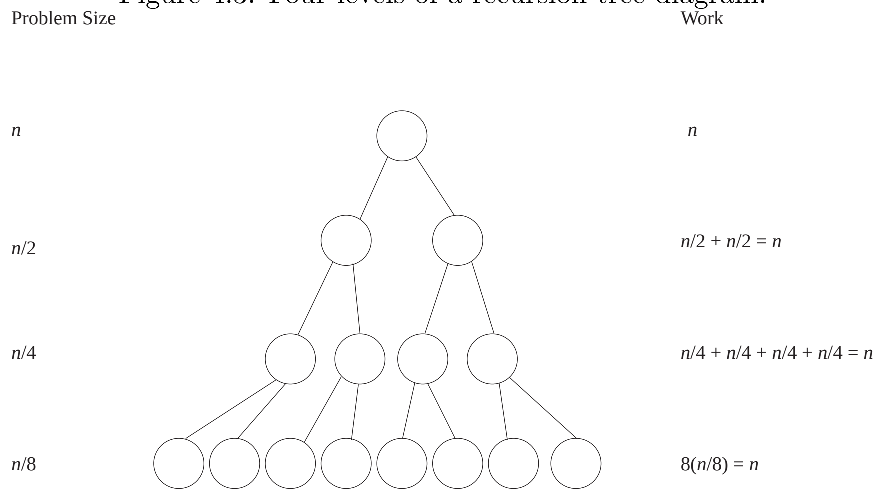
Let us summarize what the diagram tells us so far. At level zero (the top level), n units of work are done. We see that at each succeeding level, we halve the problem size and double the number of subproblems. We also see that at level 1, each of the two subproblems requires $n/2$ units of additional work, and so a total of n units of additional work are done. Similarly level 2 has 4 subproblems of size $n/4$ and so $4(n/4) = n$ units of additional work are done.

To see how iteration of the recurrence is reflected in the diagram, we iterate the recurrence once, getting

$$\begin{aligned} T(n) &= 2T(n/2) + n \\ T(n) &= 2(2T(n/4) + n/2) + n \\ T(n) &= 4T(n/4) + n + n = 4T(n/4) + 2n \end{aligned}$$

If we examine levels 0, 1, and 2 of the diagram, we see that at level 2 we have four vertices which represent four problems, each of size $n/4$ This corresponds to the recursive term that we obtained

Figure 4.3: Four levels of a recursion tree diagram.



after iterating the recurrence. However after we solve these problems we return to level 1 where we twice do $n/2$ additional units of work and to level 0 where we do another n additional units of work. In this way each time we add a level to the tree we are showing the result of one more iteration of the recurrence.

We now have enough information to be able to describe the recursion tree diagram in general. To do this, we need to determine, for each level, three things:

- the number of subproblems,
- the size of each subproblem,
- the total work done at that level.

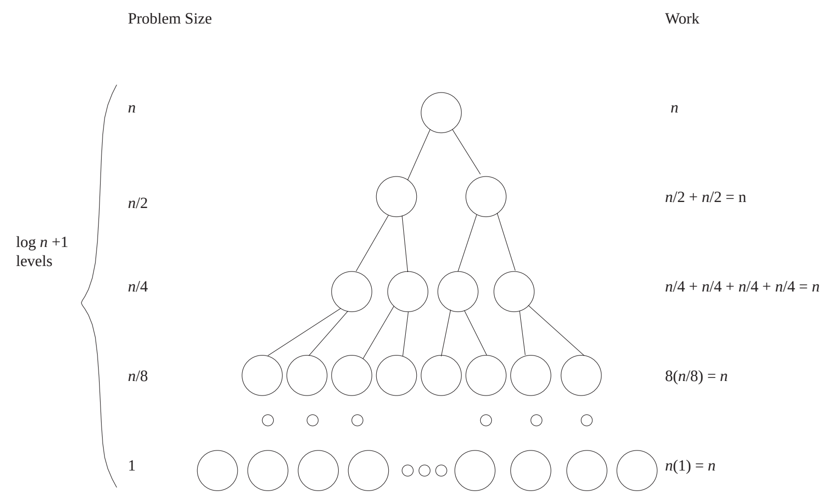
We also need to figure out how many levels there are in the recursion tree.

We see that for this problem, at level i, we have 2^i subproblems of size $n/2^i$. Further, since a problem of size 2^i requires 2^i units of additional work, there are $(2^i)[n/(2^i)] = n$ units of work done per level. To figure out how many levels there are in the tree, we just notice that at each level the problem size is cut in half, and the tree stops when the problem size is 1. Therefore there are $\log_2 n + 1$ levels of the tree, since we start with the top level and cut the problem size in half $\log_2 n$ times.[2] We can thus visualize the whole tree in Figure 4.4.

The computation of the work done at the bottom level is different from the other levels. In the other levels, the work is described by the recursive equation of the recurrence; in this case the amount of work is the n in $T(n) = 2T(n/2) + n$. At the bottom level, the work comes from the base case. Thus we must compute the number of problems of size 1 (assuming that one is the base case), and then multiply this value by $T(1) = 1$. For this particular recurrence, the nonrecursive term is n, and so when $n = 1$, we have $n = T(1)$ also. Had we chosen to say that $T(1)$ was some constant other than 1, this would not have been the case. We emphasize that

[2]To simplify notation, for the remainder of the book, if we omit the base of a logarithm, it should be assumed to be base 2.

Figure 4.4: A finished recursion tree diagram.



the correct value always comes from the base case; it is just a coincidence that it sometimes also comes from the recursive equation of the recurrence.

The bottom level of the tree represents the final stage of iterating the recurrence. We have seen that at this level we have n problems each requiring work $T(1) = 1$, giving us total work n at that level. After we solve the problems represented by the bottom level, we have to do all the additional work from all the earlier levels. For this reason, we sum the work done at all the levels of the tree to get the total work done. *Iteration of the recurrence shows us that the solution to the recurrence is the sum of all the work done at all the levels of the recursion tree.*

The important thing is that we now know how much work is done at each level. Once we know this, we can sum the total amount of work done over all the levels, giving us the solution to our recurrence. In this case, there are $\log_2 n + 1$ levels, and at each level the amount of work we do is n units. Thus we conclude that the total amount of work done to solve the problem described by recurrence (4.23) is $n(\log_2 n + 1)$. The total work done throughout the tree is the solution to our recurrence, because the tree simply models the process of iterating the recurrence. Thus the solution to recurrence (4.22) is $T(n) = n(\log n + 1)$.

Since one unit of time will vary from computer to computer, and since some kinds of work might take longer than other kinds, we are usually interested in the big-θ behavior of $T(n)$. For example, we can consider a recurrence that it identical to (4.22), except that $T(1) = a$, for some constant a. In this case, $T(n) = an + n \log n$, because an units of work are done at level 1 and n additional units of work are done at each of the remaining $\log n$ levels. It is still true that $T(n) = \Theta(n \log n)$, because the different base case did not change the solution to the recurrence by more than a constant factor[3]. Although recursion trees can give us the exact solutions (such as $T(n) = an + n \log n$ above) to recurrences, our interest in the big-Θ behavior of solutions will usually lead us to use a recursion tree to determine the big-Θ or even, in complicated cases, just the big-O behavior of the actual solution to the recurrence. In Problem 10 we explore whether the value of $T(1)$ actually influences the big-Θ behavior of the solution to a recurrence.

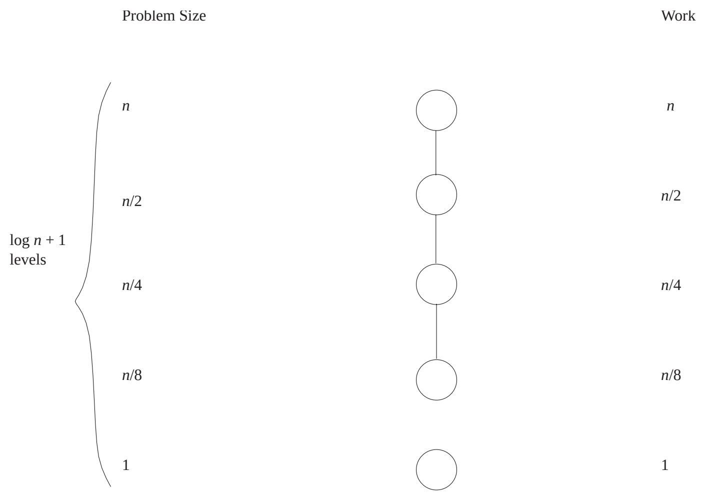
Let's look at one more recurrence.

[3]More precisely, $n \log n < an + n \log n < (a + 1)n \log n$ for any $a > 0$.

$$T(n) = \begin{cases} T(n/2) + n & \text{if } n > 1 \\ 1 & \text{if } n = 1 \end{cases} \tag{4.24}$$

Again, assume n is a power of two. We can interpret this as follows: to solve a problem of size n, we must solve one problem of size $n/2$ and do n units of additional work. We draw the tree for this problem in Figure 4.5 and see that the problem sizes are the same as in the previous tree. The remainder, however, is different. The number of subproblems does not double, rather

Figure 4.5: A recursion tree diagram for Recurrence 4.24.



it remains at one on each level. Consequently the amount of work halves at each level. Note that there are still $\log n + 1$ levels, as the number of levels is determined by how the problem size is changing, not by how many subproblems there are. So on level i, we have 1 problem of size $n/2^i$, for total work of $n/2^i$ units.

We now wish to compute how much work is done in solving a problem that gives this recurrence. Note that the additional work done is different on each level, so we have that the total amount of work is

$$n + n/2 + n/4 + \cdots + 2 + 1 = n\left(1 + \frac{1}{2} + \frac{1}{4} + \cdots + \left(\frac{1}{2}\right)^{\log_2 n}\right),$$

which is n times a geometric series. By Theorem 4.4, the value of a geometric series in which the largest term is one is $\Theta(1)$. This implies that the work done is described by $T(n) = \Theta(n)$.

We emphasize that there is exactly one solution to recurrence (4.24); it is the one we get by using the recurrence to compute $T(2)$ from $T(1)$, then to compute $T(4)$ from $T(2)$, and so on. What we have done here is show that $T(n) = \Theta(n)$. In fact, for the kinds of recurrences we have been examining, once we know $T(1)$ we can compute $T(n)$ for any relevant n by repeatedly using the recurrence, so there is no question that solutions do exist and can, in principle, be computed for any value of n. In most applications, we are not interested in the exact form of the solution, but a big-O upper bound, or Big-Θ bound on the solution.

Exercise 4.3-2 Find a big-Θ bound for the solution to the recurrence

$$T(n) = \begin{cases} 3T(n/3) + n & \text{if } n \geq 3 \\ 1 & \text{if } n < 3 \end{cases}$$

using a recursion tree. Assume that n is a power of 3.

Exercise 4.3-3 Solve the recurrence

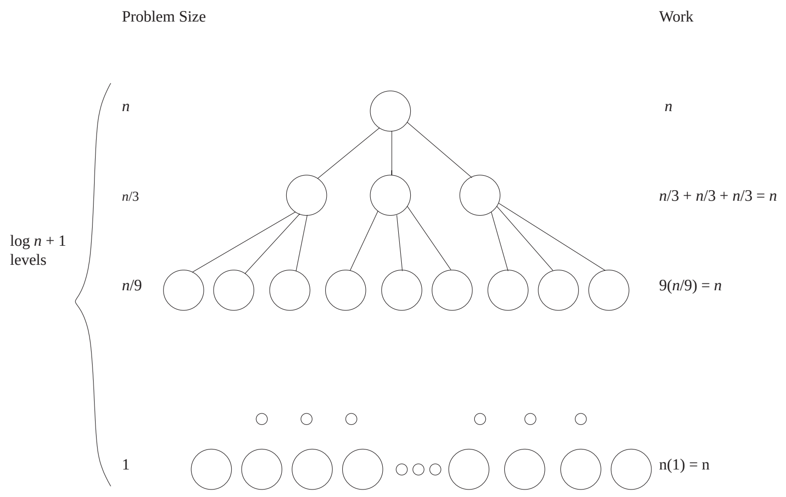
$$T(n) = \begin{cases} 4T(n/2) + n & \text{if } n \geq 2 \\ 1 & \text{if } n = 1 \end{cases}$$

using a recursion tree. Assume that n is a power of 2. Convert your solution to a big-Θ statement about the behavior of the solution.

Exercise 4.3-4 Can you give a general big-Θ bound for solutions to recurrences of the form $T(n) = aT(n/2) + n$ when n is a power of 2? You may have different answers for different values of a.

The recurrence in Exercise 4.3-2 is similar to the mergesort recurrence. One difference is that at each step we divide into 3 problems of size $n/3$. Thus we get the picture in Figure 4.6. Another difference is that the number of levels, instead of being $\log_2 n + 1$ is now $\log_3 n + 1$, so

Figure 4.6: The recursion tree diagram for the recurrence in Exercise 4.3-2.

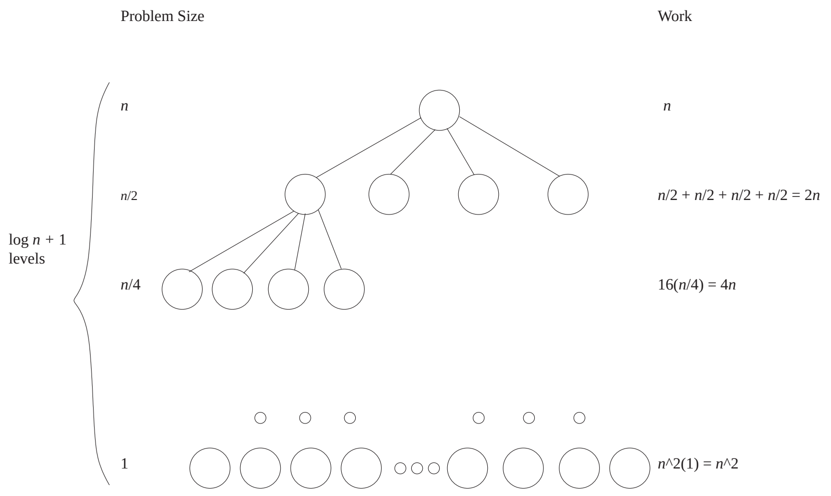


the total work is still $\Theta(n \log n)$ units. (Note that $\log_b n = \Theta(\log_2 n)$ for any $b > 1$.)

Now let's look at the recursion tree for Exercise 4.3-3. Here we have 4 children of size $n/2$, and we get Figure 4.7 Let's look carefully at this tree. Just as in the mergesort tree there are $\log_2 n + 1$ levels. However, in this tree, each node has 4 children. Thus level 0 has 1 node, level 1 has 4 nodes, level 2 has 16 nodes, and in general level i has 4^i nodes. On level i each node corresponds to a problem of size $n/2^i$ and hence requires $n/2^i$ units of additional work. Thus the total work on level i is $4^i(n/2^i) = 2^i n$ units. This formula applies on level $\log_2 n$ as well since there are $n^2 = 2^{\log_2 n} n$ nodes, each requiring $T(1) = 1$ work. Summing over the levels, we get

$$\sum_{i=0}^{\log_2 n} 2^i n = n \sum_{i=0}^{\log_2 n} 2^i.$$

Figure 4.7: The Recursion tree for Exercise 4.3-3.



There are many ways to simplify that expression, for example from our formula for the sum of a geometric series we get

$$\begin{aligned}
T(n) &= n\sum_{i=0}^{\log_2 n} 2^i \\
&= n\frac{1-2^{(\log_2 n)+1}}{1-2} \\
&= n\frac{1-2n}{-1} \\
&= 2n^2 - n \\
&= \Theta(n^2).
\end{aligned}$$

More simply, by Theorem 4.4 we have that $T(n) = n\Theta(2^{\log n}) = \Theta(n^2)$.

Three Different Behaviors

Now let's compare the recursion tree diagrams for the recurrences $T(n) = 2T(n/2) + n$, $T(n) = T(n/2) + n$ and $T(n) = 4T(n/2) + n$. Note that all three trees have depth $1 + \log_2 n$, as this is determined by the size of the subproblems relative to the parent problem, and in each case, the size of each subproblem is $1/2$ the size of of the parent problem. The trees differ, however, in the amount of work done per level. In the first case, the amount of work on each level is the same. In the second case, the amount of work done on a level decreases as you go down the tree, with the most work being at the top level. In fact, it decreases geometrically, so by Theorem 4.4 the total work done is bounded above and below by a constant times the work done at the root node. In the third case, the number of nodes per level is growing at a faster rate than the problem size is decreasing, and the level with the largest amount of work is the bottom one. Again we have a geometric series, and so by Theorem 4.4 the total work is bounded above and below by a constant times the amount of work done at the last level.

If you understand these three cases and the differences among them, you now understand the great majority of the recursion trees that arise in algorithms.

So to answer Exercise 4.3-4, which asks for a general Big-Θ bound for the solutions to recurrences of the form $T(n) = aT(n/2) + n$, we can conclude the following:

Lemma 4.7 *Suppose that we have a recurrence of the form*

$$T(n) = aT(n/2) + n,$$

where a is a positive integer and $T(1)$ is nonnegative. The we have the following big-Theta bounds on the solution.

1. *If $a < 2$ then $T(n) = \Theta(n)$.*
2. *If $a = 2$ then $T(n) = \Theta(n \log n)$*
3. *If $a > 2$ then $T(n) = \Theta(n^{\log_2 a})$*

Proof: Cases 1 and 2 follow immediately from our observations above. We can verify case 3 as follows. At each level i we have a^i nodes, each corresponding to a problem of size $n/2^i$. Thus at level i the total amount of work is $a^i(n/2^i) = n(a/2)^i$ units. Summing over the $\log_2 n$ levels, we get

$$a^{\log_2 n}T(1) + n \sum_{i=0}^{(\log_2 n)-1} (a/2)^i.$$

The sum given by the summation sign is a geometric series, so, since $a/2 \neq 1$, the sum will be big-Θ of the largest term (see Theorem 4.4). Since $a > 2$, the largest term in this case is clearly the last one, namely $n(a/2)^{(\log_2 n)-1}$, and applying rules of exponents and logarithms, we get that n times the largest term is

$$n\left(\frac{a}{2}\right)^{(\log_2 n)-1} = \frac{2}{a} \cdot \frac{n \cdot a^{\log_2 n}}{2^{\log_2 n}} = \frac{2}{a} \cdot \frac{n \cdot a^{\log_2 n}}{n} = \frac{2}{a} \cdot a^{\log_2 n} = \frac{2}{a} \cdot 2^{\log_2 a \log_2 n} = \frac{2}{a} \cdot n^{\log_2 a}. \quad (4.25)$$

Thus $T(1)a^{\log_2 n} = T(1)n^{\log_2 a}$. Since $\frac{2}{a}$ and $T(1)$ are both nonnegative, the total work done is $\Theta(n^{\log_2 a})$. ■

In fact Lemma 4.7 holds for all positive real numbers a; we can iterate the recurrence to see this. Since a recursion tree diagram is a way to visualize iterating the recurrence when a is an integer, iteration is the natural thing to try when a is not an integer.

Notice that in the last two equalities of computation we made in Equation 4.25, we showed that $a^{\log n} = n^{\log a}$. This is a useful and, perhaps, surprising fact, so we state it (in slightly more generality) as a corollary to the proof.

Corollary 4.8 *For any base b, we have $a^{\log_b n} = n^{\log_b a}$.*

Important Concepts, Formulas, and Theorems

1. *Divide and Conquer Algorithm.* A *divide and conquer algorithm* is one that solves a problem by dividing it into problems that are smaller but otherwise of the same type as the original one, recursively solves these problems, and then assembles the solution of these so-called subproblems into a solution of the original one. Not all problems can be solved by such a strategy, but a great many problems of interest in computer science can.

2. *Mergesort.* In *mergesort* we sort a list of items that have some underlying order by dividing the list in half, sorting the first half (by recursively using mergesort), sorting the second half (by recursively using mergesort), and then merging the two sorted list. For a list of length one mergesort returns the same list.

3. *Recursion Tree.* A *recursion tree diagram* for a recurrence of the form $T(n) = aT(n/b)+g(n)$ has three parts, a left, a middle, and a right. On the left, we keep track of the problem size, in the middle we draw the tree, and on right we keep track of the work done. We draw the diagram in levels, each level of the diagram representing a level of recursion. The tree has a vertex representing the initial problem and one representing each subproblem we have to solve. Each non-leaf vertex has a children. The vertices are divided into levels corresponding to (sub-)problems of the same size; to the left of a level of vertices we write the size of the problems the vertices correspond to; to the right of the vertices on a given level we write the total amount of work done at that level by an algorithm whose work is described by the recurrence, not including the work done by any recursive calls from that level.

4. *The Base Level of a Recursion Tree.* The amount of work done on the lowest level in a recursion tree is the number of nodes times the value given by the initial condition; it is not determined by attempting to make a computation of "additional work" done at the lowest level.

5. *Bases for Logarithms.* We use $\log n$ as an alternate notation for $\log_2 n$. A fundamental fact about logarithms is that $\log_b n = \Theta(\log_2 n)$ for any real number $b > 1$.

6. *An Important Fact About Logarithms.* For any $b > 0$, $a^{\log_b n} = n^{\log_b a}$.

7. *Three behaviors of solutions.* The solution to a recurrence of the form $T(n) = aT(n/2) + n$ behaves in one of the following ways:

    (a) if $a < 2$ then $T(n) = \Theta(n)$.

    (b) if $a = 2$ then $T(n) = \Theta(n \log n)$

    (c) if $a > 2$ then $T(n) = \Theta(n^{\log_2 a})$.

Problems

1. Draw recursion trees and find big-Θ bounds on the solutions to the following recurrences. For all of these, assume that $T(1) = 1$ and n is a power of the appropriate integer.

    (a) $T(n) = 8T(n/2) + n$

    (b) $T(n) = 8T(n/2) + n^3$

(c) $T(n) = 3T(n/2) + n$

(d) $T(n) = T(n/4) + 1$

(e) $T(n) = 3T(n/3) + n^2$

2. Draw recursion trees and find exact solutions to the following recurrences. For all of these, assume that $T(1) = 1$ and n is a power of the appropriate integer.

   (a) $T(n) = 8T(n/2) + n$

   (b) $T(n) = 8T(n/2) + n^3$

   (c) $T(n) = 3T(n/2) + n$

   (d) $T(n) = T(n/4) + 1$

   (e) $T(n) = 3T(n/3) + n^2$

3. Find the exact solution to Recurrence 4.24.

4. Show that $\log_b n = \Theta(\log_2 n)$, for any constant $b > 1$.

5. Prove Corollary 4.8 by showing that $a^{\log_b n} = n^{\log_b a}$ for any $b > 0$.

6. Recursion trees will still work, even if the problems do not break up geometrically, or even if the work per level is not n^c units. Draw recursion trees and and find the best big-O bounds you can for solutions to the following recurrences. For all of these, assume that $T(1) = 1$.

   (a) $T(n) = T(n-1) + n$

   (b) $T(n) = 2T(n-1) + n$

   (c) $T(n) = T(\lfloor\sqrt{n}\rfloor) + 1$ (You may assume n has the form $n = 2^{2^i}$.)

   (d) $T(n) = 2T(n/2) + n\log n$ (You may assume n is a power of 2.)

7. In each case in the previous problem, is the big-O bound you found a big-Θ bound?

8. If $S(n) = aS(n-1) + g(n)$ and $g(n) < c^n$ with $0 \le c < a$, how fast does $S(n)$ grow (in big-Θ terms)?

9. If $S(n) = aS(n-1) + g(n)$ and $g(n) = c^n$ with $0 < a \le c$, how fast does $S(n)$ grow in big-Θ terms?

10. Given a recurrence of the form $T(n) = aT(n/b) + g(n)$ with $T(1) = c > 0$ and $g(n) > 0$ for all n and a recurrence of the form $S(n) = aS(n/b) + g(n)$ with $S(1) = 0$ (and the same a. b, and $g(n)$), is there any difference in the big-Θ behavior of the solutions to the two recurrences? What does this say about the influence of the initial condition on the big-Θ behavior of such recurrences?

4.4 The Master Theorem

Master Theorem

In the last section, we saw three different kinds of behavior for recurrences of the form

$$T(n) = \begin{cases} aT(n/2) + n & \text{if } n > 1 \\ d & \text{if } n = 1. \end{cases}$$

These behaviors depended upon whether $a < 2$, $a = 2$, or $a > 2$. Remember that a was the number of subproblems into which our problem was divided. Dividing by 2 cut our problem size in half each time, and the n term said that after we completed our recursive work, we had n additional units of work to do for a problem of size n. There is no reason that the amount of additional work required by each subproblem needs to be the size of the subproblem. In many applications it will be something else, and so in Theorem 4.9 we consider a more general case. Similarly, the sizes of the subproblems don't have to be 1/2 the size of the parent problem. We then get the following theorem, our first version of a theorem called the *Master Theorem.* (Later on we will develop some stronger forms of this theorem.)

Theorem 4.9 *Let a be an integer greater than or equal to 1 and b be a real number greater than 1. Let c be a positive real number and d a nonnegative real number. Given a recurrence of the form*

$$T(n) = \begin{cases} aT(n/b) + n^c & \text{if } n > 1 \\ d & \text{if } n = 1 \end{cases}$$

then for n a power of b,

1. *if $\log_b a < c$, $T(n) = \Theta(n^c)$,*
2. *if $\log_b a = c$, $T(n) = \Theta(n^c \log n)$,*
3. *if $\log_b a > c$, $T(n) = \Theta(n^{\log_b a})$.*

Proof: In this proof, we will set $d = 1$, so that the work done at the bottom level of the tree is the same as if we divided the problem one more time and used the recurrence to compute the additional work. As in Footnote 3 in the previous section, it is straightforward to show that we get the same big-Θ bound if d is positive. It is only a little more work to show that we get the same big-Θ bound if d is zero.

Let's think about the recursion tree for this recurrence. There will be $1 + \log_b n$ levels. At each level, the number of subproblems will be multiplied by a, and so the number of subproblems at level i will be a^i. Each subproblem at level i is a problem of size (n/b^i). A subproblem of size n/b^i requires $(n/b^i)^c$ additional work and since there are a^i problems on level i, the total number of units of work on level i is

$$a^i(n/b^i)^c = n^c\left(\frac{a^i}{b^{ci}}\right) = n^c\left(\frac{a}{b^c}\right)^i. \tag{4.26}$$

Recall from Lemma 4.7 that the different cases for $c = 1$ were when the work per level was decreasing, constant, or increasing. The same analysis applies here. From our formula for work on level i, we see that the work per level is decreasing, constant, or increasing exactly when $(\frac{a}{b^c})^i$ is decreasing, constant, or increasing, respectively. These three cases depend on whether $(\frac{a}{b^c})$ less than one, equal to one, or greater than one, respectively. Now observe that

$$\begin{array}{rc} & (\frac{a}{b^c}) = 1 \\ \Leftrightarrow & a = b^c \\ \Leftrightarrow & \log_b a = c \log_b b \\ \Leftrightarrow & \log_b a = c. \end{array}$$

This shows us where the three cases in the statement of the theorem come from. Now we need to show the bound on $T(n)$ in the different cases. In the following paragraphs, we will use the facts (whose proof is a straightforward application of the definition of logarithms and rules of exponents) that for any x, y and z, each greater than 1, $x^{\log_y z} = z^{\log_y x}$ (see Corollary 4.8, Problem 5 at the end of the previous section, and Problem 3 at the end of this section) and that $\log_x y = \Theta(\log_2 y)$ (see Problem 4 at the end of the previous section).

In general, the total work done is computed by summing the expression for the work per level given in Equation 4.26 over all the levels, giving

$$\sum_{i=0}^{\log_b n} n^c \left(\frac{a}{b^c}\right)^i = n^c \sum_{i=0}^{\log_b n} \left(\frac{a}{b^c}\right)^i$$

In case 1, (part 1 in the statement of the theorem) this is n^c times a geometric series with a ratio of less than 1. Theorem 4.4 tells us that

$$n^c \sum_{i=0}^{\log_b n} \left(\frac{a}{b^c}\right)^i = \Theta(n^c).$$

Exercise 4.4-1 Prove Case 2 (part 2 of the statement) of the Master Theorem.

Exercise 4.4-2 Prove Case 3 (part 3 of the statement) of the Master Theorem.

In Case 2 we have that $\frac{a}{b^c} = 1$ and so

$$\begin{aligned} n^c \sum_{i=0}^{\log_b n} \left(\frac{a}{b^c}\right)^i &= n^c \sum_{i=0}^{\log_b n} 1^i \\ &= n^c(1 + \log_b n) \\ &= \Theta(n^c \log n). \end{aligned}$$

In Case 3, we have that $\frac{a}{b^c} > 1$. So in the series

$$\sum_{i=0}^{\log_b n} n^c \left(\frac{a}{b^c}\right)^i = n^c \sum_{i=0}^{\log_b n} \left(\frac{a}{b^c}\right)^i,$$

the largest term is the last one, so by Theorem 4.4,the sum is $\Theta\left(n^c \left(\frac{a}{b^c}\right)^{\log_b n}\right)$. But

$$\begin{aligned} n^c\left(\frac{a}{b^c}\right)^{\log_b n} &= n^c \cdot \frac{a^{\log_b n}}{(b^c)^{\log_b n}} \\ &= n^c \cdot \frac{n^{\log_b a}}{n^{\log_b b^c}} \\ &= n^c \cdot \frac{n^{\log_b a}}{n^c} \\ &= n^{\log_b a}. \end{aligned}$$

Thus the solution is $\Theta(n^{\log_b a})$. ■

We note that we may assume that a is a real number with $a > 1$ and give a somewhat similar proof (replacing the recursion tree with an iteration of the recurrence), but we do not give the details here.

Solving More General Kinds of Recurrences

Exercise 4.4-3 What can you say about the big-θ behavior of the solution to

$$T(n) = \begin{cases} 2T(n/3) + 4n^{3/2} & \text{if } n > 1 \\ d & \text{if } n = 1, \end{cases}$$

where n can be any nonnegative power of three?

Exercise 4.4-4 If $f(n) = n\sqrt{n+1}$, what can you say about the Big-Θ behavior of solutions to

$$S(n) = \begin{cases} 2S(n/3) + f(n) & \text{if } n > 1 \\ d & \text{if } n = 1, \end{cases}$$

where n can be any nonnegative power of three?

For Exercise 4.4-3, the work done at each level of the tree except for the bottom level will be four times the work done by the recurrence

$$T'(n) = \begin{cases} 2T'(n/3) + n^{3/2} & \text{if } n > 1 \\ d & \text{if } n = 1, \end{cases}$$

Thus the work done by T will be no more than four times the work done by T', but will be larger than the work done by T'. Therefore $T(n) = \Theta(T'(n))$. Thus by the master theorem, since $\log_3 2 < 1 < 3/2$, we have that $T(n) = \Theta(n^{3/2})$.

For Exercise 4.4-4, Since $n\sqrt{n+1} > n\sqrt{n} = n^{3/2}$ we have that $S(n)$ is at least as big as the solution to the recurrence

$$T'(n) = \begin{cases} 2T'(n/3) + n^{3/2} & \text{if } n > 1 \\ d & \text{if } n = 1, \end{cases}$$

where n can be any nonnegative power of three. But the solution to the recurrence for S will be no more than the solution to the recurrence in Exercise 4.4-3 for T, because $n\sqrt{n+1} \leq 4n^{3/2}$ for $n \geq 0$. Since $T(n) = \Theta(T'(n))$, then $S(n) = \Theta(T'(n))$ as well.

Extending the Master Theorem

As Exercise 4.4-3 and Exercise 4.4-4 suggest, there is a whole range of interesting recurrences that do not fit the master theorem but are closely related to recurrences that do. These recurrences have the same kind of behavior predicted by our original version of the Master Theorem, but the original version of the Master Theorem does not apply to them, just as it does not apply to the recurrences of Exercise 4.4-3 and Exercise 4.4-4.

We now state a second version of the Master Theorem that covers these cases. A still stronger version of the theorem may be found in *Introduction to Algorithms* by Cormen, et. al., but the version here captures much of the interesting behavior of recurrences that arise from the analysis of algorithms. The condition that $b \geq 2$ in this theorem can be replaced by $b > 1$, but then the base case depends on b and is not the case with $n = 1$.

Theorem 4.10 *Let a and b be positive real numbers with $a \geq 1$ and $b \geq 2$. Let $T(n)$ be defined for powers n of b by*

$$T(n) = \begin{cases} aT(n/b) + f(n) & \text{if } n > 1 \\ d & \text{if } n = 1. \end{cases}$$

Then

1. *if $f(n) = \Theta(x^c)$ where $\log_b a < c$, then $T(n) = \Theta(n^c) = \Theta(f(n))$.*
2. *if $f(n) = \Theta(n^c)$, where $\log_b a = c$, then $T(n) = \Theta(n^{\log_b a} \log_b n)$*
3. *if $f(n) = \Theta(n^c)$, where $\log_b a > c$, then $T(n) = \Theta(n^{\log_b a})$*

Proof: We construct a recursion tree or iterate the recurrence. Since we have assumed that $f(n) = \Theta(n^c)$, there are constants c_1 and c_2, independent of the level, so that the work at each level is between $c_1 n^c \left(\frac{a}{b^c}\right)^i$ and $c_2 n^c \left(\frac{a}{b^c}\right)^i$ so from this point on the proof is largely a translation of the original proof. ■

Exercise 4.4-5 What does the Master Theorem tell us about the solutions to the recurrence

$$T(n) = \begin{cases} 3T(n/2) + n\sqrt{n+1} & \text{if } n > 1 \\ 1 & \text{if } n = 1? \end{cases}$$

As we saw in our solution to Exercise 4.4-4 $x\sqrt{x+1} = \Theta(x^{3/2})$. Since $2^{3/2} = \sqrt{2^3} = \sqrt{8} < 3$, we have that $\log_2 3 > 3/2$. Then by conclusion 3 of version 2 of the Master Theorem, $T(n) = \Theta(n^{\log_2 3})$.

The remainder of this section is devoted to carefully analyzing divide and conquer recurrences in which n is not a power of b and $T(n/b)$ is replaced by $T(\lceil n/b \rceil)$. While the details are somewhat technical, the end result is that the big-Θ behavior of such recurrences is the same as the corresponding recurrences for functions defined on powers of b. The reader should be able to skip over the remainder of this section without loss of continuity.

More realistic recurrences (Optional)

So far, we have considered divide and conquer recurrences for functions $T(n)$ defined on integers n which are powers of b. In order to consider a more realistic recurrence in the master theorem, namely

$$T(n) = \begin{cases} aT(\lceil n/b \rceil) + n^c & \text{if } n > 1 \\ d & \text{if } n = 1, \end{cases}$$

or

$$T(n) = \begin{cases} aT(\lfloor n/b \rfloor) + n^c & \text{if } n > 1 \\ d & \text{if } n = 1, \end{cases}$$

or even

$$T(n) = \begin{cases} a'T(\lceil n/b \rceil) + (a-a')T(\lfloor n/b \rfloor) + n^c & \text{if } n > 1 \\ d & \text{if } n = 1, \end{cases}$$

it turns out to be easiest to first extend the domain for our recurrences to a much bigger set than the nonnegative integers, either the real or rational numbers, and then to work backwards.

For example, we can write a recurrence of the form

$$t(x) = \begin{cases} f(x)t(x/b) + g(x) & \text{if } x \geq b \\ k(x) & \text{if } 1 \leq x < b \end{cases}$$

for two (known) functions f and g defined on the real [or rational] numbers greater than 1 and one (known) function k defined on the real [or rational] numbers x with $1 \leq x < b$. Then so long as $b > 1$ it is possible to prove that there is a unique function t defined on the real [or rational] numbers greater than or equal to 1 that satisfies the recurrence. We use the lower case t in this situation as a signal that we are considering a recurrence whose domain is the real or rational numbers greater than or equal to 1.

Exercise 4.4-6 How would we compute $t(x)$ in the recurrence

$$t(x) = \begin{cases} 3t(x/2) + x^2 & \text{if } x \geq 2 \\ 5x & \text{if } 1 \leq x < 2 \end{cases}$$

if x were 7? How would we show that there is one and only one function t that satisfies the recurrence?

Exercise 4.4-7 Is it the case that there is one and only one solution to the recurrence

$$T(n) = \begin{cases} f(n)T(\lceil n/b \rceil) + g(n) & \text{if } n > 1 \\ k & \text{if } n = 1 \end{cases}$$

when f and g are (known) functions defined on the positive integers, and k and b are (known) constants with b an integer larger than or equal to 2?

To compute $t(7)$ in Exercise 4.4-6 we need to know $t(7/2)$. To compute $t(7/2)$, we need to know $t(7/4)$. Since $1 < 7/4 < 2$, we know that t(7/4) = 35/4. Then we may write

$$t(7/2) = 3 \cdot \frac{35}{4} + \frac{49}{4} = \frac{154}{4} = \frac{77}{2}.$$

Next we may write

$$\begin{aligned} t(7) &= 3t(7/2) + 7^2 \\ &= 3 \cdot \frac{77}{2} + 49 \\ &= \frac{329}{2}. \end{aligned}$$

Clearly we can compute $t(x)$ in this way for any x, though we are unlikely to enjoy the arithmetic. On the other hand suppose all we need to do is to show that there is a unique value of $t(x)$ determined by the recurrence, for all real numbers $x \geq 1$. If $1 \leq x < 2$, then $t(x) = 5x$, which uniquely determines $t(x)$. Given a number $x \geq 2$, there is a smallest integer i such that $x/2^i < 2$, and for this i, we have $1 \leq x/2^i$. We can now prove by induction on i that $t(x)$ is uniquely determined by the recurrence relation.

In Exercise 4.4-7 there is one and only one solution. Why? Clearly $T(1)$ is determined by the recurrence. Now assume inductively that $n > 1$ and that $T(m)$ is uniquely determined for positive integers $m < n$. We know that $n \geq 2$, so that $n/2 \leq n - 1$. Since $b \geq 2$, we know that $n/2 \geq n/b$, so that $n/b \leq n - 1$. Therefore $\lceil n/b \rceil < n$, so that we know by the inductive hypothesis that $T(\lceil n/b \rceil)$ is uniquely determined by the recurrence. Then by the recurrence,

$$T(n) = f(n)T\left(\left\lceil \frac{n}{b} \right\rceil\right) + g(n),$$

which uniquely determines $T(n)$. Thus by the principle of mathematical induction, $T(n)$ is determined for all positive integers n.

For every kind of recurrence we have dealt with, there is similarly one and only one solution. Because we know solutions exist, we don't find formulas for solutions to demonstrate that solutions exist, but rather to help us understand properties of the solutions. In this section and the last section, for example, we were interested in how fast the solutions grew as n grew large. This is why we were finding Big-O and Big-Θ bounds for our solutions.

Recurrences for general n (Optional)

We will now show how recurrences for arbitrary real numbers relate to recurrences involving floors and ceilings. We begin by showing that the conclusions of the Master Theorem apply to recurrences for arbitrary real numbers when we replace the real numbers by "nearby" powers of b.

Theorem 4.11 *Let a and b be positive real numbers with $b > 1$ and c and d be real numbers. Let $t(x)$ be the solution to the recurrence*

$$t(x) = \begin{cases} at(x/b) + x^c & \text{if } x \geq b \\ d & \text{if } 1 \leq x < b. \end{cases}$$

Let $T(n)$ be the solution to the recurrence

$$T(n) = \begin{cases} aT(n/b) + n^c & \text{if } n \geq 0 \\ d & \text{if } n = 1, \end{cases}$$

defined for n a nonnegative integer power of b. Let $m(x)$ be the largest integer power of b less than or equal to x. Then $t(x) = \Theta(T(m(x)))$

Proof: If we iterate (or, in the case that a is an integer, draw recursion trees for) the two recurrences, we can see that the results of the iterations are nearly identical. This means the solutions to the recurrences have the same big-Θ behavior. See the Appendix to this Section for details. ■

Removing Floors and Ceilings (Optional)

We have also pointed out that a more realistic Master Theorem would apply to recurrences of the form $T(n) = aT(\lfloor n/b \rfloor) + n^c$, or $T(n) = aT(\lceil n/b \rceil) + n^c$, or even $T(n) = a'T(\lceil n/b \rceil) + (a - a')T(\lfloor n/b \rfloor) + n^c$. For example, if we are applying mergesort to an array of size 101, we really break it into pieces, of size 50 and 51. Thus the recurrence we want is not really $T(n) = 2T(n/2) + n$, but rather $T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + n$.

We can show, however, that one can essentially "ignore" the floors and ceilings in typical divide-and-conquer recurrences. If we remove the floors and ceilings from a recurrence relation, we convert it from a recurrence relation defined on the integers to one defined on the rational numbers. However we have already seen that such recurrences are not difficult to handle.

The theorem below says that in recurrences covered by the master theorem, if we remove ceilings, our recurrences still have the same big-Θ bounds on their solutions. A similar proof shows that we may remove floors and still get the same big-Θ bounds. Without too much more work we can see that we can remove floors and ceilings simultaneously without changing the big-Θ bounds on our solutions. Since we may remove either floors or ceilings, that means that we may deal with recurrences of the form $T(n) = a'T(\lceil n/b \rceil) + (a - a')T(\lfloor n/b \rfloor) + n^c$. The condition that $b > 2$ can be replaced by $b > 1$, but the base case for the recurrence will depend on b.

Theorem 4.12 *Let a and b be positive real numbers with $b \geq 2$ and let c and d be real numbers. Let $T(n)$ be the function defined on the integers by the recurrence*

$$T(n) = \begin{cases} aT(\lceil n/b \rceil) + n^c & \text{if } n > 1 \\ d & n = 1, \end{cases}$$

and let $t(x)$ be the function on the real numbers defined by the recurrence

$$t(x) = \begin{cases} at(x/b) + x^c & \text{if } x \geq b \\ d & \text{if } 1 \leq x < b \end{cases}$$

Then $T(n) = \Theta(t(n))$. The same statement applies with ceilings replaced by floors.

Proof: As in the previous theorem, we can consider iterating the two recurrences. It is straightforward (though dealing with the notation is difficult) to show that for a given value of n, the iteration for computing $T(n)$ has at most two more levels than the iteration for computing $t(n)$. The work per level also has the same Big-Θ bounds at each level, and the work for the two additional levels of the iteration for $T(n)$ has the same Big-Θ bounds as the work at the bottom level of the recursion tree for $t(n)$. We give the details in the appendix at the end of this section. ■

Theorem 4.11 and Theorem 4.12 tell us that the Big-Θ behavior of solutions to our more realistic recurrences

$$T(n) = \begin{cases} aT(\lceil n/b \rceil) + n^c & \text{if } n > 1 \\ d & \text{n=1} \end{cases}$$

is determined by their Big-Θ behavior on powers of the base b.

Floors and ceilings in the stroger version of the Master Theorem (Optional)

We showed that, in our first version of the master theorem, we could ignore ceilings and assume our variables were powers of b. In fact we can ignore them in circumstances where the function telling us the "work" done at each level of our recursion tree is $\Theta(x^c)$ for some positive real number c. This lets us apply the second version of the master theorem to recurrences of the form $T(n) = aT(\lceil n/b\rceil) + f(n)$.

Theorem 4.13 *Theorems 4.11 and 4.12 apply to recurrences in which the x^c or n^c term is replaced by $f(x)$ or $f(n)$ for a function f with $f(x) = \Theta(x^c)$.*

Proof: We iterate the recurrences or construct recursion trees in the same way as in the proofs of the original theorems, and find that the condition $f(x) = \Theta(x^c)$ gives us enough information to again bound the solution above and below with multiples of the solution of the recurrence with x^c. The details are similar to those in the original proofs. ■

Appendix: Proofs of Theorems (Optional)

For convenience, we repeat the statements of the earlier theorems whose proofs we merely outlined.

Theorem 4.11 *Let a and b be positive real numbers with $b > 1$ and c and d be real numbers. Let $t(x)$ be the solution to the recurrence*

$$t(x) = \begin{cases} at(x/b) + x^c & \text{if } x \geq b \\ d & \text{if } 1 \leq x < b. \end{cases}$$

Let $T(n)$ be the solution to the recurrence

$$T(n) = \begin{cases} aT(n/b) + n^c & \text{if } n \geq 0 \\ d & \text{if } n = 1, \end{cases}$$

defined for n is a nonnegative integer power of b. Let $m(x)$ be the largest integer power of b less than or equal to x. Then $t(x) = \Theta(T(m(x)))$

Proof: By iterating each recursion 4 times (or using a four level recursion tree in the case that a is an integer), we see that

$$t(x) = a^4 t\Big(\frac{x}{b^4}\Big) + \Big(\frac{a}{b^c}\Big)^3 x^c + \Big(\frac{a}{b^c}\Big)^2 x^c + \frac{a}{b^c}x^c$$

and

$$T(n) = a^4 T\Big(\frac{n}{b^4}\Big) + \Big(\frac{a}{b^c}\Big)^3 n^c + \Big(\frac{a}{b^c}\Big)^2 n^c + \frac{a}{b^c}n^c$$

Thus, continuing until we have a solution, in both cases we get a solution that starts with a raised to an exponent that we will denote as either $e(x)$ or $e(n)$ when we want to distinguish between them and e when it is unnecessary to distinguish. The solution for t will be a^e times $t(x/b^e)$ plus x^c times a geometric series $\sum_{i=0}^{e-1}\left(\frac{a}{b^c}\right)^i$. The solution for T will be a^e times d plus n^c

times a geometric series $\sum_{i=0}^{e-1}\left(\frac{a}{b^c}\right)^i$ In both cases $t(x/b^e)$ (or $T(n/b^e)$) will be d. In both cases the geometric series will be $\Theta(1)$, $\Theta(e)$ or $\Theta\left(\frac{a}{b^c}\right)^e$, depending on whether $\frac{a}{b^c}$ is less than 1, equal to 1, or greater than one. Clearly $e(n) = \log_b n$. Since we must divide x by b an integer number greater than $\log_b x - 1$ times in order to get a value in the range from 1 to b, $e(x) = \lfloor \log_b x \rfloor$. Thus, if m is the largest integer power of b less than or equal to x, then $0 \le e(x) - e(m) < 1$. Let us use r to stand for the real number $\frac{a}{b^c}$. Then we have $r^0 \le r^{e(x)-e(m)} < r$, or $r^{e(m)} \le r^{e(x)} \le r \cdot r^{e(m)}$. Thus we have $r^{e(x)} = \Theta(r^{e(m)})$ Finally, $m^c \le x^c \le b^c m^c$, and so $x^c = \Theta(m^c)$. Therefore, every term of $t(x)$ is Θ of the corresponding term of $T(m)$. Further, there are only a fixed number of different constants involved in our Big-Θ bounds. Therefore since $t(x)$ is composed of sums and products of these terms, $t(x) = \Theta(T(m))$. ■

Theorem 4.12 *Let a and b be positive real numbers with $b \ge 2$ and let c and d be real numbers. Let $T(n)$ be the function defined on the integers by the recurrence*

$$T(n) = \begin{cases} aT(\lceil n/b \rceil) + n^c & \text{if } n \ge b \\ d & n = 1, \end{cases}$$

and let $t(x)$ be the function on the real numbers defined by the recurrence

$$t(x) = \begin{cases} at(x/b) + x^c & \text{if } x \ge b \\ d & \text{if } 1 \le x < b. \end{cases}$$

Then $T(n) = \Theta(t(n))$.

Proof: As in the previous proof, we can iterate both recurrences. Let us compare what the results will be of iterating the recurrence for $t(n)$ and the recurrence for $T(n)$ the same number of times. Note that

$$\begin{aligned} \lceil n/b \rceil &< n/b + 1 \\ \lceil \lceil n/b \rceil /b \rceil < \lceil n/b^2 + 1/b \rceil &< n/b^2 + 1/b + 1 \\ \lceil \lceil \lceil n/b \rceil /b \rceil /b \rceil < \lceil n/b^3 + 1/b^2 + 1/b \rceil &< n/b^3 + 1/b^2 + 1/b + 1 \end{aligned}$$

This suggests that if we define $n_0 = n$, and $n_i = \lceil n_{i-1}/b \rceil$, then, using the fact that $b \ge 2$, it is straightforward to prove by induction, or with the formula for the sum of a geometric series, that $n_i < n/b^i + 2$. The number n_i is the argument of T in the ith iteration of the recurrence for T. We have just seen it differs from the argument of t in the ith iteration of t by at most 2. In particular, we might have to iterate the recurrence for T twice more than we iterate the recurrence for t to reach the base case. When we iterate the recurrence for t, we get the same solution we got in the previous theorem, with n substituted for x. When we iterate the recurrence for T, we get for some integer j that

$$T(n) = a^j d + \sum_{i=0}^{j-1} a^i n_i^c,$$

with $\frac{n}{b^i} \le n_i \le \frac{n}{b^i} + 2$. But, so long as $n/b^i \ge 2$, we have $n/b^i + 2 \le n/b^{i-1}$. Since the number of iterations of T is at most two more than the number of iterations of t, and since the number of iterations of t is $\lfloor \log_b n \rfloor$, we have that j is at most $\lfloor \log_b n \rfloor + 2$. Therefore all but perhaps the last three values of n_i are less than or equal to n/b^{i-1}, and these last three values are at most b^2, b, and 1. Putting all these bounds together and using $n_0 = n$ gives us

$$\begin{aligned}\sum_{i=0}^{j-1} a^i \left(\frac{n}{b^i}\right)^c &\le \sum_{i=0}^{j-1} a^i n_i^c \\ &\le n^c + \sum_{i=1}^{j-4} a^i \left(\frac{n}{b^{i-1}}\right)^c + a^{j-2}(b^2)^c + a^{j-1}b^c + a^j 1^c,\end{aligned}$$

or

$$\begin{aligned}\sum_{i=0}^{j-1} a^i \left(\frac{n}{b^i}\right)^c &\le \sum_{i=0}^{j-1} a^i n_i^c \\ &\le n^c + b\sum_{i=1}^{j-4} a^i \left(\frac{n}{b^i}\right)^c + a^{j-2}\left(\frac{b^j}{b^{j-2}}\right)^c + a^{j-1}\left(\frac{b^j}{b^{j-1}}\right)^c + a^j\left(\frac{b^j}{b^j}\right)^c.\end{aligned}$$

As we shall see momentarily these last three "extra" terms and the b in front of the summation sign do not change the Big-Θ behavior of the right-hand side.

As in the proof of the master theorem, the Big-Θ behavior of the left hand side depends on whether a/b^c is less than 1, in which case it is $\Theta(n^c)$, equal to 1, in which case it is $\Theta(n^c \log_b n)$, or greater than one in which case it is $\Theta(n^{\log_b a})$. But this is exactly the Big-Θ behavior of the right-hand side, because $n < b^j < nb^2$, so $b^j = \Theta(n)$, which means that $\left(\frac{b^j}{b^i}\right)^c = \Theta\left(\left(\frac{n}{b^i}\right)^c\right)$, and the b in front of the summation sign does not change its Big-Θ behavior. Adding $a^j d$ to the middle term of the inequality to get $T(n)$ does not change this behavior. But this modified middle term is exactly $T(n)$. Since the left and right hand sides have the same big-Θ behavior as $t(n)$, we have T(n) = $\Theta(t(n))$. ■

Important Concepts, Formulas, and Theorems

1. *Master Theorem, simplified version.* The simplified version of the *Master Theorem* states: Let a be an integer greater than or equal to 1 and b be a real number greater than 1. Let c be a positive real number and d a nonnegative real number. Given a recurrence of the form

$$T(n) = \begin{cases} aT(n/b) + n^c & \text{if } n > 1 \\ d & \text{if } n = 1 \end{cases}$$

then for n a power of b,

   (a) if $\log_b a < c$, $T(n) = \Theta(n^c)$,

   (b) if $\log_b a = c$, $T(n) = \Theta(n^c \log n)$,

   (c) if $\log_b a > c$, $T(n) = \Theta(n^{\log_b a})$.

2. *Properties of Logarithms.* For any x, y and z, each greater than 1, $x^{\log_y z} = z^{\log_y x}$. Also, $\log_x y = \Theta(\log_2 y)$.

3. *Master Theorem, More General Version.* Let a and b be positive real numbers with $a \geq 1$ and $b \geq 2$. Let $T(n)$ be defined for powers n of b by

$$T(n) = \begin{cases} aT(n/b) + f(n) & \text{if } n > 1 \\ d & \text{if } n = 1 \end{cases}$$

   Then

   (a) if $f(n) = \Theta(n^c)$ where $\log_b a < c$, then $T(n) = \Theta(n^c) = \Theta(f(n))$.

   (b) if $f(n) = \Theta(n^c)$, where $\log_b a = c$, then $T(n) = \Theta(n^{\log_b a} \log_b n)$

   (c) if $f(n) = \Theta(n^c)$, where $\log_b a > c$, then $T(n) = \Theta(n^{\log_b a})$.

   A similar result with a base case that depends on b holds when $1 < b < 2$.

4. *Important Recurrences have Unique Solutions. (Optional.)* The recurrence

$$T(n) = \begin{cases} f(n)T(\lceil n/b \rceil) + g(n) & \text{if } n > 1 \\ k & \text{if } n = 1 \end{cases}$$

   when f and g are (known) functions defined on the positive integers, and k and b are (known) constants with b an integer larger than 2 has a unique solution.

5. *Recurrences Defined on the Positive Real Numbers and Recurrences Defined on the Positive Integers. (Optional.)* Let a and b be positive real numbers with $b > 1$ and c and d be real numbers. Let $t(x)$ be the solution to the recurrence

$$t(x) = \begin{cases} at(x/b) + x^c & \text{if } x \geq b \\ d & \text{if } 1 \leq x < b. \end{cases}$$

   Let $T(n)$ be the solution to the recurrence

$$T(n) = \begin{cases} aT(n/b) + n^c & \text{if } n \geq 0 \\ d & \text{if } n = 1, \end{cases}$$

   where n is a nonnegative integer power of b. Let $m(x)$ be the largest integer power of b less than or equal to x. Then $t(x) = \Theta(T(m(x)))$

6. *Removing Floors and Ceilings from Recurrences. (Optional.)* Let a and b be positive real numbers with $b \geq 2$ and let c and d be real numbers. Let $T(n)$ be the function defined on the integers by the recurrence

$$T(n) = \begin{cases} aT(\lceil n/b \rceil) + n^c & \text{if } n > 1 \\ d & n = 1 \end{cases},$$

   and let $t(x)$ be the function on the real numbers defined by the recurrence

$$t(x) = \begin{cases} at(x/b) + x^c & \text{if } x \geq b \\ d & \text{if } 1 \leq x < b \end{cases}.$$

   Then $T(n) = \Theta(t(n))$. The same statement applies with ceilings replaced by floors.

7. *Extending 5 and 6 (Optional.)* In the theorems summarized in 5 and 6 the n^c or x^c term may be replaced by a function f with $f(x) = \Theta(x^c)$.

8. *Solutions to Realistic Recurrences. (Optional.)* The theorems summarized in 5, 6, and 7 tell us that the Big-Θ behavior of solutions to our more realistic recurrences

$$T(n) = \begin{cases} aT(\lceil n/b \rceil) + f(n) & \text{if } n > 1 \\ d & \text{n=1,} \end{cases}$$

where $f(n) = \Theta(n^c)$, is determined by their Big-Θ behavior on powers of the base b and with $f(n) = n^c$.

Problems

1. Use the master theorem to give Big-Θ bounds on the solutions to the following recurrences. For all of these, assume that $T(1) = 1$ and n is a power of the appropriate integer.

   (a) $T(n) = 8T(n/2) + n$

   (b) $T(n) = 8T(n/2) + n^3$

   (c) $T(n) = 3T(n/2) + n$

   (d) $T(n) = T(n/4) + 1$

   (e) $T(n) = 3T(n/3) + n^2$

2. Extend the proof of the Master Theorem, Theorem 4.9 to the case $T(1) = d$.

3. Show that for any x, y and z, each greater than 1, $x^{\log_y z} = z^{\log_y x}$.

4. Show that for each real number $x \geq 0$ there is one and only one value of $t(x)$ given by the recurrence

$$t(x) = \begin{cases} 7xt(x-1) + 1 & \text{if } x \geq 1 \\ 1 & \text{if } 0 \leq x < 1. \end{cases}$$

5. Show that for each real number $x \geq 1$ there is one and only one value of $t(x)$ given by the recurrence

$$t(x) = \begin{cases} 3xT(x/2) + x^2 & \text{if } x \geq 2 \\ 1 & \text{if } 1 \leq x < 2 \end{cases}.$$

6. How many solutions are there to the recurrence

$$T(n) = \begin{cases} f(n)T(\lceil n/b \rceil) + g(n) & \text{if } n > 1 \\ k & \text{if } n = 1 \end{cases}$$

if $b < 2$? If $b = 10/9$, by what would we have to replace the condition that $T(n) = k$ if $n = 1$ in order to get a unique solution?

7. Give a big-Θ bound on the solution to the recurrence

$$T(n) = \begin{cases} 3T(\lceil n/2 \rceil) + \sqrt{n+3} & \text{if } n > 1 \\ d & \text{if } n = 1. \end{cases}$$

8. Give a big-Θ bound on the solution to the recurrence

$$T(n) = \begin{cases} 3T(\lceil n/2 \rceil) + \sqrt{n^3+3} & \text{if } n > 1 \\ d & \text{if } n = 1. \end{cases}$$

9. Give a big-Θ bound on the solution to the recurrence

$$T(n) = \begin{cases} 3T(\lceil n/2 \rceil) + \sqrt{n^4+3} & \text{if } n > 1 \\ d & \text{if } n = 1. \end{cases}$$

10. Give a big-Θ bound on the solution to the recurrence

$$T(n) = \begin{cases} 2T(\lceil n/2 \rceil) + \sqrt{n^2+3} & \text{if } n > 1 \\ d & \text{if } n = 1. \end{cases}$$

4.5 More general kinds of recurrences

Recurrence Inequalities

The recurrences we have been working with are really idealized versions of what we know about the problems we are working on. For example, in merge-sort on a list of n items, we say we divide the list into two parts of equal size, sort each part, and then merge the two sorted parts. The time it takes to do this is the time it takes to divide the list into two parts plus the time it takes to sort each part, plus the time it takes to merge the two sorted lists. We don't specify how we are dividing the list, or how we are doing the merging. (We assume the sorting is done by applying the same method to the smaller lists, unless they have size 1, in which case we do nothing.) What we do know is that any sensible way of dividing the list into two parts takes no more than some constant multiple of n time units (and might take no more than constant time if we do it by leaving the list in place and manipulating pointers) and that any sensible algorithm for merging two lists will take no more than some (other) constant multiple of n time units. Thus we know that if $T(n)$ is the amount of time it takes to apply merge sort to n data items, then there is a constant c (the sum of the two constant multiples we mentioned) such that

$$T(n) \leq 2T(n/2) + cn. \tag{4.27}$$

Thus real world problems often lead us to *recurrence inequalities* rather than recurrence equations. These are inequalities that state that $T(n)$ is less than or equal to some expression involving values of $T(m)$ for $m < n$. (We could also include inequalities with a greater than or equal to sign, but they do not arise in the applications we are studying.) A *solution* to a recurrence inequality is a function T that satisfies the inequality. For simplicity we will expand what we mean by the word recurrence to include either recurrence inequalities or recurrence equations.

In Recurrence 4.27 we are implicitly assuming that T is defined only on positive integer values and, since we said we divided the list into two equal parts each time, our analysis only makes sense if we assume that n is a power of 2.

Note that there are actually infinitely many solutions to Recurrence 4.27. (For example for any $c' < c$, the unique solution to

$$T(n) = \begin{cases} 2T(n/2) + c'n & \text{if } n \geq 2 \\ k & \text{if } n = 1 \end{cases} \tag{4.28}$$

satisfies Inequality 4.27 for any constant k.) The idea that Recurrence 4.27 has infinitely many solutions, while Recurrence 4.28 has exactly one solution is analogous to the idea that $x - 3 \leq 0$ has infinitely many solutions while $x - 3 = 0$ has one solution. Later in this section we shall see how to show that all the solutions to Recurrence 4.27 satisfy $T(n) = O(n \log_2 n)$. In other words, no matter how we sensibly implement merge sort, we have a $O(n \log_2 n)$ time bound on how long the merge sort process takes.

Exercise 4.5-1 Carefully prove by induction that for any function T defined on the non-negative powers of 2, if

$$T(n) \leq 2T(n/2) + cn$$

for some constant c, then $T(n) = O(n \log n)$.

A Wrinkle with induction

We can analyze recurrence inequalities via a recursion tree. The process is virtually identical to our previous use of recursion trees. We must, however, keep in mind that on each level, we are really computing an upper bound on the work done on that level. We can also use a variant of the method we used a few sections ago, guessing an upper bound and verifying by induction. We use this method for the recurrence in Exercise 4.5-1. Here we wish to show that $T(n) = O(n \log n)$. From the definition of Big-O, we can see that we wish to show that $T(n) \le kn \log n$ for some positive constant k (so long as n is larger than some value n_0).

We are going to do something you may find rather curious. We will consider the possibility that we have a value of k for which the inequality holds. Then in analyzing the consequences of this possibility, we will discover that there are assumptions that we need to make about k in order for such a k to exist. What we will really be doing is experimenting to see how we will need to choose k to make an inductive proof work.

We are given that $T(n) \le 2T(n/2) + cn$ for all positive integers n that are powers of 2. We want to prove there is another positive real number $k > 0$ and an $n_0 > 0$ such that for $n > n_0$, $T(n) \le kn \log n$. We cannot expect to have the inequality $T(n) \le kn \log n$ hold for $n = 1$, because $\log 1 = 0$. To have $T(2) \le k \cdot 2 \log 2 = k \cdot 2$, we must choose $k \ge \frac{T(2)}{2}$. This is the first assumption we must make about k. Our inductive hypothesis will be that if n is a power of 2 and m is a power of 2 with $2 < m < n$ then $T(m) \le km \log m$. Now $n/2 < n$, and since n is a power of 2 greater than 2, we have that $n/2 \ge 2$, so $(n/2) \log n/2 \ge 2$. By the inductive hypothesis, $T(n/2) \le k(n/2) \log n/2$. But then

$$\begin{aligned}
T(n) \le 2T(n/2) + cn &\le 2k\frac{n}{2}\log\frac{n}{2} + cn && (4.29)\\
&= kn\log\frac{n}{2} + cn && (4.30)\\
&= kn\log n - kn\log 2 + cn && (4.31)\\
&= kn\log n - kn + cn. && (4.32)
\end{aligned}$$

Recall that we are trying to show that $T(n) \le kn \log n$. But that is not quite what Line 4.32 tells us. This shows that we need to make another assumption about k, namely that $-kn+cn \le 0$, or $k \ge c$. Then if both our assumptions about k are satisfied, we will have $T(n) < kn \log n$, and we can conclude by the principle of mathematical induction that for all $n > 1$ (so our n_0 is 2), $T(n) \le kn \log n$, so that $T(n) = O(n \log n)$.

A full inductive proof that $T(n) = O(n \log n)$ is actually embedded in the discussion above, but since it might not appear to everyone to be a proof, below we will summarize our observations in a more traditional looking proof. However you should be aware that some authors and teachers prefer to write their proofs in a style that shows why we make the choices about k that we do, and so you should learn how to to read discussions like the one above as proofs.

We want to show that if $T(n) \le T(n/2) + cn$, then $T(n) = O(n \log n)$. We are given a real number $c > 0$ such that $T(n) \le 2T(n/2) + cn$ for all $n > 1$. Choose k to be larger than or equal to $\frac{T(2)}{2}$ and larger than or equal to c. Then

$$T(2) \le k \cdot 2 \log 2$$

because $k \geq T(n_0)/2$ and $\log 2 = 1$. Now assume that $n > 2$ and assume that for m with $2 \leq m < n$, we have $T(m) \leq km \log m$. Since n is a power of 2, we have $n \geq 4$, so that $n/2$ is an m with $2 \leq m < n$. Thus, by the inductive hypothesis,

$$T\left(\frac{n}{2}\right) \leq k\frac{n}{2}\log\frac{n}{2.}$$

Then by the recurrence,

$$\begin{aligned} T(n) &\leq 2k\frac{n}{2}\log\frac{n}{2} + cn \\ &= kn(\log n - 1) + cn \\ &= kn\log n + cn - kn \\ &\leq kn\log n, \end{aligned}$$

since $k \geq c$. Thus by the principle of mathematical induction, $T(n) \leq kn \log n$ for all $n > 2$, and therefore $T(n) = O(n \log n)$.

There are three things to note about this proof. First without the preceding discussion, the choice of k seems arbitrary. Second, without the preceding discussion, the implicit choice of 2 for the n_0 in the big-O statement also seems arbitrary. Third, the constant k is chosen in terms of the previous constant c. Since c was given to us by the recurrence, it may be used in choosing the constant we use to prove a Big-O statement about solutions to the recurrence. If you compare the formal proof we just gave with the informal discussion that preceded it, you will find each step of the formal proof actually corresponds to something we said in the informal discussion. Since the informal discussion explained why we were making the choices we did, it is natural that some people prefer the informal explanation to the formal proof.

Further Wrinkles in Induction Proofs

Exercise 4.5-2 Suppose that c is a real number greater than zero. Show by induction that any solution $T(n)$ to the recurrence

$$T(n) \leq T(n/3) + cn$$

with n restricted to integer powers of 3 has $T(n) = O(n)$.

Exercise 4.5-3 Suppose that c is a real number greater than zero. Show by induction that any solution $T(n)$ to the recurrence

$$T(n) \leq 4T(n/2) + cn$$

with n restricted to integer powers of 2 has $T(n) = O(n^2)$.

In Exercise 4.5-2 we are given a constant c such that $T(n) \leq T(n/3) + cn$ if $n > 1$. Since we want to show that $T(n) = O(n)$, we want to find two more constants n_0 and k such that $T(n) \leq kn$ whenever $n > n_0$.

We will choose $n_0 = 1$ here. (This was not an arbitrary choice; it is based on observing that $T(1) \leq kn$ is not an impossible condition to satisfy when $n = 1$.) In order to have $T(n) \leq kn$ for

$n = 1$, we must assume $k \geq T(1)$. Now assuming inductively that $T(m) \leq km$ when $1 \leq m < n$ we can write

$$\begin{aligned} T(n) &\leq T(n/3) + cn \\ &\leq k(n/3) + cn \\ &= kn + \left(c - \frac{2k}{3}\right) n \end{aligned}$$

Thus, as long as $c - \frac{2k}{3} \leq 0$, i.e. $k \geq \frac{3}{2}c$, we may conclude by mathematical induction that $T(n) \leq kn$ for all $n \geq 1$. Again, the elements of an inductive proof are in the preceding discussion. Again you should try to learn how to read the argument we just finished as a valid inductive proof. However, we will now present something that looks more like an inductive proof.

We choose k to be the maximum of $T(1)$ and $3c/2$ and we choose $n_0 = 1$. To prove by induction that $T(x) \leq kx$ we begin by observing that $T(1) \leq k \cdot 1$. Next we assume that $n > 1$ and assume inductively that for m with $1 \leq m < n$ we have $T(m) \leq km$. Now we may write

$$T(n) \leq T(n/3) + cn \leq kn/3 + cn = kn + (c - 2k/3)n \leq kn,$$

because we chose k to be at least as large as $3c/2$, making $c - 2k/3$ negative or zero. Thus by the principle of mathematical induction we have $T(n) \leq kn$ for all $n \geq 1$ and so $T(n) = O(n)$.

Now let's analyze Exercise 4.5-3. We won't dot all the i's and cross all the t's here because there is only one major difference between this exercise and the previous one. We wish to prove there are an n_0 and a k such that $T(n) \leq kn^2$ for $n > n_0$. Assuming that we have chosen n_0 and k so that the base case holds, we can bound $T(n)$ inductively by assuming that $T(m) \leq km^2$ for $m < n$ and reasoning as follows:

$$\begin{aligned} T(n) &\leq 4T\left(\frac{n}{2}\right) + cn \\ &\leq 4\left(k\left(\frac{n}{2}\right)^2\right) + cn \\ &= 4\left(\frac{kn^2}{4}\right) + cn \\ &= kn^2 + cn. \end{aligned}$$

To proceed as before, we would like to choose a value of k so that $cn \leq 0$. But we see that we have a problem because both c and n are always positive! What went wrong? We have a statement that we know is true, and we have a proof method (induction) that worked nicely for similar problems.

The usual way to describe the problem we are facing is that, while the statement is true, it is too weak to be proved by induction. To have a chance of making the inductive proof work, we will have to make an inductive hypothesis that puts some sort of negative quantity, say a term like $-kn$, into the last line of our display above. Let's see if we can prove something that is actually stronger than we were originally trying to prove, namely that for some positive constants k_1 and k_2, $T(n) \leq k_1n^2 - k_2n$. Now proceeding as before, we get

$$T(n) \leq 4T(n/2) + cn$$

$$\begin{aligned}
&\le 4\left(k_1\left(\frac{n}{2}\right)^2 - k_2\left(\frac{n}{2}\right)\right) + cn \\
&= 4\left(\frac{k_1 n^2}{4} - k_2\left(\frac{n}{2}\right)\right) + cn \\
&= k_1 n^2 - 2k_2 n + cn \\
&= k_1 n^2 - k_2 n + (c - k_2)n.
\end{aligned}$$

Now we have to make $(c - k_2)n \le 0$ for the last line to be at most $k_1 n^2 - k_2 n$, and so we just choose $k_2 \ge c$ (and greater than whatever we need in order to make a base case work). Since $T(n) \le k_1 n^2 - k_2 n$ for some constants k_1 and k_2, then $T(n) = O(n^2)$.

At first glance, this approach seems paradoxical: why is it easier to prove a stronger statement than it is to prove a weaker one? This phenomenon happens often in induction: a stronger statement is often easier to prove than a weaker one. Think carefully about an inductive proof where you have assumed that a bound holds for values smaller than n and you are trying to prove a statement for n. You use the bound you have assumed for smaller values to help prove the bound for n. Thus if the bound you used for smaller values is actually weak, then that is hindering you in proving the bound for n. In other words when you want to prove something about $p(n)$ you are using $p(1) \wedge \ldots \wedge p(n-1)$. Thus if these are stronger, they will be of greater help in proving $p(n)$. In the case above, the problem was that the statements, $p(1), \ldots, p(n-1)$ were too weak, and thus we were not able to prove $p(n)$. By using a stronger $p(1), \ldots, p(n-1)$, however, we were able to prove a stronger $p(n)$, one that implied the original $p(n)$ we wanted. When we give an induction proof in this way, we say that we are using a *stronger inductive hypothesis.*

Dealing with functions other than n^c

Our statement of the Master Theorem involved a recursive term plus an added term that was $\Theta(n^c)$. Sometimes algorithmic problems lead us to consider other kinds of functions. The most common such is example is when that added function involves logarithms. For example, consider the recurrence:

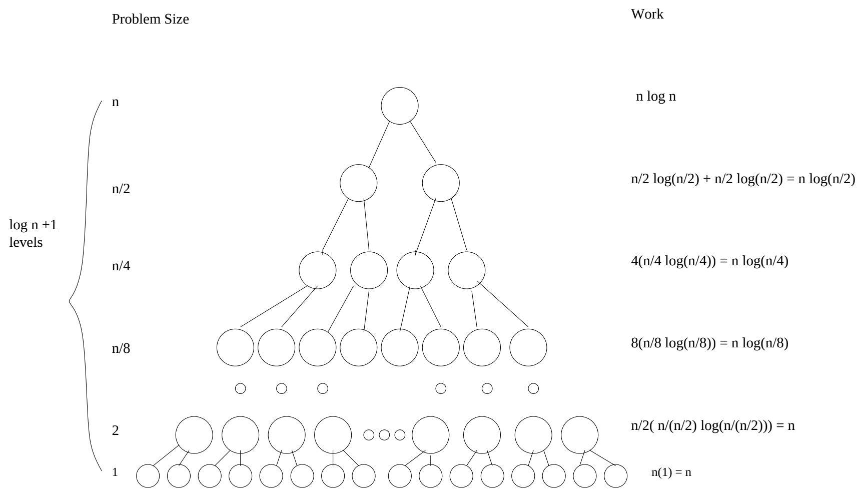
$$T(n) = \begin{cases} 2T(n/2) + n\log n & \text{if } n > 1 \\ 1 & \text{if } n = 1, \end{cases}$$

where n is a power of 2. Just as before, we can draw a recursion tree; the whole methodology works, but our sums may be a little more complicated. The tree for this recurrence is shown in Figure 4.8.

This is similar to the tree for $T(n) = 2T(n/2) + n$, except that the work on level i is $n\log\left(\frac{n}{2^i}\right)$ for $i \ge 2$, and, for the bottom level, it is n, the number of subproblems, times 1. Thus if we sum the work per level we get

$$\begin{aligned}
\sum_{i=0}^{\log n - 1} n\log\left(\frac{n}{2^i}\right) + n &= n\sum_{i=0}^{\log n - 1} \log\left(\frac{n}{2^i}\right) + n \\
&= n\sum_{i=0}^{\log n - 1} (\log n - \log 2^i) + n
\end{aligned}$$

Figure 4.8: The recursion tree for $T(n) = 2T(n/2) + n\log n$ if $n > 1$ and $T(1) = 1$.


Problem Size
Work
n
n log n
n/2
n/2 log(n/2) + n/2 log(n/2) = n log(n/2)
log n +1 levels
n/4
4(n/4 log(n/4)) = n log(n/4)
n/8
8(n/8 log(n/8)) = n log(n/8)
2
n/2(n/(n/2) log(n/(n/2))) = n
1
n(1) = n

$$\begin{aligned}
&= n\left(\sum_{i=0}^{\log n-1}\log n - \sum_{i=0}^{\log n-1} i\right) + n \\
&= n\left((\log n)(\log n) - \frac{(\log n)(\log n - 1)}{2}\right) + n \\
&= O(n\log^2 n)\ .
\end{aligned}$$

A bit of mental arithmetic in the second last line of our equations shows that the $\log^2 n$ will not cancel out, so our solution is in fact $\Theta(n\log^2 n)$.

Exercise 4.5-4 Find the best big-O bound you can on the solution to the recurrence

$$T(n) = \begin{cases} T(n/2) + n\log n & \text{if } n > 1 \\ 1 & \text{if } n = 1, \end{cases}$$

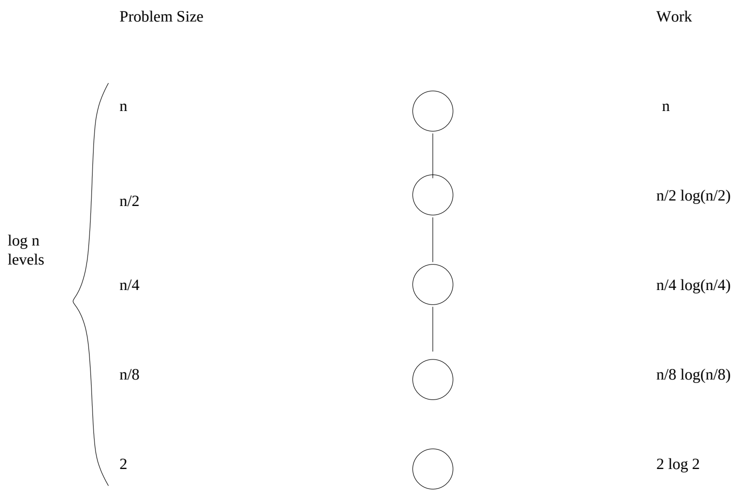
assuming n is a power of 2. Is this bound a big-Θ bound?

The tree for this recurrence is in Figure 4.9

Notice that the work done at the bottom nodes of the tree is determined by the statement $T(1) = 1$ in our recurrence; it is not $1\log 1$. Summing the work, we get

$$\begin{aligned}
1 + \sum_{i=0}^{\log n-1}\frac{n}{2^i}\log\left(\frac{n}{2^i}\right) &= 1 + n\left(\sum_{i=0}^{\log n-1}\frac{1}{2^i}(\log n - \log 2^i)\right) \\
&= 1 + n\left(\sum_{i=0}^{\log n-1}\left(\frac{1}{2}\right)^i(\log(n) - i)\right) \\
&\leq 1 + n\left(\log n\sum_{i=0}^{\log n-1}\left(\frac{1}{2}\right)^i\right)
\end{aligned}$$

Figure 4.9: The recursion tree for the recurrence $T(n) = T(n/2) + n\log n$ if $n > 1$ and $T(1) = 1$.



$$\begin{aligned} &\leq 1 + n(\log n)(2) \\ &= O(n\log n). \end{aligned}$$

Note that. the largest term in the sum in our second line of equations is $\log(n)$, and none of the terms in the sum are negative. This means that n times the sum is at least $n\log n$. Therefore, we have $T(n) = \Theta(n\log n)$.

Removing Ceilings and Using Powers of b. (Optional)

We showed that in our versions of the master theorem, we could ignore ceilings and assume our variables were powers of b. It might appear that the two theorems we used do not apply to the more general functions we have studied in this section any more than the master theorem does. However, they actually only depend on properties of the powers n^c and not the three different kinds of cases, so it turns out we can extend them.

Notice that $(xb)^c = b^c x^c$, and this proportionality holds for all values of x with constant of proportionality b^c. Putting this just a bit less precisely, we can write $(xb)^c = O(x^c)$. This suggests that we might be able to obtain Big-Θ bounds on $T(n)$ when T satisfies a recurrence of the form

$$T(n) = aT(n/b) + f(n)$$

with $f(nb) = \Theta(f(n))$, and we might be able to obtain Big-O bounds on T when T satisfies a recurrence of the form

$$T(n) \leq aT(n/b) + f(n)$$

with $f(nb) = O(f(n))$. But are these conditions satisfied by any functions of practical interest? Yes. For example if $f(x) = \log(x)$, then

$$f(bx) = \log(b) + \log(x) = \Theta(\log(x)).$$

Exercise 4.5-5 Show that if $f(x) = x^2\log x$, then $f(bx) = \Theta(f(x))$.

Exercise 4.5-6 If $f(x) = 3^x$ and $b = 2$, is $f(bx) = \Theta(f(x))$? Is $f(b(x)) = O(f(x))$?

For Exercise 4.5-5 if $f(x) = x^2 \log x$, then

$$f(bx) = (bx)^2 \log bx = b^2x^2(\log b + \log x) = \Theta(x^2 \log x).$$

However, if $f(x) = 3^x$, then

$$f(2x) = 3^{2x} = (3^x)^2 = 3^x \cdot 3^x,$$

and there is no way that this can be less than or equal to a constant multiple of 3^x, so it is neither $\Theta(3^x)$ nor $O(3^x)$. Our exercises suggest the kinds of functions that satisfy the condition $f(bx) = O(f(x))$ might include at least some of the kinds of functions of x which arise in the study of algorithms. They certainly include the power functions and thus polynomial functions and root functions, or functions bounded by such functions.

There was one other property of power functions n^c that we used implicitly in our discussions of removing floors and ceilings and assuming our variables were powers of b. Namely, if $x > y$ (and $c \geq 0$) then $x^c \geq y^c$. A function f from the real numbers to the real numbers is called *(weakly) increasing* if whenever $x > y$, then $f(x) \geq f(y)$. Functions like $f(x) = \log x$ and $f(x) = x \log x$ are increasing functions. On the other hand, the function defined by

$$f(x) = \begin{cases} x & \text{if } x \text{ is a power of } b \\ x^2 & \text{otherwise} \end{cases}$$

is not increasing even though it does satisfy the condition $f(bx) = \Theta(f(x))$.

Theorem 4.14 *Theorems 4.11 and 4.12 apply to recurrences in which the x^c term is replaced by an increasing function f for which $f(bx) = \Theta(f(x))$.*

Proof: We iterate the recurrences in the same way as in the proofs of the original theorems, and find that the condition $f(bx) = \Theta(f(x))$ applied to an increasing function gives us enough information to again bound the solution to one kind of recurrence above and below with a multiple of the solution of the other kind. The details are similar to those in the original proofs so we omit them. ■

In fact there are versions of Theorems 4.11 and 4.12 for recurrence inequalities also. The proofs involve a similar analysis of iterated recurrences or recursion trees, and so we omit them.

Theorem 4.15 *Let a and b be positive real numbers with $b > 2$ and let $f : R^+ \to R^+$ be an increasing function such that $f(bx) = O(f(x))$. Then every solution $t(x)$ to the recurrence*

$$t(x) \leq \begin{cases} at(x/b) + f(x) & \text{if } x \geq b \\ c & \text{if } 1 \leq x < b, \end{cases}$$

where a, b, and c are constants, satisfies $t(x) = O(h(x))$ if and only if every solution $T(n)$ to the recurrence

$$T(n) \leq \begin{cases} aT(n/b) + f(n) & \text{if } n > 1 \\ d & \text{if } n = 1, \end{cases}$$

where n is restricted to powers of b, satisfies $T(n) = O(h(n))$.

Theorem 4.16 *Let a and b be positive real numbers with $b \geq 2$ and let $f : R^+ \to R^+$ be an increasing function such that $f(bx) = O(f(x))$. Then every solution $T(n)$ to the recurrence*

$$T(n) \leq \begin{cases} at(\lceil n/b \rceil) + f(n) & \text{if } n > 1 \\ d & \text{if } n = 1, \end{cases}$$

satisfies $T(n) = O(h(n))$ if and only if every solution $t(x)$ to the recurrence

$$t(x) \leq \begin{cases} aT(x/b) + f(x) & \text{if } x \geq b \\ d & \text{if } 1 \leq x < b, \end{cases}$$

satisfies $t(x) = O(h(x))$.

Important Concepts, Formulas, and Theorems

1. *Recurrence Inequality.* Recurrence inequalities are inequalities that state that $T(n)$ is less than or equal to some expression involving values of $T(m)$ for $m < n$. A *solution* to a recurrence inequality is a function T that satisfies the inequality.

2. *Recursion Trees for Recurrence Inequalities.* We can analyze recurrence inequalities via a recursion tree. The process is virtually identical to our previous use of recursion trees. We must, however, keep in mind that on each level, we are really computing an upper bound on the work done on that level.

3. *Discovering Necessary Assumptions for an Inductive Proof.* If we are trying to prove a statement that there is a value k such that an inequality of the form $f(n) \leq kg(n)$ or some other statement that involves the parameter k is true, we may start an inductive proof without knowing a value for k and determine conditions on k by assumptions that we need to make in order for the inductive proof to work. When written properly, such an explanation is actually a valid proof.

4. *Making a Stronger Inductive Hypothesis.* If we are trying to prove by induction a statement of the form $p(n) \Rightarrow q(n)$ and we have a statement $s(n)$ such that $s(n) \Rightarrow q(n)$, it is sometimes useful to try to prove the statement $p(n) \Rightarrow s(n)$. This process is known as proving a *stronger* statement or making an *stronger* inductive hypothesis. It sometimes works because it gives us an inductive hypothesis which suffices to prove the stronger statement even though our original statement $q(n)$ did not give an inductive hypothesis sufficient to prove the original statement. However we must be careful in our choice of $s(n)$, because we have to be able to succeed in proving $p(n) \Rightarrow s(n)$.

5. *When the Master Theorem does not Apply.* To deal with recurrences of the form

$$T(n) = \begin{cases} aT(\lceil n/b \rceil) + f(n) & \text{if } n > 1 \\ d & \text{if } n = 1 \end{cases}$$

   where $f(n)$ is not $\Theta(n^c)$, recursion trees and iterating the recurrence are appropriate tools even though the Master Theorem does not apply. The same holds for recurrence inequalities.

6. *Increasing function. (Optional.)* A function $f : R \to R$ is said to be *(weakly) increasing* if whenever $x > y$, $f(x) \geq f(y)$

7. *Removing Floors and Ceilings when the Master Theorem does not Apply. (Optional.)* To deal with big-Θ bounds with recurrences of the form

$$T(n) = \begin{cases} aT(\lceil n/b \rceil) + f(n) & \text{if } n > 1 \\ d & \text{if } n = 1 \end{cases}$$

where $f(n)$ is not $\Theta(n^c)$, we may remove floors and ceilings and replace n by powers of b if f is increasing and satisfies the condition $f(nb) = \Theta(f(n))$. To deal with big-O bounds for a similar recurrence inequality we may remove floors and ceilings if f is increasing and satisfies the condition that $f(nb) = O(f(n))$.

Problems

1. (a) Find the best big-O upper bound you can to any solution to the recurrence

$$T(n) = \begin{cases} 4T(n/2) + n \log n & \text{if } n > 1 \\ 1 & \text{if } n = 1. \end{cases}$$

   (b) Assuming that you were able to guess the result you got in part (a), prove by induction that your answer is correct.

2. Is the big-O upper bound in the previous problem actually a big-Θ bound?

3. Show by induction that

$$T(n) = \begin{cases} 8T(n/2) + n \log n & \text{if } n > 1 \\ d & \text{if } n = 1 \end{cases}$$

   has $T(n) = O(n^3)$ for any solution $T(n)$.

4. Is the big-O upper bound in the previous problem actually a big-Θ bound?

5. Show by induction that any solution to a recurrence of the form

$$T(n) \le 2T(n/3) + c \log_3 n$$

   is $O(n \log_3 n)$. What happens if you replace 2 by 3 (explain why)? Would it make a difference if we used a different base for the logarithm (only an intuitive explanation is needed here).

6. What happens if you replace the 2 in Problem 5 by 4? (Hint: one way to attack this is with recursion trees.)

7. Is the big-O upper bound in Problem 5 actually a big Θ bound?

8. Give an example (different from any in the text) of a function for which $f(bx) = O(f(x))$. Give an example (different from any in the text) of a function for which $f(bx)$ is not $O(f(x))$.

9. Give the best big O upper bound you can for the solution to the recurrence $T(n) = 2T(n/3 - 3) + n$, and then prove by induction that your upper bound is correct.

10. Find the best big-O upper bound you can to any solution to the recurrence defined on nonnegative integers by
$$T(n) \leq 2T(\lceil n/2 \rceil + 1) + cn.$$
Prove by induction that your answer is correct.

4.6 Recurrences and Selection

One common problem that arises in algorithms is that of *selection*. In this problem you are given n distinct data items from some set which has an underlying order. That is, given any two items a and b, you can determine whether $a < b$. (Integers satisfy this property, but colors do not.) Given these n items, and some value i, $1 \leq i \leq n$, you wish to find the ith smallest item in the set. For example in the set

$$\{3, 1, 8, 6, 4, 11, 7\}, \tag{4.33}$$

the first smallest ($i = 1$) is 1, the third smallest ($i = 3$) is 4 and the seventh smallest ($i = n = 7$) is 11. An important special case is that of finding the *median*, which is the case of $i = \lceil n/2 \rceil$. Another important special case is finding percentiles; for example the 90th percentile is the case $i = \lceil .9n \rceil$. As this suggests, i is frequently given as some fraction of n.

Exercise 4.6-1 How do you find the minimum ($i = 1$) or maximum ($i = n$) in a set? What is the running time? How do you find the second smallest element? Does this approach extend to finding the ith smallest? What is the running time?

Exercise 4.6-2 Give the fastest algorithm you can to find the median ($i = \lceil n/2 \rceil$).

In Exercise 4.6-1, the simple $O(n)$ algorithm of going through the list and keeping track of the minimum value seen so far will suffice to find the minimum. Similarly, if we want to find the second smallest, we can go through the list once, find the smallest, remove it and then find the smallest in the new list. This also takes $O(n + n - 1) = O(n)$ time. If we extend this to finding the ith smallest, the algorithm will take $O(in)$ time. Thus for finding the median, this method takes $O(n^2)$ time.

A better idea for finding the median is to first sort the items, and then take the item in position $n/2$. Since we can sort in $O(n \log n)$ time, this algorithm will take $O(n \log n)$ time. Thus if $i = O(\log n)$ we might want to run the algorithm of the previous paragraph, and otherwise run this algorithm.[4]

All these approaches, when applied to the median, take at least some multiple of $(n \log n)$ units of time.[5] The best sorting algorithms take $O(n \log n)$ time also, and one can prove every comparison-based sorting algorithm takes $\Omega(n \log n)$ time. This raises the natural question of whether it is possible to do selection any faster than sorting. In other words, is the problem of finding the median element, or ith smallest element of a set significantly easier than the problem of ordering (sorting) the whole set?

Recursive Selection Algorithm

Suppose for a minute that we magically knew how to find the median in $O(n)$ time. That is, we have a routine MagicMedian, that given as input a set A, returns the median. We could then use this in a divide and conquer algorithm for Select as follows;

[4]We also note that the running time can be improved to $O(n + i \log n)$ by first creating a *heap*, which takes $O(n)$ time, and then performing a Delete-Min operation i times.

[5]An alternate notation for $f(x) = O(g(x))$ is $g(x) = \Omega(f(x))$. Notice the change in roles of f and g. In this notation, we say that all of these algorithms take $\Omega(n \log n)$ time.

```
Select($A, i, n$)
(selects the $i$th smallest element in set $A$, where $n = |A|$)
(1)  if ($n = 1$)
(2)       return the one item in $A$
(3)  else
(4)       $p = MagicMedian(A)$
(5)       Let $H$ be the set of elements greater than $p$
(6)       Let $L$ be the set of elements less than or equal to $p$
(7)       if ($i \leq |L|$)
(8)            Return Select($L, i, |L|$)
(9)       else
(10)           Return Select($H, i - |L|, |H|$)
```

By H we do not mean the elements that come after p in the list, but the elements of the list which are larger than p in the underlying ordering of our set. This algorithm is based on the following simple observation. If we could divide the set A up into a "lower half" (L) and an "upper" half (H), then we know in which of these two sets the ith smallest element in A will be. Namely, if $i \leq \lceil n/2 \rceil$, it will be in L, and otherwise it will be in H. Thus, we can recursively look in one or the other set. We can easily partition the data into two sets by making two passes, in the first we copy the numbers smaller than p into L, and in the second we copy the numbers larger than p into H.[6]

The only additional detail is that if we look in H, then instead of looking for the ith smallest, we look for the $i - \lceil n/2 \rceil$th smallest, as H is formed by removing the $\lceil n/2 \rceil$ smallest elements from A.

For example, if the input is the set given in 4.33, and p is 6, the set L would be $\{3, 1, 6, 4\}$, and H would be $\{8, 11, 7\}$. If i were 2, we would recurse on the set L, with $i = 2$. On the other hand, if i were 6, we would recurse on the set H, with $i = 6 - 4 = 2$. Observe that the second smallest element in H is 8, as is the sixth smallest element in the original set.

We can express the running time of Select by the following recurrence:

$$T(n) \leq T(n/2) + cn \ . \tag{4.34}$$

From the master theorem, we know any function which satisfies this recurrence has $T(n) = O(n)$.

So we can conclude that if we already know how to find the median in linear time, we can design a divide and conquer algorithm that will solve the selection problem in linear time. This is nothing to write home about (yet)!

Sometimes a knowledge of solving recurrences can help us design algorithms. What kinds of recurrences do we know about that have solutions $T(n)$ with $T(n) = O(n)$? In particular, consider recurrences of the form $T(n) \leq T(n/b) + cn$, and ask when they have solutions with $T(n) = O(n)$. Using the master theorem, we see that as long as $\log_b 1 < 1$ (and since $\log_b 1 = 0$ for any b, this means than any b allowed by the master theorem works; that is, any $b > 1$ will work), all solutions to this recurrence will have $T(n) = O(n)$. (Note that b does not have to be an integer.) Letting $b' = 1/b$, this says that as long as we can solve a problem of size n by solving

[6]We can do this more efficiently, and "in place", using the partition algorithm of quicksort.

(recursively) a problem of size $b'n$, for some $b' < 1$, and also doing $O(n)$ additional work, our algorithm will run in $O(n)$ time. Interpreting this in the selection problem, it says that as long as we can, in $O(n)$ time, choose p to ensure that both L and H have size at most $b'n$, we will have a linear time algorithm. (You might ask "What about actually dividing our set into L and H, doesn't that take some time too?" The answer is yes it does, but we already know we can do the division into H and L in time $O(n)$, so if we can find p in time $O(n)$ also, then we can do both these things in time $O(n)$.)

In particular, suppose that, in $O(n)$ time, we can choose p to ensure that both L and H have size at most $(3/4)n$. Then the running time is described by the recurrence $T(n) = T(3n/4)+O(n)$ and we will be able to solve the selection problem in linear time.

To see why $(3/4)n$ is relevant, suppose instead of the "black box" MagicMedian, we have a much weaker magic black box, one which only guarantees that it will return some number in the middle half of our set in time $O(n)$. That is, it will return a number that is guaranteed to be somewhere between the $n/4$th smallest number and the $3n/4$th smallest number. If we use the number given by this magic box to divide our set into H and L, then neither will have size more than $3n/4$. We will call this black box a MagicMiddle box, and can use it in the following algorithm:

```
Select1(A,i,n)
(selects the $i$th smallest element in set $A$, where $n = |A|$ )
(1)  if ($n = 1$)
(2)       return the one item in $A$
(3)  else
(4)       $p = MagicMiddle(A)$
(5)       Let $H$ be the set of elements greater than $p$
(6)       Let $L$ be the set of elements less than or equal to $p$
(7)       if ($i \leq |L|$)
(8)            Return Select1($L, i, |L|$)
(9)       else
(10)           Return Select1($H, i - |L|, |H|$)
```

The algorithm Select1 is similar to Select. The only difference is that p is now only guaranteed to be in the middle half. Now, when we recurse, the decision of the set on which to recurse is based on whether i is less than or equal to $|L|$. The element p is called a *partition element*, because it is used to partition our set A into the two sets L and H.

This is progress, as we now don't need to assume that we can find the median in order to have a linear time algorithm, we only need to assume that we can find one number in the middle half of the set. This problem seems simpler than the original problem, and in fact it is. Thus our knowledge of which recurrences have solutions which are $O(n)$ led us toward a more plausible algorithm.

Even though the problem is simpler, we don't know a straightforward way to even find an item in the middle half. We will now describe a way to find it, however, in which we first choose a subset of the numbers and then *recursively* find the median of that subset.

More precisely, consider the following algorithm (in which we assume that $|A|$ is a multiple of 5.)

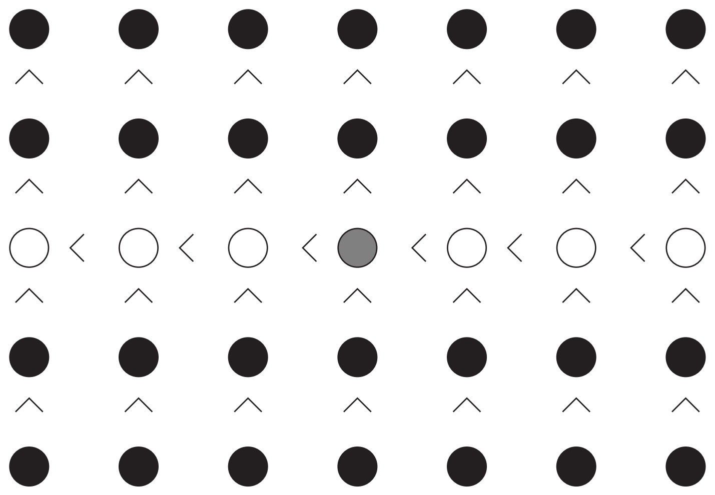
```
MagicMiddle(A)
(1)  Let $n = |A|$
(2)  if $(n < 60)$
(3)       use Select1 to return the median of $A$
(4)  else
(5)       Break $A$ into $k = n/5$ groups of size 5, $G_1, \ldots, G_k$
(6)       for $i = 1$ to $k$
(7)            find $m_i$, the median of $G_i$
(8)       Let $M = \{m_1, \ldots, m_k\}$
(9)       return Select1 $(M, \lceil k/2 \rceil, k)$
```

In this algorithm, we break A into $n/5$ sets of size 5, and then find the median of each set. We then (using Select1 recursively) find the median of medians and return this as our p.

Lemma 4.17 *The value returned by MagicMiddle(A) is in the middle half of A.*

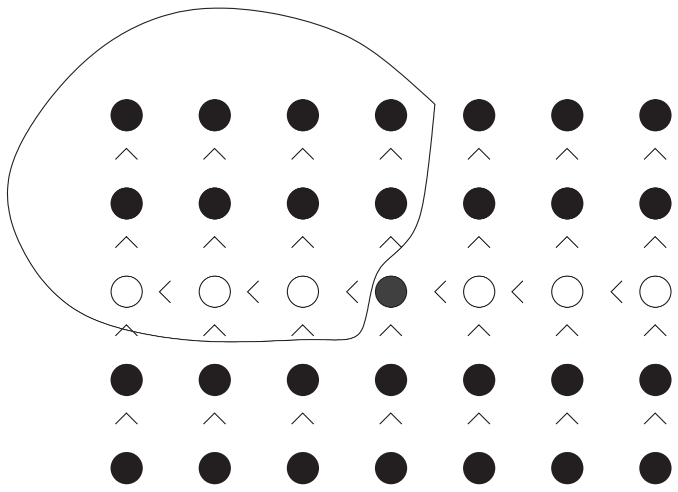
Proof: Consider arranging the elements in the following manner. For each set of 5, list them in sorted order, with the smallest element on top. Then line up all $n/5$ of these lists, ordered by their medians, smallest on the left. We get the picture in Figure 4.10. In this picture, the medians

Figure 4.10: Dividing a set into $n/5$ parts of size 5, finding the median of each part and the median of the medians.



are in white, the median of medians is cross-hatched, and we have put in all the inequalities that we know from the ordering information that we have. Now, consider how many items are less than or equal to the median of medians. Every smaller median is clearly less than the median of medians and, in its 5 element set, the elements smaller than the median are also smaller than the median of medians. Now in Figure 4.11 we circle a set of elements that is guaranteed to be smaller than the median of medians. In one fewer (or in the case of an odd number of columns as in Figure 4.11, one half fewer) than half the columns, we have circled 3 elements and in one

Figure 4.11: The circled elements are less than the median of the medians.



column we have circled 2 elements. Therefore, we have circled at least[7]

$$\left(\frac{1}{2}\left(\frac{n}{5}\right)-1\right)3+2=\frac{3n}{10}-1$$

elements.

So far we have assumed n is an exact multiple of 5, but we will be using this idea in circumstances when it is not. If it is not an exact multiple of 5, we will have $\lceil n/5\rceil$ columns (in particular more than $n/5$ columns), but in one of them we might have only one element. It is possible that column is one of the ones we counted on for 3 elements, so our estimate could be two elements too large.[8] Thus we have circled at least

$$\frac{3n}{10}-1-2=\frac{3n}{10}-3$$

elements. It is a straightforward argument with inequalities that as long as $n \geq 60$, this quantity is at least $n/4$. So if at least $n/4$ items are guaranteed to be less than the median, then at most $3n/4$ items can be greater than the median, and hence $|H| \leq 3n/4$.

A set of elements that is guaranteed to be larger than the median of medians is circled in the Figure 4.12. We can make the same argument about the number of larger elements circled when the number of columns is odd; when the number of columns is even, a similar argument shows that we circle even more elements. By the same argument as we used with $|H|$, this shows that the size of L is at most $3n/4$. ■

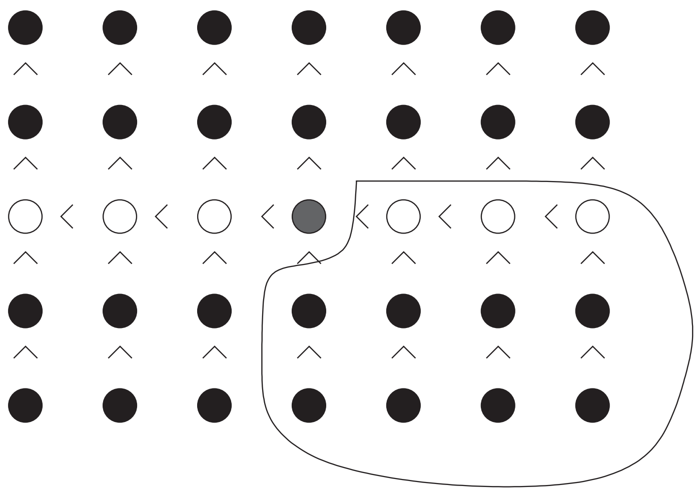
Note that we don't actually identify all the nodes that are guaranteed to be, say, less than the median of medians, we are just guaranteed that the proper number exists.

Since we only have the guarantee that MagicMiddle gives us an element in the middle half of the set if the set has at least sixty elements, we modify Select1 to start out by checking to see if

[7]We say "at least" because our argument applies exactly when n is even, but underestimates the number of circled elements when n is odd.

[8]A bit less than 2 because we have more than $n/5$ columns.

Figure 4.12: The circled elements are greater than the median of the medians.



$n < 60$, and sorting the set to find the element in position i if $n < 60$. Since 60 is a constant, sorting and finding the desired element takes at most a constant amount of time.

Exercise 4.6-3 Let $T(n)$ be the running time of Select1 on n items. How can you express the running time of Magic Middle in terms of $T(n)$?

Exercise 4.6-4 What is a recurrence for the running time of Select1? Hint: how could Exercise 4.6-3 help you?

Exercise 4.6-5 Can you prove by induction that each solution to the recurrence for Select1 is $O(n)$?

For Exercise 4.6-3, the first step of MagicMiddle is to divide the items into sets of five; this takes $O(n)$ time. We then have to find the median of each five-element set. (We can find this median by any straightforward method we choose and still only take at most a constant amount of time; we don't use recursion here.) There are $n/5$ sets and we spend no more than some constant time per set, so the total time is $O(n)$. Next we recursively call Select1 to find the median of medians; this takes $T(n/5)$ time. Finally, we partition A into those elements less than or equal to the "magic middle" and those that are not, which takes $O(n)$ time. Thus the total running time is $T(n/5) + O(n)$, which implies that for some n_0 there is a constant $c_0 > 0$ such that, for all $n > n_0$, the running time is no more than $c_0 n$. Even if $n_0 > 60$, there are only finitely many cases between 60 and n_0 so there is a constant c such that for $n \geq 60$, the running time of Magic Middle is no more than $T(n/5) + cn$.

Now Select1 has to call Magic Middle and then recurse on either L or H, each of which has size at most $3n/4$. Adding in a base case that it takes time no more than some constant amount d of time to cover sets of size less than 60, we get the following recurrence for the running time of Select1:

$$T(n) \leq \begin{cases} T(3n/4) + T(n/5) + c'n & \text{if } n \geq 60 \\ d & \text{if } n < 60. \end{cases} \tag{4.35}$$

This answers Exercise 4.6-4.

As Exercise 4.6-5 requests, we can now verify by induction that $T(n) = O(n)$. What we want to prove is that there is a constant k such that $T(n) \leq kn$. What the recurrence tells us is that there are constants c and d such that $T(n) \leq T(3n/4) + T(n/5) + cn$ if $n \geq 60$, and otherwise $T(n) \leq d$. For the base case we have $T(n) \leq d \leq dn$ for $n < 60$, so we choose k to be at least d and then $T(n) \leq kn$ for $n < 60$. We now assume that $n \geq 60$ and $T(m) \leq km$ for values $m < n$, and get

$$\begin{aligned} T(n) &\leq T(3n/4) + T(n/5) + cn \\ &\leq 3kn/4 + 2kn/5 + cn \\ &= 19/20kn + cn \\ &= kn + (c - k/20)n \ . \end{aligned}$$

As long as $k \geq 20c$, this is at most kn; so we simply choose k this big and by the principle of mathematical induction, we have $T(n) < kn$ for all positive integers n.

Uneven Divisions

The kind of recurrence we found for the running time of Select1 is actually an instance of a more general class which we will now explore.

Exercise 4.6-6 We already know that when $g(n) = O(n)$, then every solution of $T(n) = T(n/2) + g(n)$ satisfies $T(n) = O(n)$. Use the master theorem to find a Big-O bound to the solution of $T(n) = T(cn) + g(n)$ for any constant $c < 1$, assuming that $g(n) = O(n)$.

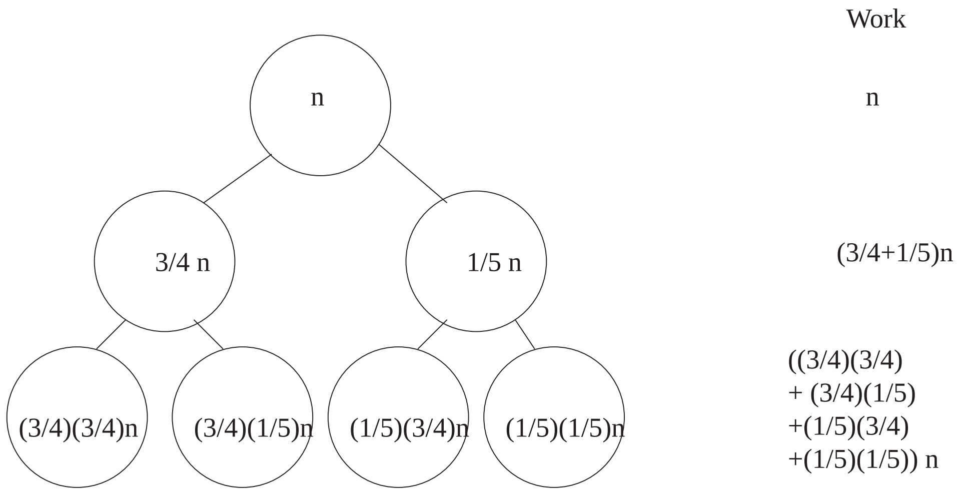
Exercise 4.6-7 Use the master theorem to find Big-O bounds to all solutions of $T(n) = 2T(cn) + g(n)$ for any constant $c < 1/2$, assuming that $g(n) = O(n)$.

Exercise 4.6-8 Suppose $g(n) = O(n)$ and you have a recurrence of the form $T(n) = T(an) + T(bn) + g(n)$ for some constants a and b. What conditions on a and b guarantee that all solutions to this recurrence have $T(n) = O(n)$?

Using the master theorem for Exercise 4.6-6, we get $T(n) = O(n)$, since $\log_{1/c} 1 < 1$. We also get $T(n) = O(n)$ for Exercise 4.6-7, since $\log_{1/c} 2 < 1$ for $c < 1/2$. You might now guess that as long as $a + b < 1$, any solution to the recurrence $T(n) \leq T(an) + T(bn) + cn$ has $T(n) = O(n)$. We will now see why this is the case.

First, let's return to the recurrence we had, $T(n) = T(3/4n)+T(n/5)+g(n)$, were $g(n) = O(n)$ and let's try to draw a recursion tree. This recurrence doesn't quite fit our model for recursion trees, as the two subproblems have unequal size (thus we can't even write down the problem size on the left), but we will try to draw a recursion tree anyway and see what happens. As we draw levels one and two, we see that at the level one, we have $(3/4 + 1/5)n$ work. At the level two we have $((3/4)^2 + 2(3/4)(1/5) + (1/5)^2)n$ work. Were we to work out the third level we would see that we have $((3/4)^3 + 3(3/4)^2(1/5) + 3(3/4)(1/5)^2 + (1/5)^3)n$. Thus we can see a pattern emerging. At level one we have $(3/4 + 1/5)n$ work. At level 2 we have, by the binomial theorem, $(3/4 + 1/5)^2n$ work. At level 3 we have, by the binomial theorem, $(3/4 + 1/5)^3n$ work. And,

Figure 4.13: Attempting a recursion tree for $T(n) = T(3/4n) + T(n/5) + g(n)$.



similarly, at level i of the tree, we have $\left(\frac{3}{4}+\frac{1}{5}\right)^i n = \left(\frac{19}{20}\right)^i$ work. Thus summing over all the levels, the total amount of work is

$$\sum_{i=0}^{O(\log n)} \left(\frac{19}{20}\right)^i n \leq \left(\frac{1}{1-19/20}\right) n = 20n.$$

We have actually ignored one detail here. In contrast to a recursion tree in which all subproblems at a level have equal size, the "bottom" of the tree is more complicated. Different branches of the tree will reach problems of size 1 and terminate at different levels. For example, the branch that follows all 3/4's will bottom out after $\log_{4/3} n$ levels, while the one that follows all 1/5's will bottom out after $\log_5 n$ levels. However, the analysis above *overestimates the work*, that is, it assumes that nothing bottoms out until everything bottoms out, i.e. at $\log_{20/19} n$ levels, and in fact, the upper bound we gave on the sum "assumes" that the recurrence never bottoms out.

We see here something general happening. It seems as if to understand a recurrence of the form $T(n) = T(an) + T(bn) + g(n)$, with $g(n) = O(n)$, we can study the simpler recurrence $T(n) = T((a+b)n) + g(n)$ instead. This simplifies things (in particular, it lets us use the Master Theorem) and allows us to analyze a larger class of recurrences. Turning to the median algorithm, it tells us that the important thing that happened there was that the sizes of the two recursive calls, namely $3/4n$ and $n/5$, summed to less than 1. As long as that is the case for an algorithm with two recursive calls and an $O(n)$ additional work term, whose recurrence has the form $T(n) = T(an) + T(bn) + g(n)$, with $g(n) = O(n)$, the algorithm will work in $O(n)$ time.

Important Concepts, Formulas, and Theorems

1. *Median.* The *median* of a set (with an underlying order) of n elements is the element that would be in position $\lceil n/2 \rceil$ if the set were sorted into a list in order.

2. *Percentile.* The pth percentile of a set (with an underlying order) is the element that would be in position $\frac{\lceil p}{100\rceil} n$ if the set were sorted into a list in order.

3. *Selection.* Given an n-element set with some underlying order, the problem of *selection* of the ith smallest element is that of finding the element that would be in the ith position if the set were sorted into a list in order. Note that often i is expressed as a fraction of n.

4. *Partition Element.* A *partition element* in an algorithm is an element of a set (with an underlying order) which is used to divide the set into two parts, those that come before or are equal to the element (in the underlying order), and the remaining elements. Notice that the set as given to the algorithm is not necessarily (in fact not usually) given in the underlying order.

5. *Linear Time Algorithms.* If the running time of an algorithm satisfies a recurrence of the form $T(n) \leq T(an) + cn$ with $0 \leq a < 1$, or a recurrence of the form $T(n) \leq T(an) + T(bn) + cn$ with a and b nonnegative and $a + b < 1$, then $T(n) = O(n)$.

6. *Finding a Good Partition Element.* If a set (with an underlying order) has sixty or more elements, then the procedure of breaking the set into pieces of size 5 (plus one leftover piece if necessary), finding the median of each piece and the finding the median of the medians gives an element guaranteed to be in the middle half of the set.

7. *Selection algorithm.* The Selection algorithm with a linear time running guarantee sorts a set of size less than sixty to find the element in the ith position; otherwise it recursively uses the median of medians of five to find a partition element, uses that partition element to divide the set into two pieces and looks for the appropriate element in the appropriate piece recursively.

Problems

1. In the MagicMiddle algorithm, suppose we broke our data up into $n/3$ sets of size 3. What would the running time of Select1 be?

2. In the MagicMiddle algorithm, suppose we broke our data up into $n/7$ sets of size 7. What would the running time of Select1 be?

3. Let
$$T(n) = \begin{cases} T(n/3) + T(n/2) + n & \text{if } n \geq 6 \\ 1 & \text{otherwise,} \end{cases}$$
and let
$$S(n) = \begin{cases} S(5n/6) + n & \text{if } n \geq 6 \\ 1 & \text{otherwise.} \end{cases}$$
Draw recursion trees for T and S. What are the big-O bounds we get on solutions to the recurrences? Use the recursion trees to argue that, for all n, $T(n) \leq S(n)$.

4. Find a (big-O) upper bound (the best you know how to get) on solutions to the recurrence $T(n) = T(n/3) + T(n/6) + T(n/4) + n$.

5. Find a (big-O) upper bound (the best you know how to get) on solutions the recurrence $T(n) = T(n/4) + T(n/2) + n^2$.

6. Note that we have chosen the median of an n-element set to be the element in position $\lceil n/2 \rceil$. We have also chosen to put the median of the medians into the set L of algorithm Select1. Show that this lets us prove that $T(n) \le T(3n/4) + T(n/5) + cn$ for $n \ge 40$ rather than $n \ge 60$. (You will need to analyze the case where $\lceil n/5 \rceil$ is even and the case where it is odd separately.) Is 40 the least value possible?

Chapter 5

Probability

5.1 Introduction to Probability

Why do we study probability?

You have studied hashing as a way to store data (or keys to find data) in a way that makes it possible to access that data quickly. Recall that we have a table in which we want to store keys, and we compute a function h of our key to tell us which location (also known as a "slot" or a "bucket") in the table to use for the key. Such a function is chosen with the hope that it will tell us to put different keys in different places, but with the realization that it might not. If the function tells us to put two keys in the same place, we might put them into a linked list that starts at the appropriate place in the table, or we might have some strategy for putting them into some other place in the table itself. If we have a table with a hundred places and fifty keys to put in those places, there is no reason in advance why all fifty of those keys couldn't be assigned (hashed) to the same place in the table. However someone who is experienced with using hash functions and looking at the results will tell you you'd never see this in a million years. On the other hand that same person would also tell you that you'd never see all the keys hash into different locations in a million years either. In fact, it is far less likely that all fifty keys would hash into one place than that all fifty keys would hash into different places, but both events are quite unlikely. Being able to understand just how likely or unlikely such events are is our reason for taking up the study of probability.

In order to assign probabilities to events, we need to have a clear picture of what these events are. Thus we present a model of the kinds of situations in which it is reasonable to assign probabilities, and then recast our questions about probabilities into questions about this model. We use the phrase *sample space* to refer to the set of possible outcomes of a process. For now, we will deal with processes that have finite sample spaces. The process might be a game of cards, a sequence of hashes into a hash table, a sequence of tests on a number to see if it fails to be a prime, a roll of a die, a series of coin flips, a laboratory experiment, a survey, or any of many other possibilities. A set of elements in a sample space is called an *event*. For example, if a professor starts each class with a 3 question true-false quiz the sample space of all possible patterns of correct answers is

$$\{TTT, TTF, TFT, FTT, TFF, FTF, FFT, FFF\}.$$

The event of the first two answers being true is $\{TTT, TTF\}$. In order to compute probabilities we assign a *probability weight* $p(x)$ to each element of the sample space so that the weight represents what we believe to be the relative likelihood of that outcome. There are two rules we must follow in assigning weights. First the weights must be nonnegative numbers, and second the sum of the weights of all the elements in a sample space must be one. We define the *probability* $P(E)$ of the event E to be the sum of the weights of the elements of E. Algebraically we can write

$$P(E) = \sum_{x:x\in E} p(x). \tag{5.1}$$

We read this as $p(E)$ equals he sum over all x such that x is in E of $p(x)$.

Notice that a probability function P on a sample space S satisfies the rules[1]

1. $P(A) \geq 0$ for any $A \subseteq S$.
2. $P(S) = 1$.
3. $P(A \cup B) = P(A) + P(B)$ for any two disjoint events A and B.

The first two rules reflect our rules for assigning weights above. We say that two events are disjoint if $A \cap B = \emptyset$. The third rule follows directly from the definition of disjoint and our definition of the probability of an event. A function P satisfying these rules is called a *probability distribution* or a *probability measure*.

In the case of the professor's three question quiz, it is natural to expect each sequence of trues and falses to be equally likely. (A professor who showed any pattern of preferences would end up rewarding a student who observed this pattern and used it in educated guessing.) Thus it is natural to assign equal weight 1/8 to each of the eight elements of our quiz sample space. Then the *probability* of an event E, which we denote by $P(E)$, is the sum of the weights of its elements. Thus the probability of the event "the first answer is T" is $\frac{1}{8} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8} = \frac{1}{2}$. The event "There is at exactly one True" is $\{TFF, FTF, FFT\}$, so P(there is exactly one True) is 3/8.

Exercise 5.1-1 Try flipping a coin five times. Did you get at least one head? Repeat five coin flips a few more times! What is the probability of getting at least one head in five flips of a coin? What is the probability of no heads?

Exercise 5.1-2 Find a good sample space for rolling two dice. What weights are appropriate for the members of your sample space? What is the probability of getting a 6 or 7 total on the two dice? Assume the dice are of different colors. What is the probability of getting less than 3 on the red one and more than 3 on the green one?

Exercise 5.1-3 Suppose you hash a list of n keys into a hash table with 20 locations. What is an appropriate sample space, and what is an appropriate weight function? (Assume the keys and the hash function are not in any special relationship to the number 20.) If n is three, what is the probability that all three keys hash to different locations? If you hash ten keys into the table, what is the probability that at least

[1]These rules are often called "the axioms of probability." For a finite sample space, we could show that if we started with these axioms, our definition of probability in terms of the weights of individual elements of S is the only definition possible. That is, for any other definition, the probabilities we would compute would still be the same if we take $w(x) = P(\{x\})$.

two keys have hashed to the same location? We say two keys *collide* if they hash to the same location. How big does n have to be to insure that the probability is at least one half that has been at least one collision?

In Exercise 5.1-1 a good sample space is the set of all 5-tuples of Hs and Ts. There are 32 elements in the sample space, and no element has any reason to be more likely than any other, so a natural weight to use is $\frac{1}{32}$ for each element of the sample space. Then the event of at least one head is the set of all elements but $TTTTT$. Since there are 31 elements in this set, its probability is $\frac{31}{32}$. This suggests that you should have observed at least one head pretty often!

The probability of no heads is the weight of the set $\{TTTTT\}$, which is $\frac{1}{32}$. Notice that the probabilities of the event of "no heads" and the opposite event of "at least one head" add to one. This observation suggests a theorem. The *complement* of an event E in a sample space S, denoted by $S-E$, is the set of all outcomes in S but not E. The theorem tells us how to compute the probability of the complement of an event from the probability of the event.

Theorem 5.1 *If two events E and F are complementary, that is they have nothing in common ($E \cap F = \emptyset$) and their union is the whole sample space ($E \cup F = S$), then*

$$P(E) = 1 - P(F).$$

Proof: The sum of all the probabilities of all the elements of the sample space is one, and since we can break this sum into the sum of the probabilities of the elements of E plus the sum of the probabilities of the elements of F, we have

$$P(E) + P(F) = 1,$$

which gives us $P(E) = 1 - P(F)$.■

For Exercise 5.1-2 a good sample space would be pairs of numbers (a, b) where $(1 \leq a, b \leq 6)$. By the product principle[2], the size of this sample space is $6 \cdot 6 = 36$. Thus a natural weight for each ordered pair is $\frac{1}{36}$. How do we compute the probability of getting a sum of six or seven? There are 5 ways to roll a six and 6 ways to roll a seven, so our event has eleven elements each of weight $1/36$. Thus the probability of our event is is $11/36$. For the question about the red and green dice, there are two ways for the red one to turn up less than 3, and three ways for the green one to turn up more than 3. Thus, the event of getting less than 3 on the red one and greater than 3 on the green one is a set of size $2 \cdot 3 = 6$ by the product principle. Since each element of the event has weight $1/36$, the event has probability $6/36$ or $1/6$.

In Exercise 5.1-3 an appropriate sample space is the set of n-tuples of numbers between 1 and 20. The first entry in an n-tuple is the position our first key hashes to, the second entry is the position our second key hashes to, and so on. Thus each n tuple represents a possible hash function, and each hash function, applied to our keys, would give us one n-tuple. The size of the sample space is 20^n (why?), so an appropriate weight for an n-tuple is $1/20^n$. To compute the probability of a collision, we will first compute the probability that all keys hash to different locations and then apply Theorem 5.1 which tells us to subtract this probability from 1 to get the probability of collision.

[2]from Section 1.1

n	Prob of empty slot	Prob of no collisions
1	1	1
2	0.95	0.95
3	0.9	0.855
4	0.85	0.72675
5	0.8	0.5814
6	0.75	0.43605
7	0.7	0.305235
8	0.65	0.19840275
9	0.6	0.11904165
10	0.55	0.065472908
11	0.5	0.032736454
12	0.45	0.014731404
13	0.4	0.005892562
14	0.35	0.002062397
15	0.3	0.000618719
16	0.25	0.00015468
17	0.2	3.09359E-05
18	0.15	4.64039E-06
19	0.1	4.64039E-07
20	0.05	2.3202E-08

Table 5.1: The probabilities that all elements of a set hash to different entries of a hash table of size 20.

To compute the probability that all keys hash to different locations we consider the event that all keys hash to different locations. This is the set of n tuples in which all the entries are different. (In the terminology of functions, these n-tuples correspond to one-to-one hash functions). There are 20 choices for the first entry of an n-tuple in our event. Since the second entry has to be different, there are 19 choices for the second entry of this n-tuple. Similarly there are 18 choices for the third entry (it has to be different from the first two), 17 for the fourth, and in general $20 - i + 1$ possibilities for the ith entry of the n-tuple. Thus we have

$$20 \cdot 19 \cdot 18 \cdot \cdots \cdot (20 - n + 1) = 20^{\underline{n}}$$

elements of our event.[3] Since each element of this event has weight $1/20^n$, the probability that all the keys hash to different locations is

$$\frac{20 \cdot 19 \cdot 18 \cdot \cdots \cdot (20 - n + 1)}{20^n} = \frac{20^{\underline{n}}}{20^n}.$$

In particular if n is 3 the probability is $(20 \cdot 19 \cdot 18)/20^3 = .855$.

We show the values of this function for n between 0 and 20 in Table 5.1. Note how quickly the probability of getting a collision grows. As you can see with $n = 10$, the probability that there have been no collisions is about .065, so the probability of at least one collision is .935.

If $n = 5$ this number is about .58, and if $n = 6$ this number is about .43. By Theorem 5.1 the probability of a collision is one minus the probability that all the keys hash to different locations.

[3]using the notation for falling factorial powers that we introduced in Section 1.2.

Thus if we hash six items into our table, the probability of a collision is more than 1/2. Our first intuition might well have been that we would need to hash ten items into our table to have probability 1/2 of a collision. This example shows the importance of supplementing intuition with careful computation!

The technique of computing the probability of an event of interest by first computing the probability of its complementary event and then subtracting from 1 is very useful. You will see many opportunities to use it, perhaps because about half the time it is easier to compute directly the probability that an event doesn't occur than the probability that it does. We stated Theorem 5.1 as a theorem to emphasize the importance of this technique.

The Uniform Probability Distribution

In all three of our exercises it was appropriate to assign the same weight to all members of our sample space. We say P is the *uniform probability measure* or *uniform probability distribution* when we assign the same probability to all members of our sample space. The computations in the exercises suggest another useful theorem.

Theorem 5.2 *Suppose P is the uniform probability measure defined on a sample space S. Then for any event E,*

$$P(E) = |E|/|S|,$$

the size of E divided by the size of S.

Proof: Let $S = \{x_1, x_2, \ldots, x_{|S|}\}$. Since P is the uniform probability measure, there must be some value p such that for each $x_i \in S, P(x_i) = p$. Combining this fact with the second and third probability rules, we obtain

$$\begin{aligned} 1 &= P(S) \\ &= P(x_1 \cup x_2 \cup \cdots \cup x_{|S|}) \\ &= P(x_1) + P(x_2) + \ldots + P(x_{|S|}) \\ &= p|S| \ . \end{aligned}$$

Equivalently

$$p = \frac{1}{|S|} \ . \tag{5.2}$$

E is a subset of S with $|E|$ elements and therefore

$$P(E) = \sum_{x_i \in E} p(x_i) = |E|p \ . \tag{5.3}$$

Combining equations 5.2 and 5.3 gives that $P(E) = |E|p = |E|(1/|S|) = |E|/|S|$. ■

Exercise 5.1-4 What is the probability of an odd number of heads in three tosses of a coin? Use Theorem 5.2.

Using a sample space similar to that of first example (with "T" and "F" replaced by "H" and "T"), we see there are three sequences with one H and there is one sequence with three H's. Thus we have four sequences in the event of "an odd number of heads come up." There are eight sequences in the sample space, so the probability is $\frac{4}{8} = \frac{1}{2}$.

It is comforting that we got one half because of a symmetry inherent in this problem. In flipping coins, heads and tails are equally likely. Further if we are flipping 3 coins, an odd number of heads implies an even number of tails. Therefore, the probability of an odd number of heads, even number of heads, odd number of tails and even number of tails must all be the same. Applying Theorem 5.1 we see that the probability must be 1/2.

A word of caution is appropriate here. Theorem 5.2 applies only to probabilities that come from the equiprobable weighting function. The next example shows that it does not apply in general.

Exercise 5.1-5 A sample space consists of the numbers 0, 1, 2 and 3. We assign weight $\frac{1}{8}$ to 0, $\frac{3}{8}$ to 1, $\frac{3}{8}$ to 2, and $\frac{1}{8}$ to 3. What is the probability that an element of the sample space is positive? Show that this is not the result we would obtain by using the formula of Theorem 5.2.

The event "x is positive" is the set $E = \{1, 2, 3\}$. The probability of E is

$$P(E) = P(1) + P(2) + P(3) = \frac{3}{8} + \frac{3}{8} + \frac{1}{8} = \frac{7}{8}\ .$$

However, $\frac{|E|}{|S|} = \frac{3}{4}$.

The previous exercise may seem to be "cooked up" in an unusual way just to prove a point. In fact that sample space and that probability measure could easily arise in studying something as simple as coin flipping.

Exercise 5.1-6 Use the set $\{0, 1, 2, 3\}$ as a sample space for the process of flipping a coin three times and counting the number of heads. Determine the appropriate probability weights $P(0)$, $P(1)$, $P(2)$, and $P(3)$.

There is one way to get the outcome 0, namely tails on each flip. There are, however, three ways to get 1 head and three ways to get two heads. Thus $P(1)$ and $P(2)$ should each be three times $P(0)$. There is one way to get the outcome 3—heads on each flip. Thus $P(3)$ should equal $P(0)$. In equations this gives $P(1) = 3P(0)$, $P(2) = 3P(0)$, and $P(3) = P(0)$. We also have the equation saying all the weights add to one, $P(0) + P(1) + P(2) + P(3) = 1$. There is one and only one solution to these equations, namely $P(0) = \frac{1}{8}$, $P(1) = \frac{3}{8}$, $P(2) = \frac{3}{8}$, and $P(3) = \frac{1}{8}$. Do you notice a relationship between $P(x)$ and the binomial coefficient $\binom{3}{x}$ here? Can you predict the probabilities of 0, 1, 2, 3, and 4 heads in four flips of a coin?

Together, the last two exercises demonstrate that we must be careful not to apply Theorem 5.2 unless we are using the uniform probability measure.

Important Concepts, Formulas, and Theorems

1. *Sample Space.* We use the phrase *sample space* to refer to the set of possible outcomes of a process.

2. *Event.* A set of elements in a sample space is called an *event*.

3. *Probability.* In order to compute probabilities we assign a *weight* to each element of the sample space so that the weight represents what we believe to be the relative likelihood of that outcome. There are two rules we must follow in assigning weights. First the weights must be nonnegative numbers, and second the sum of the weights of all the elements in a sample space must be one. We define the *probability* $P(E)$ of the event E to be the sum of the weights of the elements of E.

4. *The axioms of Probability.* Three rules that a probability measure on a finite sample space must satisfy could actually be used to define what we mean by probability.

   (a) $P(A) \geq 0$ for any $A \subseteq S$.

   (b) $P(S) = 1$.

   (c) $P(A \cup B) = P(A) + P(B)$ for any two disjoint events A and B.

5. *Probability Distribution.* A function which assigns a probability to each member of a sample space is called a (discrete) *probability distribution.*

6. *Complement.* The *complement* of an event E in a sample space S, denoted by $S - E$, is the set of all outcomes in S but not E.

7. The Probabilities of Complementary Events. If two events E and F are complementary, that is they have nothing in common ($E \cap F = \emptyset$), and their union is the whole sample space ($E \cup F = S$), then
$$P(E) = 1 - P(F).$$

8. *Collision, Collide (in Hashing).* We say two keys *collide* if they hash to the same location.

9. *Uniform Probability Distribution.* We say P is the *uniform probability measure* or *uniform probability distribution* when we assign the same probability to all members of our sample space.

10. *Computing Probabilities with the Uniform Distribution.* Suppose P is the uniform probability measure defined on a sample space S. Then for any event E,
$$P(E) = |E|/|S|,$$
the size of E divided by the size of S. This *does not* apply to general probability distributions.

Problems

1. What is the probability of exactly three heads when you flip a coin five times? What is the probability of three or more heads when you flip a coin five times?

2. When we roll two dice, what is the probability of getting a sum of 4 or less on the tops?

3. If we hash 3 keys into a hash table with ten slots, what is the probability that all three keys hash to different slots? How big does n have to be so that if we hash n keys to a hash table with 10 slots, the probability is at least a half that some slot has at least two keys hash to it? How many keys do we need to have probability at least two thirds that some slot has at least two keys hash to it?

4. What is the probability of an odd sum when we roll three dice?

5. Suppose we use the numbers 2 through 12 as our sample space for rolling two dice and adding the numbers on top. What would we get for the probability of a sum of 2, 3, or 4, if we used the equiprobable measure on this sample space. Would this make sense?

6. Two pennies, a nickel and a dime are placed in a cup and a first coin and a second coin are drawn.

   (a) Assuming we are sampling without replacement (that is, we don't replace the first coin before taking the second) write down the sample space of all ordered pairs of letters P, N, and D that represent the outcomes. What would you say are the appropriate weights for the elements of the sample space?

   (b) What is the probability of getting eleven cents?

7. Why is the probability of five heads in ten flips of a coin equal to $\frac{63}{256}$?

8. Using 5-element sets as a sample space, determine the probability that a "hand" of 5 cards chosen from an ordinary deck of 52 cards will consist of cards of the same suit.

9. Using 5 element permutations as a sample space, determine the probability that a "hand" of 5 cards chosen from an ordinary deck of 52 cards will have all the cards from the same suit

10. How many five-card hands chosen from a standard deck of playing cards consist of five cards in a row (such as the nine of diamonds, the ten of clubs, jack of clubs, queen of hearts, and king of spades)? Such a hand is called a straight. What is the probability that a five-card hand is a straight? Explore whether you get the same answer by using five element sets as your model of hands or five element permutations as your model of hands.

11. A student taking a ten-question, true-false diagnostic test knows none of the answers and must guess at each one. Compute the probability that the student gets a score of 80 or higher. What is the probability that the grade is 70 or lower?

12. A die is made of a cube with a square painted on one side, a circle on two sides, and a triangle on three sides. If the die is rolled twice, what is the probability that the two shapes we see on top are the same?

13. Are the following two events equally likely? Event 1 consists of drawing an ace and a king when you draw two cards from among the thirteen spades in a deck of cards and event 2 consists of drawing an ace and a king when you draw two cards from the whole deck.

14. There is a retired professor who used to love to go into a probability class of thirty or more students and announce "I will give even money odds that there are two people in this classroom with the same birthday." With thirty students in the room, what is the probability that all have different birthdays? What is the minimum number of students

that must be in the room so that the professor has at least probability one half of winning the bet? What is the probability that he wins his bet if there are 50 students in the room. Does this probability make sense to you? (There is no wrong answer to that question!) Explain why or why not.

5.2 Unions and Intersections

The probability of a union of events

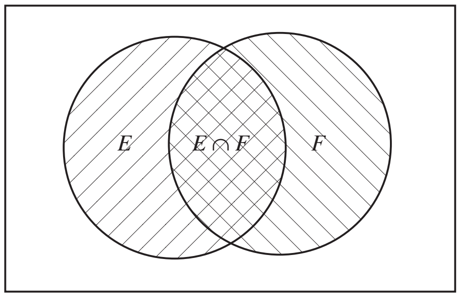
Exercise 5.2-1 If you roll two dice, what is the probability of an even sum or a sum of 8 or more?

Exercise 5.2-2 In Exercise 5.2-1, let E be the event "even sum" and let F be the event "8 or more." We found the probability of the union of the events E and F. Why isn't it the case that $P(E \cup F) = P(E) + P(F)$? What weights appear twice in the sum $P(E) + P(F)$? Find a formula for $P(E \cup F)$ in terms of the probabilities of E, F, and $E \cap F$. Apply this formula to Exercise 5.2-1. What is the value of expressing one probability in terms of three?

Exercise 5.2-3 What is $P(E \cup F \cup G)$ in terms of probabilities of the events E, F, and G and their intersections?

In the sum $P(E) + P(F)$ the weights of elements of $E \cap F$ each appear twice, while the weights of all other elements of $E \cup F$ each appear once. We can see this by looking at a diagram called a Venn Diagram, as in Figure 5.1. In a *Venn diagram*, the rectangle represents the sample space, and the circles represent the events. If we were to shade both E and F, we would wind

Figure 5.1: A Venn diagram for two events.



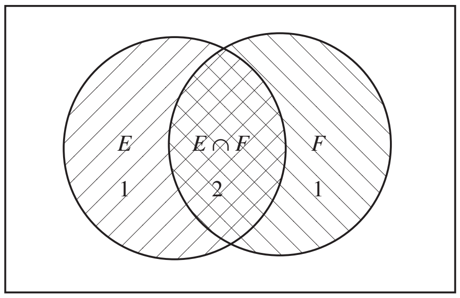
up shading the region $E \cap F$ twice. In Figure 5.2, we represent that by putting numbers in the regions, representing how many times they are shaded. This illustrates why the sum $P(E)+P(F)$ includes the probability weight of each element of $E \cap F$ twice. Thus to get a sum that includes the probability weight of each element of $E \cup F$ exactly once, we have to subtract the weight of $E \cup F$ from the sum $P(E) + P(F)$. This is why

$$P(E \cup F) = P(E) + P(F) - P(E \cap F) \tag{5.4}$$

We can now apply this to Exercise 5.2-1 by noting that the probability of an even sum is 1/2, while the probability of a sum of 8 or more is

$$\frac{1}{36} + \frac{2}{36} + \frac{3}{36} + \frac{4}{36} + \frac{5}{36} = \frac{15}{36}.$$

Figure 5.2: If we shade each of E and F once, then we shade $E \cap F$ twice



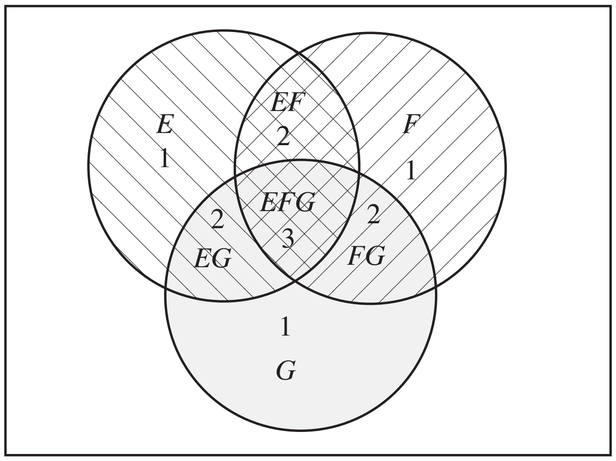
From a similar sum, the probability of an even sum of 8 or more is 9/36, so the probability of a sum that is even or is 8 or more is

$$\frac{1}{2} + \frac{15}{36} - \frac{9}{36} = \frac{2}{3}.$$

(In this case our computation merely illustrates the formula; with less work one could add the probability of an even sum to the probability of a sum of 9 or 11.) In many cases, however, probabilities of individual events and their intersections are more straightforward to compute than probabilities of unions (we will see such examples later in this section), and in such cases our formula is quite useful.

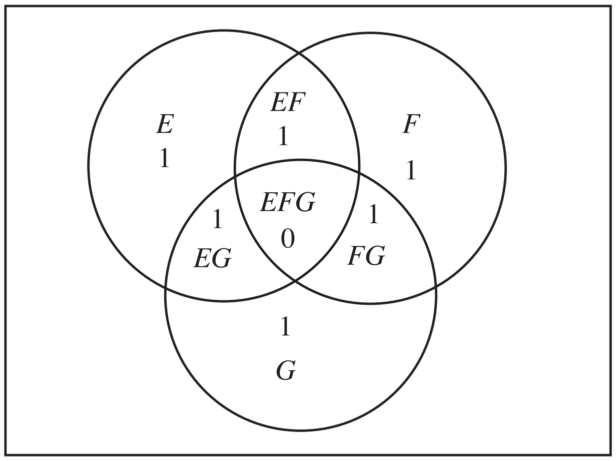
Now let's consider the case for three events and draw a Venn diagram and fill in the numbers for shading all E, F, and G. So as not to crowd the figure we use EF to label the region corresponding to $E \cap F$, and similarly label other regions. Doing so we get Figure 5.3. Thus we

Figure 5.3: The number of ways the intersections are shaded when we shade E, F, and G.



have to figure out a way to subtract from $P(E) + P(F) + P(G)$ the weights of elements in the regions labeled EF, FG and EG once, and the the weight of elements in the region labeled EFG twice. If we subtract out the weights of elements of each of $E \cap F$, $F \cap G$, and $E \cap G$, this does more than we wanted to do, as we subtract the weights of elements in EF, FG and EG once but the weights of elements in of EFG three times, leaving us with Figure 5.4. We then see that

Figure 5.4: The result of removing the weights of each intersection of two sets.



all that is left to do is to add weights of elements in the $E \cap F \cap G$ back into our sum. Thus we have that

$$P(E \cup F \cup G) = P(E) + P(F) + P(G) - P(E \cap F) - P(E \cap G) - P(F \cap G) + P(E \cap F \cap G).$$

Principle of inclusion and exclusion for probability

From the last two exercises, it is natural to guess the formula

$$P(\bigcup_{i=1}^{n} E_i) = \sum_{i=1}^{n} P(E_i) - \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} P(E_i \cap E_j) + \sum_{i=1}^{n-2} \sum_{j=i+1}^{n-1} \sum_{k=j+1}^{n} P(E_i \cap E_j \cap E_k) - \ldots. \tag{5.5}$$

All the sum signs in this notation suggest that we need some new notation to describe sums. We are now going to make a (hopefully small) leap of abstraction in our notation and introduce notation capable of compactly describing the sum described in the previous paragraph. This notation is an extension of the one we introduced in Equation 5.1. We use

$$\sum_{\substack{i_1, i_2, \ldots, i_k: \\ 1 \le i_1 < i_2 < \cdots < i_k \le n}} P(E_{i_1} \cap E_{i_2} \cap \cdots E_{i_k}) \tag{5.6}$$

to stand for the sum, over all sequences $i_1, i_2, \ldots i_k$ of integers between 1 and n of the probabilities of the sets $E_{i_1} \cap E_{i_2} \ldots \cap E_{i_k}$. More generally, $\sum_{\substack{i_1, i_2, \ldots, i_k: \\ 1 \le i_1 < i_2 < \cdots < i_k \le n}} f(i_1, i_2, \ldots, i_k)$ is the sum of $f(i_1, i_2, \ldots, i_k)$ over all increasing sequences of k numbers between 1 and n.

Exercise 5.2-4 To practice with notation, what is $\sum_{\substack{i_1, i_2, i_3: \\ 1 \le i_1 < i_2 < i_3 \le 4}} i_1 + i_2 + i_3$?

The sum in Exercise 5.2-4 is $1+2+3+1+2+4+1+3+4+2+3+4 = 3(1+2+3+4) = 30$.

With this understanding of the notation in hand, we can now write down a formula that captures the idea in Equation 5.5 more concisely. Notice that in Equation 5.5 we include probabilities of single sets with a plus sign, probabilities of intersections of two sets with a minus sign, and in general, probabilities of intersections of any even number of sets with a minus sign and probabilities of intersections of any odd number of sets (including the odd number one) with a plus sign. Thus if we are intersecting k sets, the proper coefficient for the probability of the intersection of these sets is $(-1)^{k+1}$ (it would be equally good to use $(-1)^{k-1}$, and correct but silly to use $(-1)^{k+3}$). This lets us translate the formula of Equation 5.5 to Equation 5.7 in the theorem, called the *Principle of Inclusion and Exclusion for Probability*, that follows. We will give two completely different proofs of the theorem, one of which is a nice counting argument but is a bit on the abstract side, and one of which is straightforward induction, but is complicated by the fact that it takes a lot of notation to say what is going on.

Theorem 5.3 (Principle of Inclusion and Exclusion for Probability) *The probability of the union $E_1 \cup E_2 \cup \cdots \cup E_n$ of events in a sample space S is given by*

$$P\left(\bigcup_{i=1}^{n} E_i\right) = \sum_{k=1}^{n} (-1)^{k+1} \sum_{\substack{i_1,i_2,\ldots,i_k:\\ 1\le i_1<i_2<\cdots<i_k\le n}} P\left(E_{i_1} \cap E_{i_2} \cap \cdots \cap E_{i_k}\right). \tag{5.7}$$

First Proof: Consider an element x of $\bigcup_{i=1}^{n} E_i$. Let $E_{i_1}, E_{i_2}, \ldots E_{i_k}$ be the set of all events E_i of which x is a member. Let $K = \{i_1, i_2, \ldots, i_k\}$. Then x is in the event $E_{j_1} \cap E_{j_2} \cap \cdots \cap E_{j_m}$ if and only if $\{j_1, j_2 \ldots j_m\} \subseteq K$. Why is this? If there is a j_r that is not in K, then $x \notin E_j$ and thus $x \notin E_{j_1} \cap E_{j_2} \cap \cdots \cap E_{j_m}$. Notice that every x in $\bigcup_{i=1}^{n} E_i$ is in at least one E_i, so it is in at least one of the sets $E_{i_1} \cap E_{i_2} \cap \cdots \cap E_{i_k}$.

Recall that we define $P\left(E_{j_1} \cap E_{j_2} \cap \cdots \cap E_{j_m}\right)$ to be the sum of the probability weights $p(x)$ for $x \in E_{j_1} \cap E_{j_2} \cap \cdots \cap E_{j_m}$. Suppose we substitute this sum of probability weights for $P\left(E_{j_1} \cap E_{j_2} \cap \cdots \cap E_{j_m}\right)$ on the right hand side of Equation 5.7. Then the right hand side becomes a sum of terms each of with is plus or minus a probability weight. The sum of all the terms involving $p(x)$ on the right hand side of Equation 5.7 includes a term involving $p(x)$ for each nonempty subset $\{j_1, j_2, \ldots, j_m\}$ of K, and no other terms involving $p(x)$. The coefficient of the probability weight $p(x)$ in the term for the subset $\{j_1, j_2, \ldots, j_m\}$ is $(-1)^{m+1}$. Since there are $\binom{k}{m}$ subsets of K of size m, the sum of the terms involving $p(x)$ will therefore be

$$\sum_{m=1}^{k} (-1)^{m+1}\binom{k}{m}p(x) = \left(-\sum_{m=0}^{k}(-1)^m\binom{k}{m}p(x)\right) + p(x) = 0\cdot p(x) + p(x) = p(x),$$

because $k \geq 1$ and thus by the binomial theorem, $\sum_{j=0}^{k}\binom{k}{j}(-1)^j = (1-1)^k = 0$. This proves that for each x, the sum of all the terms involving $p(x)$ after we substitute the sum of probability weights into Equation 5.7 is exactly $p(x)$. We noted above that for every x in $\cup_{i=1}^{n} E_i$ appears in at least one of the sets $E_{i_1} \cap E_{i_2} \cap \cdots \cap E_{i_k}$. Thus the right hand side of Equation 5.7 is the sum of every $p(x)$ such that x is in $\cup_{i=1}^{n} E_i$. By definition, this is the left-hand side of Equation 5.7. ■

Second Proof: The proof is simply an application of mathematical induction using Equation 5.4. When $n = 1$ the formula is true because it says $P(E_1) = P(E_1)$. Now suppose inductively that for any family of $n-1$ sets $F_1, F_2, \ldots, F_{n-1}$

$$P\left(\bigcup_{i=1}^{n-1} F_i\right) = \sum_{k=1}^{n-1} (-1)^{k+1} \sum_{\substack{i_1,i_2,i\ldots,i_k:\\ 1\le i_1<i_2<\cdots<i_k\le n-1}} P(F_{i_1}\cap F_{i_2}\cap F_{i_k}) \tag{5.8}$$

If in Equation 5.4 we let $E = E_1 \cup \ldots \cup E_{n-1}$ and $F = E_m$, we may apply Equation 5.4 to to compute $P\left(\cup_{i=1}^n E_i\right)$ as follows:

$$P\left(\bigcup_{i=1}^{n} E_i\right) = P\left(\bigcup_{i=1}^{n-1} E_i\right) + P(E_n) - P\left(\left(\bigcup_{i=1}^{n-1} E_i\right)\cap E_n\right). \tag{5.9}$$

By the distributive law,

$$\left(\bigcup_{i=1}^{n-1} E_i\right)\cap E_n = \bigcup_{i=1}^{n-1}\left(E_i\cap E_n\right),$$

and substituting this into Equation 5.9 gives

$$P\left(\bigcup_{i=1}^{n} E_i\right) = P\left(\bigcup_{i=1}^{n-1} E_i\right) + P\left(E_n\right) - P\left(\bigcup_{i=1}^{n-1}(E_i\cap E_n)\right).$$

Now we use the inductive hypothesis (Equation 5.8) in two places to get

$$\begin{aligned} P\left(\bigcup_{i=1}^{n} E_i\right) &= \sum_{k=1}^{n-1} (-1)^{k+1} \sum_{\substack{i_1,i_2,i\ldots,i_k:\\ 1\le i_1<i_2<\cdots<i_k\le n-1}} P(E_{i_1}\cap E_{i_2}\cap\cdots\cap E_{i_k}) \\ &+ P(E_n) \\ &- \sum_{k=1}^{n-1} (-1)^{k+1} \sum_{\substack{i_1,i_2,i\ldots,i_k:\\ 1\le i_1<i_2<\cdots<i_k\le n-1}} P(E_{i_1}\cap E_{i_2}\cap\cdots\cap E_{i_k}\cap E_n). \end{aligned} \tag{5.10}$$

The first summation on the right hand side sums $(-1)^{k+1}P\left(E_{i_1}\cap E_{i_2}\cap\cdots\cap E_{i_k}\right)$ over all lists $i_1, i_2, \ldots, i_k$ that *do not* contain n, while the $P(E_n)$ and the second summation work together to sum $(-1)^{k+1}P\left(E_{i_1}\cap E_{i_2}\cap\cdots\cap E_{i_k}\right)$ over all lists $i_1, i_2, \ldots, i_k$ that *do* contain n. Therefore,

$$P\left(\bigcup_{i=1}^{n} E_i\right) = \sum_{k=1}^{n} (-1)^{k+1} \sum_{\substack{i_1,i_2,i\ldots,i_k:\\ 1\le i_1<i_2<\cdots<i_k\le n}} P(E_{i_1}\cap E_{i_2}\cap\cdots\cap E_{i_k}).$$

Thus by the principle of mathematical induction, this formula holds for all integers $n > 0$. ■

Exercise 5.2-5 At a fancy restaurant n students check their backpacks. They are the only ones to check backpacks. A child visits the checkroom and plays with the check tickets for the backpacks so they are all mixed up. If there are 5 students named Judy, Sam, Pat, Jill, and Jo, in how many ways may the backpacks be returned so that Judy gets her own backpack (and maybe some other students do, too)? What is the probability that this happens? What is the probability that Sam gets his backpack (and maybe some other students do, too)? What is the probability that Judy and Sam both get their own backpacks (and maybe some other students do, too)? For any particular two element set of students, what is the probability that these two students get their own backpacks (and maybe some other students do, too)? What is the probability that at least one student gets his or her own backpack? What is the probability that no students get their own backpacks? What do you expect the answer will be for the last two questions for n students? This classic problem is often stated using hats rather than backpacks (quaint, isn't it?), so it is called the *hatcheck problem.* It is also known as the *derangement problem*; a *derangement* of a set being a one-to-one function from a set onto itself (i.e., a bijection) that sends each element to something not equal to it.

For Exercise 5.2-5, let E_i be the event that person i on our list gets the right backpack. Thus E_1 is the event that Judy gets the correct backpack and E_2 is the event that Sam gets the correct backpack. The event $E_1 \cap E_2$ is the event that Judy *and* Sam get the correct backpacks (and maybe some other people do). In Exercise 5.2-5, there are 4! ways to pass back the backpacks so that Judy gets her own, as there are for Sam or any other single student. Thus $P(E_1) = P(E_i) = \frac{4!}{5!}$. For any particular two element subset, such as Judy and Sam, there are 3! ways that these two people may get their backpacks back. Thus, for each i and j, $P(E_i \cap E_j) = \frac{3!}{5!}$. For a particular k students the probability that each one of these k students gets his or her own backpack back is $(5-k)!/5!$. If E_i is the event that student i gets his or her own backpack back, then the probability of an intersection of k of these events is $(5-k)!/5!$ The probability that at least one person gets his or her own backpack back is the probability of $E_1 \cup E_2 \cup E_3 \cup E_4 \cup E_5$. Then by the principle of inclusion and exclusion, the probability that at least one person gets his or her own backpack back is

$$P(E_1 \cup E_2 \cup E_3 \cup E_4 \cup E_5) = \sum_{k=1}^{5} (-1)^{k+1} \sum_{\substack{i_1, i_2, \ldots, i_k: \\ 1 \le i_1 < i_2 < \cdots < i_k \le 5}} P(E_{i_1} \cap E_{i_2} \cap \cdots \cap E_{i_k}). \tag{5.11}$$

As we argued above, for a set of k people, the probability that all k people get their backpacks back is $(5-k)!/5!$. In symbols, $P(E_{i_1} \cap E_{i_2} \cap \cdots \cap E_{i_k}) = \frac{(5-k)!}{5!}$. Recall that there are $\binom{5}{k}$ sets of k people chosen from our five students. That is, there are $\binom{5}{k}$ lists $i_1, i_2, \ldots i_k$ with $1 < i_1 < i_2 < \cdots < i_k \le 5$. Thus, we can rewrite the right hand side of the Equation 5.11 as

$$\sum_{k=1}^{5} (-1)^{k+1} \binom{5}{k} \frac{(5-k)!}{5!}.$$

This gives us

$$\begin{aligned}
P(E_1 \cup E_2 \cup E_3 \cup E_4 \cup E_5) &= \sum_{k=1}^{5} (-1)^{k-1} \binom{5}{k} \frac{(5-k)!}{5!} \\
&= \sum_{k=1}^{5} (-1)^{k-1} \frac{5!}{k!(5-k)!} \frac{(5-k)!}{5!} \\
&= \sum_{k=1}^{5} (-1)^{k-1} \frac{1}{k!} \\
&= 1 - \frac{1}{2} + \frac{1}{3!} - \frac{1}{4!} + \frac{1}{5!}.
\end{aligned}$$

The probability that nobody gets his or her own backpack is 1 minus the probability that someone does, or

$$\frac{1}{2} - \frac{1}{3!} + \frac{1}{4!} - \frac{1}{5!}$$

To do the general case of n students, we simply substitute n for 5 and get that the probability that at least one person gets his or her own backpack is

$$\sum_{i=1}^{n}(-1)^{i-1}\frac{1}{i!} = 1 - \frac{1}{2} + \frac{1}{3!} - \cdots + \frac{(-1)^{n-1}}{n!}$$

and the probability that nobody gets his or her own backpack is 1 minus the probability above, or

$$\sum_{i=2}^{n}(-1)^{i}\frac{1}{i!} = \frac{1}{2} - \frac{1}{3!} + \cdots + \frac{(-1)^{n}}{n!} \tag{5.12}$$

Those who have had power series in calculus may recall the power series representation of e^x, namely

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots = \sum_{i=0}^{\infty}\frac{x^i}{i!}.$$

Thus the expression in Equation 5.12 is the approximation to e^{-1} we get by substituting -1 for x in the power series and stopping the series at $i = n$. Note that the result depends very "lightly" on n; so long as we have at least four or five people, no matter how many people we have, the probability that no one gets their hat back remains at roughly e^{-1}. Our intuition might have suggested that as the number of students increases, the probability that *someone* gets his or her own backpack back approaches 1 rather than $1 - e^{-1}$. Here is another example of why it is important to use computations with the rules of probability instead of intuition!

The Principle of Inclusion and Exclusion for Counting

Exercise 5.2-6 How many functions are there from an n-element set N to a k-element set $K = \{y_1, y_2, \ldots y_k\}$ that map nothing to y_1? Another way to say this is if I have n distinct candy bars and k children Sam, Mary, Pat, etc., in how ways may I pass out the candy bars so that Sam doesn't get any candy (and maybe some other children don't either)?

Exercise 5.2-7 How many functions map nothing to a j-element subset J of K? Another way to say this is if I have n distinct candy bars and k children Sam, Mary, Pat, etc., in how ways may I pass out the candy bars so that some particular j-element subset of the children don't get any (and maybe some other children don't either)?

Exercise 5.2-8 What is the number of functions from an n-element set N to a k element set K that map nothing to at least one element of K? Another way to say this is if I have n distinct candy bars and k children Sam, Mary, Pat, etc., in how ways may I pass out the candy bars so that some child doesn't get any (and maybe some other children don't either)?

Exercise 5.2-9 On the basis of the previous exercises, how many functions are there from an n-element set onto a k element set?

The number of functions from an n-element set to a k-element set $K = \{y_1, y_2, \ldots y_k\}$ that map nothing to y_1 is simply $(k-1)^n$ because we have $k-1$ choices of where to map each of our n elements. Similarly the number that map nothing to a particular set J of j elements will be $(k-j)^n$. This warms us up for Exercise 5.2-8.

In Exercise 5.2-8 we need an analog of the principle of inclusion and exclusion for the size of a union of k sets (set i being the set of functions that map nothing to element i of the set K). Because we can make the same argument about the size of the union of two or three sets that we made about probabilities of unions of two or three sets, we have a very natural analog. That analog is the *Principle of Inclusion and Exclusion for Counting*

$$\left|\bigcup_{i=1}^{n} E_i\right| = \sum_{k=1}^{n}(-1)^{k+1} \sum_{\substack{i_1,i_2,\ldots,i_k:\\ 1\le i_1<i_2<\cdots<i_k\le n}} \left|E_{i_1}\cap E_{i_2}\cap\cdots\cap E_{i_k}\right|. \tag{5.13}$$

In fact, this formula is proved by induction or a counting argument in virtually the same way. Applying this formula to the number of functions from N that map nothing to at least one element of K gives us

$$|\bigcup_{i=1}^{k} E_i| = \sum_{k=1}^{n}(-1)^{k+1} \sum_{\substack{i_1,i_2,\ldots,i_k:\\ 1\le i_1<i_2<\cdots<i_k\le n}} |E_{i_1}\cap E_{i_2}\cap\cdots\cap E_{i_k}| = \sum_{j=1}^{k}(-1)^{j-1}\binom{k}{j}(k-j)^n.$$

This is the number of functions from N that map nothing to at least one element of K. The total number of functions from N to K is k^n. Thus the number of onto functions is

$$k^n - \sum_{j=1}^{k}(-1)^{j-1}\binom{k}{j}(k-j)^n = \sum_{j=0}^{k}(-1)^{j}\binom{k}{j}(k-j)^n,$$

where the second equality results because $\binom{k}{0}$ is 1 and $(k-0)^n$ is k^n.

Important Concepts, Formulas, and Theorems

1. *Venn diagram.* To draw a *Venn diagram*, for two or three sets, we draw a rectangle that represents the sample space, and two or three mutually overlapping circles to represent the events.

2. *Probability of a union of two events.* $P(E\cup F) = P(E) + P(F) - P(E\cap F)$

3. *Probability of a union of three events.* $P(E\cup F\cup G) = P(E) + P(F) + P(G) - P(E\cap F) - P(E\cap G) - P(F\cap G) + P(E\cap F\cap G)$.

4. A summation notation. $\displaystyle\sum_{\substack{i_1,i_2,\ldots,i_k:\\ 1\le i_1<i_2<\cdots<i_k\le n}} f(i_1,i_2,\ldots,i_k)$ is the sum of $f(i_1,i_2,\ldots,i_k)$ over all increasing sequences of k numbers between 1 and n.

5. *Principle of Inclusion and Exclusion for Probability.* The probability of the union $E_1\cup E_2\cup\cdots\cup E_n$ of events in a sample space S is given by

$$P\left(\bigcup_{i=1}^{n} E_i\right) = \sum_{k=1}^{n}(-1)^{k+1} \sum_{\substack{i_1,i_2,\ldots,i_k:\\ 1\le i_1<i_2<\cdots<i_k\le n}} P\left(E_{i_1}\cap E_{i_2}\cap\cdots\cap E_{i_k}\right).$$

6. *Hatcheck Problem.* The *hatcheck problem* or *derangement problem* asks for the probability that a bijection of an n element set maps no element to itself. The answer is

$$\sum_{i=2}^{n}(-1)^i\frac{1}{i!} = \frac{1}{2} - \frac{1}{3!} + \cdots + \frac{(-1)^n}{n!},$$

the result of truncating the power series expansion of e^{-1} at the $\frac{(-1)^n}{n!}$. Thus the result is very close to $\frac{1}{e}$, even for relatively small values of n.

7. *Principle of Inclusion and Exclusion for Counting.* The *Principle of inclusion and exclusion for counting* says that

$$\left|\bigcup_{i=1}^{n} E_i\right| = \sum_{k=1}^{n}(-1)^{k+1} \sum_{\substack{i_1,i_2,\ldots,i_k:\\ 1\le i_1<i_2<\cdots<i_k\le n}} \left|E_{i_1} \cap E_{i_2} \cap \cdots \cap E_{i_k}\right|.$$

Problems

1. Compute the probability that in three flips of a coin the coin comes heads on the first flip or on the last flip.

2. The eight kings and queens are removed from a deck of cards and then two of these cards are selected. What is the probability that the king or queen of spades is among the cards selected?

3. Two dice are rolled. What is the probability that we see a die with six dots on top?

4. A bowl contains two red, two white and two blue balls. We remove two balls. What is the probability that at least one is red or white? Compute the probability that at least one is red.

5. From an ordinary deck of cards, we remove one card. What is the probability that it is an Ace, is a diamond, or is black?

6. Give a formula for the probability of $P(E \cup F \cup G \cup H)$ in terms of the probabilities of E,F, G, and H, and their intersections.

7. What is

$$\sum_{\substack{i_1,i_2,i_3:\\ 1\le i_1<i_2<i_3\le 4}} i_1 i_2 i_3 \ ?$$

8. What is

$$\sum_{\substack{i_1,i_2,i_3:\\ 1\le i_1<i_2<i_3\le 5}} i_1 + i_2 + i_3 \ ?$$

9. The boss asks the secretary to stuff n letters into envelopes forgetting to mention that he has been adding notes to the letters and in the process has rearranged the letters but not the envelopes. In how many ways can the letters be stuffed into the envelopes so that nobody gets the letter intended for him or her? What is the probability that nobody gets the letter intended for him or her?

10. If we are hashing n keys into a hash table with k locations, what is the probability that every location gets at least one key?

11. From the formula for the number of onto functions, find a formula for $S(n,k)$ which is defined in Problem 12 of Section 1.4. These numbers are called *Stirling numbers (of the second kind).*

12. If we roll 8 dice, what is the probability that each of the numbers 1 through 6 appear on top at least once? What about with 9 dice?

13. Explain why the number of ways of distributing k identical apples to n children is $\binom{n+k-1}{k}$. In how many ways may you distribute the apples to the children so that Sam gets more than m? In how many ways may you distribute the apples to the children so that no child gets more than m?

14. A group of n married couples sits a round a circular table for a group discussion of marital problems. The counselor assigns each person to a seat at random. What is the probability that no husband and wife are side by side?

15. Suppose we have a collection of m objects and a set P of p "properties," an undefined term, that the objects may or may not have. For each subset S of the set P of all properties, define $N_a(S)$ (a is for "at least") to be the number of objects in the collection that have at least the properties in S. Thus, for example, $N_a(\emptyset) = m$. In a typical application, formulas for $N_a(S)$ for other sets $S \subseteq P$ are not difficult to figure out. Define $N_e(S)$ to be the number of objects in our collection that have exactly the properties in S. Show that

$$N_e(\emptyset) = \sum_{K:K\subseteq P} (-1)^{|K|} N_a(K).$$

Explain how this formula could be used for computing the number of onto functions in a more direct way than we did it using unions of sets. How would this formula apply to Problem 9 in this section?

16. In Problem 14 of this section we allow two people of the same sex to sit side by side. If we require in addition to the condition that no husband and wife are side by side the condition that no two people of the same sex are side by side, we obtain a famous problem known as the *mènage* problem. Solve this problem.

17. In how many ways may we place n distinct books on j shelves so that shelf one gets at least m books? (See Problem 7 in Section 1.4.) In how many ways may we place n distinct books on j shelves so that no shelf gets more than m books?

18. In Problem 15 in this section, what is the probability that an object has none of the properties, assuming all objects to be equally likely? How would this apply the Problem 5-10?

5.3 Conditional Probability and Independence

Conditional Probability

Two cubical dice each have a triangle painted on one side, a circle painted on two sides and a square painted on three sides. Applying the principal of inclusion and exclusion, we can compute that the probability that we see a circle on at least one top when we roll them is $1/3 + 1/3 - 1/9 = 5/9$. We are experimenting to see if reality agrees with our computation. We throw the dice onto the floor and they bounce a few times before landing in the next room.

Exercise 5.3-1 Our friend in the next room tells us both top sides are the same. Now what is the probability that our friend sees a circle on at least one top?

Intuitively, it may seem as if the chance of getting circles ought to be four times the chance of getting triangles, and the chance of getting squares ought to be nine times as much as the chance of getting triangles. We could turn this into the algebraic statements that $P(\text{circles}) = 4P(\text{triangles})$ and $P(\text{squares}) = 9P(\text{triangles})$. These two equations and the fact that the probabilities sum to 1 would give us enough equations to conclude that the probability that our friend saw two circles is now $2/7$. But does this analysis make sense? To convince ourselves, let us start with a sample space for the original experiment and see what natural assumptions about probability we can make to determine the new probabilities. In the process, we will be able to replace intuitive calculations with a formula we can use in similar situations. This is a good thing, because we have already seen situations where our intuitive idea of probability might not always agree with what the rules of probability give us.

Let us take as our sample space for this experiment the ordered pairs shown in Table 5.2 along with their probabilities.

Table 5.2: Rolling two unusual dice

TT	TC	TS	CT	CC	CS	ST	SC	SS
$\frac{1}{36}$	$\frac{1}{18}$	$\frac{1}{12}$	$\frac{1}{18}$	$\frac{1}{9}$	$\frac{1}{6}$	$\frac{1}{12}$	$\frac{1}{6}$	$\frac{1}{4}$

We know that the event {TT, CC, SS} happened. Thus we would say while it used to have probability

$$\frac{1}{36} + \frac{1}{9} + \frac{1}{4} = \frac{14}{36} = \frac{7}{18} \tag{5.14}$$

this event now has probability 1. Given that, what probability would we now assign to the event of seeing a circle? Notice that the event of seeing a circle now has become the event CC. Should we expect CC to become more or less likely in comparison than TT or SS just because we know now that one of these three outcomes has occurred? Nothing has happened to make us expect that, so whatever new probabilities we assign to these two events, they should have the same ratios as the old probabilities.

Multiplying all three old probabilities by $\frac{18}{7}$ to get our new probabilities will preserve the ratios and make the three new probabilities add to 1. (Is there any other way to get the three new probabilities to add to one and make the new ratios the same as the old ones?) This gives us that the probability of two circles is $\frac{1}{9} \cdot \frac{18}{7} = \frac{2}{7}$. Notice that nothing we have learned about probability so far told us what to do; we just made a decision based on common sense. When faced with similar situations in the future, it would make sense to use our common sense in the same way. However, do we really need to go through the process of constructing a new sample space and reasoning about its probabilities again? Fortunately, our entire reasoning process can be captured in a formula. We wanted the probability of an event E given that the event F happened. We figured out what the event $E \cap F$ was, and then multiplied its probability by $1/P(F)$. We summarize this process in a definition.

We define the *conditional probability* of E given F, denoted by $P(E|F)$ and read as "the probability of E given F" by

$$P(E|F) = \frac{P(E \cap F)}{P(F)}. \tag{5.15}$$

Then whenever we want the probability of E knowing that F has happened, we compute $P(E|F)$. (If $P(F) = 0$, then we cannot divide by $P(F)$, but F gives us no new information about our situation. For example if the student in the next room says "A pentagon is on top," we have no information except that the student isn't looking at the dice we rolled! Thus we have no reason to change our sample space or the probability weights of its elements, so we define $P(E|F) = P(E)$ when $P(F) = 0$.)

Notice that we did not prove that the probability of E given F is what we said it is; we simply defined it in this way. That is because in the process of making the derivation we made an additional assumption that the relative probabilities of the outcomes in the event F don't change when F happens. This assumption led us to Equation 5.15. Then we chose that equation as our definition of the new concept of the conditional probability of E given F.[4]

In the example above, we can let E be the event that there is more than one circle and F be the event that both dice are the same. Then $E \cap F$ is the event that both dice are circles, and $P(E \cap F)$ is , from the table above, $\frac{1}{9}$. $P(F)$ is, from Equation 5.14, $\frac{7}{18}$. Dividing, we get the probability of $P(E|F)$, which is $\frac{1}{9}/\frac{7}{18} = \frac{2}{7}$.

Exercise 5.3-2 When we roll two ordinary dice, what is the probability that the sum of the tops comes out even, given that the sum is greater than or equal to 10? Use the definition of conditional probability in solving the problem.

Exercise 5.3-3 We say E is *independent* of F if $P(E|F) = P(E)$. Show that when we roll two dice, one red and one green, the event "The total number of dots on top is odd" is independent of the event "The red die has an odd number of dots on top."

Exercise 5.3-4 Sometimes information about conditional probabilities is given to us indirectly in the statement of a problem, and we have to derive information about other probabilities or conditional probabilities. Here is such an example. If a student knows 80% of the material in a a course, what do you expect her grade to be on a (well-balanced) 100 question short-answer test about the course? What is the probability that she answers a question correctly on a 100 question true-false test if she guesses at each question she does not know the answer to? (We assume that she knows what

[4]For those who like to think in terms of axioms of probability, we could give an axiomatic definition of conditional probability, and one of our axioms would be that for events E_1 and E_2 that are subsets of F, the ratio of the conditional probabilities $P(E_1|F)$ and $P(E_2|F)$ is the same as the ratio of $P(E)$ and $P(F)$.

she knows, that is, if she thinks that she knows the answer, then she really does.) What do you expect her grade to be on a 100 question True-False test to be?

For Exercise 5.3-2 let's let E be the event that the sum is even and F be the event that the sum is greater than or equal to 10. Thus referring to our sample space in Exercise 5.3-2, $P(F) = 1/6$ and $P(E \cap F) = 1/9$, since it is the probability that the roll is either 10 or 12. Dividing these two we get 2/3.

In Exercise 5.3-3, the event that the total number of dots is odd has probability 1/2. Similarly, given that the red die has an odd number of dots, the probability of an odd sum is 1/2 since this event corresponds exactly to getting an even roll on the green die. Thus, by the definition of independence, the event of an odd number of dots on the red die and the event that the total number of dots is odd are independent.

In Exercise 5.3-4, if a student knows 80% of the material in a course, we would hope that her grade on a well-designed test of the course would be around 80%. But what if the test is a True-False test? Let R be the event that she gets the right answer, K be the event that she knows that right answer and $\overline{K}$ be the event that she guesses. Then $R = P(R \cap K) + P(R \cap \overline{K})$. Since R is a union of two disjoint events, its probability would be the sum of the probabilities of these two events. How do we get the probabilities of these two events? The statement of the problem gives us implicitly the conditional probability that she gets the right answer given that she knows the answer, namely one, and the probability that she gets the right answer if she doesn't know the answer, namely 1/2. Using Equation 5.15, we see that we use the equation

$$P(E \cap F) = P(E|F)P(F) \tag{5.16}$$

to compute $P(R \cap K)$ and $P(R \cap \overline{K})$, since the problem tells us directly that $P(K) = .8$ and $P(\overline{K}) = .2$. In symbols,

$$\begin{aligned} P(R) &= P(R \cap K) + P(R \cap \overline{K}) \\ &= P(R|K)P(K) + P(R|\overline{K})P(\overline{K}) \\ &= 1 \cdot .8 + .5 \cdot .2 = .9 \end{aligned}$$

We have shown that the probability that she gets the right answer is .9. Thus we would expect her to get a grade of 90%.

Independence

We said in Exercise 5.3-3 that E is independent of F if $P(E|F) = P(E)$. The *product principle for independent probabilities* (Theorem 5.4) gives another test for independence.

Theorem 5.4 *Suppose E and F are events in a sample space. Then E is independent of F if and only if $P(E \cap F) = P(E)P(F)$.*

Proof: First consider the case when F is non-empty. Then, from our definition in Exercise 5.3-3

$$E \text{ is independent of } F \quad \Leftrightarrow \quad P(E|F) = P(E).$$

(Even though the definition only has an "if", recall the convention of using "if" in definitions, even though "if and only if" is meant.) Using the definition of $P(E|F)$ in Equation 5.15, in the right side of the above equation we get

$$\begin{aligned} & P(E|F) = P(E) \\ \Leftrightarrow \quad & \frac{P(E \cap F)}{P(F)} = P(E) \\ \Leftrightarrow \quad & P(E \cap F) = P(E)P(F). \end{aligned}$$

Since every step in this proof was an if and only if statement we have completed the proof for the case when F is non-empty.

If F is empty, then E is independent of F and both $P(E)P(F)$ and $P(E \cap F)$ are zero. Thus in this case as well, E is independent of F if and only if $P(E \cap F) = P(E)P(F)$. ■

Corollary 5.5 *E is independent of F if and only if F is independent of E.*

When we flip a coin twice, we think of the second outcome as being independent of the first. It would be a sorry state of affairs if our definition of independence did not capture this intuitive idea! Let's compute the relevant probabilities to see if it does. For flipping a coin twice our sample space is $\{HH, HT, TH, TT\}$ and we weight each of these outcomes 1/4. To say the second outcome is independent of the first, we must mean that getting an H second is independent of whether we get an H or a T first, and same for getting a T second. This gives us that $P(H \text{ first}) = 1/4 + 1/4 = 1/2$ and $P(H \text{ second}) = 1/2$, while $P(\text{H first and H second}) = 1/4$. Note that

$$P(H \text{ first})P(H \text{ second}) = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4} = P(\text{H first and H second}).$$

By Theorem 5.4, this means that the event "H second" is independent of the event "H first." We can make a similar computation for each possible combination of outcomes for the first and second flip, and so we see that our definition of independence captures our intuitive idea of independence in this case. Clearly the same sort of computation applies to rolling dice as well.

Exercise 5.3-5 What sample space and probabilities have we been using when discussing hashing? Using these, show that the event "key i hashes to position p" and the event "key j hashes to position q" are independent when $i \neq j$. Are they independent if $i = j$?

In Exercise 5.3-5 if we have a list of n keys to hash into a table of size k, our sample space consists of all n-tuples of numbers between 1 and k. The event that key i hashes to some number p consists of all n-tuples with p in the ith position, so its probability is $\left(\frac{1}{k}\right)^{n-1} / \left(\frac{1}{k}\right)^{n} = \frac{1}{k}$. The probability that key j hashes to some number q is also $\frac{1}{k}$. If $i \neq j$, then the event that key i hashes to p and key j hashes to q has probability $\left(\frac{1}{k}\right)^{n-2} / \left(\frac{1}{k}\right)^{n} = \left(\frac{1}{k}\right)^{2}$, which is the product of the probabilities that key i hashes to p and key j hashes to q, so these two events are independent. However if $i = j$ the probability of key i hashing to p and key j hashing to q is zero unless $p = q$, in which case it is 1. Thus if $i = j$, these events are not independent.

Independent Trials Processes

Coin flipping and hashing are examples of what are called "independent trials processes." Suppose we have a process that occurs in stages. (For example, we might flip a coin n times.) Let us use x_i to denote the outcome at stage i. (For flipping a coin n times, $x_i = H$ means that the outcome of the ith flip is a head.) We let S_i stand for the set of possible outcomes of stage i. (Thus if we flip a coin n times, $S_i = \{H, T\}$.) A process that occurs in stages is called an **independent trials process** if for each sequence $a_1, a_2, \ldots, a_n$ with $a_i \in S_i$,

$$P(x_i = a_i | x_1 = a_1, \ldots, x_{i-1} = a_{i-1}) = P(x_i = a_i).$$

In other words, if we let E_i be the event that $x_i = a_i$, then

$$P(E_i | E_1 \cap E_2 \cap \cdots \cap E_{i-1}) = P(E_i).$$

By our product principle for independent probabilities, this implies that

$$P(E_1 \cap E_2 \cap \cdots E_{i-1} \cap E_i) = P(E_1 \cap E_2 \cap \cdots E_{i-1}) P(E_i). \tag{5.17}$$

Theorem 5.6 *In an independent trials process the probability of a sequence $a_1, a_2, \ldots, a_n$ of outcomes is $P(\{a_1\})P(\{a_2\}) \cdots P(\{a_n\})$.*

Proof: We apply mathematical induction and Equation 5.17. ■

How do independent trials relate to coin flipping? Here our sample space consists of sequences of n Hs and Ts, and the event that we have an H (or T) on the ith flip is independent of the event that we have an H (or T) on each of the first $i-1$ flips. In particular, the probability of an H on the ith flip is $2^{n-1}/2^n = .5$, and the probability of an H on the ith flip, given a particular sequence on the first $i-1$ flips is $2^{n-i-1}/2^{n-i} = .5$.

How do independent trials relate to hashing a list of keys? As in Exercise 5.3-5 if we have a list of n keys to hash into a table of size k, our sample space consists of all n-tuples of numbers between 1 and k. The probability that key i hashes to p and keys 1 through $i-1$ hash to q_1, q_2,… q_{i-1} is $\left(\frac{1}{k}\right)^{n-i} / \left(\frac{1}{k}\right)^n$ and the probability that keys 1 through $i-1$ hash to q_1, q_2,… q_{i-1} is $\left(\frac{1}{k}\right)^{n-i+1} / \left(\frac{1}{k}\right)^n$. Therefore

$$P(\text{key } i \text{ hashes to } p | \text{keys 1 through } i-1 \text{ hash to } q_1,\ q_2, \ldots q_{i-1}) = \frac{\left(\frac{1}{k}\right)^{n-i} / \left(\frac{1}{k}\right)^n}{\left(\frac{1}{k}\right)^{n-i+1} / \left(\frac{1}{k}\right)^n} = \frac{1}{k}.$$

Therefore, the event that key i hashes to some number p is independent of the event that the first $i-1$ keys hash to some numbers q_1, q_2,… q_{i-1}. Thus our model of hashing is an independent trials process.

Exercise 5.3-6 Suppose we draw a card from a standard deck of 52 cards, replace it, draw another card, and continue for a total of ten draws. Is this an independent trials process?

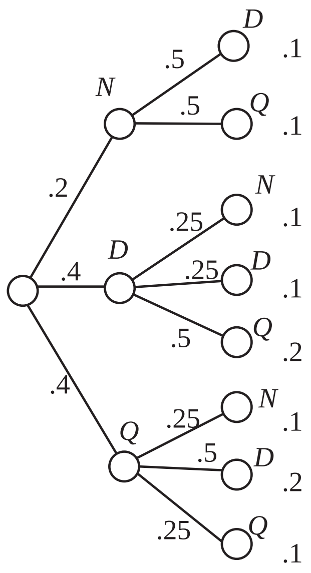
Exercise 5.3-7 Suppose we draw a card from a standard deck of 52 cards, discard it (i.e. we do not replace it), draw another card and continue for a total of ten draws. Is this an independent trials process?

In Exercise 5.3-6 we have an independent trials process, because the probability that we draw a given card at one stage does not depend on what cards we have drawn in earlier stages. However, in Exercise 5.3-7, we don't have an independent trials process. In the first draw, we have 52 cards to draw from, while in the second draw we have 51. In particular, we do not have the same cards to draw from on the second draw as the first, so the probabilities for each possible outcome on the second draw depend on whether that outcome was the result of the first draw.

Tree diagrams

When we have a sample space that consists of sequences of outcomes, it is often helpful to visualize the outcomes by a tree diagram. We will explain what we mean by giving a tree diagram of the following experiment. We have one nickel, two dimes, and two quarters in a cup. We draw a first and second coin. In Figure 5.3 you see our diagram for this process. Notice that in probability theory it is standard to have trees open to the right, rather than opening up or down.

Figure 5.5: A tree diagram illustrating a two-stage process.



Each level of the tree corresponds to one stage of the process of generating a sequence in our sample space. Each vertex is labeled by one of the possible outcomes at the stage it represents. Each edge is labeled with a conditional probability, the probability of getting the outcome at its right end given the sequence of outcomes that have occurred so far. Since no outcomes have occurred at stage 0, we label the edges from the root to the first stage vertices with the probabilities of the outcomes at the first stage. Each path from the root to the far right of the tree represents a possible sequence of outcomes of our process. We label each leaf node with the probability of the sequence that corresponds to the path from the root to that node. By the definition of conditional probabilities, the probability of a path is the product of the probabilities along its edges. We draw a probability tree for any (finite) sequence of successive trials in this way.

Sometimes a probability tree provides a very effective way of answering questions about a process. For example, what is the probability of having a nickel in our coin experiment? We see there are four paths containing an N, and the sum of their weights is .4, so the probability that one of our two coins is a nickel is .4.

Exercise 5.3-8 How can we recognize from a probability tree whether it is the probability tree of an independent trials process?

Exercise 5.3-9 In Exercise 5.3-4 we asked (among other things), if a student knows 80% of the material in a a course, what is the probability that she answers a question correctly on a 100 question True-False test (assuming that she guesses on any question she does not know the answer to)? (We assume that she knows what she knows, that is, if she thinks that she knows the answer, then she really does.) Show how we can use a probability tree to answer this question.

Exercise 5.3-10 A test for a disease that affects 0.1% of the population is 99% effective on people with the disease (that is, it says they have it with probability 0.99). The test gives a false reading (saying that a person who does not have the disease is affected with it) for 2% of the population without the disease. We can think of choosing someone and testing them for the disease as a two stage process. In stage 1, we either choose someone with the disease or we don't. In stage two, the test is either positive or it isn't. Give a probability tree for this process. What is the probability that someone selected at random and given a test for the disease will have a positive test? What is the probability that someone who has positive test results in fact has the disease?

A tree for an independent trials process has the property that at each level, for each node at that level, the (labeled) tree consisting of that node and all its children is identical to each labeled tree consisting of another node at that level and all its children. If we have such a tree, then it automatically satisfies the definition of an independent trials process.

In Exercise 5.3-9, if a student knows 80% of the material in a course, we expect that she has probability .8 of knowing the answer to any given question of a well-designed true-false test. We regard her work on a question as a two stage process; in stage 1 she determines whether she knows the answer, and in stage 2, she either answers correctly with probability 1, or she guesses, in which case she answers correctly with probability 1/2 and incorrectly with probability 1/2. Then as we see in Figure 5.3 there are two root-leaf paths corresponding to her getting a correct answer. One of these paths has probability .8 and the other has probability .1. Thus she actually has probability .9 of getting a right answer if she guesses at each question she does not know the answer to.

In Figure 5.3 we show the tree diagram for thinking of Exercise 5.3-10 as a two stage process. In the first stage, a person either has or doesn't have the disease. In the second stage we administer the test, and its result is either positive or not. We use D to stand for having the disease and ND to stand for not having the disease. We use "pos" to stand for a positive test and "neg" to stand for a negative test, and assume a test is either positive or negative. The question asks us for the conditional probability that someone has the disease, given that they test positive. This is

$$P(D|\text{pos}) = \frac{P(D \cap \text{pos})}{P(\text{pos})}.$$

Figure 5.6: The probability of getting a right answer is .9.

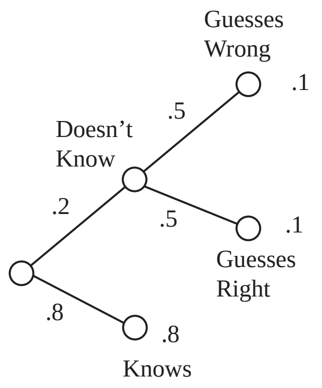
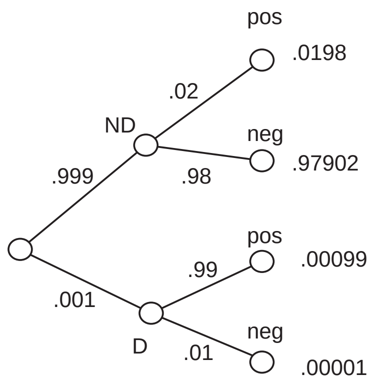


Figure 5.7: A tree diagram illustrating Exercise 5.3-10.



From the tree, we read that $P(D \cap \text{pos}) = .00099$ because this event consists of just one root-leaf paths. The event "pos" consists of two root-leaf paths whose probabilities total $.0198 + .00099 = .02097$. Thus $P(D|\text{pos}) = P(D \cap \text{pos})/P(\text{pos}) = .00099/.02097 = .0472$. Thus, given a disease this rare and a test with this error rate, a positive result only gives you roughly a 5% chance of having the disease! Here is another instance where a probability analysis shows something we might not have expected initially. This explains why doctors often don't want to administer a test to someone unless that person is already showing some symptoms of the disease being tested for.

We can also do Exercise 5.3-10 purely algebraically. We are given that

$$P(\text{disease}) = .001, \tag{5.18}$$
$$P(\text{positive test result}|\text{disease}) = .99, \tag{5.19}$$
$$P(\text{positive test result}|\text{no disease}) = .02. \tag{5.20}$$

We wish to compute

$$P(\text{disease}|\text{positive test result}).$$

We use Equation 5.15 to write that

$$P(\text{disease}|\text{positive test result}) = \frac{P(\text{disease} \cap \text{positive test result})}{P(\text{positive test result})}. \qquad (5.21)$$

How do we compute the numerator? Using the fact that $P(\text{disease} \cap \text{positive test result}) = P(\text{positive test result} \cap \text{disease})$ and Equation 5.15 again, we can write

$$P(\text{positive test result}|\text{disease}) = \frac{P(\text{positive test result} \cap \text{disease})}{P(\text{disease})}.$$

Plugging Equations 5.19 and 5.18 into this equation, we get

$$.99 = \frac{P(\text{positive test result} \cap \text{disease})}{.001}$$

or $P(\text{positive test result} \cap \text{disease}) = (.001)(.99) = .00099$.

To compute the denominator of Equation 5.21, we observe that since each person either has the disease or doesn't, we can write

$$P(\text{positive test}) = P(\text{positive test} \cap \text{disease}) + P(\text{positive test} \cap \text{no disease}). \qquad (5.22)$$

We have already computed $P(\text{positive test result} \cap \text{disease})$, and we can compute the probability $P(\text{positive test result} \cap \text{no disease})$ in a similar manner. Writing

$$P(\text{positive test result}|\text{no disease}) = \frac{P(\text{positive test result} \cap \text{no disease})}{P(\text{no disease})},$$

observing that $P(\text{no disease}) = 1 - P(\text{disease})$ and plugging in the values from Equations 5.18 and 5.20, we get that $P(\text{positive test result} \cap \text{no disease}) = (.02)(1 - .001) = .01998$ We now have the two components of the right hand side of Equation 5.22 and thus $P(\text{positive test result}) = .00099 + .01998 = .02097$. Finally, we have all the pieces in Equation 5.21, and conclude that

$$P(\text{disease}|\text{positive test result}) = \frac{P(\text{disease} \cap \text{positive test result})}{P(\text{positive test result})} = \frac{.00099}{.02097} = .0472.$$

Clearly, using the tree diagram mirrors these computations, but it both simplifies the thought process and reduces the amount we have to write.

Important Concepts, Formulas, and Theorems

1. *Conditional Probability.* We define the *conditional probability* of E given F, denoted by $P(E|F)$ and read as "the probability of E given F" by
$$P(E|F) = \frac{P(E \cap F)}{P(F)}.$$
2. *Independent.* We say E is *independent* of F if $P(E|F) = P(E)$.
3. *Product Principle for Independent Probabilities.* The *product principle for independent probabilities* (Theorem 5.4) gives another test for independence. Suppose E and F are events in a sample space. Then E is independent of F if and only if $P(E \cap F) = P(E)P(F)$.

4. *Symmetry of Independence.* The event E is independent of the event F if and only if F is independent of E.

5. *Independent Trials Process.* A process that occurs in stages is called an *independent trials process* if for each sequence $a_1, a_2, \ldots, a_n$ with $a_i \in S_i$,

$$P(x_i = a_i | x_1 = a_1, \ldots, x_{i-1} = a_{i-1}) = P(x_i = a_i).$$

6. *Probabilities of Outcomes in Independent Trials.* In an independent trials process the probability of a sequence $a_1, a_2, \ldots, a_n$ of outcomes is $P(\{a_1\})P(\{a_2\}) \cdots P(\{a_n\})$.

7. *Coin Flipping.* Repeatedly flipping a coin is an independent trials process.

8. *Hashing.* Hashing a list of n keys into k slots is an independent trials process with n stages.

9. *Probability Tree.* In a probability tree for a multistage process, each level of the tree corresponds to one stage of the process. Each vertex is labeled by one of the possible outcomes at the stage it represents. Each edge is labeled with a conditional probability, the probability of getting the outcome at its right end given the sequence of outcomes that have occurred so far. Each path from the root to a leaf represents a sequence of outcomes and is labelled with the product of the probabilities along that path. This is the probability of that sequence of outcomes.

Problems

1. In three flips of a coin, what is the probability that two flips in a row are heads, given that there is an even number of heads?

2. In three flips of a coin, is the event that two flips in a row are heads independent of the event that there is an even number of heads?

3. In three flips of a coin, is the event that we have at most one tail independent of the event that not all flips are identical?

4. What is the sample space that we use for rolling two dice, a first one and then a second one? Using this sample space, explain why it is that if we roll two dice, the event "i dots are on top of the first die" and the event "j dots are on top of the second die" are independent.

5. If we flip a coin twice, is the event of having an odd number of heads independent of the event that the first flip comes up heads? Is it independent of the event that the second flip comes up heads? Would you say that the three events are mutually independent? (This hasn't been defined, so the question is one of opinion. However you should back up your opinion with a reason that makes sense!)

6. Assume that on a true-false test, students will answer correctly any question on a subject they know. Assume students guess at answers they do not know. For students who know 60% of the material in a course, what is the probability that they will answer a question correctly? What is the probability that they will know the answer to a question they answer correctly?

7. A nickel, two dimes, and two quarters are in a cup. We draw three coins, one at a time, without replacement. Draw the probability tree which represents the process. Use the tree to determine the probability of getting a nickel on the last draw. Use the tree to determine the probability that the first coin is a quarter, given that the last coin is a quarter.

8. Write down a formula for the probability that a bridge hand (which is 13 cards, chosen from an ordinary deck) has four aces, given that it has one ace. Write down a formula for the probability that a bridge hand has four aces, given that it has the ace of spades. Which of these probabilities is larger?

9. A nickel, two dimes, and three quarters are in a cup. We draw three coins, one at a time without replacement. What is the probability that the first coin is a nickel? What is the probability that the second coin is a nickel? What is the probability that the third coin is a nickel?

10. If a student knows 75% of the material in a course, and a 100 question multiple choice test with five choices per question covers the material in a balanced way, what is the student's probability of getting a right answer to a given question, given that the student guesses at the answer to each question whose answer he or she does not know?

11. Suppose E and F are events with $E \cap F = \emptyset$. Describe when E and F are independent and explain why.

12. What is the probability that in a family consisting of a mother, father and two children of different ages, that the family has two girls, given that one of the children is a girl? What is the probability that the children are both boys, given that the older child is a boy?

5.4 Random Variables

What are Random Variables?

A **random variable** for an experiment with a sample space S is a function that assigns a number to each element of S. Typically instead of using f to stand for such a function we use X (at first, a random variable was conceived of as a variable related to an experiment, explaining the use of X, but it is very helpful in understanding the mathematics to realize it actually is a function on the sample space).

For example, if we consider the process of flipping a coin n times, we have the set of all sequences of n Hs and Ts as our sample space. The "number of heads" random variable takes a sequence and tells us how many heads are in that sequence. Somebody might say "Let X be the number of heads in 5 flips of a coin." In that case $X(HTHHT) = 3$ while $X(THTHT) = 2$. It may be rather jarring to see X used to stand for a function, but it is the notation most people use.

For a sequence of hashes of n keys into a table with k locations, we might have a random variable X_i which is the number of keys that are hashed to location i of the table, or a random variable X that counts the number of collisions (hashes to a location that already has at least one key). For an n question test on which each answer is either right or wrong (a short answer, True-False or multiple choice test for example) we could have a random variable that gives the number of right answers in a particular sequence of answers to the test. For a meal at a restaurant we might have a random variable that gives the price of any particular sequence of choices of menu items.

Exercise 5.4-1 Give several random variables that might be of interest to a doctor whose sample space is her patients.

Exercise 5.4-2 If you flip a coin six times, how many heads do you expect?

A doctor might be interested in patients' ages, weights, temperatures, blood pressures, cholesterol levels, etc.

For Exercise 5.4-2, in six flips of a coin, it is natural to expect three heads. We might argue that if we average the number of heads over all possible outcomes, the average should be half the number of flips. Since the probability of any given sequence equals that of any other, it is reasonable to say that this average is what we expect. Thus we would say we expect the number of heads to be half the number of flips. We will explore this more formally later.

Binomial Probabilities

When we study an independent trials process with two outcomes at each stage, it is traditional to refer to those outcomes as successes and failures. When we are flipping a coin, we are often interested in the number of heads. When we are analyzing student performance on a test, we are interested in the number of correct answers. When we are analyzing the outcomes in drug trials, we are interested in the number of trials where the drug was successful in treating the disease. This suggests a natural random variable associated with an independent trials process with two outcomes at each stage, namely the number of successes in n trials. We will analyze in general the probability of exactly k successes in n independent trials with probability p of success (and thus probability $1-p$ of failure) on each trial. It is standard to call such an independent trials process a *Bernoulli trials process.*

Exercise 5.4-3 Suppose we have 5 Bernoulli trials with probability p of success on each trial. What is the probability of success on the first three trials and failure on the last two? Failure on the first two trials and success on the last three? Success on trials 1, 3, and 5, and failure on the other two? Success on any particular three trials, and failure on the other two?

Since the probability of a sequence of outcomes is the product of the probabilities of the individual outcomes, the probability of any sequence of 3 successes and 2 failures is $p^3(1-p)^2$. More generally, in n Bernoulli trials, the probability of a given sequence of k successes and $n-k$ failures is $p^k(1-p)^{n-k}$. However this is not the probability of having k successes, because many different sequences could have k successes.

How many sequences of n successes and failures have exactly k successes? The number of ways to choose the k places out of n where the successes occur is $\binom{n}{k}$, so the number of sequences with k successes is $\binom{n}{k}$. This paragraph and the last together give us Theorem 5.7.

Theorem 5.7 *The probability of having exactly k successes in a sequence of n independent trials with two outcomes and probability p of success on each trial is*

$$P(\textit{exactly } k \textit{ successes}) = \binom{n}{k} p^k(1-p)^{n-k}$$

.

Proof: The proof follows from the two paragraphs preceding the theorem.■

Because of the connection between these probabilities and the binomial coefficients, the probabilities of Theorem 5.7 are called **binomial probabilities**, or the **binomial probability distribution**.

Exercise 5.4-4 A student takes a ten question objective test. Suppose that a student who knows 80% of the course material has probability .8 of success an any question, independently of how the student did on any other problem. What is the probability that this student earns a grade of 80 or better?

Exercise 5.4-5 Recall the primality testing algorithm from Section 2.4. Here we said that we could, by choosing a random number less than or equal to n, perform a test on n that, if n was not prime, would certify this fact with probability 1/2. Suppose we perform 20 of these tests. It is reasonable to assume that each of these tests is independent of the rest of them. What is the probability that a non-prime number is certified to be non-prime?

Since a grade of 80 or better on a ten question test corresponds to 8, 9, or 10 successes in ten trials, in Exercise 5.4-4 we have

$$P(80 \text{ or better}) = \binom{10}{8}(.8)^8(.2)^2 + \binom{10}{9}(.8)^9(.2)^1 + (.8)^{10}.$$

Some work with a calculator gives us that this sum is approximately .678.

In Exercise 5.4-5, we will first compute the probability that a non-prime number is not certified to be non-prime. If we think of success as when the number is certified non-prime and failure when it isn't, then we see that the only way to fail to certify a number is to have 20 failures. Using our formula we see that the probability that a non-prime number is not certified non-prime is just $\binom{20}{20}(.5)^{20} = 1/1048576$. Thus the chance of this happening is less than one in a million, and the chance of certifying the non-prime as non-prime is 1 minus this. Therefore the probability that a non-prime number will be certified non-prime is 1048575/1048576, which is more than .999999, so a non-prime number is almost sure to be certified non-prime.

A Taste of Generating Functions We note a nice connection between the probability of having exactly k successes and the binomial theorem. Consider, as an example, the polynomial $(H+T)^3$. Using the binomial theorem, we get that this is

$$(H+T)^3 = \binom{3}{0}H^3 + \binom{3}{1}H^2T + \binom{3}{2}HT^2 + \binom{3}{3}T^3.$$

We can interpret this as telling us that if we flip a coin three times, with outcomes heads or tails each time, then there are $\binom{3}{0} = 1$ way of getting 3 heads, $\binom{3}{2} = 3$ ways of getting two heads and one tail, $\binom{3}{1} = 3$ ways of getting one head and two tails and $\binom{3}{3} = 1$ way of getting 3 tails.

Similarly, if we replace H and T by px and $(1-p)y$ we would get the following:

$$(px+(1-p)y)^3 = \binom{3}{0}p^3x^3 + \binom{3}{1}p^2(1-p)x^2y + \binom{3}{2}p(1-p)^2xy^2 + \binom{3}{3}(1-p)^3y^3.$$

Generalizing this to n repeated trials where in each trial the probability of success is p, we see that by taking $(px+(1-p)y)^n$ we get

$$(px+(1-p)y)^n = \sum_{k=0}^{k} \binom{n}{k} p^k(1-p)^{n-k}x^ky^{n-k}.$$

Taking the coefficient of x^ky^{n-k} from this sum, we get exactly the result of Theorem 5.7. This connection is a simple case of a very powerful tool known as *generating functions*. We say that the polynomial $(px+(1-p)y)^n$ *generates* the binomial probabilities. In fact, we don't even need the y, because

$$(px+1-p)^n = \sum_{i=0}^{n} \binom{n}{i} p^i(1-p)^{n-i}x^i.$$

In general, the *generating function* for the sequence $a_0, a_1, a_2, \ldots, a_n$ is $\sum_{i=1}^{n} a_ix^i$, and the *generating function* for an infinite sequence $a_0, a_1, a_2, \ldots, a_n, \ldots$ is the infinite series $\sum_{i=1}^{\infty} a_ix^i$.

Expected Value

In Exercise 5.4-4 and Exercise 5.4-2 we asked about the value you would expect a random variable(in these cases, a test score and the number of heads in six flips of a coin) to have. We haven't yet defined what we mean by the value we expect, and yet it seems to make sense in the places we asked about it. If we say we expect 1 head if we flip a coin twice, we can explain our reasoning by taking an average. There are four outcomes, one with no heads, two with one head, and one with two heads, giving us an average of

$$\frac{0+1+1+2}{4} = 1.$$

Notice that using averages compels us to have some expected values that are impossible to achieve. For example in three flips of a coin the eight possibilities for the number of heads are 0, 1, 1, 1, 2, 2, 2, 3, giving us for our average

$$\frac{0+1+1+1+2+2+2+3}{8} = 1.5.$$

Exercise 5.4-6 An interpretation in games and gambling makes it clear that it makes sense to expect a random variable to have a value that is not one of the possible outcomes. Suppose that I proposed the following game. You pay me some money, and then you flip three coins. I will pay you one dollar for every head that comes up. Would you play this game if you had to pay me \$2.00? How about if you had to pay me \$1? How much do you think it should cost, in order for this game to be fair?

Since you expect to get 1.5 heads, you expect to make \$1.50. Therefore, it is reasonable to play this game as long as the cost is at most \$1.50.

Certainly averaging our variable over all elements of our sample space by adding up one result for each element of the sample space as we have done above is impractical even when we are talking about something as simple as ten flips of a coin. However we can ask how many times each possible number of heads arises, and then multiply the number of heads by the number of times it arises to get an average number of heads of

$$\frac{0\binom{10}{0}+1\binom{10}{1}+2\binom{10}{2}+\cdots+9\binom{10}{9}+10\binom{10}{10}}{1024} = \frac{\sum_{i=0}^{10} i\binom{10}{i}}{1024}. \tag{5.23}$$

Thus we wonder whether we have seen a formula for $\sum_{i=0}^{n} i\binom{n}{i}$. Perhaps we have, but in any case the binomial theorem and a bit of calculus or a proof by induction show that

$$\sum_{i=0}^{n} i\binom{n}{i} = 2^{n-1}n,$$

giving us $512 \cdot 10/1024 = 5$ for the fraction in Equation 5.23. If you are asking "Does it have to be that hard?" then good for you. Once we know a bit about the theory of expected values of random variables, computations like this will be replaced by far simpler ones.

Besides the nasty computations that a simple question lead us to, the average value of a random variable on a sample space need not have anything to do with the result we expect. For instance if we replace heads and tails with right and wrong, we get the sample space of possible results that a student will get when taking a ten question test with probability .9 of getting the right answer on any one question. Thus if we compute the average number of right answers in all the possible patterns of test results we get an average of 5 right answers. This is not the number of right answers we expect because averaging has nothing to do with the underlying process that gave us our probability! If we analyze the ten coin flips a bit more carefully, we can resolve this disconnection. We can rewrite Equation 5.23 as

$$0\frac{\binom{10}{0}}{1024}+1\frac{\binom{10}{1}}{1024}+2\frac{\binom{10}{2}}{1024}+\cdots+9\frac{\binom{10}{9}}{1024}+10\frac{\binom{10}{10}}{1024}=\sum_{i=0}^{10} i\frac{\binom{10}{i}}{1024}. \tag{5.24}$$

In Equation 5.24 we see we can compute the average number of heads by multiplying each value of our "number of heads" random variable by the probability that we have that value for our random variable, and then adding the results. This gives us a "weighted average" of the values of our random variable, each value weighted by its probability. Because the idea of weighting a random variable by its probability comes up so much in Probability Theory, there is a special notation that has developed to use this weight in equations. We use $P(X = x_i)$ to stand for the probability that the random variable X equals the value x_i. We call the function that assigns $P(x_i)$ to the event $P(X = x_i)$ the *distribution function* of the random variable X. Thus, for example, the binomial probability distribution is the distribution function for the "number of successes" random variable in Bernoulli trials.

We define the **expected value** or **expectation** of a random variable X whose values are the set $\{x_1, x_2, \ldots x_k\}$ to be

$$E(X) = \sum_{i=1}^{k} x_i P(X = x_i).$$

Then for someone taking a ten-question test with probability .9 of getting the correct answer on each question, the expected number of right answers is

$$\sum_{i=0}^{10} i\binom{10}{i}(.9)^i(.1)^{10-i}.$$

In the end of section exercises we will show a technique (that could be considered an application of generating functions) that allows us to compute this sum directly by using the binomial theorem and calculus. We now proceed to develop a less direct but easier way to compute this and many other expected values.

Exercise 5.4-7 Show that if a random variable X is defined on a sample space S (you may assume X has values x_1, x_2, $\ldots x_k$ as above) then the expected value of X is given by

$$E(X) = \sum_{s:s\in S} X(s)P(s).$$

(In words, we take each member of the sample space, compute its probability, multiply the probability by the value of the random variable and add the results.)

In Exercise 5.4-7 we asked for a proof of a fundamental lemma

Lemma 5.8 *If a random variable X is defined on a (finite) sample space S, then its expected value is given by*

$$E(X) = \sum_{s:s\in S} X(s)P(s).$$

Proof: Assume that the values of the random variable are x_1, x_2, $\ldots x_k$. Let F_i stand for the event that the value of X is x_i, so that $P(F_i) = P(X = x_i)$. Then, in the sum on the right-hand side of the equation in the statement of the lemma, we can take the items in the sample space, group them together into the events F_i and and rework the sum into the definition of expectation, as follows:

$$\begin{aligned}
\sum_{s:s\in S} X(s)P(s) &= \sum_{i=1}^{k} \sum_{s:s\in F_i} X(s)P(s) \\
&= \sum_{i=1}^{k} \sum_{s:s\in F_i} x_i P(s) \\
&= \sum_{i=1}^{k} x_i \sum_{s:s\in F_i} P(s) \\
&= \sum_{i=1}^{k} x_i P(F_i) \\
&= \sum_{i=1}^{k} x_i P(X = x_i) = E(X).
\end{aligned}$$

■

The proof of the lemma need not be so formal and symbolic as what we wrote; in English, it simply says that when we compute the sum in the Lemma, we can group together all elements of the sample space that have X-value x_i and add up their probabilities; this gives us $x_iP(x_i)$, which leads us to the definition of the expected value of X.

Expected Values of Sums and Numerical Multiples

Another important point about expected value follows naturally from what we think about when we use the word "expect" in English. If a paper grader expects to earn ten dollars grading papers today and expects to earn twenty dollars grading papers tomorrow, then she expects to earn thirty dollars grading papers in these two days. We could use X_1 to stand for the amount of money she makes grading papers today and X_2 to stand for the amount of money she makes grading papers tomorrow, so we are saying

$$E(X_1 + X_2) = E(X_1) + E(X_2).$$

This formula holds for any sum of a pair of random variables, and more generally for any sum of random variables on the same sample space.

Theorem 5.9 *Suppose X and Y are random variables on the (finite) sample space S. Then*

$$E(X + Y) = E(X) + E(Y).$$

Proof: From Lemma 5.8 we may write

$$E(X + Y) = \sum_{s:s\in S} (X(s) + Y(s))P(s) = \sum_{s:s\in S} X(s)P(s) + \sum_{s:s\in S} Y(s)P(s) = E(X) + E(Y).$$

■

If we double the credit we give for each question on a test, we would expect students' scores to double. Thus our next theorem should be no surprise. In it we use the notation cX for the random variable we get from X by multiplying all its values by the number c.

Theorem 5.10 *Suppose X is a random variable on a sample space S. Then for any number c, $E(cX) = cE(X)$.*

Proof: Left as a problem.■

Theorems 5.9 and 5.10 are very useful in proving facts about random variables. Taken together, they are typically called *linearity of expectation.* (The idea that the expectation of a sum is the same as the sum of expectations is called the *additivity of expectation.*) The idea of linearity will often allow us to work with expectations much more easily than if we had to work with the underlying probabilities.

For example, on one flip of a coin, our expected number of heads is .5. Suppose we flip a coin n times and let X_i be the number of heads we see on flip i, so that X_i is either 0 or 1. (For example in five flips of a coin, $X_2(HTHHT) = 0$ while $X_3(HTHHT) = 1$.) Then X, the total number of heads in n flips is given by

$$X = X_1 + X_2 + \cdots X_n, \tag{5.25}$$

the sum of the number of heads on the first flip, the number on the second, and so on through the number of heads on the last flip. But the expected value of each X_i is .5. We can take the expectation of both sides of Equation 5.25 and apply Lemma 5.9 repeatedly (or use induction) to get that

$$\begin{aligned} E(X) &= E(X_1 + X_2 + \cdots + X_n) \\ &= E(X_1) + E(X_2) + \cdots + E(X_n) \\ &= .5 + .5 + \cdots + .5 \\ &= .5n \end{aligned}$$

Thus in n flips of a coin, the expected number of heads is $.5n$. Compare the ease of this method with the effort needed earlier to deal with the expected number of heads in ten flips! Dealing with probability .9 or, in general with probability p poses no problem.

Exercise 5.4-8 Use the additivity of expectation to determine the expected number of correct answers a student will get on an n question "fill in the blanks" test if he or she knows 90% of the material in the course and the questions on the test are an accurate and uniform sampling of the material in the course.

In Exercise 5.4-8, since the questions sample the material in the course accurately, the most natural probability for us to assign to the event that the student gets a correct answer on a given question is .9. We can let X_i be the number of correct answers on question i (that is, either 1 or 0 depending on whether or not the student gets the correct answer). Then the expected number of right answers is the expected value of the sum of the variables X_i. From Theorem 5.9 see that in n trials with probability .9 of success, we expect to have $.9n$ successes. This gives that the expected number of right answers on a ten question test with probability .9 of getting each question right is 9, as we expected. This is a special case of our next theorem, which is proved by the same kind of computation.

Theorem 5.11 *In a Bernoulli trials process, in which each experiment has two outcomes and probability p of success, the expected number of successes is np.*

Proof: Let X_i be the number of successes in the ith of n independent trials. The expected number of successes on the ith trial (i.e. the expected value of X_i) is, by definition,

$$p \cdot 1 + (1-p) \cdot 0 = p.$$

The number of successes X in all n trials is the sum of the random variables X_i. Then by Theorem 5.9 the expected number of successes in n independent trials is the sum of the expected values of the n random variables X_i and this sum is np.■

The Number of Trials until the First Success

Exercise 5.4-9 How many times do you expect to have to flip a coin until you first see a head? Why? How many times to you expect to have to roll two dice until you see a sum of seven? Why?

Our intuition suggests that we should have to flip a coin twice to see a head. However we could conceivably flip a coin forever without seeing a head, so should we really expect to see a head in two flips? The probability of getting a seven on two dice is 1/6. Does that mean we should expect to have to roll the dice six times before we see a seven?

In order to analyze this kind of question we have to realize that we are stepping out of the realm of independent trials processes on finite sample spaces. We will consider the process of repeating independent trials with probability p of success until we have a success and then stopping. Now the possible outcomes of our multistage process are the infinite set

$$\{S, FS, FFS, \ldots, F^iS, \ldots\},$$

in which we have used the notation F^iS to stand for the sequence of i failures followed by a success. Since we have an infinite sequence of outcomes, it makes sense to think about whether we can assign an infinite sequence of probability weights to its members so that the resulting sequence of probabilities adds to one. If so, then all our definitions make sense, and in fact the proofs of all our theorems remain valid.[5] There is only one way to assign weights that is consistent with our knowledge of (finite) independent trials processes, namely

$$P(S) = p, \quad P(FS) = (1-p)p, \quad \ldots, \quad P(F^iS) = (1-p)^ip, \quad \ldots.$$

Thus we have to hope these weights add to one; in fact their sum is

$$\sum_{i=0}^{\infty}(1-p)^ip = p\sum_{i=0}^{\infty}(1-p)^i = p\frac{1}{1-(1-p)} = \frac{p}{p} = 1.$$

[5]for those who are familiar with the concept of convergence for infinite sums (i.e. infinite series), it is worth noting that it is the fact that probability weights cannot be negative and must add to one that makes all the sums we need to deal with for all the theorems we have proved so far converge. That doesn't mean all sums we might want to deal with will converge; some random variables defined on the sample space we have described will have infinite expected value. However those we need to deal with for the expected number of trials until success do converge.

Therefore we have a legitimate assignment of probabilities and the set of sequences

$$\{F, FS, FFS, FFFS, \ldots, F^iS, \ldots\}$$

is a sample space with these probability weights. This probability distribution, $P(F^iS) = (1-p)^ip$, is called a *geometric* distribution because of the geometric series we used in proving the probabilities sum to 1.

Theorem 5.12 *Suppose we have a sequence of trials in which each trial has two outcomes, success and failure, and where at each step the probability of success is p. Then the expected number of trials until the first success is $1/p$.*

Proof:

We consider the random variable X which is i if the first success is on trial i. (In other words, $X(F^{i-1}S)$ is i.) The probability that the first success is on trial i is $(1-p)^{i-1}p$, since in order for this to happen there must be $i-1$ failures followed by 1 success. The expected number of trials is the expected value of X, which is, by the definition of expected value and the previous two sentences,

$$\begin{aligned}
E[\text{number of trials}] &= \sum_{i=0}^{\infty} p(1-p)^{i-1}i \\
&= p\sum_{i=0}^{\infty}(1-p)^{i-1}i \\
&= \frac{p}{1-p}\sum_{i=0}^{\infty}(1-p)^{i}i \\
&= \frac{p}{1-p}\frac{1-p}{p^2} \\
&= \frac{1}{p}
\end{aligned}$$

To go from the third to the fourth line we used the fact that

$$\sum_{j=0}^{\infty} jx^j = \frac{x}{(1-x)^2}, \tag{5.26}$$

true for x with absolute value less than one. We proved a finite version of this equation as Theorem 4.6; the infinite version is even easier to prove. ■

Applying this theorem, we see that the expected number of times you need to flip a coin until you get heads is 2, and the expected number of times you need to roll two dice until you get a seven is 6.

Important Concepts, Formulas, and Theorems

1. *Random Variable.* A *random variable* for an experiment with a sample space S is a function that assigns a number to each element of S.

2. Bernoulli Trials Process. An independent trials process with two outcomes, success and failure, at each stage and probability p of success and $1-p$ of failure at each stage is called a *Bernoulli trials process.*

3. Probability of a Sequence of Bernoulli Trials. In n Bernoulli trials with probability p of success, the probability of a given sequence of k successes and $n-k$ failures is $p^k(1-p)^{n-k}$.

4. *The Probability of k Successes in n Bernoulli Trials* The probability of having exactly k successes in a sequence of n independent trials with two outcomes and probability p of success on each trial is
$$P(\text{exactly } k \text{ successes}) = \binom{n}{k} p^k(1-p)^{n-k}$$
.

5. Binomial Probability Distribution. The probabilities of of k successes in n Bernoulli trials, $\binom{n}{k}p^k(1-p)^{n-k}$, are called *binomial probabilities*, or the *binomial probability distribution.*

6. *Generating Function.* The *generating function* for the sequence $a_0, a_1, a_2, \dots, a_n$ is $\sum_{i=1}^{n} a_i x^i$, and the *generating function* for an infinite sequence $a_0, a_1, a_2, \dots, a_n, \dots$ is the infinite series $\sum_{i=1}^{\infty} a_i x^i$. The polynomial $(px+1-p)^n$ is the generating function for the binomial probabilities for n Bernoulli trials with probability p of success.

7. *Distribution Function.* We call the function that assigns $P(x_i)$ to the event $P(X = x_i)$ the *distribution function* of the random variable X.

8. *Expected Value.* We define the *expected value* or *expectation* of a random variable X whose values are the set $\{x_1, x_2, \dots x_k\}$ to be
$$E(X) = \sum_{i=1}^{k} x_i P(X = x_i).$$

9. *Another Formula for Expected Values.* If a random variable X is defined on a (finite) sample space S, then its expected value is given by
$$E(X) = \sum_{s:s\in S} X(s)P(s).$$

10. *Expected Value of a Sum.* Suppose X and Y are random variables on the (finite) sample space S. Then
$$E(X+Y) = E(X) + E(Y).$$
This is called the *additivity of expectation.*

11. *Expected Value of a Numerical Multiple.* Suppose X is a random variable on a sample space S. Then for any number c, $E(cX) = cE(X)$. This result and the additivity of expectation together are called the *linearity of expectation.*

12. *Expected Number of Successes in Bernoulli Trials.* In a Bernoulli trials process, in which each experiment has two outcomes and probability p of success, the expected number of successes is np.

13. *Expected Number of Trials Until Success.* Suppose we have a sequence of trials in which each trial has two outcomes, success and failure, and where at each step the probability of success is p. Then the expected number of trials until the first success is $1/p$.

Problems

1. Give several random variables that might be of interest to someone rolling five dice (as one does, for example, in the game Yatzee).

2. Suppose I offer to play the following game with you if you will pay me some money. You roll a die, and I give you a dollar for each dot that is on top. What is the maximum amount of money a rational person might be willing to pay me in order to play this game?

3. How many sixes do we expect to see on top if we roll 24 dice?

4. What is the expected sum of the tops of n dice when we roll them?

5. In an independent trials process consisting of six trials with probability p of success, what is the probability that the first three trials are successes and the last three are failures? The probability that the last three trials are successes and the first three are failures? The probability that trials 1, 3, and 5 are successes and trials 2, 4, and 6 are failures? What is the probability of three successes and three failures?

6. What is the probability of exactly eight heads in ten flips of a coin? Of eight or more heads?

7. How many times do you expect to have to role a die until you see a six on the top face?

8. Assuming that the process of answering the questions on a five-question quiz is an independent trials process and that a student has a probability of .8 of answering any given question correctly, what is the probability of a sequence of four correct answers and one incorrect answer? What is the probability that a student answers exactly four questions correctly?

9. What is the expected value of the constant random variable X that has $X(s) = c$ for every member s of the sample space? We frequently just use c to stand for this random variable, and thus this question is asking for $E(c)$.

10. Someone is taking a true-false test and guessing when they don't know the answer. We are going to compute a score by subtracting a percentage of the number of incorrect answers from the number of correct answers. When we convert this "corrected score" to a percentage score we want its expected value to be the percentage of the material being tested that the test-taker knows. How can we do this?

11. Do Problem 10 of this section for the case that someone is taking a multiple choice test with five choices for each answer and guesses randomly when they don't know the answer.

12. Suppose we have ten independent trials with three outcomes called good, bad, and indifferent, with probabilities p, q, and r, respectively. What is the probability of three goods, two bads, and five indifferents? In n independent trials with three outcomes A, B, and C, with probabilities p, q, and r, what is the probability of i As, j Bs, and k Cs? (In this problem we assume $p + q + r = 1$ and $i + j + k = n$.)

13. In as many ways as you can, prove that
$$\sum_{i=0}^{n} i\binom{n}{i} = 2^{n-1}n.$$

14. Prove Theorem 5.10.

15. Two nickels, two dimes, and two quarters are in a cup. We draw three coins, one after the other, without replacement. What is the expected amount of money we draw on the first draw? On the second draw? What is the expected value of the total amount of money we draw? Does this expected value change if we draw the three coins all together?

16. In this exercise we will evaluate the sum
$$\sum_{i=0}^{10} i\binom{10}{i}(.9)^i(.1)^{10-i}$$
that arose in computing the expected number of right answers a person would have on a ten question test with probability .9 of answering each question correctly. First, use the binomial theorem and calculus to show that
$$10(.1+x)^9 = \sum_{i=0}^{10} i\binom{10}{i}(.1)^{10-i}x^{i-1}$$
Substituting in $x = .9$ gives us almost the sum we want on the right hand side of the equation, except that in every term of the sum the power on .9 is one too small. Use some simple algebra to fix this and then explain why the expected number of right answers is 9.

17. Give an example of two random variables X and Y such that $E(XY) \neq E(X)E(Y)$. Here XY is the random variable with $(XY)(s) = X(s)Y(s)$.

18. Prove that if X and Y are independent in the sense that the event that $X = x$ and the event that $Y = y$ are independent for each pair of values x of X and y of Y, then $E(XY) = E(X)E(Y)$. See Exercise 5-17 for a definition of XY.

19. Use calculus and the sum of a geometric series to show that
$$\sum_{j=0}^{\infty} jx^j = \frac{x}{(1-x)^2}$$
as in Equation 5.26.

20. Give an example of a random variable on the sample space $\{S, FS, FFS, \ldots, F^iS, \ldots\}$ with an infinite expected value.

5.5 Probability Calculations in Hashing

We can use our knowledge of probability and expected values to analyze a number of interesting aspects of hashing including:

1. expected number of items per location,
2. expected time for a search,
3. expected number of collisions,
4. expected number of empty locations,
5. expected time until all locations have at least one item,
6. expected maximum number of items per location.

Expected Number of Items per Location

Exercise 5.5-1 We are going to compute the expected number of items that hash to any particular location in a hash table. Our model of hashing n items into a table of size k allows us to think of the process as n independent trials, each with k possible outcomes (the k locations in the table). On each trial we hash another key into the table. If we hash n items into a table with k locations, what is the probability that any one item hashes into location 1? Let X_i be the random variable that counts the number of items that hash to location 1 in trial i (so that X_i is either 0 or 1). What is the expected value of X_i? Let X be the random variable $X_1 + X_2 + \cdots + X_n$. What is the expected value of X? What is the expected number of items that hash to location 1? Was the fact that we were talking about location 1 special in any way? That is, does the same expected value apply to every location?

Exercise 5.5-2 Again we are hashing n items into k locations. Our model of hashing is that of Exercise 5.5-1. What is the probability that a location is empty? What is the expected number of empty locations? Suppose we now hash n items into the same number n of locations. What limit does the expected fraction of empty places approach as n gets large?

In Exercise 5.5-1, the probability that any one item hashes into location 1 is $1/k$, because all k locations are equally likely. The expected value of X_i is then $1/k$. The expected value of X is then n/k, the sum of n terms each equal to $1/k$. Of course the same expected value applies to any location. Thus we have proved the following theorem.

Theorem 5.13 *In hashing n items into a hash table of size k, the expected number of items that hash to any one location is n/k.*

Expected Number of Empty Locations

In Exercise 5.5-2 the probability that position i will be empty after we hash 1 item into the table will be $1 - \frac{1}{k}$. (Why?) In fact, we can think of our process as an independent trials process with two outcomes: the key hashes to slot i or it doesn't. From this point of view, it is clear that the probability of nothing hashing to slot i in n trials is $(1 - \frac{1}{k})^n$. Now consider the original sample space again and let X_i be 1 if slot i is empty for a given sequence of hashes or 0 if it is not. Then the number of empty slots for a given sequence of hashes is $X_1 + X_2 + \cdots + X_k$ evaluated at that sequence. Therefore, the expected number of empty slots is, by Theorem 5.9, $k(1 - \frac{1}{k})^n$. Thus we have proved another nice theorem about hashing.

Theorem 5.14 *In hashing n items into a hash table with k locations, the expected number of empty locations is $k(1 - \frac{1}{k})^n$.*

Proof: Given above.■

If we have the same number of slots as places, the expected number of empty slots is $n(1-\frac{1}{n})^n$, so the expected fraction of empty slots is $(1 - \frac{1}{n})^n$. What does this fraction approach as n grows? You may recall that $\lim_{n\to\infty}(1 + \frac{1}{n})^n$ is e, the base for the natural logarithm. In the problems at the end of the section, we show you how to derive from this that $\lim_{n\to\infty}(1-\frac{1}{n})^n$ is e^{-1}. Thus for a reasonably large hash table, if we hash in as many items as we have slots, we expect a fraction $1/e$ of those slots to remain empty. In other words, we expect n/e empty slots. On the other hand, we expect $\frac{n}{n}$ items per location, which suggests that we should expect each slot to have an item and therefore expect to have no empty locations. Is something wrong? No, but we simply have to accept that our expectations about expectation just don't always hold true. What went wrong in that apparent contradiction is that our definition of expected value doesn't imply that if we have an expectation of one key per location then every location must have a key, but only that empty locations have to be balanced out by locations with more than one key. When we want to make a statement about expected values, we must use either our definitions or theorems to back it up. This is another example of why we have to back up intuition about probability with careful analysis.

Expected Number of Collisions

We say that we have a *collision* when we hash an item to a location that already contains an item. How can we compute the expected number of collisions? The number of collisions will be the number n of keys hashed minus the number of occupied locations because each occupied location will contain one key that will not have collided in the process of being hashed. Thus, by Theorems 5.9 and 5.10,

$$E(\text{collisions}) = n - E(\text{occupied locations}) = n - k + E(\text{empty locations}) \tag{5.27}$$

where the last equality follows because the expected number of occupied locations is k minus the expected number of unoccupied locations. This gives us yet another theorem.

Theorem 5.15 *In hashing n items into a hash table with k locations, the expected number of collisions is $n - k + k(1 - \frac{1}{k})^n$.*

Proof: We have already shown in Theorem 5.14 that the expected number of empty locations is $k(1-\frac{1}{k})^n$. Substituting this into Equation 5.27 gives our formula. ■

Exercise 5.5-3 In real applications, it is often the case that the hash table size is not fixed in advance, since you don't know, in advance, how many items you will insert. The most common heuristic for dealing with this is to start k, the hash table size, at some reasonably small value, and then when n, the number of items gets to be greater than $2k$, you double the hash table size. In this exercise, we propose a different idea. Suppose you waited until every single slot in the hash table had at least one item in it, and then you increased the table size. What is the expected number of items that will be in the table when you increase the size? In other words, how many items do you expect to insert into a hash table in order to ensure that every slot has at least one item? (Hint: Let X_i be the number of items added between the time that there are $i-1$ occupied slots for the first time and the first time that there are i occupied slots.)

For Exercise 5.5-3, the key is to let X_i be the number of items added between the time that there are $i-1$ full slots for the first time and i full slots for the first time. Let's think about this random variable. $E(X_1)=1$, since after one insertion there is one full slot. In fact X_1 itself is equal to 1.

To compute the expected value of X_2, we note that X_2 can take on any value greater than 1. In fact if we think about it, what we have here (until we actually hash an item to a new slot) is an independent trials process with two outcomes, with success meaning our item hashes to an unused slot. X_2 counts the number of trials until the first success. The probability of success is $(k-1)/k$. In asking for the expected value of X_2, we are asking for expected number of steps until the first success. Thus we can apply Lemma 5.12 to get that it is $k/(k-1)$.

Continuing, X_3 similarly counts the number of steps in an independent trials process (with two outcomes) that stops at the first success and has probability of success $(k-2)/k$. Thus the expected number of steps until the first success is $k/(k-2)$.

In general, we have that X_i counts the number of trials until success in an independent trials process with probability of success $(k-i+1)/k$ and thus the expected number of steps until the first success is $k/(k-i+1)$, which is the expected value of X_i.

The total time until all slots are full is just $X = X_1 + \cdots + X_k$. Taking expectations and using Lemma 5.12 we get

$$\begin{aligned}
E(X) &= \sum_{j=1}^{k} E(X_j) \\
&= \sum_{j=1}^{k} \frac{k}{k-j+1} \\
&= k\sum_{j=1}^{k} \frac{1}{k-j+1} \\
&= k\sum k-j+1 = 1^k \frac{1}{k-j+1} \\
&= k\sum_{i=1}^{k} \frac{1}{i},
\end{aligned}$$

where the last line follows just by switching the variable of the summation, that is, letting $k-j+1=i$ and summing over i.[6] Now the quantity $\sum_{i=1}^{k} \frac{1}{i}$ is known as a *harmonic number*, and is sometimes denoted by H_k. It is well known (and you can see why in the problems at the end of the section) that $\sum_{i=1}^{k} \frac{1}{i} = \Theta(\log k)$, and more precisely

$$\frac{1}{4} + \ln k \le H_k \le 1 + \ln k, \tag{5.28}$$

and in fact,

$$\frac{1}{2} + \ln k \le H_k \le 1 + \ln k, \tag{5.29}$$

when k is large enough. As n gets large, $H_n - \ln n$ approaches a limit called *Euler's constant*; Euler's constant is about .58. Equation 5.28 gives us that $E(X) = O(k \log k)$.

Theorem 5.16 *The expected number of items needed to fill all slots of a hash table of size k is between $k \ln k + \frac{1}{2}k$ and $k \ln k + k$.*

Proof: Given above.■.

So in order to fill every slot in a hash table of size k, we need to hash roughly $k \ln k$ items. This problem is sometimes called the *coupon collectors problem.*

Expected maximum number of elements in a slot of a hash table

In a hash table, the time to find an item is related to the number of items in the slot where you are looking. Thus an interesting quantity is the expected maximum length of the list of items in a slot in a hash table. This quantity is more complicated than many of the others we have been computing, and hence we will only try to upper bound it, rather than compute it exactly. In doing so, we will introduce a few upper bounds and techniques that appear frequently and are useful in many areas of mathematics and computer science. We will be able to prove that if we hash n items into a hash table of size n, the expected length of the longest list is $O(\log n / \log\log n)$. One can also prove, although we won't do it here, that with high probability, there will be some list with $\Omega(\log n / \log\log n)$ items in it, so our bound is, up to constant factors, the best possible.

Before we start, we give some useful upper bounds. The first allows us to bound terms that look like $(1+\frac{1}{x})^x$, for any positive x, by e.

Lemma 5.17 *For all $x > 0$, $(1+\frac{1}{x})^x \le e$.*

Proof: $\lim_{x\to\infty}(1+\frac{1}{x})^x = e$, and $1+(\frac{1}{x})^x$ has positive first derivative. ■

Second, we will use an approximation called Stirling's formula,

$$x! = \left(\frac{x}{e}\right)^x \sqrt{2\pi x}(1+\Theta(1/n)),$$

[6] note that $k-j+1$ runs from k to 1 as j runs from 1 to k, so we are describing exactly the same sum.

which tells us, roughly, that $(x/e)^x$ is a good approximation for $x!$. Moreover the constant in the $\Theta(1/n)$ term is extremely small, so for our purposes we will just say that

$$x! = \left(\frac{x}{e}\right)^x \sqrt{2\pi x}.$$

(We use this equality only in our proof of Lemma 5.18. You will see in that Lemma that we make the statement that $\sqrt{2\pi} > 1$. In fact, $\sqrt{2\pi} > 2$, and this is more than enough to make up for any lack of accuracy in our approximation.) Using Stirling's formula, we can get a bound on $\binom{n}{t}$,

Lemma 5.18 *For $n > t > 0$,*

$$\binom{n}{t} \le \frac{n^n}{t^t(n-t)^{n-t}}.$$

Proof:

$$\begin{aligned}\binom{n}{t} &= \frac{n!}{t!(n-t)!} & (5.30)\\ &= \frac{(n/e)^n\sqrt{2\pi n}}{(t/e)^t\sqrt{2\pi t}((n-t)/e)^{n-t}\sqrt{2\pi(n-t)}} & (5.31)\\ &= \frac{n^n\sqrt{n}}{t^t(n-t)^{n-t}\sqrt{2\pi}\sqrt{t(n-t)}} & (5.32)\end{aligned}$$

Now if $1 < t < n-1$, we have $t(n-t) \ge n$, so that $\sqrt{t(n-t)} \ge \sqrt{n}$. Further $\sqrt{2\pi} > 1$. We can use these to facts to upper bound the quantity marked 5.32 by

$$\frac{n^n}{t^t(n-t)^{n-t}}$$

When $t = 1$ or $t = n-1$, the inequality in the statement of the lemma is $n \le n^n/(n-1)^{n-1}$ which is true since $n - 1 < n$. ■

We are now ready to attack the problem at hand, the expected value of the maximum list size. Let's start with a related quantity that we already know how to compute. Let H_{it} be the event that t keys hash to slot i. $P(H_{it})$ is just the probability of t successes in an independent trials process with success probability $1/n$, so

$$P(H_{it}) = \binom{n}{t}\left(\frac{1}{n}\right)^t\left(1-\frac{1}{n}\right)^{n-t}. \qquad (5.33)$$

Now we relate this known quantity to the probability of the event M_t that the maximum list size is t.

Lemma 5.19 *Let M_t be the event that t is the maximum list size in hashing n items into a hash table of size n. Let H_{1t} be the event that t keys hash to position 1. Then*

$$P(M_t) \le nP(H_{1t})$$

Proof: We begin by letting M_{it} be the event that the maximum list size is t and this list appears in slot i. Observe that that since M_{it} is a subset of H_{it},

$$P(M_{it}) \leq P(H_{it}). \tag{5.34}$$

We know that, by definition,

$$M_t = M_{1t} \cup \cdots \cup M_{nt},$$

and so

$$P(M_t) = P(M_{1t} \cup \cdots \cup M_{nt}).$$

Therefore, since the sum of the probabilities of the individual events must be at least as large as the probability of the union,

$$P(M_t) \leq P(M_{1t}) + P(M_{2t}) + \cdots + P(M_{nt}). \tag{5.35}$$

(Recall that we introduced the Principle of Inclusion and Exclusion because the right hand side overestimated the probability of the union. Note that the inequality in Equation 5.35 holds for any union, not just this one: it is sometimes called *Boole's inequality*.)

In this case, for any i and j, $P(M_{it}) = P(M_{jt})$, since there is no reason for slot i to be more likely than slot j to be the maximum. We can therefore write that

$$P(M_t) = nP(M_{1t}) \leq nP(H_{1t}).$$

■

Now we can use Equation 5.33 for $P(H_{1t})$ and then apply Lemma 5.18 to get that

$$\begin{aligned} P(H_{1t}) &= \binom{n}{t}\left(\frac{1}{n}\right)^t\left(1-\frac{1}{n}\right)^{n-t} \\ &\leq \frac{n^n}{t^t(n-t)^{n-t}}\left(\frac{1}{n}\right)^t\left(1-\frac{1}{n}\right)^{n-t}. \end{aligned}$$

We continue, using algebra, the fact that $(1-\frac{1}{n})^{n-t} \leq 1$ and Lemma 5.17 to get

$$\begin{aligned} &\leq \frac{n^n}{t^t(n-t)^{n-t}n^t} \\ &= \frac{n^{n-t}}{t^t(n-t)^{n-t}} \\ &= \left(\frac{n}{n-t}\right)^{n-t}\frac{1}{t^t} \\ &= \left(1+\frac{t}{n-t}\right)^{n-t}\frac{1}{t^t} \\ &= \left(\left(1+\frac{t}{n-t}\right)^{\frac{n-t}{t}}\right)^t\frac{1}{t^t} \\ &\leq \frac{e^t}{t^t}. \end{aligned}$$

We have shown the following:

Lemma 5.20 *$P(M_t)$, the probability that the maximum list length is t, is at most ne^t/t^t.*

Proof: Our sequence of equations and inequalities above showed that $P(H_{1t}) \leq \frac{e^t}{t^t}$. Multiplying by n and applying Lemma 5.19 gives our result.■

Now that we have a bound on $P(M_t)$ we can compute a bound on the expected length of the longest list, namely

$$\sum_{t=0}^{n} P(M_t)t.$$

However, if we think carefully about the bound in Lemma 5.20, we see that we have a problem. For example when $t = 1$, the lemma tells us that $P(M_1) \leq ne$. This is vacuous, as we know that any probability is at most 1, We could make a stronger statement that $P(M_t) \leq \max\{ne^t/t^t, 1\}$, but even this wouldn't be sufficient, since it would tell us things like $P(M_1) + P(M_2) \leq 2$, which is also vacuous. All is not lost however. Our lemma causes this problem only when t is small. We will split the sum defining the expected value into two parts and bound the expectation for each part separately. The intuition is that when we restrict t to be small, then $\sum P(M_t)t$ is small because t is small (and over all t, $\sum P(M_t) \leq 1$). When t gets larger, Lemma 5.20 tells us that $P(M_t)$ is very small and so the sum doesn't get big in that case either. We will choose a way to split the sum so that this second part of the sum is bounded by a constant. In particular we split the sum up by

$$\sum_{t=0}^{n} P(M_t)t \leq \sum_{t=0}^{\lfloor 5\log n/\log\log n \rfloor} P(M_t)t + \sum_{t=\lceil 5\log n/\log\log n \rceil}^{n} P(M_t)t \tag{5.36}$$

For the sum over the smaller values of t, we just observe that in each term $t \leq 5\log n/\log\log n$ so that

$$\begin{aligned}
\sum_{t=0}^{5\log n/\log\log n} P(M_t)t &\leq \sum_{t=0}^{5\log n/\log\log n} P(M_t)5\log n/\log\log n & (5.37)\\
&= 5\log n/\log\log n \sum_{t=0}^{5\log n/\log\log n} P(M_t) & (5.38)\\
&\leq 5\log n/\log\log n & (5.39)
\end{aligned}$$

(Note that we are not using Lemma 5.20 here; only the fact that the probabilities of disjoint events cannot add to more than 1.) For the rightmost sum in Equation 5.36, we want to first compute an upper bound on $P(M_t)$ for $t = (5\log n/\log\log n)$. Using Lemma 5.20, and doing a bit of calculation we get that in this case $P(M_t) \leq 1/n^2$. Since the bound on $P(M_t)$ from Lemma 5.20 decreases as t grows, and $t \leq n$, we can bound the right sum by

$$\sum_{t=5\log n/\log\log n}^{n} P(M_t)t \leq \sum_{t=5\log n/\log\log n}^{n} \frac{1}{n^2}n \leq \sum_{t=5\log n/\log\log n}^{n} \frac{1}{n} \leq 1. \tag{5.40}$$

Combining Equations 5.39 and 5.40 with 5.36 we get the desired result.

Theorem 5.21 *If we hash n items into a hash table of size n, the expected maximum list length is $O(\log n/\log\log n)$.*

The choice to break the sum into two pieces here—and especially the breakpoint we chose—may have seemed like magic. What is so special about $\log n/\log\log n$? Consider the bound on $P(M_t)$. If you asked what is the value of t for which the bound equals a certain value, say $1/n^2$, you get the equation $ne^t/t^t = n^{-2}$. If we try to solve the equation $ne^t/t^t = n^{-2}$ for t, we quickly see that we get a form that we do not know how to solve. (Try typing this into Mathematica or Maple, to see that it can't solve this equation either.) The equation we need to solve is somewhat similar to the simpler equation $t^t = n$. While this equation does not have a closed form solution, one can show that the t that satisfies this equation is roughly $c\log n/\log\log n$, for some constant c. This is why some multiple of $\log n/\log\log n$ made sense to try as the the magic value. For values much less than $\log n/\log\log n$ the bound provided on $P(M_t)$ is fairly large. Once we get past $\log n/\log\log n$, however, the bound on $P(M_t)$ starts to get significantly smaller. The factor of 5 was chosen by experimentation to make the second sum come out to be less than 1. We could have chosen any number between 4 and 5 to get the same result; or we could have chosen 4 and the second sum would have grown no faster than the first.

Important Concepts, Formulas, and Theorems

1. *Expected Number of Keys per Slot in Hash Table.* In hashing n items into a hash table of size k, the expected number of items that hash to any one location is n/k.

2. Expected Number of Empty Slots in Hash Table. In hashing n items into a hash table with k locations, the expected number of empty locations is $k(1-\frac{1}{k})^n$.

3. Collision in Hashing. We say that we have a *collision* when we hash an item to a location that already contains an item.

4. *The Expected Number of Collisions in Hashing.* In hashing n items into a hash table with k locations, the expected number of collisions is $n-k+k(1-\frac{1}{k})^n$.

5. *Harmonic Number.* The quantity $\sum_{i=1}^{k}\frac{1}{i}$ is known as a *harmonic number*, and is sometimes denoted by H_k. It is a fact that that $\sum_{i=1}^{k}\frac{1}{i} = \Theta(\log k)$, and more precisely

$$\frac{1}{2} + \ln k \le H_k \le 1 + \ln k.$$

6. *Euler's Constant.* As n gets large, $H_n - \ln n$ approaches a limit called *Euler's constant*; Euler's constant is about .58.

7. *Expected Number of Hashes until all Slots of a Hash Table Are Occupied.* The expected number of items needed to fill all slots of a hash table of size k is between $k\ln k + \frac{1}{2}k$ and $k\ln k + k$.

8. *Expected Maximum Number of Keys per Slot.* If we hash n items into a hash table of size n, the expected maximum list length is $O(\log n/\log\log n)$.

Problems

1. A candy machine in a school has d different kinds of candy. Assume (for simplicity) that all these kinds of candy are equally popular and there is a large supply of each. Suppose that c children come to the machine and each purchases one package of candy. One of the kinds of candy is a Snackers bar. What is the probability that any given child purchases a Snackers bar? Let Y_i be the number of Snackers bars that child i purchases, so that Y_i is either 0 or 1. What is the expected value of Y_i? Let Y be the random variable $Y_1 + Y_2 + \cdots + Y_c$. What is the expected value of Y? What is the expected number of Snackers bars that is purchased? Does the same result apply to any of the varieties of candy?

2. Again as in the previous exercise, we have c children choosing from among ample supplies of d different kinds of candy, one package for each child, and all choices equally likely. What is the probability that a given variety of candy is chosen by no child? What is the expected number of kinds of candy chosen by no child? Suppose now that $c = d$. What happens to the expected number of kinds of candy chosen by no child?

3. How many children do we expect to have to observe buying candy until someone has bought a Snackers bar?

4. How many children to we expect to have to observe buying candy until each type of candy has been selected at least once?

5. If we have 20 kinds of candy, how many children have to buy candy in order for the probability to be at least one half that (at least) two children buy the same kind of candy?

6. What is the expected number of duplications among all the candy the children have selected?

7. Compute the values on the left-hand and right-hand side of the inequality in Lemma 5.18 for $n = 2$, $t = 0, 1, 2$ and for $n = 3$, $t = 0, 1, 2, 3$.

8. When we hash n items into k locations, what is the probability that all n items hash to different locations? What is the probability that the ith item is the first collision? What is the expected number of items we must hash until the first collision? Use a computer program or spreadsheet to compute the expected number of items hashed into a hash table until the first collision with $k = 20$ and with $k = 100$.

9. We have seen a number of occasions when our intuition about expected values or probability in general fails us. When we wrote down Equation 5.27 we said that the expected number of occupied locations is k minus the expected number of unoccupied locations. While this seems obvious, there is a short proof. Give the proof.

10. Write a computer program that prints out a table of values of the expected number of collisions with n keys hashed into a table with k locations for interesting values of n and k. Does this value vary much as n and k change?

11. Suppose you hash n items into a hash table of size k. It is natural to ask about the time it takes to find an item in the hash table. We can divide this into two cases, one when the item is not in the hash table (an unsuccessful search), and one when the item is in the hash table (a successful search). Consider first the unsuccessful search. Assume the keys hashing

to the same location are stored in a list with the most recent arrival at the beginning of the list. Use our expected list length to bound the expected time for an unsuccessful search. Next consider the successful search. Recall that when we insert items into a hash table, we typically insert them at the beginning of a list, and so the time for a successful search for item i should depend on how many entries were inserted after item i. Carefully compute the expected running time for a successful search. Assume that the item you are searching for is randomly chosen from among the items already in the table. (Hint: The unsuccessful search should take roughly twice as long as the successful one. Be sure to explain why this is the case.)

12. Suppose I hash $n \log n$ items into n buckets. What is the expected maximum number of items in a bucket?

13. The fact that $\lim_{n\to\infty}(1+\frac{1}{n})^n = e$ (where n varies over integers) is a consequence of the fact that $\lim_{h\to 0}(1+h)^{\frac{1}{h}} = e$ (where h varies over real numbers). Thus if h varies over negative real numbers, but approaches 0, the limit still exists and equals e. What does this tell you about $\lim_{n\to-\infty}(1+\frac{1}{n})^n$? Using this and rewriting $(1-\frac{1}{n})^n$ as $(1+\frac{1}{-n})^n$ show that

14. What is the expected number of empty slots when we hash $2k$ items into a hash table with k slots? What is the expected fraction of empty slots close to when k is reasonably large?

15. Using whatever methods you like (hand calculations or computer), give upper and/or lower bounds on the value of the x satisfying $x^x = n$.

16. Professor Max Weinberger decides that the method proposed for computing the maximum list size is much too complicated. He proposes the following solution. Let X_i be the size of list i. Then what we want to compute is $E(\max_i(X_i))$. Well

$$E(\max_i(X_i)) = \max_i(E(X_i)) = \max_i(1) = 1.$$

What is the flaw in his solution?

17. Prove as tight upper and lower bounds as you can on $\sum_{i=1}^{k}\frac{1}{i}$. For this purpose it is useful to remember the definition of the natural logarithm as an integral involving $1/x$ and to draw rectangles and other geometric figures above and below the curve.

18. Notice that $\ln n! = \sum_{i=1}^{n}\ln i$. Sketch a careful graph of $y = \ln x$, and by drawing in geometric figures above and below the graph, show that

$$\sum_{i=1}^{n}\ln i - \frac{1}{2}\ln n \le \int_1^n \ln x\,dx \le \sum_{i=1}^{n}\ln i.$$

Based on your drawing, which inequality do you think is tighter? Use integration by parts to evaluate the integral. What bounds on $n!$ can you get from these inequalities? Which one do you think is tighter? How does it compare to Stirling's approximation? What big Oh bound can you get on $n!$?

5.6 Conditional Expectations, Recurrences and Algorithms

Probability is a very important tool in algorithm design. We have already seen two important examples in which it is used—primality testing and hashing. In this section we will study several more examples of probabilistic analysis in algorithms. We will focus on computing the running time of various algorithms. When the running time of an algorithm is different for different inputs of the same size, we can think of the running time of the algorithm as a random variable on the sample space of inputs and analyze the expected running time of the algorithm. This us a different understanding from studying just the worst case running time for an input of a given size. We will then consider *randomized algorithms*, algorithms that depend on choosing something randomly, and see how we can use recurrences to give bounds on their expected running times as well.

For randomized algorithms, it will be useful to have access to a function which generates random numbers. We will assume that we have a function `randint(i,j)`, which generates a random integer uniformly between i and j (inclusive) [this means it is equally likely to be any number between i and j] and `rand01()`, which generates a random real number, uniformly between 0 and 1 [this means that given any two pairs of real numbers (r_1, r_2) and (s_1, s_2) with $r_2 - r_1 = s_2 - s_1$ and r_1, r_2, s_1 and s_2 all between 0 and 1, our random number is just as likely to be between r_1 and r_2 as it is to be between s_1 and s_2]. Functions such as `randint` and `rand01` are called *random number generators*. A great deal of number theory goes into the construction of good random number generators.

When Running Times Depend on more than Size of Inputs

Exercise 5.6-1 Let A be an array of length $n-1$ (whose elements are chosen from some ordered set), sorted into increasing order. Let b be another element of that ordered set that we want to insert into A to get a sorted array of length n. Assuming that the elements of A and b are chosen randomly, what is the expected number of elements of A that have to be shifted one place to the right to let us insert b?

Exercise 5.6-2 Let $A(1:n)$ denote the elements in positions 1 to n of the array A. A recursive description of insertion sort is that to sort $A(1:n)$, first we sort $A(1:n-1)$, and then we insert $A(n)$, by shifting the elements greater than $A(n)$ each one place to the right and then inserting the original value of $A(n)$ into the place we have opened up. If $n = 1$ we do nothing. Let $S_j(A(1:j))$ be the time needed to sort the portion of A from place 1 to place j, and let $I_j(A(1:j), b)$ be the time needed to insert the element b into a sorted list originally in the first j positions of A to give a sorted list in the first $j+1$ positions of A. Note that S_j and I_j depend on the actual array A, and not just on the value of j. Use S_j and I_j to describe the time needed to use insertion sort to sort $A(1:n)$ in terms of the time needed to sort $A(1:n-1)$. Don't forget that it is necessary to copy the element in position i of A into a variable b before we move elements of $A(1:i-1)$ to the right to make a place for it, because this moving process will write over $A(i)$. Let $T(n)$ be the expected value of S_n; that is, the expected running time of insertion sort on a list of n items. Write a recurrence for $T(n)$ in terms of $T(n-1)$ by taking expected values in the equation that corresponds to your previous description of the time needed to use insertion sort on a particular array. Solve your recurrence relation in big-Θ terms.

If X is the random variable with $X(A, b)$ equal to the number of items we need to move one place to the right in order to insert b into the resulting empty slot in A, then X takes on the values $0, 1, \ldots, n-1$ with equal probability $1/n$. Thus we have

$$E(x) = \sum_{i=0}^{n-1} i\frac{1}{n} = \frac{1}{n}\sum_{i=0}^{n-1} i = \frac{1}{n}\frac{(n-1)n}{2} = \frac{n-1}{2}.$$

Using $S_j(A(1:j))$ to stand for the time to sort the portion of the array A from places 1 to j by insertion sort, and $I_j(A(1:j), b)$ to stand for the time needed to insert b into a sorted list in the first j positions of the array A, moving all items larger than j to the right one place and putting b into the empty slot, we can write that for insertion sort

$$S_n(A(1:n)) = S_{n-1}(A(1:n-1)) + I_{n-1}(A(1:n-1), A(n)) + c_1.$$

We have included the constant term c_1 for the time it takes to copy the value of $A(n)$ into some variable b, because we will overwrite $A(n)$ in the process of moving items one place to the right. Using the additivity of expected values, we get

$$E(S_n) = E(S_{n-1}) + E(I_{n-1}) + E(c_1).$$

Using $T(n)$ for the expected time to sort $A(1:n)$ by insertion sort, and the result of the previous exercise, we get

$$T(n) = T(n-1) + c_2\frac{n-1}{2} + c_1.$$

where we include the constant c_2 because the time needed to do the insertion is going to be proportional to the number of items we have to move plus the time needed to copy the value of $A(n)$ into the appropriate slot (which we will assume we have included in c_1). We can say that $T(1) = 1$ (or some third constant) because with a list of size 1 we have to realize it has size 1, and then do nothing. It might be more realistic to write

$$T(n) \leq T(n-1) + cn$$

and

$$T(n) \geq T(n-1) + c'n,$$

because the time needed to do the insertion may not be exactly proportional to the number of items we need to move, but might depend on implementation details. By iterating the recurrence or drawing a recursion tree, we see that $T(n) = \Theta(n^2)$. (We could also give an inductive proof.) Since the best-case time of insertion sort is $\Theta(n)$ and the worst-case time is $\Theta(n^2)$, it is interesting to know that the expected case is much closer to the worst-case than the best case.

Conditional Expected Values

Our next example is cooked up to introduce an idea that we often use in analyzing the expected running times of algorithms, especially randomized algorithms.

Exercise 5.6-3 I have two nickels and two quarters in my left pocket and 4 dimes in my right pocket. Suppose I flip a penny and take two coins from my left pocket if it is heads, and two coins from my right pocket if it is tails. Assuming I am equally likely to choose any coin in my pocket at any time, what is the expected amount of money that I draw from my pocket?

You could do this problem by drawing a tree diagram or by observing that the outcomes can be modeled by three tuples in which the first entry is heads or tails, and the second and third entries are coins. Thus our sample space is HNQ, HQN, HQQ, HNN, TDD The probabilities of these outcomes are $\frac{1}{6}$, $\frac{1}{6}$, $\frac{1}{12}$, $\frac{1}{12}$, and $\frac{1}{2}$ respectively. Thus our expected value is

$$30\frac{1}{6} + 30\frac{1}{6} + 50\frac{1}{12} + 10\frac{1}{12} + 20\frac{1}{2} = 25.$$

Here is a method that seems even simpler. If the coin comes up heads, I have an expected value of 15 cents on each draw, so with probability 1/2, our expected value is 30 cents. If the coin comes up tails, I have an expected value of ten cents on each draw, so with probability 1/2 our expected value is 20 cents. Thus it is natural to expect that our expected value is $\frac{1}{2}30 + \frac{1}{2}20 = 25$ cents. In fact, if we group together the 4 outcomes with an H first, we see that their contribution to the expected value is 15 cents, which is 1/2 times 30, and if we look at the single element which has a T first, then its contribution to the sum is 10 cents, which is half of 20 cents.

In this second view of the problem, we took the probability of heads times the expected value of our draws, given that the penny came up heads, plus the probability of tails times the expected value of our draws, given that the penny came came up tails. In particular, we were using a new (and as yet undefined) idea of *conditional expected value.* To get the conditional expected value if our penny comes up heads, we could create a new sample space with four outcomes, NQ, QN, NN, QQ, with probabilities $\frac{1}{3}$, $\frac{1}{3}$, $\frac{1}{6}$, and $\frac{1}{6}$. In this sample space the expected amount of money we draw in two draws is 30 cents (15 cents for the first draw plus 15 cents for the second), so we would say the conditional expected value of our draws, given that the penny came up heads, was 30 cents. With a one-element sample space $\{DD\}$, we see that we would say that the conditional expected value of our draws, given that the penny came up tails, is 20 cents.

How do we define conditional expected value? Rather than create a new sample space as we did above, we use the idea of a new sample space (as we did in discovering a good definition for conditional probability) to lead us to a good definition for conditional expected value. Namely, to get the conditional expected value of X given that an event F has happened we use our conditional probability weights for the elements of F, namely $P(x)/P(F)$ is the weight for the element x of F, and pretend F is our sample space. Thus we define the **conditional expected value** of X given F by

$$E(X|F) = \sum_{x:x\in F} X(x)\frac{P(x)}{P(F)}. \tag{5.41}$$

Remember that we defined the expected value of a random variable X with values $x_1, x_2, \ldots x_k$ by

$$E(X) = \sum_{i=1}^{k} x_i P(X = x_i),$$

where $X = x_i$ stands for the event that X has the value x_i. Using our standard notation for conditional probabilities, $P(X = x_i|F)$ stands for the conditional probability of the event $X = x_i$

given the event F. This lets us rewrite Equation 5.41 as

$$E(X|F) = \sum_{i=1}^{k} x_i P(X = x_i|F).$$

Theorem 5.22 *Let X be a random variable defined on a sample space S and let F_1, F_2, $\ldots F_n$ be disjoint events whose union is S (i.e. a partition of S). Then*

$$E(X) = \sum_{i=1}^{n} E(X|F_i)P(F_i).$$

Proof: The proof is simply an exercise in applying definitions. ■

Randomized algorithms

Exercise 5.6-4 Consider an algorithm that, given a list of n numbers, prints them all out. Then it picks a random integer between 1 and 3. If the number is 1 or 2, it stops. If the number is 3 it starts again from the beginning. What is the expected running time of this algorithm?

Exercise 5.6-5 Consider the following variant on the previous algorithm:

```
funnyprint(n)
if $(n == 1)$
     return
for $i = 1$ to $n$
     print $i$
x = randint(1,n)
if $(x > n/2)$
     funnyprint(n/2)
else
     return
```

What is the expected running time of this algorithm?

For Exercise 5.6-4, with probability 2/3 we will print out the numbers and quit, and with probability 1/3 we will run the algorithm again. Using Theorem 5.22, we see that if $T(n)$ is the expected running time on a list of length n, then there is a constant c such that

$$T(n) = \frac{2}{3}cn + \frac{1}{3}(cn + T(n)),$$

which gives us $\frac{2}{3}T(n) = cn$. This simplifies to $T(n) = \frac{3}{2}cn$.

Another view is that we have an independent trials process, with success probability 2/3 where we stop at the first success, and for each round of the independent trials process we spend

$\Theta(n)$ time. Letting T be the running time (note that T is a random variable on the sample space $1, 2, 3$ with probabilities $\frac{1}{3}$ for each member) and R be the number of rounds, we have that

$$T = R \cdot \Theta(n)$$

and so

$$E(T) = E(R)\Theta(n).$$

Note that we are applying Theorem 5.10 since in this context $\Theta(n)$ behaves as if it were a constant[7], since n does not depend on R. By Lemma 5.12, we have that $E(R) = 3/2$ and so $E(T) = \Theta(n)$.

In Exercise 5.6-5, we have a recursive algorithm, and so it is appropriate to write down a recurrence. We can let $T(n)$ stand for the *expected* running time of the algorithm on an input of size n. Notice how we are changing back and forth between letting T stand for the running time of an algorithm and the expected running time of an algorithm. Usually we use T to stand for the quantity of most interest to us, either running time if that makes sense, or expected running time (or maybe worst-case running time) if the actual running time might vary over different inputs of size n. The nice thing will be that once we write down a recurrence for the expected running time of an algorithm, the methods for solving it will be those for we have already learned for solving recurrences. For the problem at hand, we immediately get that with probability 1/2 we will be spending n units of time (we should really say $\Theta(n)$ time), and then terminating, and with probability 1/2 we will spend n units of time and then recurse on a problem of size $n/2$. Thus using Theorem 5.22, we get that

$$T(n) = n + \frac{1}{2}T(n/2)$$

Including a base case of $T(1) = 1$, we get that

$$T(n) = \begin{cases} \frac{1}{2}T(n/2) + n & \text{if } n > 1 \\ 1 & \text{if } n = 1 \end{cases}.$$

A simple proof by induction shows that $T(n) = \Theta(n)$. Note that the Master Theorem (as we originally stated it) doesn't apply here, since $a < 1$. However, one could also observe that the solution to this recurrence is no more than the solution to the recurrence $T(n) = T(n/2) + n$, and then apply the Master Theorem.

Selection revisited

We now return to the selection algorithm from Section 4.6. The purpose of the algorithm is to select the ith smallest element in a set with some underlying order. Recall that in this algorithm, we first picked an an element p in the middle half of the set, that is, one whose value was simultaneously larger than at least 1/4 of the items and smaller than at least 1/4 of the items. We used p to partition the items into two sets and then recursed on one of the two sets. If you recall, we worked very hard to find an item in the middle half, so that our partitioning would work well. It is natural to try instead to just pick a partition element at random, because, with probability 1/2, this element will be in the middle half. We can extend this idea to the following algorithm:

[7]What we mean here is that $T \geq Rc_1n$ for some constant c_1 and $T \leq Rc_2n$ for some other constant c_2. Then we apply Theorem 5.10 to both these inequalities, using the fact that if $X > Y$, then $E(X) > E(Y)$ as well.

```
RandomSelect(A,i,n)
(selects the $i$th smallest element in set $A$, where $n = |A|$ )
if ($n = 1$)
    return the one item in $A$
else
    $p = randomElement(A)$
    Let $H$ be the set of elements greater than $p$
    Let $L$ be the set of elements less than or equal to $p$
    If $H$ is empty
        put $p$ in $H$
    if ($i \le |L|$)
        Return RandomSelect($L, i, |L|$)
    else
        Return RandomSelect($H, i - |L|, |H|$)
```

Here randomElement(A) returns one element from A uniformly at random. We use this element as our partition element; that is, we use it to divide A into sets L and H with every element less than the partition element in L and every element greater than it in H. We add the special case when H is empty, to ensure that both recursive problems have size strictly less than n. This simplifies a detailed analysis, but is not strictly necessary. At the end of this section we will show how to get a recurrence that describes fairly precisely the time needed to carry out this algorithm. However, by being a bit less precise, we can still get the same big-O upper bound with less work.

When we choose our partition element, half the time it will be between $\frac{1}{4}n$ and $\frac{3}{4}n$. Then when we partition our set into H and L, each of these sets will have no more than $\frac{3}{4}n$ elements. The other half of the time each of H and L will have no more than n elements. In any case, the time to partition our set into H and L is $O(n)$. Thus we may write

$$T(n) \le \begin{cases} \frac{1}{2}T(\frac{3}{4}n) + \frac{1}{2}T(n) + bn & \text{if } n > 1 \\ d & \text{if } n = 1. \end{cases}$$

We may rewrite the recursive part of the recurrence as

$$\frac{1}{2}T(n) \le \frac{1}{2}T\left(\frac{3}{4}n\right) + bn,$$

or

$$T(n) \le T\left(\frac{3}{4}n\right) + 2bn = T\left(\frac{3}{4}n\right) + b'n.$$

Notice that it is possible (but unlikely) that each time our algorithm chooses a pivot element, it chooses the worst one possible, in which case the selection process could take n rounds, and thus take time $\Theta(n^2)$. Why, then, is it of interest? If involves far less computation than finding the median of medians, and its expected running time is still $\Theta(n)$. Thus it is reasonable to suspect that on the average, it would be significantly faster than the deterministic process. In fact, with good implementations of both algorithms, this will be the case.

Exercise 5.6-6 Why does every solution to the recurrence

$$T(n) \le T\left(\frac{3}{4}n\right) + b'n.$$

have $T(n) = O(n)$?

By the master theorem we know that any solution to this recurrence is $O(n)$, giving a proof of our next Theorem.

Theorem 5.23 *Algorithm RandomSelect has expected running time $O(n)$.*

Quicksort

There are many algorithms that will efficiently sort a list of n numbers. The two most common sorting algorithms that are guaranteed to run in $O(n \log n)$ time are MergeSort and HeapSort. However, there is another algorithm, Quicksort, which, while having a worst-case running time of $O(n^2)$, has an expected running time of $O(n \log n)$. Moreover, when implemented well, it tends to have a faster running time than MergeSort or HeapSort. Since many computer operating systems and programs come with quicksort built in, it has become the sort of choice in many applications. In this section, we will see why it has expected running time $O(n \log n)$. We will not concern ourselves with the low-level implementation issues that make this algorithm the fastest one, but just with a high-level description.

Quicksort actually works similarly to the RecursiveSelect algorithm of the previous subsection. We pick a random element, and then use it to partition the set of items into two sets L and H. In this case, we don't recurse on one or the other, but recurse on both, sorting each one. After both L and H have been sorted, we just concatenate them to get a sorted list. (In fact, quicksort is usually done "in place" by pointer manipulation and so the concatenation just happens.) Here is a pseudocode description of quicksort.

```
 Quicksort(A,n)
if ($n = 1$)
     return the one item in $A$
else
     $p = randomElement(A)$
     Let $H$ be the set of elements greater than $p$; Let $h = |H|$
     Let $L$ be the set of elements less than or equal to $p$; Let $\ell = |L|$
     If $H$ is empty
          put $p$ in $H$
     $A_1$ = QuickSort(H,h)
     $A_2$ = QuickSort(L,$\ell$)
     return the concatenation of $A_1$ and $A_2$
```

There is an analysis of quicksort similar to the detailed analysis of RecursiveSelect at the end of the section, and this analysis is a problem at the end of the section. Instead, based on the preceding analysis of RandomSelect we will think about modifying the algorithm a bit in order to make the analysis easier. First, consider what would happen if the random element was the median each time. Then we would be solving two subproblems of size $n/2$, and would have the recurrence

$$T(n) = \begin{cases} 2T(n/2) + O(n) & \text{if } n > 1 \\ O(1) & \text{if } n = 1 \end{cases}$$

and we know by the master theorem that all solutions to this recurrence have $T(n) = O(n \log n)$. In fact, we don't need such an even division to guarantee such performance.

Exercise 5.6-7 Suppose you had a recurrence of the form

$$T(n) = \begin{cases} T(a_n n) + T((1-a_n)n) + O(n) & \text{if } n > 1 \\ O(1) & \text{if } n = 1 \end{cases},$$

where a_n is between $1/4$ and $3/4$. Show that all solutions of a recurrence of this form have $T(n) = O(n \log n)$. What do we really need to assume about a_n in order to prove this upper bound?

We can prove that $T(n) = O(n \log n)$ by induction, or via a recursion tree, noting that there are $O(\log n)$ levels, and each level has at most $O(n)$ work. (The details of the recursion tree are complicated somewhat by the fact that a_n varies with n, while the details of an inductive proof simply use the fact that a_n and $1 - a_n$ are both no more than $3/4$.) So long as we know there is some positive number $a < 1$ such that $a_n < a$ for every n, then we know we have at most $\log_{(1/a)} n$ levels in a recursion tree, with at most cn units of work per level for some constant c, and thus we have the same upper bound in big-O terms.

What does this tell us? As long as our problem splits into two pieces, each having size at least $1/4$ of the items, quicksort will run in $O(n \log n)$ time. Given this, we will modify our algorithm to enforce this condition. That is, if we choose a pivot element p that is not in the middle half, we will just pick another one. This leads to the following algorithm:

```
Slower Quicksort(A,n)
if ($n = 1$)
    return the one item in $A$
else
    Repeat
        $p = randomElement(A)$
        Let $H$ be the set of elements greater than $p$; Let $h = |H|$
        Let $L$ be the set of elements less than or equal to $p$; Let $\ell = |L|$
    Until ($|H| \geq n/4$) and ($|L| \geq n/4$)
    $A_1$ = QuickSort(H,h)
    $A_2$ = QuickSort(L,$\ell$)
    return the concatenation of $A_1$ and $A_2$
```

Now let's analyze this algorithm. Let r be the number of times we execute the loop to pick p, and let $a_n n$ be the position of the pivot element. Then if $T(n)$ is the expected running time for a list of length n, then for some constant b

$$T(n) \leq E(r)bn + T(a_n n) + T((1-a_n)n),$$

since each iteration of the loop takes $O(n)$ time. Note that we take the expectation of r, because $T(n)$ stands for the expected running time on a problem of size n. Fortunately, $E(r)$ is simple to compute, it is just the expected time until the first success in an independent trials process with success probability 1/2. This is 2. So we get that the running time of Slower Quicksort satisfies the recurrence

$$T(n) \le \begin{cases} T(a_n n) + T((1-a_n))n + b'n & \text{if } n > 1 \\ d & \text{if } n = 1 \end{cases},$$

where a_n is between 1/4 and 3/4. Thus by Exercise 5.6-7 the running time of this algorithm is $O(n \log n)$.

As another variant on the same theme, observe that looping until we have($|H| \ge n/4$ and $|L| \ge n/4$, is effectively the same as choosing p, finding H and L and then calling Slower Quicksort(A,n) once again if either H or L has size less than $n/4$. Then since with probability 1/2 the element p is between $n/4$ and $3n/4$, we can write

$$T(n) \le \frac{1}{2}T(n) + \frac{1}{2}(T(a_n n) + T((1-a_n)n) + bn),$$

which simplifies to

$$T(n) \le T(a_n n) + T((1-a_n)n) + 2bn,$$

or

$$T(n) \le T(a_n n) + T((1-a_n)n) + b'n.$$

Again by Exercise 5.6-7 the running time of this algorithm is $O(n \log n)$.

Further, it is straightforward to see that the expected running time of Slower Quicksort is no less than half that of Quicksort (and, incidentally, no more than twice that of quicksort) and so we have shown:

Theorem 5.24 *Quicksort has expected running time $O(n \log n)$.*

A more exact analysis of RandomSelect

Recall that our analysis of the RandomSelect was based on using $T(n)$ as an upper bound for $T(|H|)$ or $T(|L|)$ if either the set H or the set L had more than $3n/4$ elements. Here we show how one can avoid this assumption. The kinds of computations we do here are the kind we would need to do if we wanted to try to actually get bounds on the constants implicit in our big-O bounds.

Exercise 5.6-8 Explain why, if we pick the kth element as the random element in RandomSelect ($k \ne n$), our recursive problem is of size no more than $\max\{k, n-k\}$.

If we pick the kth element, then we recurse either on the set L, which has size k, or on the set H which has size $n-k$. Both of these sizes are at most $\max\{k, n-k\}$. (If we pick the nth element, then $k = n$ and thus L actually has size $k-1$ and H has size $n-k+1$.)

Now let X be the random variable equal to the rank of the chosen random element (e.g. if the random element is the third smallest, $X = 3$.) Using Theorem 5.22 and the solution to Exercise 5.6-8, we can write that

$$T(n) \le \begin{cases} \sum_{k=1}^{n-1} P(X=k)(T(\max\{k, n-k\}) + bn) + P(X=n)(T(\max\{1, n-1\} + bn) & \text{if } n > 1 \\ d & \text{if } n = 1. \end{cases}$$

Since X is chosen uniformly between 1 and n, $P(X = k) = 1/n$ for all k. Ignoring the base case for a minute, we get that

$$\begin{aligned} T(n) &\leq \sum_{k=1}^{n-1} \frac{1}{n}(T(\max\{k, n-k\}) + bn) + \frac{1}{n}(T(n-1) + bn) \\ &= \frac{1}{n}\left(\sum_{k=1}^{n-1} T(\max\{k, n-k\})\right) + bn + \frac{1}{n}(T(n-1) + bn). \end{aligned}$$

Now if n is odd and we write out $\sum_{k=1}^{n-1} T(\max\{k, n-k\})$, we get

$$T(n-1) + T(n-2) + \cdots + T(\lceil n/2 \rceil) + T\lceil n/2 \rceil) + \cdots + T(n-2) + T(n-1),$$

which is just $2\sum_{k=\lceil n/2 \rceil}^{n-1} T(k)$. If n is even we write out $\sum_{k=1}^{n-1} T(\max\{k, n-k\})$, we get

$$T(n-1) + T(n-2) + \cdots + T(n/2) + T(1 + n/2) + \cdots + T(n-2) + T(n-1),$$

which is less than $2\sum_{k=n/2}^{n-1} T(k)$. Thus we can replace our recurrence by

$$T(n) \leq \begin{cases} \frac{2}{n}\left(\sum_{k=n/2}^{n-1} T(k)\right) + \frac{1}{n}T(n-1) + bn & \text{if } n > 1 \\ d & \text{if } n = 1. \end{cases} \tag{5.42}$$

If n is odd, the lower limit of the sum is a half-integer, so the possible integer values of the dummy variable k run from $\lceil n/2 \rceil$ to $n - 1$. Since this is the natural way to interpret a fractional lower limit, and since it corresponds to what we wrote in both the n even and n odd case above, we adopt this convention.

Exercise 5.6-9 Show that every solution to the recurrence in Equation 5.42 has $T(n) = O(n)$.

We can prove this by induction. We try to prove that $T(n) \leq cn$ for some constant c. By the natural inductive hypothesis, we get that

$$\begin{aligned} T(n) &\leq \frac{2}{n}\left(\sum_{k=n/2}^{n-1} ck\right) + \frac{1}{n}c(n-1) + bn \\ &= \frac{2}{n}\left(\sum_{k=1}^{n-1} ck - \sum_{k=1}^{n/2-1} ck\right) + \frac{1}{n}c(n-1) + bn \\ &\leq \frac{2c}{n}\left(\frac{(n-1)n}{2} - \frac{(\frac{n}{2}-1)\frac{n}{2}}{2}\right) + c + bn \\ &= \frac{2c}{n}\frac{\frac{3n^2}{4} - \frac{n}{2}}{2} + c + bn \\ &= \frac{3}{4}cn + \frac{c}{2} + bn \\ &= cn - (\frac{1}{4}cn - bn - \frac{c}{2}) \end{aligned}$$

Notice that so far, we have only assumed that there is some constant c such that $T(k) < ck$ for $k < n$. We can choose a larger c than the one given to us by this assumption without changing the inequality $T(k) < ck$. By choosing c so that $\frac{1}{4}cn - bn - \frac{c}{2}$ is nonnegative (for example $c \geq 8b$ makes this term at least $bn - 2b$ which is nonnegative for $n \geq 2$), we conclude the proof, and have another proof of Theorem 5.23.

This kind of careful analysis arises when we are trying to get an estimate of the constant in a big-O bound (which we decided not to do in this case).

Important Concepts, Formulas, and Theorems

1. *Expected Running Time.* When the running time of an algorithm is different for different inputs of the same size, we can think of the running time of the algorithm as a random variable on the sample space of inputs and analyze the expected running time of the algorithm. This us a different understanding from studying just the worst case running time.

2. *Randomized Algorithm.* A*randomized algorithm* is an algorithm that depends on choosing something randomly.

3. *Random Number Generator.* A random number generator is a procedure that generates a number that appears to be chosen at random. Usually the designer of a random number generator tries to generate numbers that appear to be uniformly distributed.

4. *Insertion Sort.* A recursive description of insertion sort is that to sort $A(1 : n)$, first we sort $A(1 : n-1)$, and then we insert $A(n)$, by shifting the elements greater than $A(n)$ each one place to the right and then inserting the original value of $A(n)$ into the place we have opened up. If $n = 1$ we do nothing.

5. *Expected Running Time of Insertion Sort.* If $T(n)$ is the expected time to use insertion sort on a list of length n, then there are constants c and c' such that $T(n) \leq T(n-1) + cn$ and $T(n) \geq T(n-1) + c'n,$. This means that $T(n) = \Theta(n^2)$. However the best case running time of insertion sort is $\Theta(n)$.

6. *Conditional Expected Value.* We define the *conditional expected value* of X given F by $E(X|F) = \sum_{x:x\in F} X(x)\frac{P(x)}{P(F)}$. This is equivalent to $E(X|F) = \sum_{i=1}^{k} x_i P(X = x_i|F)$.

7. *Randomized Selection Algorithm.* In the randomized selection algorithm to select the ith smallest element of a set A, we randomly choose a pivot element p in A, divide the rest of A into those elements that come before p (in the underlying order of A) and those that come after, put the pivot into the smaller set, and then recursively apply the randomized selection algorithm to find the appropriate element of the appropriate set.

8. *Running Time of Randomized Select.* Algorithm RandomSelect has expected running time $O(n)$. Because it does less computation than the deterministic selection algorithm, on the average a good implementation will run faster than a good implementation of the deterministic algorithm, but the worst case behavior is $\Theta(n^2)$.

9. *Quicksort.* Quicksort is a sorting algorithm in which we randomly choose a pivot element p in A, divide the rest of A into those elements that come before p (in the underlying order of A) and those that come after, put the pivot into the smaller set, and then recursively apply the Quicksort algorithm to sort each of the smaller sets, and concatenate the two sorted lists. We do nothing if a set has size one.

10. *Running Time of Quicksort.* Quicksort has expected running time $O(n \log n)$. It has worst case running time $\Theta(n^2)$. Good implementations of Quicksort have proved to be faster on the average than good implementations of other sorting algorithms.

Problems

1. Given an array A of length n (chosen from some set that has an underlying ordering), we can select the largest element of the array by starting out setting $L = A(1)$, and then comparing L to the remaining elements of the array one at a time, replacing L by $A(i)$ if $A(i)$ is larger than L. Assume that the elements of A are randomly chosen. For $i > 1$, let X_i be 1 if element i of A is larger than any element of $A(1 : i - 1)$. Let $X_1 = 1$. Then what does $X_1 + X_2 + \cdots + X_n$ have to do with the number of times we assign a value to L? What is the expected number of times we assign a value to L?

2. Let $A(i : j)$ denote the array of items in positions i through j of the Array A. In selection sort, we use the method of Exercise 5.6-1 to find the largest element of the array A and its position k in the array, then we exchange the elements in position k and n of Array A, and we apply the same procedure recursively to the array $A(1 : n - 1)$. (Actually we do this if $n > 1$; if $n = 1$ we do nothing.) What is the expected total number of times we assign a value to L in the algorithm selection sort?

3. Show that if H_n stands for the nth harmonic number, then
$$H_n + H_{n-1} + \cdots + H_2 = \Theta(n \log n).$$

4. In a card game, we remove the Jacks, Queens, Kings, and Aces from a deck of ordinary cards and shuffle them. You draw a card. If it is an Ace, you are paid a dollar and the game is repeated. If it is a Jack, you are paid two dollars and the game ends; if it is a Queen, you are paid three dollars and the game ends; and if it is a King, you are paid four dollars and the game ends. What is the maximum amount of money a rational person would pay to play this game?

5. Why does every solution to $T(n) \le T(\frac{2}{3}n) + bn$ have $T(n) = O(n)$?

6. Show that if in Algorithm Random Select we remove the instruction

```
If $H$ is empty
    put $p$ in $H$,
```

then if $T(n)$ is the expected running time of the algorithm, there is a constant b such that $T(n)$ satisfies the recurrence
$$T(n) \le \frac{2}{n-1} \sum_{k=n/2}^{n-1} T(k) + bn.$$
Show that if $T(n)$ satisfies this recurrence, then $T(n) = O(n)$.

7. Suppose you have a recurrence of the form

$$T(n) \leq T(a_n n) + T((1-a_n)n) + bn \text{ if } n > 1,$$

where a_n is between $\frac{1}{5}$ and $\frac{4}{5}$. Show that all solutions to this recurrence are of the form $T(n) = O(n \log n)$.

8. Prove Theorem 5.22.

9. A tighter (up to constant factors) analysis of quicksort is possible by using ideas very similar to those that we used for the randomized selection algorithm. More precisely, we use Theorem 5.6.1, similarly to the way we used it for select. Write down the recurrence you get when you do this. Show that this recurrence has solution $O(n \log n)$. In order to do this, you will probably want to prove that $T(n) \leq c_1 n \log n - c_2 n$ for some constants c_1 and c_2.

10. It is also possible to write a version of rhe randomized Selection algorithm analogous to Slower Quicksort. That is, when we pick out the random pivot element, we check if it is in the middle half and discard it if it is not. Write this modified selection algorithm, give a recurrence for its running time, and show that this recurrence has solution $O(n)$.

11. One idea that is often used in selection is that instead of choosing a random pivot element, we choose three random pivot elements and then use the median of these three as our pivot. What is the probability that a randomly chosen pivot element is in the middle half? What is the probability that the median of three randomly chosen pivot elements is in the middle half? Does this justify the choice of using the median of three as pivot?

12. Is the expected running time of Quicksort $\Omega(n \log n)$?

13. A random binary search tree on n keys is formed by first randomly ordering the keys, and then inserting them in that order. Explain why in at least half the random binary search trees, both subtrees of the root have between $\frac{1}{4}n$ and $\frac{3}{4}n$ keys. If $T(n)$ is the expected height of a random binary search tree on n keys, explain why $T(n) \leq \frac{1}{2}T(n) + \frac{1}{2}T(\frac{3}{4}n) + 1$. (Think about the **definition** of a binary tree. It has a root, and the root has two subtrees! What did we say about the possible sizes of those subtrees?) What is the expected height of a one node binary search tree? Show that the expected height of a random binary search tree is $O(\log n)$.

14. The expected time for an unsuccessful search in a random binary search tree on n keys (see Problem 13 for a definition) is the expected depth of a leaf node. Arguing as in Problem 13 and the second proof of Theorem 5.6.2, find a recurrence that gives an upper bound on the expected depth of a leaf node in a binary search tree and use it to find a big Oh upper bound on the expected depth of a leaf node.

15. The expected time for a successful search in a random binary search tree on n nodes (see problem 13 for a definition) is the expected depth of a node of the tree. With probability $\frac{1}{n}$ the node is the root, which has depth 0; otherwise the expected depth is one plus the expected depth of one of its subtrees. Argue as in Problem 13 and the first proof of Theorem 5.23 to show that if $T(n)$ is the expected depth of a node in a binary search tree, then $T(n) \leq \frac{n-1}{n}(\frac{1}{2}T(n) + \frac{1}{2}T(\frac{3}{4}n)) + 1$. What big Oh upper bound does this give you on the expected depth of a node in a random binary search tree on n nodes?

16. Consider the following code for searching an array A for the maximum item:

```
$max = -\infty$
for $i = 1$ to $n$
    if $(A[i] > max)$
        $max = A[i]$
```

If A initially consists of n nodes in a random order, what is the expected number of times that the line $max = A[i]$ is executed? (Hint: Let X_i be the number of times that $max = A[i]$ is executed in the ith iteration of the loop.)

17. You are a contestant in the game show "Let's make a Deal." In this game show, there are three curtains. Behind one of the curtains is a new car, and behind the other two are cans of spam. You get to pick one of the curtains. After you pick that curtain, the emcee, Monte Hall, who we assume knows where the car is, reveals what is behind one of the curtains that you did not pick, showing you some cans of spam. He then asks you if you would like to switch your choice of curtain. Should you switch? Why or why not? Please answer this question carefully. You have all the tools needed to answer it, but several math Ph.D.'s are on record (in Parade Magazine) giving the wrong answer.

5.7 Probability Distributions and Variance

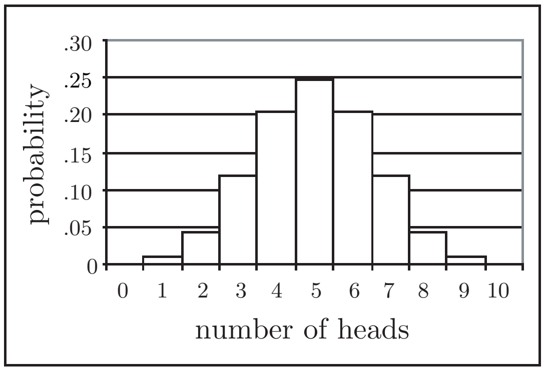
Distributions of random variables

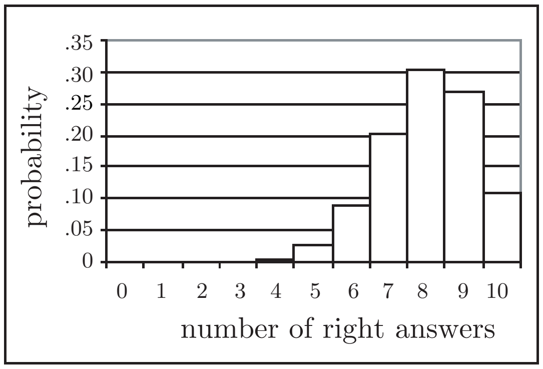
We have given meaning to the phrase expected value. For example, if we flip a coin 100 times, the expected number of heads is 50. But to what extent do we expect to see 50 heads. Would it be surprising to see 55, 60 or 65 heads instead? To answer this kind of question, we have to analyze how much we expect a random variable to deviate from its expected value. We will first see how to analyze graphically how the values of a random variable are distributed around its expected value. The *distribution function* D of a random variable X is the function on the values of X defined by

$$D(x) = P(X = x).$$

You probably recognize the distribution function from the role it played in the definition of expected value. The distribution function of X assigns to each value of X the probability that X achieves that value. When the values of X are integers, it is convenient to visualize the distribution function with a diagram called a *histogram.* In Figure 5.8 we show histograms for the distribution of the "number of heads" random variable for ten flips of a coin and the "number of right answers" random variable for someone taking a ten question test with probability .8 of getting a correct answer. What is a histogram? Those we have drawn are graphs which show for for each integer value x of X a rectangle of width 1, centered at x, whose height (and thus area) is proportional to the probability $P(X = x)$. Histograms can be drawn with non-unit width rectangles. When people draw a rectangle with a base ranging from $x = a$ to $x = b$, the area of the rectangle is the probability that X is between a and b.

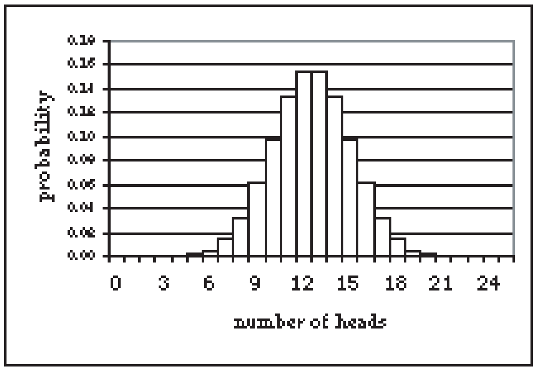
Figure 5.8: Two histograms.

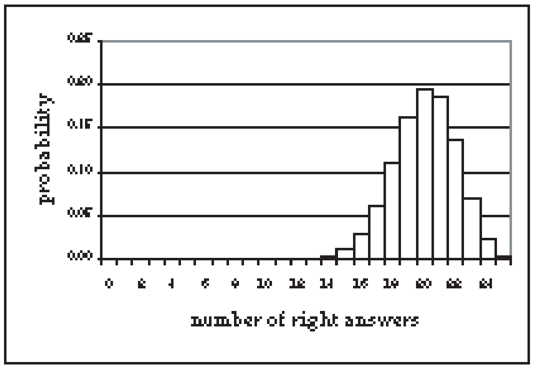




From the histograms you can see the difference in the two distributions. You can also see that we can expect the number of heads to be somewhat near the expected number, though as few heads as 2 or as many as 8 are not out of the question. We see that the number of right answers tends to be clustered between 6 and ten, so in this case we can expect to be reasonably close to the expected value. With more coin flips or more questions, however, will the results spread out? Relatively speaking, should we expect to be closer to or farther from the expected value? In Figure 5.9 we show the results of 25 coin flips or 25 questions. The expected number of heads is 12.5. The histogram makes it clear that we can expect the vast majority of our results to have between 9 and 16 heads. Essentially all the results lie between 4 and 20 Thus the results are not spread as broadly (relatively speaking) as they were with just ten flips. Once again the test score histogram seems even more tightly packed around its expected value. Essentially all the scores lie between 14 and 25. While we can still tell the difference between the shapes of the histograms, they have become somewhat similar in appearance.

Figure 5.9: Histograms of 25 trials

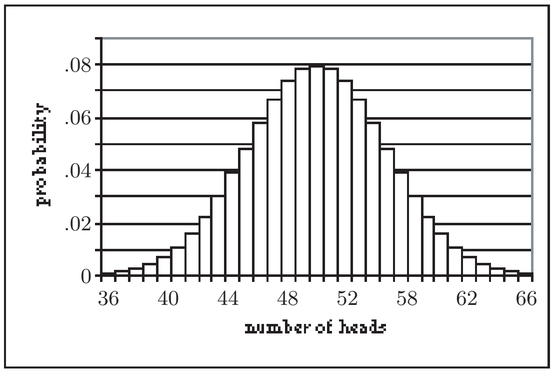




In Figure 5.10 we have shown the thirty most relevant values for 100 flips of a coin and a 100 question test. Now the two histograms have almost the same shape, though the test histogram is still more tightly packed around its expected value. The number of heads has virtually no chance of deviating by more than 15 from its expected value, and the test score has almost no chance of deviating by more than 11 from the expected value. Thus the spread has only doubled, even though the number of trials has quadrupled. In both cases the curve formed by the tops of the rectangles seems quite similar to the bell shaped curve called the normal curve that arises in so many areas of science. In the test-taking curve, though, you can see a bit of difference between the lower left-hand side and the lower right hand side.

Since we needed about 30 values to see the most relevant probabilities for these curves, while we needed 15 values to see most of the relevant probabilities for independent trials with 25 items,

Figure 5.10: One hundred independent trials



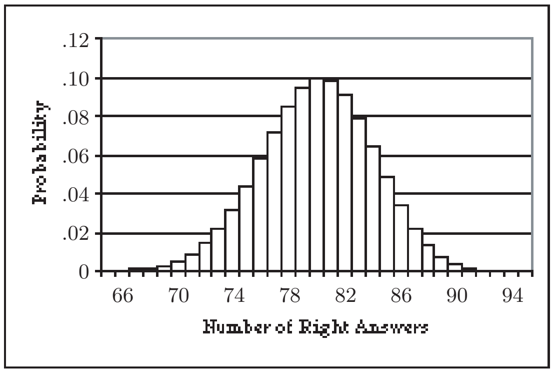
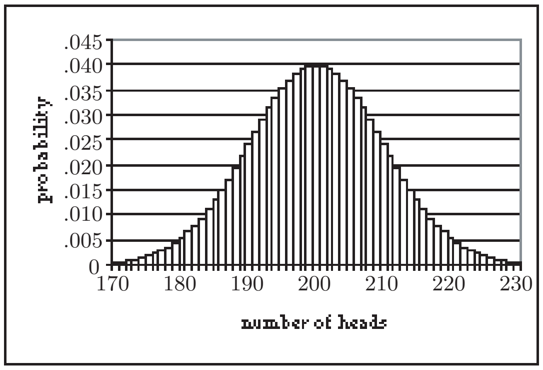
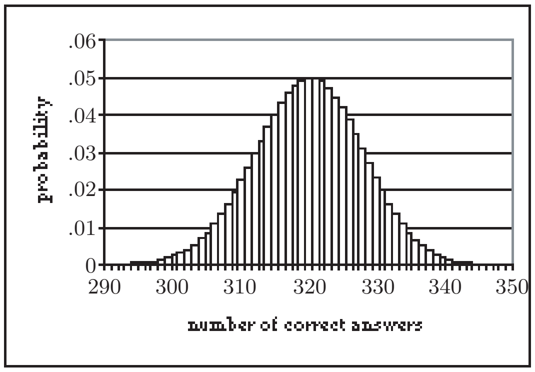


Figure 5.11: Four hundred independent trials





we might predict that we would need only about 60 values to see essentially all the results in four hundred trials. As Figure 5.11 shows, this is indeed the case. The test taking distribution is still more tightly packed than the coin flipping distribution, but we have to examine it closely to find any asymmetry. These experiments are convincing, and they suggest that the spread of a distribution (for independent trials) grows as the square root of the number of trials, because each time we quadruple the number of elements, we double the spread. They also suggest there is some common kind of bell-shaped limiting distribution function for at least the distribution of successes in independent trials with two outcomes. However without a theoretical foundation we don't know how far the truth of our observations extends. Thus we seek an algebraic expression of our observations. This algebraic measure should somehow measure the difference between a random variable and its expected value.

Variance

Exercise 5.7-1 Suppose the X is the number of heads in four flips of a coin. Let Y be the random variable $X-2$, the difference between X and its expected value. Compute $E(Y)$. Does it effectively measure how much we expect to see X deviate from its expected value? Compute $E(Y^2)$. Try repeating the process with X being the number of heads in ten flips of a coin and Y being $X-5$.

Before answering these questions, we state a trivial, but useful lemma (which appeared as Problem 9 in Section 4 of this chapter and corollary showing that the expected value of an expectation is that expectation.

Lemma 5.25 *If X is a random variable that always takes on the value c, then $E(X) = c$.*

Proof: $E(X) = P(X = c) \cdot c = 1 \cdot c = c.$ ■

We can think of a constant c as a random variable that always takes on the value c. When we do, we will just write $E(c)$ for the expected value of this random variable, in which case our lemma says that $E(c) = c$. This lemma has an important corollary.

Corollary 5.26 $E(E(X)) = E(X)$.

Proof: When we think of $E(X)$ as a random variable, it has a constant value, μ. By Lemma 5.25 $E(E(x)) = E(\mu) = \mu = E(x)$. ■

Returning to Exercise 5.7-1, we can use linearity of expectation and Corollary 5.26 to show that

$$E(X - E(X)) = E(X) - E(E(X)) = E(X) - E(X) = 0. \tag{5.43}$$

Thus this is not a particularly useful measure of how close a random variable is to its expectation. If a random variable is sometimes above its expectation and sometimes below, you would like these two differences to somehow add together, rather than cancel each other out. This suggests we try to convert the values of $X - E(X)$ to positive numbers in some way and then take the expectation of these positive numbers as our measure of spread. There are two natural ways to make numbers positive, taking their absolute value and squaring them. It turns our that to prove things about the spread of expected values, squaring is more useful. Could we have guessed that? Perhaps, since we see that the spread seems to grow with the square root, and the square root isn't related to the absolute value in the way it is related to the squaring function. On the other hand, as you saw in the example, computing expected values of these squares from what we know now is time consuming. A bit of theory will make it easier.

We define the **variance** $V(X)$ of a random variable X as the expected value of $(X - E(X))^2$. We can also express this as a sum over the individual elements of the sample space S and get that

$$V(X) = E(X - E(X))^2 = \sum_{s:s\in S} P(s)(X(s) - E(X))^2. \tag{5.44}$$

Now let's apply this definition and compute the variance in the number X of heads in four flips of a coin. We have

$$V(X) = (0-2)^2 \cdot \frac{1}{16} + (1-2)^2 \cdot \frac{1}{4} + (2-2)^2 \cdot \frac{3}{8} + (3-2)^2 \cdot \frac{1}{4} + (4-2)^2 \cdot \frac{1}{16} = 1.$$

Computing the variance for ten flips of a coin involves some very inconvenient arithmetic. It would be nice to have a computational technique that would save us from having to figure out large sums if we want to compute the variance for ten or even 100 or 400 flips of a coin to check our intuition about how the spread of a distribution grows. We saw before that the expected value of a sum of random variables is the sum of the expected values of the random variables. This was very useful in making computations.

Exercise 5.7-2 What is the variance for the number of heads in one flip of a coin? What is the sum of the variances for four independent trials of one flip of a coin?

Exercise 5.7-3 We have a nickel and quarter in a cup. We withdraw one coin. What is the expected amount of money we withdraw? What is the variance? We withdraw two coins, one after the other without replacement. What is the expected amount of money we withdraw? What is the variance? What is the expected amount of money and variance for the first draw? For the second draw?

Exercise 5.7-4 Compute the variance for the number of right answers when we answer one question with probability .8 of getting the right answer (note that the number of right answers is either 0 or 1, but the expected value need not be). Compute the variance for the number of right answers when we answer 5 questions with probability .8 of getting the right answer. Do you see a relationship?

In Exercise 5.7-2 we can compute the variance

$$V(X) = (0 - \frac{1}{2})^2 \cdot \frac{1}{2} + (1 - \frac{1}{2})^2 \cdot \frac{1}{2} = \frac{1}{4}.$$

Thus we see that the variance for one flip is 1/4 and sum of the variances for four flips is 1. In Exercise 5.7-4 we see that for one question the variance is

$$V(X) = .2(0 - .8)^2 + .8(1 - .8)^2 = .16$$

For five questions the variance is

$$4^2 \cdot (.2)^5 + 3^2 \cdot 5 \cdot (.2)^4 \cdot (.8) + 2^2 \cdot 10 \cdot (.2)^3 \cdot (.8)^2 + 1^2 \cdot 10 \cdot (.2)^2 \cdot (.8)^3 + \\ 0^2 \cdot 5 \cdot (.2)^1 \cdot (.8)^4 + 1^2 \cdot (.8)^5 = .8$$

The result is five times the variance for one question.

For Exercise 5.7-3 the expected amount of money for one draw is \$.15. The variance is

$$(.05 - .15)^2 \cdot .5 + (.25 - .15)^2 \cdot .5 = .01.$$

For removing both coins, one after the other, the expected amount of money is \$.30 and the variance is 0. Finally the expected value and variance on the first draw are \$.15 and .01 and the expected value and variance on the second draw are \$.15 and .01.

It would be nice if we had a simple method for computing variance by using a rule like "the expected value of a sum is the sum of the expected values." However Exercise 5.7-3 shows that the variance of a sum is not always the sum of the variances. On the other hand, Exercise 5.7-2 and Exercise 5.7-4 suggest such a result might be true for a sum of variances in independent trials processes. In fact slightly more is true. We say random variables X and Y are *independent* when the event that X has value x is independent of the event that Y has value y, regardless of the choice of x and y. For example, in n flips of a coin, the number of heads on flip i (which is 0 or 1) is independent of the number of heads on flip j. To show that the variance of a sum of independent random variables is the sum of their variances, we first need to show that the expected value of the product of two *independent* random variables is the product of their expected values.

Lemma 5.27 *If X and Y are independent random variables on a sample space S with values $x_1, x_2, \ldots, x_k$ and $y_1, y_2, \ldots, y_m$ respectively, then*

$$E(XY) = E(X)E(Y).$$

Proof: We prove the lemma by the following series of equalities. In going from (5.45) to (5.46), we use the fact that X and Y are independent; the rest of the equations follow from definitions and algebra.

$$\begin{aligned}
E(X)E(Y) &= \sum_{i=1}^{k} x_i P(X = x_i) \sum_{j=1}^{m} y_j P(Y = y_j) \\
&= \sum_{i=1}^{k}\sum_{j=1}^{m} x_i y_j P(X = x_i)P(y = y_j) \\
&= \sum_{z:\, z \text{is a value of } XY} z \sum_{(i,j):x_iy_j=z} P(X = x_i)P(Y = y_j) \qquad (5.45) \\
&= \sum_{z:\, z \text{is a value of } XY} z \sum_{(i,j):x_iy_j=z} P((X = x_i) \wedge (Y = y_j)) \qquad (5.46) \\
&= \sum_{z:\, z \text{is a value of } XY} zP(XY = z) \\
&= E(XY)
\end{aligned}$$

■

Theorem 5.28 *If X and Y are independent random variables then*

$$V(X + Y) = V(X) + V(Y).$$

Proof: Using the definitions, algebra and linearity of expectation we have

$$\begin{aligned}
V(X + Y) &= E((X + Y) - E(X + Y))^2 \\
&= E(X - E(X) + Y - E(Y))^2 \\
&= E((X - E(X))^2 + 2(X - E(X))(Y - E(Y)) + (Y - E(Y))^2) \\
&= E(X - E(X))^2 + 2E((X - E(X))(Y - E(Y)) + E(Y - E(Y))^2
\end{aligned}$$

Now the first and last terms and just the definitions of $V(X)$ and $V(Y)$ respectively. Note also that if X and Y are independent and b and c are constants, then $X-b$ and $Y-c$ are independent (See Problem 8 at the end of this section.) Thus we can apply Lemma 5.27 to the middle term to obtain

$$= V(X) + 2E(X - E(X))E(Y - E(Y)) + V(Y).$$

Now we apply Equation 5.43 to the middle term to show that it is 0. This proves the theorem. ■

With this theorem, computing the variance for ten flips of a coin is easy; as usual we have the random variable X_i that is 1 or 0 depending on whether or not the coin comes up heads. We saw that the variance of X_i is 1/4, so the variance for $X_1 + X_2 + \cdots + X_{10}$ is $10/4 = 2.5$.

Exercise 5.7-5 Find the variance for 100 flips of a coin and 400 flips of a coin.

Exercise 5.7-6 The variance in the previous problem grew by a factor of four when the number of trials grew by a factor of 4, while the spread we observed in our histograms grew by a factor of 2. Can you suggest a natural measure of spread that fixes this problem?

For Exercise 5.7-5 recall that the variance for one flip was $1/4$. Therefore the variance for 100 flips is 25 and the variance for 400 flips is 100. Since this measure grows linearly with the size, we can take its square root to give a measure of spread that grows with the square root of the quiz size, as our observed "spread" did in the histograms. Taking the square root actually makes intuitive sense, because it "corrects" for the fact that we were measuring expected squared spread rather than expected spread.

The square root of the variance of a random variable is called the *standard deviation* of the random variable and is denoted by σ, or $\sigma(X)$ when there is a chance for confusion as to what random variable we are discussing. Thus the standard deviation for 100 flips is 5 and for 400 flips is 10. Notice that in both the 100 flip case and the 400 flip case, the "spread" we observed in the histogram was ± 3 standard deviations from the expected value. What about for 25 flips? For 25 flips the standard deviation will be $5/2$, so ± 3 standard deviations from the expected value is a range of 15 points, again what we observed. For the test scores the variance is .16 for one question, so the standard deviation for 25 questions will be 2, giving us a range of 12 points. For 100 questions the standard deviation will be 4, and for 400 questions the standard deviation will be 8. Notice again how three standard deviations relate to the spread we see in the histograms.

Our observed relationship between the spread and the standard deviation is no accident. A consequence of a theorem of probability known as the central limit theorem is that the percentage of results within one standard deviation of the mean in a relatively large number of independent trials with two outcomes is about 68%; the percentage within two standard deviations of the mean is about 95.5%, and the percentage within three standard deviations of the mean is about 99.7%.

What the central limit theorem says is that the sum of independent random variables with the same distribution function is approximated well by saying that the probability that the sum is between a and b is an appropriately chosen multiple of $\int_a^b e^{-cx^2}\,dx$ (where c is an appropriate constant) when the number of random variables we are adding is sufficiently large.[8] The distribution given by that multiple of the integral is called the *normal distribution*. Since many of the things we observe in nature can be thought of as the outcome of multistage processes, and the quantities we measure are often the result of adding some quantity at each stage, the central limit theorem "explains" why we should expect to see normal distributions for so many of the things we do measure. While weights can be thought of as the sum of the weight change due to eating and exercise each week, say, this is not a natural interpretation for blood pressures. Thus while we shouldn't be particularly surprised that weights are normally distributed, we don't have the same basis for predicting that blood pressures would be normally distributed, even though they are!

Exercise 5.7-7 If we want to be 95% sure that the number of heads in n flips of a coin is within $\pm 1\%$ of the expected value, how big does n have to be?

Exercise 5.7-8 What is the variance and standard deviation for the number of right answers for someone taking a 100 question short answer test where each answer is graded either correct or incorrect if the person knows 80% of the subject material for the test the test and answers correctly each question she knows? Should we be surprised if such a student scores 90 or above on the test?

[8]Still more precisely, if we let μ be the expected value of the random variable X_i and σ be its standard deviation (all X_i have the same expected value and standard distribution since they have the same distribution) and scale the sum of our random variables by $Z = \frac{X_1+X_2+\ldots X_n - n\mu}{\sigma\sqrt{n}}$, then the probability that $a \le Z \le b$ is $\int_a^b \frac{1}{\sqrt{2\pi}} e^{-x^2/2}\,dx$.

Recall that for one flip of a coin the variance is $1/4$, so that for n flips it is $n/4$. Thus for n flips the standard deviation is $\sqrt{n}/2$. We expect that 95% of our outcomes will be within 2 standard deviations of the mean (people always round 95.5 to 95) so we are asking when two standard deviations are 1% of $n/2$. Thus we want an n such that $2\sqrt{n}/2 = .01(.5n)$, or such that $\sqrt{n} = 5 \cdot 10^{-3}n$, or $n = 25 \cdot 10^{-6}n^2$. This gives us $n = 10^6/25 = 40,000$.

For Exercise 5.7-8, the expected number of correct answers on any given question is .8. The variance for each answer is $.8(1-.8)^2 + .2(0-.8)^2 = .8 \cdot .04 + .2 \cdot .64 = .032 + .128 = .16$. Notice this is $.8 \cdot (1-.8)$. The total score is the sum of the random variables giving the number of points on each question, and assuming the questions are independent of each other, the variance of their sum is the sum of their variances, or 16. Thus the standard deviation is 4. Since 90% is 2.5 standard deviations above the expected value, the probability of getting that a score that far from the expected value is somewhere between .05 and .003 by the Central Limit Theorem. (In fact it is just a bit more than .01). Assuming that someone is just as likely to be 2.5 standard deviations below the expected score as above, which is not exactly right but close, we see that it is quite unlikely that someone who knows 80% of the material would score 90% or above on the test. Thus we should be surprised by such a score, and take the score as evidence that the student likely knows more than 80% of the material.

Coin flipping and test taking are two special cases of Bernoulli trials. With the same kind of computations we used for the test score random variable, we can prove the following.

Theorem 5.29 *In Bernoulli trials with probability p of success, the variance for one trial is $p(1-p)$ and for n trials is $np(1-p)$, so the standard deviation for n trials is $\sqrt{np(1-p)}$.*

Proof: You are asked to give the proof in Problem 7.■

Important Concepts, Formulas, and Theorems

1. *Histogram. Histograms* are graphs which show for for each integer value x of a random variable X a rectangle of width 1, centered at x, whose height (and thus area) is proportional to the probability $P(X = x)$. Histograms can be drawn with non-unit width rectangles. When people draw a rectangle with a base ranging from $x = a$ to $x = b$, the area of the rectangle is the probability that X is between a and b.

2. *Expected Value of a Constant.* If X is a random variable that always takes on the value c, then $E(X) = c$. In particular, $E(E(X)) = E(X)$.

3. *Variance.* We define the *variance* $V(X)$ of a random variable X as the expected value of $(X - E(X))^2$. We can also express this as a sum over the individual elements of the sample space S and get that $V(X) = E(X - E(X))^2 = \sum_{s:s\in S} P(s)(X(s) - E(X))^2$.

4. *Independent Random Variables.* We say random variables X and Y are *independent* when the event that X has value x is independent of the event that Y has value y, regardless of the choice of x and y.

5. *Expected Product of Independent Random Variables.* If X and Y are independent random variables on a sample space S, then $E(XY) = E(X)E(Y)$.

6. *Variance of Sum of Independent Random Variables.* If X and Y are independent random variables then $V(X+Y) = V(X) + V(Y)$.

7. *Standard deviation.* The square root of the variance of a random variable is called the *standard deviation* of the random variable and is denoted by σ, or $\sigma(X)$ when there is a chance for confusion as to what random variable we are discussing.

8. *Variance and Standard Deviation for Bernoulli Trials.* In Bernoulli trials with probability p of success, the variance for one trial is $p(1-p)$ and for n trials is $np(1-p)$, so the standard deviation for n trials is $\sqrt{np(1-p)}$.

9. *Central Limit Theorem.* The central limit theorem says that the sum of independent random variables with the same distribution function is approximated well by saying that the probability that the random variable is between a and b is an appropriately chosen multiple of $\int_a^b e^{-cx^2}\,dx$, for some constant c, when the number of random variables we are adding is sufficiently large. This implies that the probability that a sum of independent random variables is within one, two, or three standard deviations of its expected value is approximately .68, .955, and .997.

Problems

1. Suppose someone who knows 60% of the material covered in a chapter of a textbook is taking a five question objective (each answer is either right or wrong, not multiple choice or true-false) quiz. Let X be the random variable that for each possible quiz, gives the number of questions the student answers correctly. What is the expected value of the random variable $X-3$? What is the expected value of $(X-3)^2$? What is the variance of X?

2. In Problem 1 let X_i be the number of correct answers the student gets on question i, so that X_i is either zero or one. What is the expected value of X_i? What is the variance of X_i? How does the sum of the variances of X_1 through X_5 relate to the variance of X for Problem 1?

3. We have a dime and a fifty cent piece in a cup. We withdraw one coin. What is the expected amount of money we withdraw? What is the variance? Now we draw a second coin, without replacing the first. What is the expected amount of money we withdraw? What is the variance? Suppose instead we consider withdrawing two coins from the cup together. What is the expected amount of money we withdraw, and what is the variance? What does this example show about whether the variance of a sum of random variables is the sum of their variances.

4. If the quiz in Problem 1 has 100 questions, what is the expected number of right answers, the variance of the expected number of right answers, and the standard deviation of the number of right answers?

5. Estimate the probability that a person who knows 60% of the material gets a grade strictly between 50 and 70 in the test of Exercise 5.7-4

6. What is the variance in the number of right answers for someone who knows 80% of the material on which a 25 question quiz is based? What if the quiz has 100 questions? 400 questions? How can we "correct" these variances for the fact that the "spread" in the histogram for the number of right answers random variable only doubled when we multiplied the number of questions in a test by 4?

7. Prove Theorem 5.29.

8. Show that if X and Y are independent and b and c are constant, then $X - b$ and $Y - c$ are independent.

9. We have a nickel, dime and quarter in a cup. We withdraw two coins, first one and then the second, without replacement. What is the expected amount of money and variance for the first draw? For the second draw? For the sum of both draws?

10. Show that the variance for n independent trials with two outcomes and probability p of success is given by $np(1-p)$. What is the standard deviation? What are the corresponding values for the number of failures random variable?

11. What are the variance and standard deviation for the sum of the tops of n dice that we roll?

12. How many questions need to be on a short answer test for us to be 95% sure that someone who knows 80% of the course material gets a grade between 75% and 85%?

13. Is a score of 70% on a 100 question true-false test consistent with the hypothesis that the test taker was just guessing? What about a 10 question true-false test? (This is not a plug and chug problem; you have to come up with your own definition of "consistent with.")

14. Given a random variable X, how does the variance of cX relate to that of X?

15. Draw a graph of the equation $y = x(1 - x)$ for x between 0 and 1. What is the maximum value of y? Why does this show that the variance (see Problem 10 in this section) of the "number of successes" random variable for n independent trials is less than or equal to $n/4$?

16. This problem develops an important law of probability known as *Chebyshev's law.* Suppose we are given a real number $r > 0$ and we want to estimate the probability that the difference $|X(x) - E(X)|$ of a random variable from its expected value is more than r.

    (a) Let $S = \{x_1, x_2, \ldots, x_n\}$ be the sample space, and let $E = \{x_1, x_2, \ldots, x_k\}$ be the set of all x such that $|X(x) - E(X)| > r$. By using the formula that defines $V(X)$, show that
    $$V(X) > \sum_{i=1}^{k} P(x_i)r^2 = P(E)r^2$$

    (b) Show that the probability that $|X(x) - E(X)| \geq r$ is no more than $V(X)/r^2$. This is called *Chebyshev's law.*

17. Use Problem 15 of this section to show that in n independent trials with probability p of success,
$$P\left(\left|\frac{\#\text{ of successes} - np}{n}\right| \geq r\right) \leq \frac{1}{4nr^2}$$

18. This problem derives an intuitive law of probability known as the *law of large numbers* from Chebyshev's law. Informally, the law of large numbers says if you repeat an experiment many times, the fraction of the time that an event occurs is very likely to be close to the probability of the event. In particular, we shall prove that for any positive number s, no matter how small, by making the number n independent trials in a sequence of independent trials large enough, we can make the probability that the number X of successes is between $np - ns$ and $np + ns$ as close to 1 as we choose. For example, we can make the probability that the number of successes is within 1% (or 0.1 per cent) of the expected number as close to 1 as we wish.

    (a) Show that the probability that $|X(x) - np| \geq sn$ is no more than $p(1-p)/s^2n$.

    (b) Explain why this means that we can make the probability that $X(x)$ is between $np - sn$ and $np + sn$ as close to 1 as we want by making n large.

19. On a true-false test, the score is often computed by subtracting the number of wrong answers from the number of right ones and converting that number to a percentage of the number of questions. What is the expected score on a true-false test graded this way of someone who knows 80% of the material in a course? How does this scheme change the standard deviation in comparison with an objective test? What must you do to the number of questions to be able to be a certain percent sure that someone who knows 80% gets a grade within 5 points of the expected percentage score?

20. Another way to bound the deviance from the expectation is known as Markov's inequality. This inequality says that if X is a random variable taking only non-negative values, then, for any $k \geq 1$,
$$P(X > kE(X)) \leq \frac{1}{k}.$$
Prove this inequality.

Chapter 6

Graphs

6.1 Graphs

In this chapter we introduce a fundamental structural idea of discrete mathematics, that of a graph. Many situations in the applications of discrete mathematics may be modeled by the use of a graph, and many algorithms have their most natural description in terms of graphs. It is for this reason that graphs are important to the computer scientist. Graph theory is an ideal subject for developing a deeper understanding of proof by induction because induction, especially strong induction, seems to enter into the majority of proofs in graph theory.

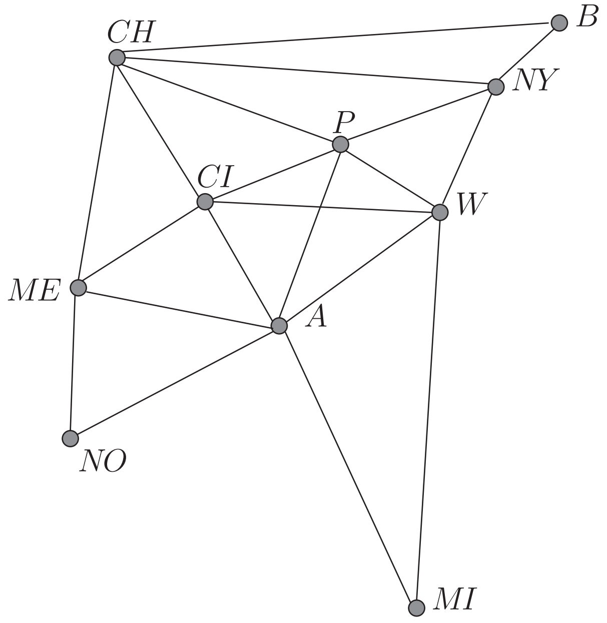
Exercise 6.1-1 In Figure 6.1, you see a stylized map of some cities in the eastern United States (Boston, New York, Pittsburgh, Cincinnati, Chicago, Memphis, New Orleans, Atlanta, Washington DC, and Miami). A company has major offices with data processing centers in each of these cities, and as its operations have grown, it has leased dedicated communication lines between certain pairs of these cities to allow for efficient communication among the computer systems in the various cities. Each grey dot in the figure stands for a data center, and each line in the figure stands for a dedicated communication link. What is the minimum number of links that could be used in sending a message from B (Boston) to NO (New Orleans)? Give a route with this number of links.

Exercise 6.1-2 Which city or cities has or have the most communication links emanating from them?

Exercise 6.1-3 What is the total number of communication links in the figure?

The picture in Figure 6.1 is a drawing of what we call a "graph". A **graph** consists of a set of *vertices* and a set of *edges* with the property that each edge has two (not necessarily different) vertices associated with it and called its *endpoints*. We say the edge *joins* the endpoints, and we say two endpoints are *adjacent* if they are joined by an edge. When vertex is an endpoint of an edge, we say the edge and the vertex are *incident*. Several more examples of graphs are given in Figure 6.2. To *draw* a graph, we draw a point (in our case a grey circle) in the plane for each vertex, and then for each edge we draw a (possibly curved) line between the points that correspond to the endpoints of the edge. The only vertices that may be touched by the line

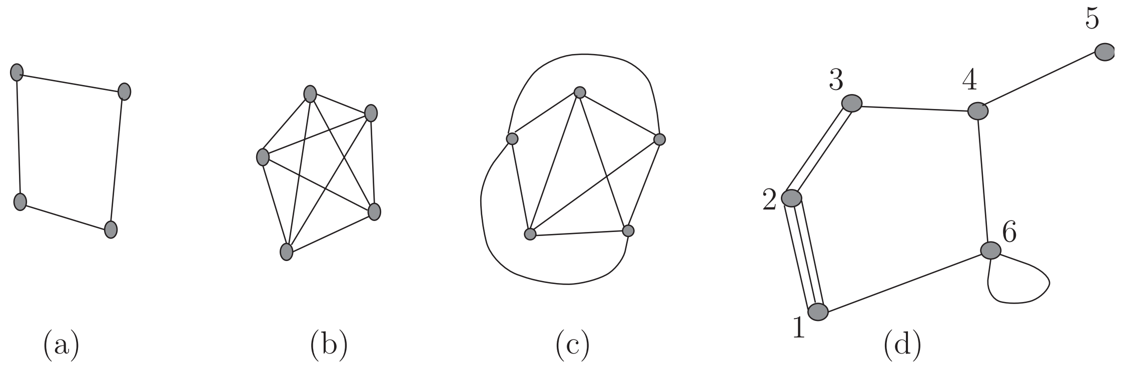
Figure 6.1: A stylized map of some eastern US cities.



representing an edge are the endpoints of the edge. Notice that in graph (d) of Figure 6.2 we have three edges joining the vertices marked 1 and 2 and two edges joining the vertices marked 2 and 3. We also have one edge that joins the vertex marked 6 to itself. This edge has two identical endpoints. The graph in Figure 6.1 and the first three graphs in Figure 6.2 are called simple graphs. A *simple graph* is one that has at most one edge joining each pair of distinct vertices, and no edges joining a vertex to itself.[1] You'll note in Figure 6.2 that we sometimes label the vertices of the graph and we sometimes don't. We label the vertices when we want to give them meaning,

[1]The terminology of graph theory has not yet been standardized, because it is a relatively young subject. The terminology we are using here is the most popular terminology in computer science, but some graph theorists would reserve the word graph for what we have just called a simple graph and would use the word multigraph for what we called a graph.

Figure 6.2: Some examples of graphs



as in Figure 6.1 or when we know we will want to refer to them as in graph (d) of Figure 6.2. We say that graph (d) in Figure 6.2 has a "loop" at vertex 6 and multiple edges joining vertices 1 and 2 and vertices 2 and 3. More precisely, an edge that joins a vertex to itself is called a *loop* and we say we have *multiple edges* between vertices x and y if there is more than one edge joining x and y. If there is an edge from vertex x to vertex y in a simple graph, we denote it by $\{x, y\}$. Thus $\{P, W\}$ denotes the edge between Pittsburgh and Washington in Figure 6.1 Sometimes it will be helpful to have a symbol to stand for a graph. We use the phrase "Let $G = (V, E)$" as a shorthand for "Let G stand for a graph with vertex set V and edge set E."

The drawings in parts (b) and (c) of Figure 6.2 are different drawings of the same graph. The graph consists of five vertices and one edge between each pair of distinct vertices. It is called the complete graph on five vertices and is denoted by K_5. In general, a *complete graph* on n vertices is a graph with n vertices that has an edge between each two of the vertices. We use K_n to stand for a complete graph on n vertices. These two drawings are intended to illustrate that there are many different ways we can draw a given graph. The two drawings illustrate two different ideas. Drawing (b) illustrates the fact that each vertex is adjacent to each other vertex and suggests that there is a high degree of symmetry. Drawing (c) illustrates the fact that it is possible to draw the graph so that only one pair of edges crosses; other than that the only places where edges come together are at their endpoints. In fact, it is impossible to draw K_5 so that no edges cross, a fact that we shall explain later in this chapter.

In Exercise 6.1-1 the links referred to are edges of the graph and the cities are the vertices of the graph. It is possible to get from the vertex for Boston to the vertex for New Orleans by using three communication links, namely the edge from Boston to Chicago, the edge from Chicago to Memphis, and the edge from Memphis to New Orleans. We call an alternating sequence of vertices and edges in a graph a **path** if it starts and ends with a vertex, and each edge joins the vertex before it in the sequence to the vertex after it in the sequence.[2] If a is the first vertex in the path and b is the last vertex in the path, then we say the path is a path from a to b. Thus the path we found from Boston to New Orleans is $B\{B, CH\}CH\{CH, ME\}, ME\{ME, NO\}NO$. Because the graph is simple, we can also use the shorter notation B, CH, ME, NO to describe the same path, because there is exactly one edge between successive vertices in this list. The *length* of a path is the number of edges it has, so our path from Boston to New Orleans has length 3. The length of a shortest path between two vertices in a graph is called the *distance* between them. Thus the distance from Boston to New Orleans in the graph of Figure 6.1 is three. By inspecting the map we see that there is no shorter path from Boston to New Orleans. Notice that no vertex or edge is repeated on our path from Boston to New Orleans. A path is called a **simple path** if it has no repeated vertices or edges.[3]

The degree of a vertex

In Exercise 6.1-2, the city with the most communication links is Atlanta (A). We say the vertex A has "degree" 6 because 6 edges emanate from it. More generally the *degree* of a vertex in a graph is the number of times it is incident with edges of the graph; that is, the degree of a vertex x is the number of edges from x to other vertices plus twice the number of loops at vertex x. In

[2]Again, the terminology we are using here is the most popular terminology in computer science, but what we just defined as a path would be called a walk by most graph theorists.

[3]Most graph theorists reserve the word path for what we are calling a simple path, but again we are using the language most popular in computer science.

graph (d) of Figure 6.2 vertex 2 has degree 5, and vertex 6 has degree 4. In a graph like the one in Figure 6.1, it is somewhat difficult to count the edges just because you can forget which ones you've counted and which ones you haven't.

Exercise 6.1-4 Is there a relationship between the number of edges in a graph and the degrees of the vertices? If so, find it. Hint: computing degrees of vertices and number of edges in some relatively small examples of graphs should help you discover a formula. To find one proof, imagine a wild west movie in which the villain is hiding under the front porch of a cabin. A posse rides up and is talking to the owner of the cabin, and the bad guy can just barely look out from underneath the porch and count the horses hoofs. If he counts the hooves accurately, what can he do to figure out the number of horses, and thus presumably the size of the posse?

In Exercise 6.1-4, examples such as those in Figure 6.2 convince us that the sum of the degrees of the vertices is twice the number of edges. How can we prove this? One way is to count the total number of incidences between vertices and edges (similar to counting the horses hooves in the hint). Each edge has exactly two incidences, so the total number of incidences is twice the number of edges. But the degree of a vertex is the number of incidences it has, so the sum of the degrees of the vertices is also the total number of of incidences. Therefore the sum of the degrees of the vertices of a graph is twice the number of edges. Thus to compute the number of edges of a graph, we can sum the degrees of the vertices and divide by two. (In the case of the hint, the horses correspond to edges and the hooves to endpoints.) There is another proof of this result that uses induction.

Theorem 6.1 *Suppose a graph has a finite number of edges. Then the sum of the degrees of the vertices is twice the number of edges.*

Proof: We induct on the number of edges of the graph. If a graph has no edges, then each vertex has degree zero and the sum of the degrees is zero, which is twice the number of edges. Now suppose $e > 0$ and the theorem is true whenever a graph has fewer than e edges. Let G be a graph with e edges and let ϵ be an edge of G.[4] Let G' be the graph (on the same vertex set as G) we get by deleting ϵ from the edge set E of G. Then G has $e-1$ edges, and so by our inductive hypothesis, the sum of the degrees of the vertices of G' is twice $e-1$. Now there are two possible cases. Either e was a loop, in which case one vertex of G' has degree two less in G' than it has in G. Otherwise e has two distinct endpoints, in which case exactly two vertices of G' have degree one less than their degree in G. Thus in both cases the sum of the degrees of the vertices in G' is two less than the sum of the degrees of the vertices in G, so the sum of the degrees of the vertices in G is $(2e-2)+2 = 2e$. Thus the truth of the theorem for graphs with $e-1$ edges implies the truth of the theorem for graphs with e edges. Therefore, by the principle of mathematical induction, the theorem is true for a graph with any finite number of edges. ■

There are a couple instructive points in the proof of the theorem. First, since it wasn't clear from the outset whether we would need to use strong or weak induction, we made the inductive hypothesis we would normally make for strong induction. However in the course of the proof, we

[4]Since it is very handy to have e stand for the number of edges of a graph, we will use Greek letters such as epsilon (ϵ) to stand for edges of a graph. It is also handy to use v to stand for the number of vertices of a graph, so we use other letters near the end of the alphabet, such as w, x, y,and z to stand for vertices.

saw that we only needed to use weak induction, so that is how we wrote our conclusion. This is not a mistake, because we used our inductive hypothesis correctly. We just didn't need to use it for every possible value it covered.

Second, instead of saying that we would take a graph with $e-1$ edges and add an edge to get a graph with e edges, we said that we would take a graph with e edges and remove an edge to get a graph with $e-1$ edges. This is because we need to prove that the result holds for *every* graph with e edges. By using the second approach we avoided the need to say that "every graph with e edges may be built up from a graph with $e-1$ edges by adding an edge," because in the second approach we started with an arbitrary graph on e edges. In the first approach, we would have proved that the theorem was true for all graphs that could be built from an $e-1$ edge graph by adding an edge, and we would have had to explicitly say that every graph with e edges could be built in this way.

In Exercise 3 the sum of the degrees of the vertices is (working from left to right)

$$2+4+5+6+5+2+5+4+2=40,$$

and so the graph has 20 edges.

Connectivity

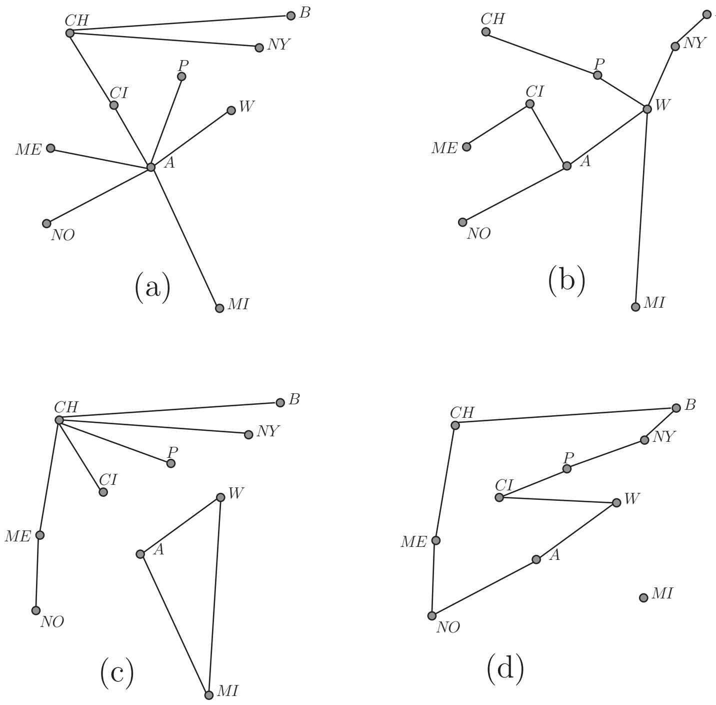
All of the examples we have seen so far have a property that is not common to all graphs, namely that there is a path from every vertex to every other vertex.

Exercise 6.1-5 The company with the computer network in Figure 6.1 needs to reduce its expenses. It is currently leasing each of the communication lines shown in the Figure. Since it can send information from one city to another through one or more intermediate cities, it decides to only lease the minimum number of communication lines it needs to be able to send a message from any city to any other city by using any number of intermediate cities. What is the minimum number of lines it needs to lease? Give two examples of subsets of the edge set with this number of edges that will allow communication between any two cities and two examples of a subset of the edge set with this number of edges that will not allow communication between any two cities.

Some experimentation with the graph convinces that if we keep eight or fewer edges, there is no way we can communicate among the cities (we will explain this more precisely later on), but that there are quite a few sets of nine edges that suffice for communication among all the cities. In Figure 6.3 we show two sets of nine edges each that allow us to communicate among all the cities and two sets of nine edges that do not allow us to communicate among all the cities.

Notice that in graphs (a) and (b) it is possible to get from any vertex to any other vertex by a path. A graph is called *connected* there is a path between each two vertices of the graph. Notice that in graph (c) it is not possible to find a path from Atlanta to Boston, for example, and in graph (d) it is not possible to find a path from Miami to any of the other vertices. Thus these graphs are not connected; we call them disconnected. In graph (d) we say that Miami is an *isolated vertex.*

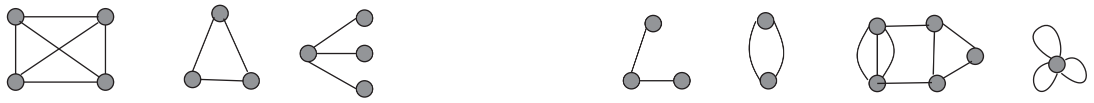
Figure 6.3: Selecting nine edges from the stylized map of some eastern US cities.



We say two vertices are *connected* if there is a path between them, so a graph is connected if each two of its vertices are connected. Thus in Graph (c) the vertices for Boston and New Orleans are connected. The relationship of being connected is an equivalence relation (in the sense of Section 1.4). To show this we would have to show that this relationship divides the set of vertices up into mutually exclusive classes; that is, that it partitions the vertices of the graph. The class containing Boston, for example is all vertices connected to Boston. If two vertices are in that set, they both have paths to Boston, so there is a path between them using Boston as an intermediate vertex. If a vertex x is in the set containing Boston and another vertex y is not, then they cannot be connected or else the path from y to x and then on to Boston would connect y to Boston, which would mean y was in the class containing Boston after all. Thus the relation of being connected partitions the vertex set of the graph into disjoint classes, so it is an equivalence relation. Though we made this argument with respect to the vertex Boston in the specific case of graph (c) of Figure 6.3, it is a perfectly general argument that applies to arbitrary vertices in arbitrary graphs. We call the equivalence relation of being connected to the *connectivity* relation. There can be no edge of a graph between two vertices in different equivalence classes of the connectivity relation because then everything in one class would be connected to everything in the other class, so the two classes would have to be the same. Thus we also end up with a partition of the edges into disjoint sets. If a graph has edge set E, and C is an equivalence class of the connectivity relation, then we use $E(C)$ to denote the set of edges whose endpoints are both in C. Since no edge connects vertices in different equivalence classes, each edge must be in some set $E(C)$. The graph consisting of an equivalence class C of the connectivity relation together with the edges $E(C)$ is called a *connected component* of our original graph. From now on our emphasis will be on connected components rather than on equivalence classes of the connectivity relation. Notice that graphs (c) and (d) of Figure 6.3

each have two connected components. In graph (c) the vertex sets of the connected components are $\{NO, ME, CH, CI, P, NY, B\}$ and $\{A, W, MI\}$. In graph (d) the connected components are $\{NO, ME, CH, B, NY, P, CI, W, A\}$ and $\{MI\}$. Two other examples of graphs with multiple connected components are shown in Figure 6.4.

Figure 6.4: A simple graph with three connected components and a graph with four connected components.



Cycles

In graphs (c) and (d) of Figure 6.3 we see a feature that we don't see in graphs (a) and (b), namely a path that leads from a vertex back to itself. A path that starts and ends at the same vertex is called a *closed path*. A closed path with at least one edge is called a *cycle* if, except for the last vertex, all of its vertices are different. The closed paths we see in graphs (c) and (d) of Figure 6.3 are cycles. Not only do we say that $\{NO, ME, CH, B, NY, P, CI, W, A, NO\}$ is a cycle in in graph (d) of Figure 6.3, but we also say it is a cycle in the graph of Figure 6.1. The way we distinguish between these situations is to say the cycle $\{NO, ME, CH, B, NY, P, CI, W, A, NO\}$ is an induced cycle in Figure 6.3 but not in Figure 6.1. More generally, a graph H is called a *subgraph* of the graph G if all the vertices and edges of H are vertices and edges of G, and we call H an *induced subgraph* of G if every vertex of H is a vertex of G, and every edge of G connecting vertices of H is an edge of H. Thus the first graph of Figure 6.4 has an induced K_4 and an induced cycle on three vertices.

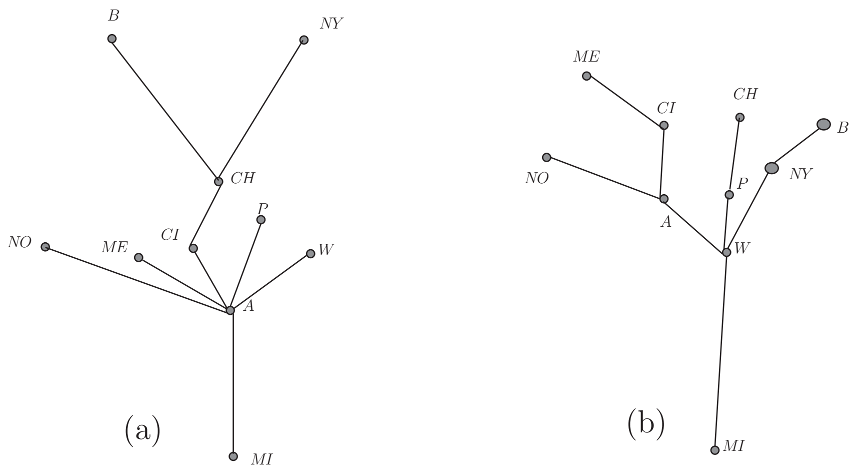
We don't normally distinguish which point on a cycle really is the starting point; for example we consider the cycle $\{A, W, MI, A\}$ to be the same as the cycle $\{W, MI, A, W\}$. Notice that there are cycles with one edge and cycles with two edges in the second graph of Figure 6.4. We call a graph G a *cycle* on n vertices or an n-cycle and denote it by C_n if it has a cycle that contains all the vertices and edges of G and a *path* on n vertices and denote it by P_n if it has a path that contains all the vertices and edges of G. Thus drawing (a) of Figure 6.2 is a drawing of C_4. The second graph of Figure 6.4 has an induced P_3 and an induced C_2 as subgraphs.

Trees

The graphs in parts (a) and (b) of Figure 6.3 are called trees. We have redrawn them slightly in Figure 6.5 so that you can see why they are called trees. We've said these two graphs are called trees, but we haven't given a definition of trees. In the examples in Figure 6.3, the graphs we have called trees are connected and have no cycles.

Definition 6.1 *A connected graph with no cycles is called a tree.*

Figure 6.5: A visual explanation of the name tree.



Other Properties of Trees

In coming to our definition of a tree, we left out a lot of other properties of trees we could have discovered by a further analysis of Figure 6.3

Exercise 6.1-6 Given two vertices in a tree, how many distinct simple paths can we find between the two vertices?

Exercise 6.1-7 Is it possible to delete an edge from a tree and have it remain connected?

Exercise 6.1-8 If $G = (V, E)$ is a graph and we add an edge that joins vertices of V, what can happen to the number of connected components?

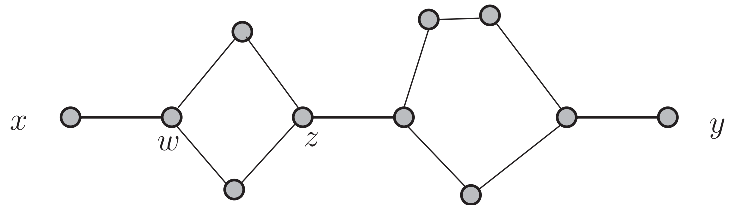
Exercise 6.1-9 How many edges does a tree with v vertices have?

Exercise 6.1-10 Does every tree have a vertex of degree 1? If the answer is yes, explain why. If the answer is no, try to find additional conditions that will guarantee that a tree satisfying these conditions has a vertex of degree 1.

For Exercise 6.1-6, suppose we had two distinct paths from a vertex x to a vertex y. They begin with the same vertex x and might have some more edges in common as in Figure 6.6. Let w be the last vertex after (or including) x the paths share before they become different. The paths must come together again at y, but they might come together earlier. Let z be the first vertex the paths have in common after w. Then there are two paths from w to z that have only w and z in common. Taking one of these paths from w to z and the other from z to w gives us a cycle, and so the graph is not a tree. We have shown that if a graph has two distinct paths from x to y, then it is not a tree. By contrapositive inference, then, if a graph is a tree, it does not have two distinct paths between two vertices x and y. We state this result as a theorem.

Theorem 6.2 *There is exactly one path between each two vertices in a tree.*

Figure 6.6: A graph with multiple paths from x to y.



Proof: By the definition of a tree, there is at least one path between each two vertices. By our argument above, there is at most one path between each two vertices. Thus there is exactly one path. ■

For Exercise 6.1-7, note that if ϵ is an edge from x to y, then x, ϵ, y is the unique path from x to y in the tree. Suppose we delete ϵ from the edge set of the tree. If there were still a path from x to y in the resulting graph, it would also be a path from x to y in the tree, which would contradict Theorem 6.2. Thus the only possibility is that there is no path between x and y in the resulting graph, so it is not connected and is therefore not a tree.

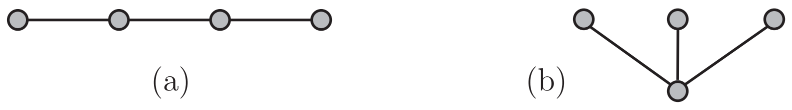
For Exercise 6.1-8, if the endpoints are in the same connected component, then the number of connected components won't change. If the endpoints of the edge are in different connected components, then the number of connected components can go down by one. Since an edge has two endpoints, it is impossible for the number of connected components to go down by more than one when we add an edge. This paragraph and the previous one lead us to the following useful lemma.

Lemma 6.3 *Removing one edge from the edge set of a tree gives a graph with two connected components, each of which is a tree.*

Proof: Suppose as before the lemma that ϵ is an edge from x to y. We have seen that the graph G we get by deleting ϵ from the edge set of the tree is not connected, so it has at least two connected components. But adding the edge back in can only reduce the number of connected compponents by one. Therefore G has exactly two connected components. Since neither has any cycles, both are trees. ■

In Exercise 6.1-9, our trees with ten vertices had nine edges. If we draw a tree on two vertices it will have one edge; if we draw a tree on three vertices it will have two edges. There are two different looking trees on four vertices as shown in Figure 6.7, and each has three edges. On the

Figure 6.7: Two trees on four vertices.



basis of these examples we conjecture that a tree on n vertices has $n - 1$ edges. One approach to proving this is to try to use induction. To do so, we have to see how to build up every tree from smaller trees or how to take a tree and break it into smaller ones. Then in either case we have to figure out how use the truth of our conjecture for the smaller trees to imply its truth for the larger trees. A mistake that people often make at this stage is to assume that every tree can be built from smaller ones by adding a vertex of degree 1. While that is true for finite trees with more than one vertex (which is the point of Exercise 6.1-10), we haven't proved it yet, so we can't yet use it in proofs of other theorems. Another approach to using induction is to ask whether there is a natural way to break a tree into two smaller trees. There is: we just showed in Lemma 6.3 that if you remove an edge ϵ from the edge set of a tree, you get two connected components that are trees. We may assume inductively that the number of edges of each of these trees is one less than its number of vertices. Thus if the graph with these two connected components has v vertices, then it has $v-2$ edges. Adding ϵ back in gives us a graph with $v-1$ edges, so except for the fact that we have not done a base case, we have proved the following theorem.

Theorem 6.4 *For all integers $v \geq 1$, a tree with v vertices has $v-1$ edges.*

Proof: If a tree has one vertex, it can have no edges, for any edge would have to connect that vertex to itself and would thus give a cycle. A tree with two or more vertices must have an edge in order to be connected. We have shown before the statement of the theorem how to use the deletion of an edge to complete an inductive proof that a tree with v vertices has $v-1$ edges, and so for all $v \geq 1$, a tree with v vertices has $v-1$ edges. ■

Finally, for Exercise 6.1-10 we can now give a contrapositive argument to show that a finite tree with more than one vertex has a vertex of degree one. Suppose instead that G is a graph that is connected and all vertices of G have degree two or more. Then the sum of the degrees of the vertices is at least 2v, and so by Theorem 6.1 the number of edges is at least v. Therefore by Theorem 6.4 G is not a tree. Then by contrapositive inference, if T is a tree, then T must have at least one vertex of degree one. This corollary to Theorem 6.4 is so useful that we state it as a corollary.

Corollary 6.5 *A finite tree with more than one vertex has at least one vertex of degree one.*

Important Concepts, Formulas, and Theorems

1. *Graph.* A *graph* consists of a set of *vertices* and a set of *edges* with the property that each edge has two (not necessarily different) vertices associated with it and called its *endpoints.*

2. *Edge; Adjacent.* We say an edge in a graph *joins* its endpoints, and we say two endpoints are *adjacent* if they are joined by an edge.

3. *Incident.* When a vertex is an endpoint of an edge, we say the edge and the vertex are *incident.*

4. *Drawing of a Graph.* To *draw* a graph, we draw a point in the plane for each vertex, and then for each edge we draw a (possibly curved) line between the points that correspond to the endpoints of the edge. Lines that correspond to edges may only touch the vertices that are their endpoints.

5. *Simple Graph.* A *simple graph* is one that has at most one edge joining each pair of distinct vertices, and no edges joining a vertex to itself.

6. *Length, Distance.* The *length* of a path is the number of edges. The *distance* between two vertices in a graph is the length of a shortest path between them.

7. *Loop; Multiple Edges.* An edge that joins a vertex to itself is called a loop and we say we have multiple edges between vertices x and y if there is more than one edge joining x and y.

8. *Notation for a Graph.* We use the phrase "Let $G = (V, E)$" as a shorthand for "Let G stand for a graph with vertex set V and edge set E."

9. *Notation for Edges.* In a simple graph we use the notation $\{x, y\}$ for an edge from x to y. In any graph, when we want to use a letter to denote an edge we use a Greek letter like ϵ so that we can save e to stand for the number of edges of the graph.

10. *Complete Graph on n vertices.* A *complete graph* on n vertices is a graph with n vertices that has an edge between each two of the vertices. We use K_n to stand for a complete graph on n vertices.

11. *Path.* We call an alternating sequence of vertices and edges in a graph a *path* if it starts and ends with a vertex, and each edge joins the vertex before it in the sequence to the vertex after it in the sequence.

12. *Simple Path.* A path is called a *simple path* if it has no repeated vertices or edges.

13. *Degree of a Vertex.* The *degree* of a vertex in a graph is the number of times it is incident with edges of the graph; that is, the degree of a vertex x is the number of edges from x to other vertices plus twice the number of loops at vertex x.

14. *Sum of Degrees of Vertices.* The sum of the degrees of the vertices in a graph with a finite number of edges is twice the number of edges.

15. *Connected.* A graph is called *connected* there is a path between each two vertices of the graph. We say two vertices are *connected* if there is a path between them, so a graph is connected if each two of its vertices are connected. The relationship of being connected is an equivalence relation on the vertices of a graph.

16. *Connected Component.* If C is a subset of the vertex set of a graph, we use $E(C)$ to stand for the set of all edges *both* of whose endpoints are in C. The graph consisting of an equivalence class C of the connectivity relation together with the edges $E(C)$ is called a *connected component* of our original graph.

17. *Closed Path.* A path that starts and ends at the same vertex is called a *closed path.*

18. *Cycle.* A closed path with at least one edge is called a *cycle* if, except for the last vertex, all of its vertices are different.

19. *Tree.* A connected graph with no cycles is called a tree.

20. *Important Properties of Trees.*

    (a) There is a unique path between each two vertices in a tree.

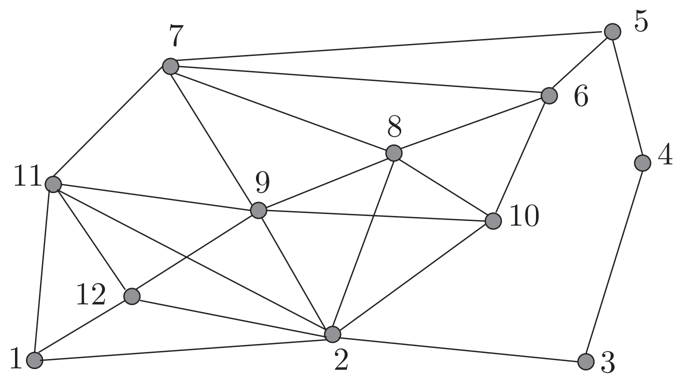
    (b) A tree on v vertices has $v - 1$ edges.

    (c) Every finite tree with at least two vertices has a vertex of degree one.

Problems

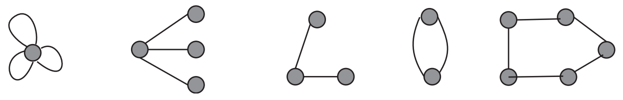
1. Find the shortest path you can from vertex 1 to vertex 5 in Figure 6.8.

Figure 6.8: A graph.



2. Find the longest simple path you can from vertex 1 to vertex 5 in Figure 6.8.
3. Find the vertex of largest degree in Figure 6.8. What is it's degree?

Figure 6.9: A graph with a number of connected components.



4. How many connected components does the graph in Figure 6.9 have?
5. Find all induced cycles in the graph of Figure 6.9.
6. What is the size of the largest induced K_n in Figure 6.9?
7. Find the largest induced K_n (in words, the largest complete subgraph) you can in Figure 6.8.
8. Find the size of the largest induced P_n in the graph in Figure 6.9.
9. A graph with no cycles is called a *forest*. Show that if a forest has v vertices, e edges, and c connected components, then $v = e + c$.
10. What can you say about a five vertex simple graph in which every vertex has degree four?
11. Find a drawing of K_6 in which only three pairs of edges cross.
12. Either prove true or find a counter-example. A graph is a tree if there is one and only one simple path between each pair of vertices.
13. Is there some number m such that if a graph with v vertices is connected and has m edges, then it is a tree? If so, what is m in terms of v?

14. Is there some number m such that a graph on n vertices is a tree if and only if it has m edges and has no cycles.

15. Suppose that a graph G is connected, but for each edge, deleting that edge leaves a disconnected graph. What can you say about G? Prove it.

16. Show that each tree on four vertices can be drawn with one of the two drawings in Figure 6.7.

17. Draw the minimum number of drawings of trees you can so that each tree on five vertices has one of those drawings. Explain why you have drawn all possible trees.

18. Draw the minimum number of drawings of trees you can so that each tree on six vertices has one of those drawings. Explaining why you have drawn all possible drawings is optional.

19. Find the longest induced cycle you can in Figure 6.8.

6.2 Spanning Trees and Rooted Trees

Spanning Trees

We introduced trees with the example of choosing a minimum-sized set of edges that would connect all the vertices in the graph of Figure 6.1. That led us to discuss trees. In fact the kinds of trees that solve our original problem have a special name. A tree whose edge set is a subset of the edge set of the graph G is called a *spanning tree* of G if the tree has exactly the same vertex set as G. Thus the graphs (a) and (b) of Figure 6.3 are spanning trees of the graph of Figure 6.1.

Exercise 6.2-1 Does every connected graph have a spanning tree? Either give a proof or a counter-example.

Exercise 6.2-2 Give an algorithm that determines whether a graph has a spanning tree, finds such a tree if it exists, and takes time bounded above by a polynomial in v and e, where v is the number of vertices, and e is the number of edges.

For Exercise 6.2-1, if the graph has no cycles but is connected, it is a tree, and thus is its own spanning tree. This makes a good base step for a proof by induction on the number of cycles of the graph that every connected graph has a spanning tree. Let $c > 0$ and suppose inductively that when a connected graph has fewer than c cycles, then the graph has a spanning tree. Suppose that G is a graph with c cycles. Choose a cycle of G and choose an edge of that cycle. Deleting that edge (but not its endpoints) reduces the number of cycles by at least one, and so our inductive hypothesis implies that the resulting graph has a spanning tree. But then that spanning tree is also a spanning tree of G. Therefore by the principle of mathematical induction, every finite connected graph has a spanning tree. We have proved the following theorem.

Theorem 6.6 *Each finite connected graph has a spanning tree.*

Proof: The proof is given before the statement of the theorem.■

In Exercise 6.2-2, we want an algorithm for determining whether a graph has a spanning tree. One natural approach would be to convert the inductive proof of Theorem 6.6 into a recursive algorithm. Doing it in the obvious way, however, would mean that we would have to search for cycles in our graph. A natural way to look for a cycle is to look at each subset of the vertex set and see if that subset is a cycle of the graph. Since there are 2^v subsets of the vertex set, we could not guarantee that an algorithm that works in this way would find a spanning tree in time which is big Oh of a polynomial in v and e. In an algorithms course you will learn a much faster (and much more sophisticated) way to implement this approach. We will use another approach, describing a quite general algorithm which we can then specialize in several different ways for different purposes.

The idea of the algorithm is to build up, one vertex at a time, a tree that is a subgraph (not necessarily an induced subgraph) of the graph $G = (V, E)$. (A subgraph of G that is a tree is called a *subtree* of G.) We start with some vertex, say x_0. If there are no edges leaving the vertex and the graph has more than one vertex, we know the graph is not connected and we therefore don't have a spanning tree. Otherwise, we can choose an edge ϵ_1 that connects x_0 to another vertex x_1. Thus $\{x_0, x_1\}$ is the vertex set of a subtree of G. Now if there are no edges that connect some vertex in the set $\{x_0, x_1\}$ to a vertex not in that set, then $\{x_0, x_1\}$ is a connected component of G. In this case, either G is not connected and has no spanning tree, or it just has two vertices and we have a spanning tree. However if there is an edge that connects some vertex in the set $\{x_0, x_1\}$ to a vertex not in that set, we can use this edge to continue building a tree. This suggests an inductive approach to building up the vertex set S of a subtree of our graph one vertex at a time. For the base case of the algorithm, we let $S = \{x_0\}$. For the inductive step, given S, we choose an edge ϵ that leads from a vertex in S to a vertex in $V - S$ and add it to the edge set E' of the subtree if such an edge exists. If no such edge exists, we stop. If $V = S$ when we stop then E' is the edge set of a spanning tree. (We can prove inductively that E' is the edge set of a tree on S, because adding a vertex of degree one to a tree gives a tree.) If $V \neq S$ when we stop, G is not connected and does not have a spanning tree.

To describe the algorithm a bit more precisely, we give pseudocode.

```
Spantree($V$,$E$)
// Assume that $V$ is the vertex set of the graph.
// Assume that $E$ is an array with $|V|$ entries, and entry $i$ of $E$ is the set of
// edges incident with the vertex in position $i$ of $V$.
(1)  $i = 0$;
(2)  Choose a vertex $x_0$ in $V$.
(3)  $S = \{x_0\}$
(4)  While there is an edge from a vertex in $S$ to a vertex not in $S$
(5)       i=i+1
(6)       Choose an edge $\epsilon_i$ from a vertex y in $S$ to a vertex $x_i$ not in $S$
(7)       $S = S \cup \{x_i\}$
(8)       $E' = E' \cup \epsilon_i$
(9)  If $i = |V| - 1$
(10)      return E'
(11) Else
(12)      Print "The graph is not connected."
```

The way in which the vertex x_i and the edge ϵ_i are chosen was deliberately left vague because there are several different ways to specify x_i and ϵ_i that accomplish several different purposes. However, with some natural assumptions, we can still give a big Oh bound on how long the algorithm takes. Presumably we will need to consider at most all v vertices of the graph in order to choose x_i, and so assuming we decide whether or not to use a vertex in constant time, this step of the algorithm will take $O(v)$ time. Presumably we will need to consider at most all e edges of our graph in order to choose ϵ_i, and so assuming we decide whether or not to use an edge in constant time, this step of the algorithm takes at most $O(e)$ time. Given the generality of the condition of the while loop that begins in line 4, determining whether that condition is true might also take $O(e)$ time. Since we repeat the While loop at most v times, all executions of the While loop should take at most $O(ve)$ time. Since line 9 requires us to compute $|V|$, it takes $O(v)$ steps, and all the other lines take constant time. Thus, with the assumptions we have made, the algorithm takes $O(ve + v + e) = O(ve)$ time.

Breadth First Search

Notice that algorithm Spantree will continue as long as a vertex in S is connected to a vertex not in S. Thus when it stops, S will be the vertex set of a connected component of the graph and E' will be the edge set of a spanning tree of this connected component. This suggests that one use that we might make of algorithm Spantree is to find connected components of graphs. If we want the connected component containing a specific vertex x, then we make this choice of x_0 in Line 2. Suppose this is our goal for the algorithm, and suppose that we also want to make the algorithm run as quickly as possible. We could guarantee a faster running time if we could arrange our choice of ϵ_i so that we examined each edge no more than some constant number of times between the start and the end of the algorithm. One way to achieve this is to first use all edges incident with x_0 as ϵ_is, then consider all edges incident with x_1, using them as ϵ_i if we can, and so on.

We can describe this process inductively. We begin by putting vertex x_0 in S and (except for loops or multiple edges) all edges incident with x_0 in E'. Then given vertices 0 through i, all of whose edges we have examined and either accepted or (permanently) rejected as an ϵ_j, we examine the edges leaving vertex $i + 1$. For each of these edges that is incident with a vertex not already in S, we add the edge and that vertex to the tree. Otherwise we reject that edge. Eventually we reach a point where we have examined all the edges leaving all the vertices in S, and we stop.

To give a pseudocode description of the algorithm, we assume that we are given an array V that contains the names of the vertices. There are a number of ways to keep track of the edge set of a graph in a computer. One way is to give a list, called an *adjacency list*, for each vertex listing all vertices adjacent to it. In the case of multiple edges, we list each adjacency as many times as there are edges that give the adjacency. In our pseudocode we implement this idea with the array E that gives in position i a list of all locations in the array V of vertices adjacent in G to vertex $V[i]$.

In our pseudocode we also use an array "Edge" to list the edges of the set we called E' in algorithm Spantree, an array "Vertex" to list the positions in V of the vertices in the set S in the algorithm Spantree, an array "Vertexname" to keep track of the names of the vertices we add to the set S, and an array "Intree" to keep track of whether the vertex in position i of V is in S. Because we want our pseudocode to be easily translatable into a computer language, we avoid subscripts, and use x to stand for the place in the array V that holds the name of the vertex where we are to start the search, i.e. the vertex x_0.

```
BFSpantree($x$,$V$,$E$)
// Assume that $V$ is an array with $v$ entries, the names of the vertices,
// and that $x$ is the location in $V$ of the name of the vertex with which we want
// to start the tree.
// Assume that $E$ is an array with $v$ entries, each a list of the positions
// in $V$ of the names of vertices adjacent to the corresponding entry of $V$.
(1)  $i = 0$ ; $k = 0$ ; Intree$[x] = 1$; Vertex$[0] = x$; Vertexname$[0] = V[x]$
(2)  While $i <= k$
(3)       i=i+1
(4)       For each $j$ in the list E[Vertex[i]]
(5)            If Intree$[j] \neq 1$
(6)                 $k = k + 1$
```

```
(7)                  Edge[$k$] = {$V$[Vertex[$i$]], $V[j]$}
(8)                  Intree[$j$] = 1
(9)                  Vertex[$k$] = $j$
(10)                 Vertexname[$k$] = $V[j]$.
(11) Print "Connected component"
(12) return Vertexname[$0 : k$]
(13) print "Spanning tree edges of connected component"
(14) return Edge[$1 : k$]
```

Notice that the pseudocode allows us to deal with loops and multiple edges through the test whether vertex j is in the tree in Line 5. However the primary purpose of this line is to make sure that we do not examine edges that point from vertex i back to a vertex that is already in the tree.

This algorithm requires that we execute the "For" loop that starts in Line 4 once for each edge incident with vertex i. The "While" loop that starts in Line 2 is executed at most once for each vertex. Thus we execute the "For" loop at most twice for each edge, and carry out the other steps of the "While" loop at most once for each vertex, so that the time to carry out this algorithm is $O(V + E)$.

The algorithm carries out what is known as a "breadth first search"[5] of the graph centered at $V[x]$. The reason for the phrase "breadth first" is because each time we start to work on a new vertex, we examine all its edges (thus exploring the graph broadly at this point) before going on to another vertex. As a result, we first add all vertices at distance 1 from $V[x]$ to S, then all vertices at distance 2 and so on. When we choose a vertex V[Vertex[k]] to put into the set S in Line 9, we are effectively labelling it as vertex k. We call k the *breadth first number* of the vertex $V[j]$ and denote it as $BFN(V[j])$[6]. The breadth first number of a vertex arises twice in the breadth first search algorithm. The breadth first search number of a vertex is assigned to that vertex when it is added to the tree, and (see Problem 7) is the number of vertices that have been previously added. But it then determines when a vertex of the tree is used to add other vertices to the tree: the vertices are taken in order of their breadth first number for the purpose of examining all incident edges to see which ones allow us to add new vertices, and thus new edges, to the tree.

This leads us to one more description of breadth first search. We create a breadth first search tree centered at x_0 in the following way. We put the vertex x_0 in the tree and give it breadth first number zero. Then we process the vertices in the tree in the order of their breadth first number as follows: We consider each edge leaving the vertex. If it is incident with a vertex z not in the tree, we put the edge into the edge set of the tree, we put z into the vertex set of the tree, and we assign z a breadth first number one more than that of the vertex most recently added to the tree. We continue in this way until all vertices in the tree have been processed.

We can use the idea of breadth first number to make our remark about the distances of vertices from x_0 more precise.

Lemma 6.7 *After a breadth first search of a graph G centered at $V[x]$, if $d(V[x], V[z]) > d(V[x], V[y])$, then $BFN(V[z]) > BFN(V[y])$.*

[5]This terminology is due to Robert Tarjan who introduced the idea in his PhD thesis.

[6]In words, we say that the breadth first number of a vertex is k if it is the kth vertex added to a breadth-first search tree, counting the initial vertex x as the zeroth vertex added to the tree

Proof: We will prove this in a way that mirrors our algorithm. We shall show by induction that for each nonnegative k, all vertices of distance k from x_0 are added to the spanning tree (that is, assigned a breadth first number and put into the set S) after all vertices of distance $k-1$ and before any vertices of distance $k+1$. When $k=1$ this follows because S starts as the set $V[x]$ and all vertices adjacent to $V[x]$ are next added to the tree before any other vertices. Now assume that $n>1$ and all vertices of distance n from $V[x]$ are added to the tree after all vertices of distance $n-1$ from $V[x]$ and before any vertices of distance $n+1$. Suppose some vertex of distance n added to the tree has breadth first number m. Then when i reaches m in Line 3 of our pseudocode we examine edges leaving vertex $V[\text{Vertex}[m]]$ in the "For loop." Since, by the inductive hypothesis, all vertices of distance $n-1$ or less from $V[x]$ are added to the tree before vertex $V[\text{Vertex}[m]]$, when we examine vertices $V[j]$ adjacent to vertex $V[\text{Vertex}[m]]$, we will have $\text{Intree}[j]=1$ for these vertices. Since each vertex of distance n from $V[x]$ is adjacent to some vertex $V[z]$ of distance $n-1$ from $V[x]$, and $\text{BFN}[V[z]]<m$ (by the inductive hypothesis), any vertex of distance n from $V[x]$ and adjacent to vertex $V[\text{Vertex}[m]]$ will have $\text{Intree}[j]=1$. Since any vertex adjacent to vertex $V[\text{Vertex}[m]]$ is of distance at most $n+1$ from $V[x]$, every vertex we add to the tree from vertex $V[\text{Vertex}[m]]$ will have distance $n+1$ from the tree. Thus every vertex added to the tree from a vertex of distance n from $V[x]$ will have distance $n+1$ from $V[x]$. Further, all vertices of distance $n+1$ are adjacent to some vertex of distance n from $V[x]$, so each vertex of distance $n+1$ is added to the tree from a vertex of distance n. Note that no vertices of distance $n+2$ from vertex $V[x]$ are added to the tree from vertices of distance n from vertex $V[x]$. Note also that all vertices of distance $n+1$ are added to the tree from vertices of distance n from vertex $V[x]$. Therefore all vertices with distance $n+1$ from $V[x]$ are added to the tree after all edges of distance n from $V[x]$ and before any edges of distance $n+2$ from $V[x]$. Therefore by the principle of mathematical induction, for every positive integer k, all vertices of distance k from $V[x]$ are added to the tree before any vertices of distance $k+1$ from vertex $V[x]$ and after all vertices of distance $k-1$ from vertex $V[x]$. Therefore since the breadth first number of a vertex is the number of the stage of the algorithm in which it was added to the tree, if $d(V[x],V[z])>d(V[x],V[y])$, then $\text{BFN}(V[z])>\text{BFN}(V[y])$. ■

Although we introduced breadth first search for the purpose of having an algorithm that quickly determines a spanning tree of a graph or a spanning tree of the connected component of a graph containing a given vertex, the algorithm does more for us.

Exercise 6.2-3 How does the distance from $V[x]$ to $V[y]$ in a breadth first search centered at $V[x]$ in a graph G relate to the distance from $V[x]$ to $V[y]$ in G?

In fact the unique path from $V[x]$ to $V[y]$ in a breadth first search spanning tree of a graph G is a shortest path in G, so the distance from $V[x]$ to another vertex in G is the same as their distance in a breadth first search spanning tree centered at $V[x]$. This makes it easy to compute the distance between a vertex $V[x]$ and all other vertices in a graph.

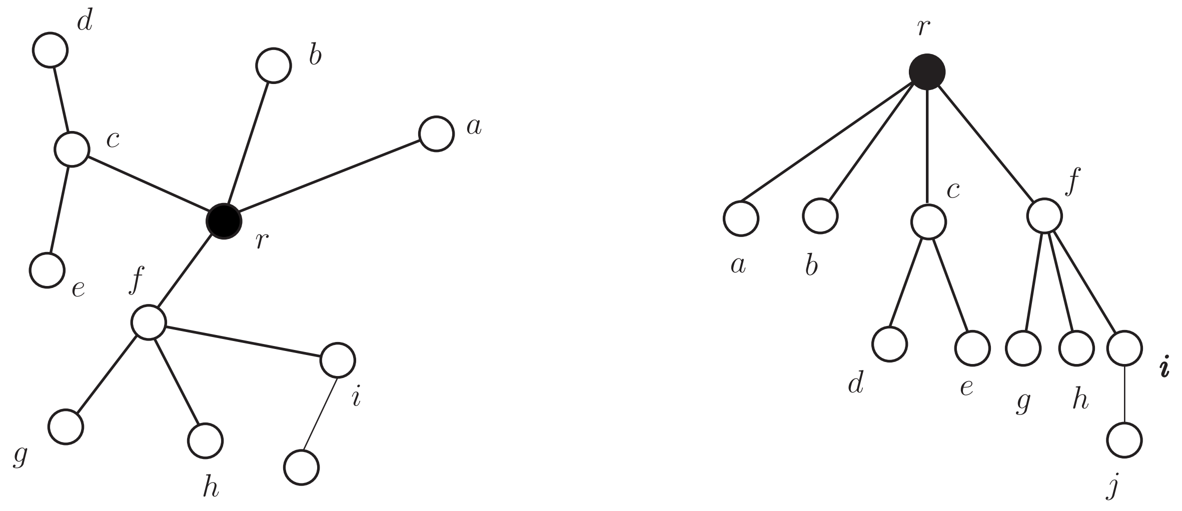
Theorem 6.8 *The unique path from $V[x]$ in a breadth first search spanning tree centered at the vertex $V[x]$ of a graph G to a vertex $V[y]$ is a shortest path from $V[x]$ to $V[y]$ in G.*

Proof: We prove the theorem by induction on the distance from $V[x]$ to $V[y]$. Fix a breadth first search tree of G centered at $V[x]$. If the distance is 0, then the single vertex $V[x]$ is a shortest path from $V[x]$ to $V[x]$ in G and the unique path in the tree. Assume that $k>0$ and that when distance from $V[x]$ to $V[y]$ is less than k, the path from $V[x]$ to $V[y]$ in the tree is a shortest path from $V[x]$ to $V[y]$ in G. Now suppose that the distance from $V[x]$ to $V[y]$ is k. Suppose that a shortest path from $V[x]$ to $V[y]$ has $V[z]$ and $V[y]$ as its last two vertices. Suppose that the unique path from $V[x]$ to $V[y]$ in the tree has $V[z']$ and $V[y]$ as its last two vertices. Then $\text{BFN}(V[z']) < \text{BFN}(V[z])$, because otherwise we would have added $V[y]$ to the tree from vertex $V[z]$. Then by the contrapositive of Lemma 6.7, the distance from $V[x]$ to $V[z']$ is less than or equal to that from $V[x]$ to $V[z]$. But then by the inductive hypothesis, the distance from $V[x]$ to $V[z']$ is the length of the unique path in the tree, and by our previous comment is less than or equal to the distance from $V[x]$ to $V[z]$. However then the length of the unique path from $V[x]$ to $V[y]$ in the tree is no more than the distance from $V[x]$ to $V[y]$, so the two are equal. By the principle of mathematical induction, the distance from $V[x]$ to $V[y]$ is the length of the unique path in the tree for every vertex y of the graph. ■

Rooted Trees

A breadth first search spanning tree of a graph is not simply a tree, but a tree with a selected vertex, namely $V[x]$. It is one example of what we call a rooted tree. A *rooted tree* consists of a tree with a selected vertex, called a *root*, in the tree. Another kind of rooted tree you have likely seen is a binary search tree. It is fascinating how much additional structure is provided to a tree when we select a vertex and call it a root. In Figure 6.10 we show a tree with a chosen vertex and the result of redrawing the tree in a more standard way, with the root at the top and all the edges sloping down, as you might expect to see with a family tree.

Figure 6.10: Two different views of the same rooted tree.



We adopt the language of family trees—ancestor, descendant, parent, and child—to describe rooted trees in general. In Figure 6.10, we say that vertex j is a child of vertex i, and a descendant of vertex r as well as a descendant of vertices f and i. We say vertex f is an ancestor of vertex i. Vertex r is the parent of vertices a, b, c, and f. Each of those four vertices is a child of vertex r. Vertex r is an ancestor of all the other vertices in the tree. In general, in a rooted tree with root r, a vertex x is an *ancestor* of a vertex y, and vertex y is a *descendant* of vertex x if x and y are different and x is on the unique path from the root to y. Vertex x is a *parent* of vertex y and y is a *child* of vertex x in a rooted tree if x is the unique vertex adjacent to y on the unique path from r to y. A vertex can have only one parent, but many ancestors. A vertex with no children is called a *leaf* vertex or an *external vertex*; other vertices are called *internal vertices*.

Exercise 6.2-4 Prove that a vertex in a rooted tree can have at most one parent. Does every vertex in a rooted tree have a parent?

In Exercise 6.2-4, suppose x is not the root. Then, because there is a unique path between a vertex x and the root of a rooted tree and there is a unique vertex on that path adjacent to x, each vertex other than the root has a unique parent. The root, however, has no parent.

Exercise 6.2-5 A binary tree is a special kind of rooted tree that has some additional structure that makes it tremendously useful as a data structure. In order to describe the idea of a binary tree it is useful to think of a tree with no vertices, which we call the null tree or empty tree. Then we can recursively describe a *binary tree* as

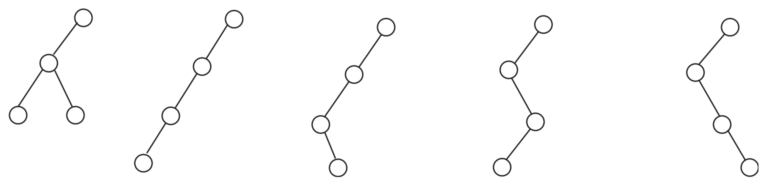
- an empty tree, or
- a structure consisting of a root vertex, a binary tree called the left subtree of the root and a disjoint binary tree called the right subtree of the root, with an edge connecting the root of the left subtree to the root vertex and an edge connecting the root of the right subtree to the root vertex.

Then a single vertex is a binary tree with an empty right subtree and an empty left subtree. A rooted tree with two vertices can occur in two ways as a binary tree, either with a root and a left subtree consisting of one vertex or as a root and a right subtree consisting of one vertex. Draw all binary trees on four vertices in which the root node has an empty right child. Draw all binary trees on four vertices in which the root has a nonempty left child and a nonempty right child.

Exercise 6.2-6 A binary tree is a *full* binary tree if each vertex has either two nonempty children or two empty children (a vertex with two empty children is called a *leaf*.) Are there any full binary trees on an even number of vertices? Prove that what you say is correct.

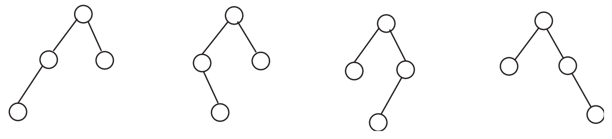
For Exercise 6.2-5 we have five binary trees shown in Figure 6.11 for the first question. Then

Figure 6.11: The four-vertex binary trees whose root has an empty right child.



in Figure 6.12 we have four more trees as the answer to the second question.

Figure 6.12: The four-vertex binary trees whose root has both a left and a right child.



For Exercise 6.2-6, it is possible to have a full binary tree with zero vertices, so there is one such binary tree. But, if a full binary tree is not empty, it must have odd number of vertices. We can prove this inductively. A full binary tree with 1 vertex has an odd number of vertices. Now suppose inductively that $n > 1$ and any full binary tree with fewer than n vertices has an odd number of vertices. For a full binary tree with $n > 1$ vertices, the root must have two nonempty children. Thus removing the root gives us two binary trees, rooted at the children of the original root, each with fewer than n vertices. By the definition of full, each of the subtrees rooted in the two children must be full binary tree. The number of vertices of the original tree is one more than the total number of vertices of these two trees. This is a sum of three odd numbers, so it must be odd. Thus, by the principle of mathematical induction, if a full binary tree is not empty, it must have odd number of vertices.

The definition we gave of a binary tree was a inductive one, because the inductive definition makes it easy for us to prove things about binary trees. We remove the root, apply the inductive hypothesis to the binary tree or trees that result, and then use that information to prove our result for the original tree. We could have defined a binary tree as a special kind of rooted tree, such that

- each vertex has at most two children,
- each child is specified to be a left or right child, and
- a vertex has at most one of each kind of child.

While it works, this definition is less convenient than the inductive definition.

There is a similar inductive definition of a rooted tree. Since we have already defined rooted trees, we will call the object we are defining an r-tree. The inductive definition states that an r-tree is either a single vertex, called a root, or a graph consisting of a vertex called a root and a set of disjoint r-trees, each of which has its root attached by an edge to the original root. We can then prove as a theorem that a graph is an r-tree if and only if it is a rooted tree. Sometimes inductive proofs for rooted trees are easier if we use the method of removing the root and applying the inductive hypothesis to the rooted trees that result, as we did for binary trees in our solution of Exercise 6.2-6.

Important Concepts, Formulas, and Theorems

1. *Spanning Tree.* A tree whose edge set is a subset of the edge set of the graph G is called a *spanning tree* of G if the tree has exactly the same vertex set as G.

2. *Breadth First Search.* We create a *breadth first search* tree centered at x_0 in the following way. We put the vertex x_0 in the tree and give it breadth first number zero. Then we

process the vertices in the tree in the order of their breadth first number as follows: We consider each edge leaving the vertex. If it is incident with a vertex z not in the tree, we put the edge into the edge set of the tree, we put z into the vertex set of the tree, and we assign z a breadth first number one more than that of the vertex most recently added to the tree. We continue in this way until all vertices in the tree have been processed.

3. *Breadth first number.* The *breadth first number* of a vertex in a breadth first search tree is the number of vertices that were already in the tree when the vertex was added to the vertex set of the tree.

4. *Breadth first search and distances.* The distance from a vertex y to a vertex x may be computed by doing a breadth first search centered at x and then computing the distance from y to x in the breadth first search tree. In particular, the path from x to y in a breadth first search tree of G centered at x is a shortest path from x to y in G.

5. *Rooted tree.* A *rooted tree* consists of a tree with a selected vertex, called a *root*, in the tree.

6. *Ancestor, Descendant.* In a rooted tree with root r, a vertex x is an *ancestor* of a vertex y, and vertex y is a *descendant* of vertex x if x and y are different and x is on the unique path from the root to y.

7. *Parent, Child.* In a rooted tree with root r, vertex x is a *parent* of vertex y and y is a *child* of vertex x in if x is the unique vertex adjacent to y on the unique path from r to y.

8. *Leaf (External) Vertex.* A vertex with no children in a rooted tree is called a *leaf* vertex or an *external vertex.*

9. *Internal Vertex* A vertex of a rooted tree that is not a leaf vertex is called an *internal* vertex.

10. *Binary Tree* We recursively describe a *binary tree* as
    - an empty tree (a tree with no vertices), or
    - a structure consisting of a root vertex, a binary tree called the left subtree of the root and a binary tree called the right subtree of the root, with an edge connecting the root of the left subtree to the root vertex and an edge connecting the root of the right subtree to the root vertex.

11. *Full Binary Tree* A binary tree is a *full* binary tree if each vertex has either two nonempty children or two empty children.

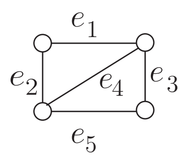
Problems

1. Find all spanning trees (list their edge sets) of the graph in Figure 6.13.

2. Show that a finite graph is connected if and only if it has a spanning tree.

3. Draw all rooted trees on 5 vertices. The order and the place in which you write the vertices down on the page is unimportant. If you would like to label the vertices (as we did in the graph in Figure 6.10), that is fine, but don't give two different ways of labelling or drawing the same tree.

Figure 6.13: A graph.



4. Draw all rooted trees on 6 vertices with four leaf vertices. If you would like to label the vertices (as we did in the graph in Figure 6.10), that is fine, but don't give two different ways of labelling or drawing the same tree.

5. Find a tree with more than one vertex that has the property that all the rooted trees you get by picking different vertices as roots are different as rooted trees. (Two rooted trees are the same (isomorphic), if they each have one vertex or if you can label them so that they have the same (labelled) root and the same (labelled) subtrees.)

6. Create a breadth first search tree centered at vertex 12 for the graph in Figure 6.8 and use it to compute the distance of each vertex from vertex 12. Give the breadth first number for each vertex.

7. It may seem clear to some people that the breadth first number of a vertex is the number of vertices previously added to the tree. However the breadth first number was not actually defined in this way. Give a proof that the breadth first number of a vertex is the number of vertices previously added to the tree.

8. A *(left, right) child* of a vertex in a binary tree is the root of a (left, right) subtree of that vertex. A binary tree is a *full* binary tree if each vertex has either two nonempty children or two empty children (a vertex with two empty children is called a *leaf*.) Draw all full binary trees on seven vertices.

9. The *depth* of a node in a rooted tree is defined to be the number of edges on the (unique) path to the root. A binary tree is *complete* if it is full (see Problem 8) and all its leaves have the same depth. How many vertices does a complete binary tree of depth 1 have? Depth 2? Depth d? (Proof required for depth d.)

10. The *height* of a rooted or binary tree with one vertex is 0; otherwise it is 1 plus the maximum of the heights of its subtrees. Based on Exercise 6.2-9, what is the minimum height of *any* binary tree on n vertices? (Please prove this.)

11. A binary tree is complete if it is full and all its leaves have the same depth (see Exercise 6.2-8 and Exercise 6.2-9). A vertex that is not a leaf vertex is called an *internal* vertex. What is the relationship between the number I of internal vertices and the number L of leaf vertices in a complete binary tree. A full binary tree? (Proof please.)

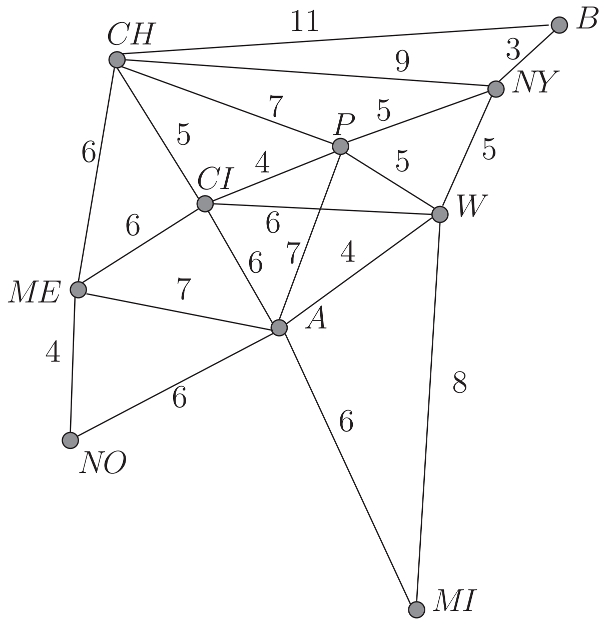
12. The *internal path length* of a binary tree is the sum, taken over all internal (see Exercise 6.2-11) vertices of the tree, of the depth of the vertex. The *external path length* of a binary tree is the sum, taken over all leaf vertices of the tree, of the depth of the vertex. Show that in a full binary tree with n internal vertices, internal path length i and external path length e, we have $e = i + 2n$.

13. Prove that a graph is an r-tree, as defined at the end of the section if and only if it is a rooted tree.

14. Use the inductive definition of a rooted tree (r-tree) given at the end of the section to prove once again that a rooted tree with n vertices has $n-1$ edges if $n \geq 1$.

15. In Figure 6.14 we have added numbers to the edges of the graph of Figure 6.1 to give what is usually called a *weighted graph*—the name for a graph with numbers, often called *weights* associated with its edges. We use $w(\epsilon)$ to stand for the weight of the edge ϵ. These numbers represent the lease fees in thousands of dollars for the communication lines the edges represent. Since the company is choosing a spanning tree from the graph to save money, it is natural that it would want to choose the spanning tree with minimum total cost. To be precise, a *minimum spanning tree* in a weighted graph is a spanning tree of the graph such that the sum of the weights on the edges of the spanning tree is a minimum among all spanning trees of the graph.

Figure 6.14: A stylized map of some eastern US cities.



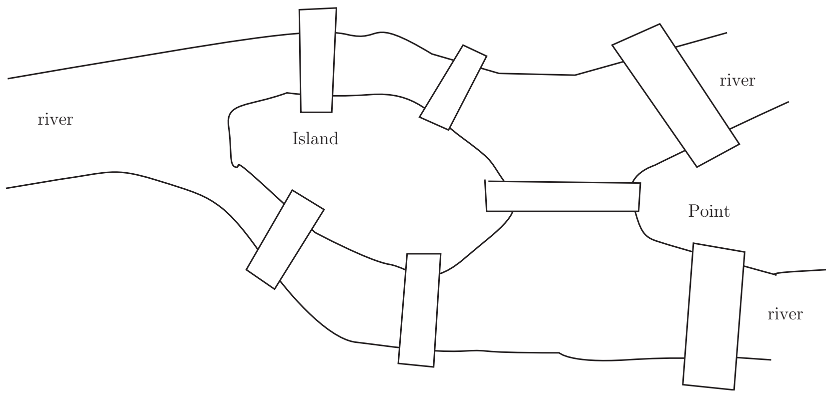
Give an algorithm to select a spanning tree of minimum total weight from a weighted graph and apply it to find a minimum spanning tree of the weighted graph in Figure 6.14. Show that your algorithm works and analyze how much time it takes.

6.3 Eulerian and Hamiltonian Paths and Tours

Eulerian Tours and Trails

Exercise 6.3-1 In an article generally acknowledged to be one of the origins of the graph theory [7] Leonhard Euler (pronounced Oiler) described a geographic problem that he offered as an elementary example of what he called "the geometry of position." The problem, known as the "Königsberg Bridge Problem," concerns the town of Königsberg in Prussia (now Kaliningrad in Russia), which is shown in a schematic map (circa 1700) in Figure 6.15. Euler tells us that the citizens amused themselves

Figure 6.15: A schematic map of Königsberg

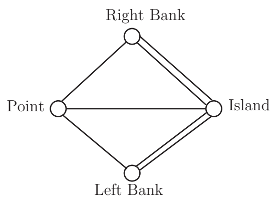


by trying to find a walk through town that crossed each of the seven bridges once and only once (and, hopefully, ended where it started). Is such a walk possible?

In Exercise 6.3-1, such a walk will enter a land mass on a bridge and leave it on a different bridge, so except for the starting and ending point, the walk requires two new bridges each time it enters and leaves a land mass. Thus each of these land masses must be at the end of an even number of bridges. However, as we see from Figure 6.15 each land mass is at the end of an odd number of bridges. Therefore no such walk is possible.

We can represent the map in Exercise 6.3-1 more compactly with the graph in Figure 6.16. In graph theoretic terminology Euler's question asks whether there is a path, starting and ending

Figure 6.16: A graph to replace the schematic map of Königsberg



[7]Reprinted in *Graph Theory 1736-1936* by Biggs, Lloyd and Wilson (Clarendon, 1976)

at the same vertex, that uses each edge exactly once.

Exercise 6.3-2 Determine whether the graph in Figure 6.1 has a closed path that includes each edge of the graph exactly once, and find one if it does.

Exercise 6.3-3 Find the strongest condition you can that has to be satisfied by a graph that has a path, starting and ending at the same place, that includes each vertex at least once and each edge once and only once. Such a path is known as an *Eulerian Tour* or *Eulerian Circuit*.

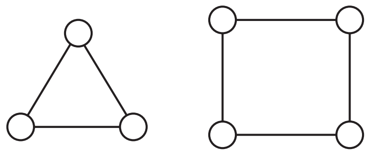
Exercise 6.3-4 Find the strongest condition you can that has to be satisfied by a graph that has a path, starting and ending at different places, that includes each vertex at least once and each edge once and only once. Such a path is known as an *Eulerian Trail*

Exercise 6.3-5 Determine whether the graph in Figure 6.1 has an Eulerain Trail and find one if it does.

The graph in Figure 6.1 cannot have a closed path that includes each edge exactly once because if the initial vertex of the path were P, then the number of edges incident with P would have to be one at the beginning of the path, plus two for each time P appears before the end of the path, plus one more for the time P would appear at the end of the path, so the degree of P would have to be even. But if P were not the initial vertex of a closed path including all the edges, each time we entered P on one edge, we would have to leave it on a second edge, so the number of edges incident with P would have to be even. Thus in Exercise 6.3-2 there is no closed path that includes each edge exactly once.

Notice that, just as we argued for a walk through Königsberg, in any graph with an Eulerian Circuit, each vertex except for the starting-finishing one will be paired with two new edges (those preceding and following it on the path) each time it appears on the path. Therefore each of these vertices is incident with an even number of edges. Further, the starting vertex is incident with one edge at the beginning of the path and is incident with a different edge at the end of the path. Each other time it occurs, it will be paired with two edges. Thus this vertex is incident with an even number of edges as well. Therefore a natural condition a graph must satisfy if it has an Eulerian Tour is that each vertex has even degree. But Exercise 6.3-3 asked us for the strongest condition we could find that a graph with an Eulerian Tour would satisfy. How do we know whether this is as strong a condition as we could devise? In fact it isn't, the graph in Figure 6.17 clearly has no Eulerian Tour because it is disconnected, but every vertex has even degree.

Figure 6.17: This graph has no Eulerian Tour, even though each vertex has even degree.



The point that Figure 6.17 makes is that in order to have an Eulerian Tour, a graph must be connected as well as having only vertices of even degree. Thus perhaps the strongest condition we can find for having an Eulerian Tour is that the graph is connected and every vertex has even degree. Again, the question comes up "How do we show this condition is as strong as possible, if indeed it is?" We showed a condition was not as strong as possible by giving an example of a graph that satisfied the condition but did not have an Eulerian Tour. What if we could show that no such example is possible, i.e. we could prove that a graph which is connected and in which every vertex has even degree does have an Eulerian Tour? Then we would have shown our condition is as strong as possible.

Theorem 6.9 *A graph has an Eulerian Tour if and only if it is connected and each vertex has even degree.*

Proof: A graph must be connected to have an Eulerian tour, because there must be a path that includes each vertex, so each two vertices are joined by a path. Similarly, as explained earlier, each vertex must have even degree in order for a graph to have an Eulerian Tour. Therefore we need only show that if a graph is connected and each vertex has even degree, then it has an Eulerain Tour. We do so with a recursive construction. If G has one vertex and no edges, we have an Eulerian tour consisting of one vertex and no edges. So suppose G is connected, has at least one edge, and each vertex of G has even degree. Now, given distinct vertices x_0, x_1, $\ldots$, x_i and edges ϵ_1, ϵ_2, $\ldots$, ϵ_i such that $x_0\epsilon_1x_1\ldots\epsilon_ix_i$ is a path, choose an edge ϵ_{i+1} from x_i to a vertex x_{j+1}. If x_{j+1} is x_0, stop. Eventually this process must stop because G is finite, and (since each vertex in G has even degree) when we enter a vertex other than x_0, there will be an edge on which we can leave it. This gives us a closed path C. Delete the edges of this closed path from the edge set of G. This gives us a graph G' in which each vertex has even degree, because we have removed two edges incident with each vertex of the closed path (or else we have removed a loop). However, G' need not be connected. Each connected component of G' is a connected graph in which each vertex has even degree. Further, each connected component of G' contains at least one element x_i. (Suppose a connected component C contained no x_i. Since G is connected, for each i, there is a path in G from each vertex in C to x_i. Choose the shortest such path, and suppose it connects a vertex y in C to x_j. Then no edge in the path can be in the closed path, or else we would have a shorter path from y to a different vertex x_i. Therefore removing the edges of the closed path leaves y connected to x_j in C, so that C contains an x_i after all, a contradiction.) We may assume inductively that each connected component has fewer edges than G, so each connected component has an Eulerian Tour. Now we may begin to recursively construct an Eulerian Tour of G by starting at x_j, and taking an Eulerian Tour of the connected component containing x_j. Then given a sequence x_j, x_1, $\ldots$, x_k such that the Eulerian tour we have constructed so far includes the vertices x_j through x_k, the vertices and edges of the connected components of G' containing the vertices x_k through x_k, the edges ϵ_{j+1} through ϵ_k, we add the edge e_{k+1} and the vertex x_{k+1} to our tour, and if the vertices and edges of the connected component of G' containing x_{k+1} are not already in our tour, we add an Eulerian Tour of the connected component of G' containing x_{k+1} to our tour. When we add the last edge and vertex of our closed path to the path we have been constructing, every vertex and edge of the graph will have to be in the path we have constructed, because every vertex is in some connected component of G', and every edge is either an edge of the first closed path or an edge of some connected component of G'. Therefore if G is connected and each vertex of G has even degree, then G has an Eulerian Tour. ■

A graph with an Eulerian Tour is called an *Eulerian Graph*.

In Exercise 6.3-4, each vertex other than the initial and final vertices of the walk must have even degree by the same reasoning we used for Eulerian tours. But the initial vertex must have odd degree, because the first time we encounter it in our Eulerian Trail it is incident with one edge in the path, but each succeeding time it is incident with two edges in the path. Similarly the final vertex must have odd degree. This makes it natural to guess the following theorem.

Theorem 6.10 *A graph G has an Eulerian Trail if and only if G is connected and all but two of the vertices of G have even degree.*

Proof: We have already shown that if G has an Eulerian Trail, then all but two vertices of G have even degree and these two vertices have odd degree.

Now suppose that G is a connected graph in which all but two vertices have even degree. Suppose the two vertices of odd degree are x and y. Add an edge ϵ joining x and y to the edge set of G to get G'. Then G' has an Eulerian Tour by Theorem 6.9. One of the edges of the tour is the added edge. We may traverse the tour starting with any vertex and any edge following that vertex in the tour, so we may begin the tour with either $x\epsilon y$ or $y\epsilon x$. By removing the first vertex and ϵ from the tour, we get an Eulerian Trail. ■

By Theorem 6.10, there is no Eulerian Trail in Exercise 6.3-5.

Notice that our proof of Theorem 6.9 gives us a recursive algorithm for constructing a Tour. Namely, we find a closed walk W starting and ending at a vertex we choose, identify the connected components of the graph $G - W$ that results from removing the closed walk, and then follow our closed walk, pausing each time we enter a new connected component of $G - W$ to recursively construct an Eulerian Tour of the component and traverse it before returning to following our closed walk. It is possible that the closed walk we remove has only one edge (or in the case of a simple graph, some very small number of edges), and the number of steps needed for a breadth first search is $\Theta(e')$, where e' the number of edges in the graph we are searching. Thus our construction could take $\Theta(e)$ steps, each of which involves examining $\Theta(e)$ edges, and therefore our algorithm takes $O(e^2)$ time. (We get a big Oh bound and not a big Theta bound because it is also possible that the closed walk we find the first time is an Eulerian tour.)

It is an interesting observation on the progress of mathematical reasoning that Euler made a big deal in his paper of explaining why it is necessary for each land mass to have an even number of bridges, but seemed to consider the process of constructing the path rather self-evident, as if it was hardly worth comment. For us, on the other hand, proving that the construction is possible if each land mass has an even number of bridges (that is, showing that the condition that each land mass has an even number of bridges is a sufficient condition for the existence of an Eulerian tour) was a much more significant effort than proving that having an Eulerian tour requires that each land mass has an even number of bridges. The standards of what is required in order to back up a mathematical claim have changed over the years.

Hamiltonian Paths and Cycles

A natural question to ask in light of our work on Eulerian tours is whether we can state necessary and sufficient conditions for a graph to have a closed path that includes each vertex exactly once (except for the beginning and end). An answer to this question would have the potential to be quite useful. For example, a salesperson might have to plan a trip through a number of cities which are connected by a network of airline routes. Planning the trip so the salesperson would travel through a city only when stopping there for a sales call would minimize the number of flights the needed. This question came up in a game, called "around the world," designed by William Rowan Hamilton. In this game the vertices of the graph were the vertices of a dodecahedron (a twelve sided solid in which each side is a pentagon), and the edges were the edges of the solid. The object was to design a trip that started at one vertex and visited each vertex once and then returned to the starting vertex along an edge. Hamilton suggested that players could take turns, one choosing the first five cities on a tour, and the other trying to complete the tour. It is because of this game that a cycle that includes each vertex of the graph exactly once (thinking of the first and last vertex of the cycle as the same) is called a *Hamiltonian Cycle*. A graph is called Hamiltonian if it has a Hamiltonian cycle.. A *Hamiltonian Path* is a simple path that includes each vertex of the graph exactly once. It turns out that nobody yet knows (and as we shall explain briefly at the end of the section, it may be reasonable to expect that nobody will find) uesful necessary and sufficient conditions for a graph to have a Hamiltonian Cycle or a Hamiltonian Path that are significantly easier to verify than trying all permutations of the vertices to see if taking the vertices in the order of that permutation to see if that order defines a Hamiltonian Cycle or Path. For this reason this branch of graph theory has evolved into theorems that give sufficient conditions for a graph to have a Hamiltonian Cycle or Path; that is theorems that say all graphs of a certain type have Hamiltonian Cycles or Paths, but do not characterize all graphs that have Hamiltonian Cycles of Paths.

Exercise 6.3-6 Describe all values of n such that a complete graph on n vertices has a Hamiltonian Path. Describe all values of n such that a complete graph on n vertices has a Hamiltonian Cycle.

Exercise 6.3-7 Determine whether the graph of Figure 6.1 has a Hamiltonian Cycle or Path, and determine one if it does.

Exercise 6.3-8 Try to find an interesting condition involving the degrees of the vertices of a simple graph that guarantees that the graph will have a Hamiltonian cycle. Does your condition apply to graphs that are not simple? (There is more than one reasonable answer to this exercise.)

In Exercise 6.3-6, the path consisting of one vertex and no edges is a Hamiltonian path but not a Hamiltonian cycle in the complete graph on one vertex. (Recall that a path consisting of one vertex and no edges is not a cycle.) Similarly, the path with one edge in the complete graph K_2 is a Hamiltonian path but not a Hamiltonian cycle, and since K_2 has only one edge, there is no Hamiltonian cycle in the K_2. In the complete graph K_n, any permutation of the vertices is a list of the vertices of a Hamiltonian path, and if $n > 3$, such a Hamiltonian Path from x_1 to x_n, followed by the edge from x_n to x_1 and the vertex x_1 is a Hamiltonian Cycle. Thus each complete graph has a Hamiltonian Path, and each complete graph with more than three vertices has a Hamiltonian Cycle.

In Exercise 6.3-7, the path with vertices NO, A, MI, W, P, NY, B, CH, CL, and ME is a Hamiltonian Path, and adding the edge from ME to NO gives a Hamiltonian Cycle.

Based on our observation that the complete graph on n vertices has a Hamiltonian Cycle if $n > 2$, we might let our condition be that the degree of each vertex is one less than the number of vertices, but this would be uninteresting since it would simply restate what we already know for complete graphs. The reason why we could say that K_n has a Hamiltonian Cycle when $n > 3$ was that when we entered a vertex, there was always an edge left on which we could leave the vertex. However the condition that each vertex has degree $n-1$ is stronger than we needed, because until we were at the second-last vertex of the cycle, we had more choices than we needed for edges on which to leave the vertex. On the other hand, it might seem that even if n were rather large, the condition that each vertex should have degree $n-2$ would not be sufficient to guarantee a Hamiltonian cycle, because when we got to the second last vertex on the cycle, all of the $n-2$ vertices it is adjacent to might already be on the cycle and different from the first vertex, so we would not have an edge on which we could leave that vertex. However there is the possibility that when we had some choices earlier, we might have made a different choice and thus included this vertex earlier on the cycle, giving us a different set of choices at the second last vertex. In fact, if $n > 3$ and each vertex has degree at least $n-2$, then we could choose vertices for a path more or less as we did for the complete graph until we arrived at vertex $n-1$ on the path. Then we could complete a Hamiltonian path unless x_{n-1} was adjacent only to the first $n-2$ vertices on the path. In this last case, the first $n-1$ vertices would form a cycle, because x_{n-1} would be adjacent to x_1. Suppose y was the vertex not yet on the path. Since y has degree $n-2$ and y is not adjacent to x_{n-1}, y would have to be adjacent to the first $n-2$ vertices on the path. Then since $n > 3$, we could take the path $x_1yx_2\ldots x_{n-1}x_1$ and we would have a Hamiltonian cycle. Of course unless n were four, we could also insert y between x_2 and x_3 (or any x_{i-1} and x_i such that $i < n-1$, so we would still have a great deal of flexibility. To push this kind of reasoning further, we will introduce a new technique that often appears in graph theory. We will point out our use of the technique after the proof.

Theorem 6.11 (Dirac) *If every vertex of a v-vertex simple graph G with at least three vertices has degree at least $v/2$, then G has a Hamiltonian cycle.*

Proof: Suppose, for the sake of contradiction that there is a graph G_1 with no Hamiltonian Cycle in which each vertex has degree at least $v/2$. If we add edges joining existing vertices to G_1, each vertex will still have degree at least $v/2$. If add all possible edges to G_1 we will get a complete graph, and it will have a Hamiltonian cycle. Thus if we continue adding edges to G_1, we will at some point reach a graph that does have a Hamiltonian cycle. Instead, we add edges to G_1 until we reach a graph G_2 that has no Hamiltonian cycle but has the property that if we add any edge to G_2, we get a Hamiltonian cycle. We say G_2 is *maximal* with respect to not having a Hamiltonian cycle. Suppose x and y are not adjacent in G_2. Then adding an edge between x and y to G_2 gives a graph with a Hamiltonian cycle, and x and y must be connected by the added edge in this Hamiltonian cycle. (Otherwise G_2 would have a Hamiltonian cycle.) Thus G_2 has a Hamiltonian path $x_1x_2\ldots x_v$ that starts at $x = x_1$ and ends at $y = x_v$. Further x and y are not adjacent.

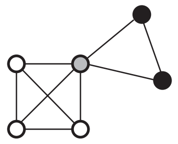
Before we stated our theorem we considered a case where we had a cycle on $f-1$ vertices and were going to put an extra vertex into it between two adjacent vertices. Now we have a path on f vertices from $x = x_1$ to $y = x_f$, and we want to convert it to a cycle. If we had that y is adjacent to some vertex x_i on the path while x is adjacent to x_{i+1}, then we could construct the Hamiltonian cycle $x_1x_{i+1}x_{i+2}\ldots x_fx_ix_{i-1}\ldots x_2x_1$. But we are assuming our graph does not have a Hamiltonian cycle. Thus for each x_i that x is adjacent to on the path $x_1x_2\ldots x_v$, y is not adjacent to x_{i-1}. Since all vertices are on the path, x is adjacent to at least $v/2$ vertices among x_2 through x_v. Thus y is not adjacent to at least $v/2$ vertices among x_1 through x_{v-1}. But there are only $v-1$ vertices, namely x_1 through x_{v-1}, vertices y could be adjacent to, since it is not adjacent to itself. Thus y is adjacent at most $v - 1 - v/2 = v/2 - 1$ vertices, a contradiction. Therefore if each vertex of a simple graph has degree at least $v/2$, the graph has a Hamiltonian Cycle. ■ The new tachnique was that of assuming we had a maximal graph (G_2) that did not have our desired property and then using this maximal graph in a proof by contradiction.

Exercise 6.3-9 Suppose $v = 2k$ and consider a graph G consisting of two complete graphs, one with k vertices, $x_1, \ldots x_k$ and one with $k + 1$ vertices, $x_k, \ldots x_{2k}$. Notice that we get a graph with exactly $2k$ vertices, because the two complete graphs have one vertex in common. How do the degrees of the vertices relate to v? Does the graph you get have a Hamiltonian cycle? What does this say about whether we can reduce the lower bound on the degree in Theorem 6.11?

Exercise 6.3-10 In the previous exercise, is there a similar example in the case $v = 2k+1$?

In Exercise 6.3-9, the vertices that lie in the complete graph with k vertices, with the exception of x_k, have degree $k - 1$. Since $v/2 = k$, this graph does not satisfy the hypothesis of Dirac's theorem which assumes that every vertex of the graph has degree at least $v/2$. We show the case in which $k = 3$ in Figure 6.18.

Figure 6.18: The vertices of K_4 are white or grey; those of K_3 are black or grey.



You can see that the graph in Figure 6.18 has no Hamiltonian cycle as follows. If an attempt at a Hamiltonian cycle begins at a white vertex, after crossing the grey vertex to include the black ones, it can never return to a white vertex without using the grey one a second time. The situation is similar if we tried to begin a Hamiltonian cycle at a black vertex. If we try to begin a Hamiltonian cycle at the grey vertex, we would next have to include all white vertices or all black vertices in our cycle and would then be stymied because we would have to take our path through the grey vertex a second time to change colors between white and black. As long as $k \geq 2$, the same argument shows that our graph has no Hamiltonian cycle. Thus the lower bound of $v/2$ in Dirac's theorem is tight; that is, we have a way to construct a graph with minimum degree $v/2 - 1$ (when v is even) for which there is no Hamiltonian cycle. If $v = 2k + 1$ we might consider two complete graphs of size $k + 1$, joined at a single vertex. Each vertex other than the one at which they are joined would have degree k, and we would have $k < k + 1/2 = v/2$, so again the minimum degree would be less than $v/2$. The same kind of argument that we used with the graph in Figure 6.18 would show that as long as $k \geq 1$, we have no Hamiltonian cycle.

If you analyze our proof of Dirac's theorem, you will see that we really used only a consequence of the condition that all vertices have degree at least $v/2$, namely that for any two vertices, the sum of their degrees is at least n.

Theorem 6.12 (Ore) *If G is a v-vertex simple graph with $n \geq 3$ such that for each two nonadjacent vertices x and y the sum of the degrees of x and y is at least v, then G has a Hamiltonian cycle.*

Proof: See Problem 13. ■

NP-Complete Problems

As we began the study of Hamiltonian Cycles, we mentioned that the problem of determining whether a graph has a Hamiltonian Cycle seems significantly more difficult than the problem of determining whether a graph has a Eulerian Tour. On the surface these two problems have significant similarities.

- Both problems whether a graph has a particular property. (Does this graph have a Hamiltonian/Eulerian closed path?) The answer is simply yes or no.
- For both problems, there is additional information we can provide that makes it relatively easy to check a yes answer if there is one. (The additional information is a closed path. We simply check whether the closed path includes each edge or each vertex exactly once.)

But there is a striking difference between the two problems as well. It is reasonably easy to find an Eulerian path in a graph that has one (we saw that the time to use the algorithm implicit in the proof of Theorem 6.9 is $O(e^2)$ where e is the number of edges of the graph. However, nobody knows how to actually find a permutation of the vertices that is a Hamiltonian path without checking essentially all permutations of the vertices.[8] This puts us in an interesting position. Although if someone gets lucky and guesses a permutation that is a Hamiltonian path, we can quickly verify the person's claim to have a Hamiltonian path, but in a graph of reasonably large size we have no practical method for finding a Hamiltonian path.

This difference is the essential difference between the class **P** of problems said to be solvable in polynomial time and the class **NP** of problems said to be solvable in nondeterministic polynomial time. We are not going to describe these problem classes in their full generality. A course in formal languages or perhaps algorithms is a more appropriate place for such a discussion. However in order to give a sense of the difference between these kinds of problems, we will talk about them in the context of graph theory. A question about whether a graph has a certain property is called a *graph decision problem.* Two examples are the question of whether a graph has an Eulerian tour and the question of whether a graph has a Hamiltonian cycle.

A graph decision problem has a yes/no answer. A **P**-*algorithm* or polynomial time algorithm for a property takes a graph as input and in time $O(n^k)$, where k is a positive integer independent of the input graph and n is a measure of the amount of information needed to specify the input graph, it outputs the answer "yes" if and only if the graph does have the property. We say the algorithm *accepts* the graph if it answers yes. (Notice we don't specify what the algorithm does if the graph does not have the property, except that it doesn't output yes.) We say a property of graphs is *in the class* **P** if there is a P-algorithm that accepts exactly the graphs with the property.

An **NP**-algorithm (non-deterministic polynomial time) for a property takes a graph and $O(n^j)$ additional information, and in time $O(n^k)$, where k and j are positive integers independent of

[8]We say essentially because one can eliminate some permutations immediately; for example if there is no edge between the first two elements of the permutation, then not only can we ignore the rest of this permutation, but we may ignore any other permutation that starts in this way. However nobody has managed to find a sub-exponential time algorithm for solving the Hamiltonian cycle problem.

the the input graph and n is a measure of the amount of information needed to specify the input graph, outputs the answer yes if and only if it can use the additional information to determine that the graph has the property. For example for the property of being Hamiltonian, the algorithm might input a graph and a permutation of the vertex set of the graph. The algorithm would then check the permutation to see if it lists the vertices in the order of a Hamiltonian cycle and output "yes" if it does. We say the algorithm *accepts* a graph if there is additional information it can use with the graph as an input to output "yes." We call such an algorithm nondeterministic, because whether or not it accepts a graph is determined not merely by the graph but by the additional information as well. In particular, the algorithm might or might not accept every graph with the given property. We say a property is *in the class* **NP** if there is an **NP**-algorithm that accepts exactly the graphs with the property. Since graph decision problems ask us to decide whether or not a graph has a given problem, we adopt the notation P and NP to describe problems as well. We say a decision problem is in **P** or **NP** if the graph property it asks us to decide is in **P** or **NP** respectively.

When we say that a nondeterministic algorithm uses the additional information, we are thinking of "use" in a very loose way. In particular, for a graph decision problem in **P**, the algorithm could simply ignore the additional information and use the polynomial time algorithm to determine whether the answer should be yes. Thus every graph property in **P** is also in **NP** as well. The question as to whether **P** and **NP** are the same class of problems has vexed computer scientists since it was introduced in 1968.

Some problems in **NP**, like the Hamiltonian path problem have an exciting feature: any instance[9] of any problem in **NP** can be translated in $O(n^k)$ steps, where n and k are as before, into $O(n^j)$ instances of the Hamilton path problem, where j is independent of n and k. In particular, the answer to the original problem is yess if and only if the answer to one of the Hamiltonian path problems is yes. The translation preserves whether or not the graph in the original instance of the problem is accepted. We say that the Hamiltonian Path problem is **NP**-complete. More generally, a problem in **NP** if called **NP**-complete if, for each other problem in **NP**, we can devise an algorithm for the second problem that has $O(n^k)$ steps (where n is a measure of the size of the input graph, and k is independent of n), including counting as one step solving an instance of the first problem, and accepts exactly the instances of the second problem that have a yes answer. The question of whether a graph has a clique (a subgraph that is a complete graph) of size j is another problem in **NP** that is **NP**-complete. In particular, if one **NP** complete problem has a polynomial time algorithm, every problem in NP is in **P**. Thus we can determine in polynomial time whether an arbitrary graph has a Hamiltonian path if and only if we can determine in polynomial time whether an arbitrary graph has a clique of (an arbitrary) size j. Literally hundreds of interesting problems are NP-complete. Thus a polynomial time solution to any one of them would provide a polynomial time solution to all of them. For this reason, many computer scientists consider a demonstration that a problem is NP-complete to be a demonstration that it is unlikely to be solved by a polynomial time algorithm.

This brief discussion of NP-completeness is intended to give the reader a sense of the nature and importance of the subject. We restricted ourselves to graph problems for two reasons. First, we expect the reader to have a sense of what a graph problem is. Second, no treatment of graph theory is complete without at least some explanation of how some problems seem to be much more intractable than others. However, there are **NP**-complete problems throughout mathematics and

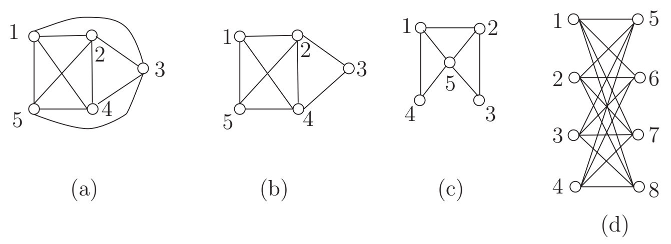
[9]An instance of a problem is a case of the problem in which all parameters are specified; for example a particular instance of the Hamiltonian Cycle problem is a case of the problem for a particular graph.

computer science. Providing a real understanding of the subject would require much more time than is available in an introductory course in discrete mathematics.

Important Concepts, Formulas, and Theorems

1. A graph that has a path, starting and ending at the same place, that includes each vertex at least once and each edge once and only once is called an *Eulerian Graph.* Such a path is known as an *Eulerian Tour* or *Eulerian Circuit.*
2. A graph has an Eulerian Tour if and only if it is connected and each vertex has even degree.
3. A path that includes each vertex of the graph at least once and each edge of the graph exactly once, but has different first and last endpoints, is known as an *Eulerian Trail*
4. A graph G has an Eulerian Trail if and only if G is connected and all but two of the vertices of G have even degree.
5. A cycle that includes each vertex of a graph exactly once (thinking of the first and last vertex of the cycle as the same) is called a *Hamiltonian Cycle.* A graph is called Hamiltonian if it has a Hamiltonian cycle.
6. A *Hamiltonian Path* is a simple path that includes each vertex of the graph exactly once.
7. (Dirac's Theorem) If every vertex of a v-vertex simple graph G with at least three vertices has degree at least $v/2$, then G has a Hamiltonian cycle.
8. (Ore's Theorem) If G is a v-vertex simple graph with $v \geq 3$ such that for each two non-adjacent vertices x and y the sum of the degrees of x and y is at least v, then G has a Hamiltonian cycle.
9. A question about whether a graph has a certain property is called a *graph decision problem.*
10. A **P**-*algorithm* or polynomial time algorithm for a property takes a graph as input and in time $O(n^k)$, where k is a positive integer independent of the input graph and n is a measure of the amount of information needed to specify the input graph, it outputs the answer "yes" if and only if the graph does have the property. We say the algorithm *accepts* the graph if it answers yes.
11. We say a property of graphs is *in the class* **P** if there is a P-algorithm that accepts exactly the graphs with the property.
12. An **NP**-algorithm (non-deterministic polynomial time) for a property takes a graph and $O(n^j)$ additional information, and in time $O(n^k)$, where k and j are positive integers independent of the the input graph and n is a measure of the amount of information needed to specify the input graph, outputs the answer yes if and only if it can use the additional information to determine that the graph has the property.
13. A graph decision problem in **NP** if called **NP**-complete if, for each other problem in **NP**, we can devise an algorithm for the second problem that has $O(n^k)$ steps (where n is a measure of the size of the input graph, and k is independent of n), including counting as one step solving an instance of the first problem, and accepts exactly the instances of the second problem that have a yes answer.

Figure 6.19: Some graphs

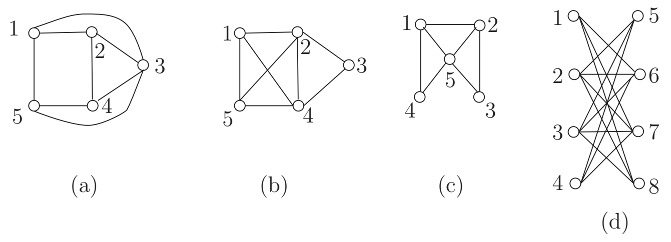


Problems

1. For each graph in Figure 6.19, either explain why the graph does not have an Eulerian circuit or find an Eulerian Circuit.

2. For each graph in Figure 6.20, either explain why the graph does not have an Eulerian Trail or find an Eulerian Trail.

Figure 6.20: Some more graphs



3. What is the minimum number of new bridges that would have to be built in Königsberg and where could they be built in order to give a graph with an Eulerian circuit?

4. If we built a new bridge in Königsberg between the Island and the top and bottom banks of the river, could we take a walk that crosses all the bridges and uses none twice? Either explain where could we start and end in that case or why we couldn't do it.

5. For which values of n does the complete graph on n vertices have an Eulerian Circuit?

6. The hypercube graph Q_n has as its vertex set the n-tuples of zeros and ones. Two of these vertices are adjacent if and only if they are different in one position. The name comes from the fact that Q_3 can be drawn in three dimensional space as a cube. For what values of n is Q_n Eulerian?

7. For what values of n is the hypercube graph Q_n (see Problem 6) Hamiltonian?

8. Give an example of a graph which has a Hamiltonian cycle but no Eulerian Circuit and a graph which has an Eulerian Circuit but no Hamiltonian Cycle.

9. The complete bipartite graph $K_{m,n}$ is a graph with $m + n$ vertices. These vertices are divided into a set of size m and a set of size n. We call these sets the *parts* of the graph. Within each of these sets there are *no* edges. But between each pair of vertices in different sets, there is an edge. The graph $K_{4,4}$ is pictured in part (d) of Figure 6.19.

   (a) For what values of m and n is $K_{m,n}$ Eulerian?

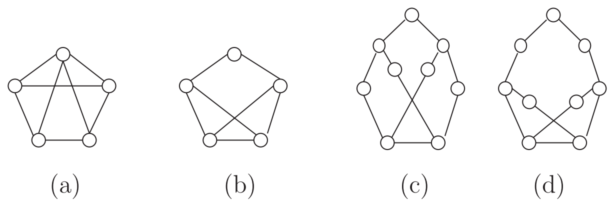
   (b) For which values of m and n is $K_{m,n}$ Hamiltonian?

10. Show that the edge set of a graph in which each vertex has even degree may be partitioned into edge sets of cycles of the graph.

11. A cut-vertex of a graph is a vertex whose removal (along with all edges incident with it) increases the number of connected components of the graph. Describe any circumstances under which a graph with a cut vertex can be Hamiltonian.

12. Which of the graphs in Figure 6.21 satisfy the hypotheses of Dirac's Theorem? of Ore's Theorem? Which have Hamiltonian cycles?

Figure 6.21: Which of these graphs have Hamiltonian Cycles?



13. Prove Theorem 6.12.

14. The Hamiltonian Path problem is the problem of determining whether a graph has a Hamiltonian Path. Explain why this problem is in **NP**. Explain why the problem of determining whether a graph has a Hamiltonian Path is **NP**-complete.

15. The k-Path problem is the problem of determining whether a graph on n vertices has a path of length k, where k *is* allowed to depend on n. Show that the k-Path problem is **NP**-complete.

16. We form the Hamiltonian closure of a graph by constructing a sequence of graphs G_i with $G_0 = G$, and G_i formed from G_{i-1} by adding an edge between two nonadjacent vertices whose degree sum is at least nv. When we reach a G_i to which we cannot add such an edge, we call it a Hamiltonian Closure of G. Prove that a Hamiltonian Closure of a simple graph G is Hamiltonian if and only if G is.

17. Show that a simple connected graph has one and only one Hamiltonian closure.

6.4 Matching Theory

The idea of a matching

Suppose a school board is deciding among applicants for faculty positions. The school board has positions for teachers in a number of different grades, a position for an assistant librarian, two coaching positions, and for high school math and English teachers. They have many applicants, each of whom can fill more than one of the positions. They would like to know whether they can fill all the positions with people who have applied for jobs and have been judged as qualified.

Exercise 6.4-1 Table 6.1 shows a sample of the kinds of applications a school district might get for its positions. An x below an applicant's number means that that applicant

Table 6.1: Some sample job application data

job\ applicant	1	2	3	4	5	6	7	8	9
assistant librarian	x		x	x					
second grade	x	x	x	x					
third grade	x	x		x					
high school math				x	x	x			
high school English				x		x	x		
asst baseball coach						x	x	x	x
asst football coach					x	x		x	

qualifies for the position to the left of the x. Thus candidate 1 is qualified to teach second grade, third grade, and be an assistant librarian. The coaches teach physical education when they are not coaching, so a coach can't also hold one of the listed teaching positions. Draw a graph in which the vertices are labelled 1 through 9 for the applicants, and s, t, l, m, e, b, and f for the positions. Draw an edge from an applicant to a position if that applicant can fill that position. Use the graph to help you decide if it is possible to fill all the positions from among the applicants deemed suitable. If you can do so, give an assignment of people to jobs. If you cannot, try to explain why not.

Exercise 6.4-2 Table 6.2 shows a second sample of the kinds of applications a school district might get for its positions. Draw a graph as before and use it to help you decide if it is possible to fill all the positions from among the applicants deemed suitable. If you can do so, give an assignment of people to jobs. If you cannot, try to explain why not.

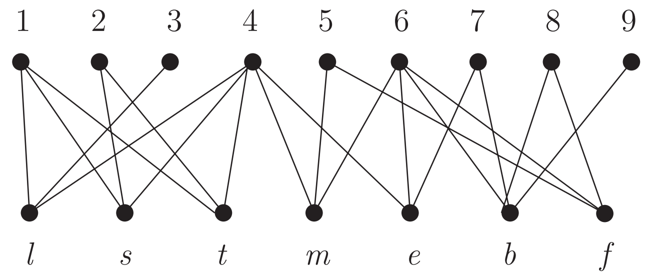
The graph of the data in Table 6.1 is shown in Figure 6.22.

From the figure it is clear that l:1, s:2, t:4, m:5, e:6, b:7, f:8 is one assignment of jobs to people that works. This assignment picks out a set of edges that share no endpoints. For example, the edge from l to 1 has no endpoint among s, t, m, e, b, f, 2, 3, 4, 5, 6, 7, or 8. A set of edges in a graph that share no endpoints is called a *matching* of the graph. Thus we have a matching between jobs and people that can fill the jobs. Since we don't want to assign two jobs to one

Table 6.2: Some other sample job application data

job\ applicant	1	2	3	4	5	6	7	8	9
library assistant				x	x				
second grade	x	x	x					8	
third grade		x	x	x			x		x
high school math				x	x	x			
high school English					x	x			
asst baseball coach						x	x	x	x
asst football coach				x	x	x			

Figure 6.22: A graph of the data from Table 6.1.

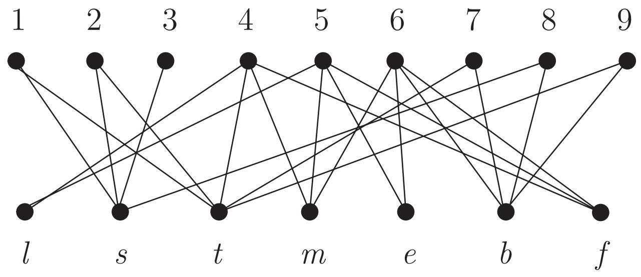


person or two people to one job, this is exactly the sort of solution we were looking for. Notice that the edge from l to 1 is a matching all by itself, so we weren't simply looking for a matching; we were looking for a matching that fills all the jobs. A matching is said to *saturate* a set X of vertices if every vertex in X is matched. We wanted a matching that saturates the jobs. In this case a matching that saturates all the jobs is a matching that is as big as possible, so it is also a *maximum* matching, that is, a matching that is at least as big as any other matching.

The graph in Figure 6.22 is an example of a "bipartite graph." A graph is called *bipartite* whenever its vertex set can be partitioned into two sets X and Y so that each edge connects a vertex in X with a vertex in Y. We can think of the jobs as the set X and the applicants as the set Y. Each of the two sets is called a *part* of the graph. A part of a bipartite graph is an example of an "independent set." A subset of the vertex set of a graph is called *independent* if no two of its vertices are connected by an edge. (In particular, a vertex connected to itself by a loop is in no independent set.) Thus a graph is bipartite if and only if its vertex set is a union of two independent sets.

In a bipartite graph, it is sometimes easy to pick a maximum matching out just by staring at a drawing of the graph. However that is not always the case. Figure 6.23 is a graph of the data in Table 6.2. Staring at this Figure gives us many matchings, but no matching that saturates the set of jobs. But staring is not a proof, unless we can describe what we are staring at very well. Perhaps you tried to construct a matching by matching l to 4, s to 2, t to 7, m to 5, e to 6, b to 7, and then were frustrated when you got to f and 4, 5 and 6 were already used. You may then have gone back and tried to redo your earlier choices so as to keep one of 4, 5, or 6 free, and found you couldn't do so. This is because jobs l, m, e, and f are adjacent only to people 4, 5, and 6. Thus there are only three people qualified for these four jobs, and so there is no way we

Figure 6.23: A graph of the data of Table 6.2.



can fill them all.

We call the set $N(S)$ of all vertices adjacent to at least one vertex of S the *neighborhood* of S or the *neighbors* of S. In these terms, there is no matching that saturates a part X of a bipartite graph if there is some subset S of X such that the set $N(S)$ of neighbors of S is smaller than S. We call the set $N(S)$ of all vertices adjacent to at least one vertex of S the *neighborhood* of S or the *neighbors* of S. In symbols, we can summarize as follows.

Lemma 6.13 *If we can find a subset S of a part X of a bipartite graph G such that $|N(S)| < |S|$, then there is no matching of G that saturates X.*

Proof: A matching that saturates X must saturate S. But if there is such a matching, each element of S must be matched to a different vertex, and this vertex cannot be in S since $S \subseteq X$. Therefore there are edges from vertices in S to at least $|S|$ different vertices not in S, so $|N(S)| > |S|$, a contradiction. Thus there is no such matching.■

This gives a proof that there is no matching that saturates all the jobs, so the matching that matches matching l to 4, s to 2, t to 7, m to 5, e to 6, b to 7 is a maximum matching for the graph in Figure 6.23.

Another method you may have used to prove that there is no larger matching than the one we found is the following. When we matched l to 4, we may have noted that 4 is an endpoint of quite a few edges. Then when we matched s to 2, we may have noted that s is an endpoint of quite a few edges, and so is t. In fact, 4, s, and t touch 12 edges of the graph, and there are only 23 edges in the graph. If we could find three more vertices that touch the remaining edges of the graph, we would have six vertices that among them are incident with every edge. A set of vertices such that at least one of them is incident with each edge of a graph G is called a *vertex cover of the edges* of G, or a *vertex cover* of G for short. What does this have to do with a matching? Each matching edge would have to touch one, or perhaps two of the vertices in a vertex cover of the edges. Thus the number of edges in a matching is always less than the number of vertices in a vertex cover of the edges of a graph. Thus if we can find a vertex cover of size six in our graph in Figure 6.23, we will know that there is no matching that saturates the set of jobs since there are seven jobs. For future reference, we state our result about the size of a matching and the size of a vertex cover as a lemma.

Lemma 6.14 *The size of a matching in a graph G is no more than the size of a vertex cover of G.*

Proof: Given in the preceding discussion.■

We have seen that 4, s, and t are good candidates for being members of a relatively small vertex cover of the graph in Figure 6.23, since they cover more than half the edges of the graph. Continuing through the edges we first examined, we see that m, 6, and b are good candidates for a small vertex cover as well. In fact, $\{4, s, t, m, 6, b\}$ do form a vertex cover. Since we have a vertex cover of size six, we know a maximum matching has size no more than six. Since we have already found a six-edge matching, that is a maximum matching. Therefore with the data in Table 6.2, it is not possible to fill all the jobs.

Making matchings bigger

Practical problems involving matchings will usually lead us to search for the largest possible matching in a graph. To see how to use a matching to create a larger one, we will assume we have two matchings of the same graph and see how they differ, especially how a larger one differs from a smaller one.

Exercise 6.4-3 In the graph G of Figure 6.22, let M_1 be the matching $\{l, 1\}$, $\{s, 2\}$, $\{t, 4\}$, $\{m, 5\}$, $\{e, 6\}$, $\{b, 9\}$, $\{f, 8\}$, and let M_2 be the matching $\{l, 4\}$, $\{s, 2\}$ $\{t, 1\}$, $\{m, 6\}$, $\{e, 7\}$ $\{b, 8\}$. Recall that for sets S_1 and S_2 the symmetric difference of S_1 and S_2, denoted by $S_1 \Delta S_2$ is $(S_1 \cup S_2) - (S_1 \cap S_2)$. Compute the set $M_1 \Delta M_2$ and draw the graph with the same vertex set as G and edge set $M_1 \Delta M_2$. Use different colors or textures for the edges from M_1 and M_2 so you can see their interaction. Describe the kinds of graphs you see as connected components as succinctly as possible.

Exercise 6.4-4 In Exercise 6.4-3, one of the connected components suggests a way to modify M_2 by removing one or more edges and substituting one or more edges from M_1 that will give you a larger matching M_2' related to M_2. In particular, this larger matching should saturate everything M_2 saturates and more. What is M_2' and what else does it saturate?

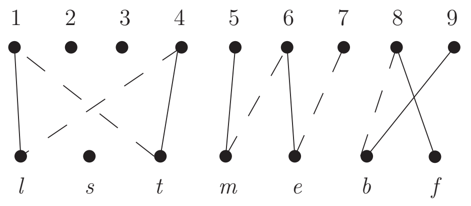
Exercise 6.4-5 Consider the matching $M = \{s, 1\}, \{t, 4\}, \{m, 6\}, \{b, 8\}$ in the graph of Figure 6.23. How does it relate to the simple path whose vertices are $3, s, 1, t, 4, m, 6, f$? Say as much as you can about the set M' that you obtain from M by deleting the edges of M that are in the path and adding to the result the edges of the path that are not in M.

In Exercise 6.4-3

$$M_1 \Delta M_2 = \{l, 1\}, \{l, 4\}, \{t, 4\}, \{t, 1\}, \{m, 5\}, \{m, 6\}, \{e, 6\}, \{e, 7\}, \{b, 8\}, \{f, 8\}, \{b, 9\}.$$

We have drawn the graph in Figure 6.24. We show the edges of M_2 as dashed. As you see, it consists of a cycle with four edges, alternating between edges of M_1 and M_2, a path with four edges, alternating between edges of M_1 and M_2, and a path with three edges, alternating between edges of M_1 and M_2. We call a simple path or cycle an *alternating path* or *alternating cycle* for a matching M of a graph G if its edges alternate between edges in M and edges not in M. Thus our connected components were alternating paths and cycles for both M_1 and M_2. The example we just discussed shows all the ways in which two matchings can differ in the following sense.

Figure 6.24: The graph for Exercise 6.4-3.



Lemma 6.15 (Berge) *If M_1 and M_2 are matchings of a graph $G = (V, E)$ then the connected components of $M_1 \Delta M_2$ are cycles with an even number of vertices and simple paths. Further, the the cycles and paths are alternating cycles and paths for both M_1 and M_2.*

Proof: Each vertex of the graph $(V, M_1 \Delta M_2)$ has degree 0, 1, or two. If a component has no cycles it is a tree, and the only kind of tree that has vertices of degree 1 and two is a simple path. If a component has a cycle, then it can not have any edges other the edges of the cycle incident with its vertices because the graph would then have a vertex of degree 3 or more. Thus the component must be a cycle. If two edges of a path or cycle in $(V, M_1 \Delta M_2)$ share a vertex, they cannot come from the same matching, since two edges in the same matching do not share a vertex. Thus alternating edges of a path or cycle of $(V, M_1 \Delta M_2)$ must come from different matchings. ■

Corollary 6.16 *If M_1 and M_2 are matchings of a graph $G = (V, E)$ and $|M_2| < |M_1|$, then there is an alternating path for M_1 and M_2 that starts and ends with vertices saturated by M_2 but not by M_1.*

Proof: Since an even alternating cycle and an even alternating path in $(V, M_1 \Delta M_2)$ have equal numbers of edges from M_1 and M_2, at least one component must be an alternating path with more edges from M_1 than M_2, because otherwise $|M_2| \le |M_1|$. Since this is a component of $(V, M_1 \Delta M_2)$, its endpoints lie only in edges of M_2, so they are saturated by M_2 but not M_1. ■

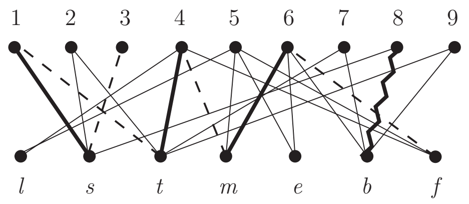
The path with three edges in Exercise 6.4-3 has two edges of M_1 and one edge of M_2. We see that if we remove $\{b, 8\}$ from M_2 and add $\{b, 9\}$ and $\{f, 8\}$, we get the matching

$$M_2' = \{\{l, 4\}, \{s, 2\}, \{t, 1\}, \{m, 6\}, \{e, 7\}, \{b, 9\}, \{f, 8\}\}.$$

This answers the question of Exercise 6.4-4. Notice that this matching saturates everything M_2 does, and also saturates vertices f and 9.

In Figure 6.25 we have shown the matching edges of the path in Exercise 6.4-5 in bold and the non-matching edges of the path as dashed. The edge of the matching not in the path is shown in zig-zag. Notice that the dashed edges and the zig-zag edge form a matching which is larger than M and saturates all the vertices that M does in addition to 3 and f. The path begins and ends with unmatched vertices, namely 3 and f, and and alternates between matching edges and non-matching edges. All but the first and last vertices of such a path lie on matching edges of the path and the endpoints of the path do not lie on matching edges. Thus no edges of the matching that are not path-edges will be incident with vertices on the path. Thus if we delete all the matching edges of the path from M and add all the other edges of the path to M, we will get

Figure 6.25: The path and matching of Exercise 6.4-5.



a new matching, because by taking every second edge of a simple path, we get edges that do not have endpoints in common. An alternating path is called an *augmenting path* for a matching M if it begins and ends with M-unsaturated vertices. That is, it is an alternating path that begins and ends with unmatched vertices. Our preceding discussion suggests the proof of the following theorem.

Theorem 6.17 (Berge) *A matching M in a graph is of maximum size if and only if M has no augmenting path. Further, if a matching M has an augmenting path P with edge set $E(P)$, then we can create a larger matching by deleting the edges in $M \cap E(P)$ from M and adding in the edges of $E(P) - M$.*

Proof: First if there is a matching M_1 larger than M, then by Corollary 6.16 there is an augmenting path for M. Thus if a matching has maximum size, it has no augmenting path. Further, as in our discussion of Exercise 6.4-5, if there is an augmenting path for M, then there is a larger matching for M. Finally, this discussion showed that if P is an augmenting path, we can get such a larger matching by deleting the edges in $M \cap E(P)$ and adding in the edges of $E(P) - M$. ■

Corollary 6.18 *While the larger matching of Theorem 6.17 may not contain M as a subset, it does saturate all the vertices that M saturates and two additional vertices.*

Proof: Every vertex incident with an edge in M is incident with some edge of the larger matching, and each of the two endpoints of the augmenting path is also incident with a matching edge. Because we may have removed edges of M to get the larger matching, it may not contain M.■

Matching in Bipartite Graphs

While our examples have all been bipartite, all our lemmas, corollaries and theorems about matchings have been about general graphs. In fact, it some of the results can be strengthened in bipartite graphs. For example, Lemma 6.14 tells us that the size of a matching is no more than the size of a vertex cover. We shall soon see that in a bipartite graph, the size of a maximum matching actually equals the size of a minimum vertex cover.

Searching for Augmenting Paths in Bipartite Graphs

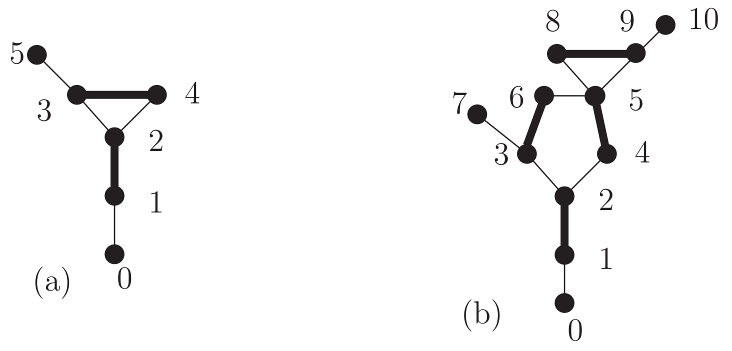
We have seen that if we can find an augmenting path for a matching M in a graph G, then we can create a bigger matching. Since our goal from the outset has been to create the largest matching possible, this helps us achieve that goal. However, you may ask, how do we find an augmenting path? Recall that a breadth-first search tree centered at a vertex x in a graph contains a path, in fact a shortest path, from x to every vertex y to which it is connected. Thus it seems that we ought to be able to alternate between matching edges and non-matching edges when doing a breadth-first search and find alternating paths. In particular, if we add vertex i to our tree by using a matching edge, then any edge we use to add a vertex *from* vertex i should be a non-matching edge. And if we add vertex i to our tree by using a non-matching edge, then any edge we use to add a vertex *from* vertex i should be a matching edge. (Thus there is just one such edge.) Because not all edges are available to us to use in adding vertices to the tree, the tree we get will not necessarily be a spanning tree of our original graph. However we can hope that if there is an augmenting path starting at vertex x and ending at vertex y, then we will find it by using breadth first search starting from x in this alternating manner.

Exercise 6.4-6 Given the matching $\{s, 2\}, \{t, 4\}, \{b, 7\}\{f, 8\}$ of the graph in Figure 6.22 use breadth-first search starting at vertex 1 in an alternating way to search for an augmenting path starting at vertex 1. Use the augmenting path you get to create a larger matching.

Exercise 6.4-7 Continue using the method of Exercise 6.4-6 until you find a matching of maximum size.

Exercise 6.4-8 Apply breadth-first search from vertex 0 in an alternating way to graph (a) in Figure 6.26. Does this method find an augmenting path? Is there an augmenting path?

Figure 6.26: Matching edges are shown in bold in these graphs.



For Exercise 6.4-6, if we begin at vertex 1, we add vertices l,s and t to our tree, giving them breadth-first numbers 1,2, and 3. Since l is not incident with a matching edge, we cannot continue the search from there. Since vertex s is incident with matching edge $\{s, 2\}$, we can use this edge to add vertex 2 to the tree and give it breadth-first number 4. This is the only vertex we can add from l since we can only use matching edges to add vertices from l. Similarly, from t we can add vertex 4 by using the matching edge $\{t, 4\}$ and giving it breadth-first number 5. All vertices adjacent to vertex 2 have already been added to the tree, but from vertex 4 we can use non-matching edges to add vertices m and e to our tree, giving them breadth first numbers 6 and 7. Now we can only use matching edges to add vertices to the tree from m or e, but there are no matching edges incident with them, so our alternating search tree stops here. Since m and e are unmatched, we know we have a path in our tree from vertex 1 to vertex m and a path from vertex 1 to vertex e. The vertex sequence of the path from 1 to m is $1s2t4m$ Thus our matching becomes $\{1, s\}, \{2, t\}, \{4, m\}, \{b, 7\}, \{f, 8\}$.

For Exercise 6.4-7 find another unmatched vertex and repeat the search. Working from vertex l, say, we start a tree by using the edges $\{l, 1\}, \{l, 3\}, \{l, 4\}$ to add vertices 1, 3, and 4. We could continue working on the tree, but since we see that $l\{l, 3\}3$ is an augmenting path, we use it to add the edge $\{l, 3\}$ to the matching, short-circuiting the tree-construction process. Thus our matching becomes $\{1, s\}, \{2, t\}, \{l, 3\}, \{4, m\}, \{b, 7\}\{f, 8\}$. The next unmatched vertex we see might be vertex 5. Starting from it, we add m and f to our tree, giving them breadth first numbers 1 and 2. From m we have the matching edge $\{m, 4\}$, and from f we have the matching edge $\{f, 8\}$, so we use them to add the vertices 4 and 8 to the tree. From vertex 4 we add l, s, t, and e to the tree, and from vertex 8 we add vertex b to the tree. All these vertices except e are in matching edges. Since e is not in a matching edge, we have discovered a vertex connected by an augmenting path to vertex 5. The path in the tree from vertex 5 to vertex e has vertex sequence $5m4e$, and using this augmenting path gives us the matching $\{1, s\}, \{2, t\}, \{l, 3\}, \{5, m\}, \{4, e\}, \{b, 7\}, \{f, 8\}$. Since we now have a matching whose size is the same as the size of a vertex cover, namely the bottom part of the graph in Figure 6.22, we have a matching of maximum size.

For Exercise 6.4-8 we start at vertex 0 and add vertex 1. From vertex 1 we use our matching edge to add vertex 2. From vertex 2 we use our two non-matching edges to add vertices 3 and 4. However, vertices 3 and 4 are incident with the same matching edge, so we cannot use that matching edge to add any vertices to the tree, and we must stop without finding an augmenting path. From staring at the picture, we see there is an augmenting path, namely 012435, and it gives us the matching $\{\{0, 1\}, \{2, 4\}, \{3, 5\}\}$. We would have similar difficulties in discovering either of the augmenting paths in part (b) of Figure 6.26.

It turns out to be the odd cycles in Figure 6.26 that prevent us from finding augmenting paths by our modification of breadth-first search. We shall demonstrate this by describing an algorithm which is a variation on the alternating breadth-first search we were using in solving our exercises. This algorithm takes a bipartite graph and a matching and either gives us an augmenting path or constructs a vertex cover whose size is the same as the size of the matching.

The Augmentation-Cover algorithm

We begin with a bipartite graph with parts X and Y and a matching M. We label the unmatched vertices in X with the label a (which stands for alternating). We number them in sequence as we label them. Starting with $i = 1$ and taking labeled vertices in order of the numbers we have assigned them, we use vertex number i to do additional labelling as follows.

1. If vertex i is in X, we label all unlabeled vertices adjacent to it with the label a and the name of vertex i. Then we number these newly labeled vertices, continuing our sequence of numbers without interruption.

2. If vertex i is in Y, and it is incident with an edge of M, its neighbor in the matching edge cannot yet be labeled. We label this neighbor with the label a and the name of vertex i.

3. If vertex i is in Y and it is not incident with an edge of M, then we have discovered an augmenting path: the path from vertex i to the vertex we used to add it (and recorded at vertex i) and so on back to one of the unlabeled vertices in X. It is alternating by our labeling method, and it starts and ends with unsaturated vertices, so it is augmenting.

If we can continue the labelling process until no more labeling is possible and we do not find an augmenting path, then we let A be the set of labeled vertices. The set $C = (X - A) \cup (Y \cap A)$ then turns out to be a vertex cover whose size is the size of M. We call this algorithm the *augmentation-cover algorithm.*

Theorem 6.19 (König and Egerváry) *In a bipartite graph with parts X and Y, the size of a maximum sized matching equals the size of a minimum-sized vertex cover.*

Proof: In light of Berge's Theorem, if the augmentation-cover algorithm gives us an augmenting path, then the matching is not maximum sized, and in light of Lemma 6.14, if we can prove that the set C the algorithm gives us when there is no augmenting path is a vertex cover whose size is the size of the matching, we will have proved the theorem. To see that C is a vertex cover, note that every edge incident with a vertex in $X \cap A$ is covered, because its endpoint in Y has been marked with an a and so is in $Y \cap A$. But every other edge must be covered by $X - A$ because in a bipartite graph, each edge must be incident with a vertex in each part. Therefore C is a vertex cover. If an element of $Y \cap A$, were not matched, it would be an endpoint of an augmenting path, and so all elements of $Y \cap A$ are incident with matching edges. But every vertex of $X - A$ is matched because A includes all unmatched vertices of X. By step 2 of the augmentation-cover algorithm, if ϵ is a matching edge with an endpoint in $Y \cap A$, then the other endpoint must be in A. Therefore each matching edge contains only one member of C. Therefore the size of a maximum matching is the size of C. ■

Corollary 6.20 *The augmentation-cover algorithm applied to a bipartite graph and a matching of that graph produces either an augmenting path for the matching or a minimum vertex cover whose size equals the size of the matching.*

Before we proved the König-Egárvary Theorem, we knew that if we could find a matching and a vertex cover of the same size, then we had a maximum sized matching and a minimum sized vertex cover. However it is possible that in some graphs we can't test for whether a matching is as large as possible by comparing its size to that of a vertex cover because a maximum sized matching might be smaller than a minimum sized vertex cover. The König-Egárvary Theorem tells us that in bipartite graphs this problem never arises, so the test always works.

We had a second technique we used to show that a matching could not saturate the set X of all jobs in Exercise 6.4-2. In Lemma 6.13 we showed that if we can find a subset S of a part X of a bipartite graph G such that $|N(S)| < |S|$, then there is no matching of G that saturates X. In other words, to have a matching that saturates X in a bipartite graph on parts X and Y, it is necessary that $|N(S)| \geq |S|$ for *every* subset S of X. (When $S = \emptyset$, then so does $N(S)$.) This necessary condition is called *Hall's condition*, and Hall's theorem says that this necessary condition is sufficient.

Theorem 6.21 (Hall) *If G is a bipartite graph with parts X and Y, then there is a matching of G that saturates X if and only if $|N(S)| \geq |S|$ for every subset $\subseteq X$.*

Proof: In Lemma 6.13 we showed (the contrapositive of the statement) that if there is a matching of G, then $|N(S)| \geq |S|$ for every subset of X. (There is no reason to use a contrapositive argument though; if there is a matching that saturates X, then because matching edges have no endpoints in common, the elements of each subset S of X will be matched to at least $|S|$ different elements, and these will all be in $N(S)$.)

Thus we need only show that if the graph satisfies Hall's condition then there is a matching that saturates S. We will do this by showing that X is a minimum-sized vertex cover. Let C be some vertex cover of G. Let $S = X - C$. If ϵ is an edge from a vertex in S to a vertex $y \in Y$, ϵ cannot be covered by a vertex in $C \cap X$. Therefore ϵ must be covered by a vertex in $C \cap Y$. This means that $N(S) \subseteq C \cap Y$, so $|C \cap Y| \geq |N(S)|$. By Hall's condition, $N(S)| > |S|$. Therefore $|C \cap Y| \geq |S|$. Since $C \cap X$ and $C \cap Y$ are disjoint sets whose union is C, we can summarize our remarks with the equation

$$|C| = |C \cap X| + |C \cap Y| \geq |C \cap X| + |N(S)| \geq |C \cap X| + |S| = |C \cap X| + |C - X| = |X|.$$

X is a vertex cover, and we have just shown that it is a vertex cover of minimum size . Therefore a matching of maximum size has size $|X|$. Thus there is a matching that saturates X. ■

Good Algorithms

While Hall's theorem is quite elegant, applying it requires that we look at every subset of X, which would take us $\Omega\left(2^{|X|}\right)$ time. Similarly, actually finding a minimum vertex cover could involve looking at all (or nearly all) subsets of $X \cup Y$, which would also take us exponential time. However, the augmentation-cover algorithm requires that we examine each edge at most some fixed number of times and then do a little extra work; certainly no more than $O(e)$ work. We need to repeat the algorithm at most X times to find a maximum matching and minimum vertex cover. Thus in time $O(ev)$, we can not only find out whether we have a matching that saturates X; we can find such a matching if it exists and a vertex cover that proves it doesn't exist if it doesn't. However this only applies to bipartite graphs. The situation is much more complicated in non-bipartite graphs. In a paper which introduced the idea that a good algorithm is one that runs in time $O(n^c)$, where n is the amount of information needed to specify the input and c is a constant, Edmunds[10] developed a more complicated algorithm that extended the idea of a search tree to a more complicated structure he called a flower. He showed that this algorithm was good in his sense, introduced the problem class **NP**, and conjectured that $\mathbf{P} \neq \mathbf{NP}$. In a wry twist of fate, the problem of finding a minimum vertex cover problem (actually the problem of determining whether there is a vertex cover of size k, where k can be a function of v) is, in fact, **NP**-complete in arbitrary graphs. It is fascinating that the matching problem for general graphs turned out to be solvable in polynomial time, while determining the "natural" upper bound on the size of a matching, an upper bound that originally seemed quite useful, remains out of our reach.

[10]Jack Edmonds. Paths, Trees and Flowers. *Canadian Journal of mathematics*, **17**, 1965 pp449-467

Important Concepts, Formulas, and Theorems

1. *Matching.* A set of edges in a graph that share no endpoints is called a *matching* of the graph.

2. *Saturate.* A matching is said to *saturate* a set X of vertices if every vertex in X is matched.

3. *Maximum Matching.* A matching in a graph is a *maximum matching* if it is at least as big as any other matching.

4. *Bipartite Graph.* A graph is called *bipartite* whenever its vertex set can be partitioned into two sets X and Y so that each edge connects a vertex in X with a vertex in Y. Each of the two sets is called a *part* of the graph.

5. *Independent Set.* A subset of the vertex set of a graph is called *independent* if no two of its vertices are connected by an edge. (In particular, a vertex connected to itself by a loop is in no independent set.) A part of a bipartite graph is an example of an 'independent set.

6. *Neighborhood.* We call the set $N(S)$ of all vertices adjacent to at least one vertex of S the *neighborhood* of S or the *neighbors* of S.

7. *Hall's theorem for a Matching in a Bipartite Graph.* If we can find a subset S of a part X of a bipartite graph G such that $|N(S)| < |S|$, then there is no matching of G that saturates X. If there is no subset $S \subseteq X$ such that $|N(S)| < |S|$, then there is a matching that saturates X.

8. *Vertex Cover.* A set of vertices such that at least one of them is incident with each edge of a graph G is called a *vertex cover of the edges* of G, or a *vertex cover* of G for short. In any graph, the size a matching is less than or equal to the size of any vertex cover.

9. *Alternating Path, Augmenting Path.* A simple path is called an *alternating path* for a matching M if, as we move along the path, the edges alternate between edges in M and edges not in M. An *augmenting path* is an alternating path that begins and ends at unmatched vertices.

10. *Berge's Lemma.* If M_1 and M_2 are matchings of a graph G then the connected components of $M_1 \Delta M_2$ are cycles with an even number of vertices and simple paths. Further, the cycles and paths are alternating cycles and paths for both M_1 and M_2.

11. *Berge's Corollary.* If M_1 and M_2 are matchings of a graph $G = (V, E)$ and $|M_1| > |M_2|$, then there is an alternating path for M_1 and M_2 that starts and ends with vertices saturated by M_1 but not by M_2.

12. *Berge's Theorem.* A matching M in a graph is of maximum size if and only if M has no augmenting path. Further, if a matching M has an augmenting path P with edge set $E(P)$, then we can create a larger matching by deleting the edges in $M \cap E(P)$ from M and adding in the edges of $E(P) - M$.

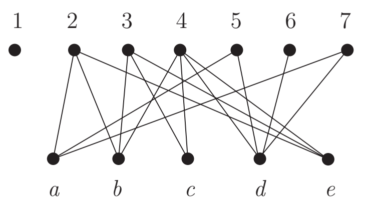
13. *Augmentation-Cover Algorithm.* The Augmentation-Cover algorithm is an algorithm that begins with a bipartite graph and a matching of that graph and produces either an augmenting path or a vertex cover whose size equals that of the matching, thus proving that the matching is a maximum matching.

14. *König-Egerváry Theorem.* In a bipartite graph with parts X and Y, the size of a maximum sized matching equals the size of a minimum-sized vertex cover.

Problems

1. Either find a maximum matching or a subset S of the set $X = \{a, b, c, d, e\}$ such that $|S| > |N(S)|$ in the graph of Figure 6.27

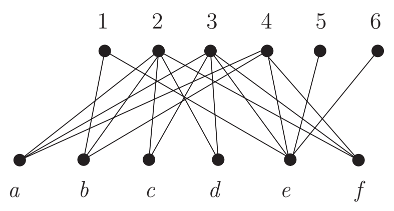
Figure 6.27: A bipartite graph



2. Find a maximum matching and a minimum vertex cover in the graph of Figure 6.27

3. Either find a matching which saturates the set $X = \{a, b, c, d, e, f\}$ in Figure 6.28 or find a set S such that $|N(S)| < |X|$.

Figure 6.28: A bipartite graph



4. Find a maximum matching and a minimum vertex cover in the graph of Figure 6.28.

5. In the previous exercises, when you were able to find a set S with $|S| > |N(S)|$, how did $N(S)$ relate to the vertex cover? Why did this work out as it did?

6. A star is a another name for a tree with one vertex connected to each of n other vertices. (So a star has $n + 1$ vertices.) What are the size of a maximum matching and a minimum vertex cover in a star with $n + 1$ vertices?

7. In Theorem 6.17 is it true that if there is an augmenting path P with edge set $E(P)$ for a matching M, then $M \Delta E(P)$ is a larger matching than M?

8. Find a maximum matching and a minimum vertex cover in graph (b) of Figure 6.26.

9. In a bipartite graph, is one of the parts always a maximum-sized independent set? What if the graph is connected?

10. Find infinitely many examples of graphs in which a maximum-sized matching is smaller than a minimum-sized vertex cover.

11. Find an example of a graph in which the maximum size of a matching is less than one quarter of the size of a minimum vertex cover.

12. Prove or give a counter-example: Every tree is a bipartite graph. (Note, a single vertex with no edges is a bipartite graph; one of the two parts is empty.)

13. Prove or give a counter-example. A bipartite graph has no odd cycles.

14. Let G be a connected graph with no odd cycles. Let x be a vertex of G. Let X be all vertices at an even distance from x, and let Y be all vertices at an odd distance from x. Prove that G is bipartite with parts X and Y.

15. What is the sum of the maximum size of an independent set and the minimum size of a vertex cover in a graph G? Hint: it is useful to think both about the independent set and its complement (relative to the vertex set).

6.5 Coloring and planarity

The idea of coloring

Graph coloring was one of the origins of graph theory. It arose from a question from Francis Guthrie, who noticed that that four colors were enough colors to color the map of the counties of England so that if two counties shared a common boundary line, then they got different colors. He wondered whether this was the case for any map. Through his brother he passed it on to Agustus DeMorgan, and in this way it seeped into the consciousness of the mathematical community. If we think of the counties as vertices and draw an edge between two vertices if their counties share some boundary line, we get a representation of the problem that is independent of such things as the shape of the counties, the amount of boundary line they share, etc. so that it captures the part of the problem we need to focus on. We now color the vertices of the graph, and for this problem we want to do so in such a way that adjacent vertices get different colors. We will return to this problem later in the section; we begin our study with another application of coloring.

Exercise 6.5-1 The executive committee of the board of trustees of a small college has seven members, Kim, Smith, Jones, Gupta, Ramirez, Wang, and Chernov. It has six subcommittees with the following membership

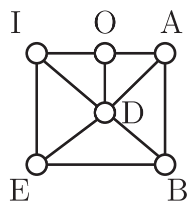
- Investments: W, R, G
- Operations: G, J, S, K
- Academic affairs: S, W, C
- Fund Raising: W, C, K
- Budget: G, R, C
- Enrollment: R, S, J, K

Each time the executive committee has a meeting, first each of the subcommittees meets with appropriate college officers, and then the executive committee gets together as a whole to go over subcommittee recommendations and make decisions. Two committees cannot meet at the same time if they have a member in common, but committees that don't have a member in common can meet at the same time. In this exercise you will figure out the minimum number of time slots needed to schedule all the subcommittee meetings. Draw a graph in which the vertices are named by the initials of the committee names and two vertices are adjacent if they have a member in common. Then assign numbers to the vertices in such a way that two adjacent vertices get different numbers. The numbers represent time slots, so they need not be distinct unless they are on adjacent vertices. What is the minimum possible number of numbers you need?

Because the problem of map coloring motivated much of graph theory, it is traditional to refer to the process of assigning labels to the vertices of a graph as coloring the graph. An assignment of labels, that is a function from the vertices to some set, is called a *coloring*. The set of possible labels (the range of the coloring function) is often referred to as a set of *colors*. Thus in Exercise 6.5-1 we are asking for a coloring of the graph. However, as with the map problem, we want a coloring in which adjacent vertices have different colors. A coloring of a graph is called a *proper coloring* if it assigns different colors to adjacent vertices.

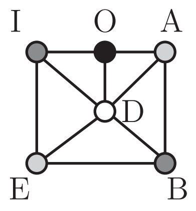
We have drawn the graph of Exercise 6.5-1 in Figure 6.29. We call this kind of graph an *intersection graph*, which means its vertices correspond to sets and it has an edge between two vertices if and only if the corresponding sets intersect.

Figure 6.29: The "intersection" graph of the committees.



The problem asked us to color the graph with as few colors possible, regarding the colors as 1,2,3, etc. We will represent 1 as a white vertex, 2 as a light grey vertex, 3 as a dark grey vertex and 4 as a black vertex. The triangle on the bottom requires three colors simply because all three vertices are adjacent. Since it doesn't matter which three colors we use, we choose arbitrarily to make them white, light grey, and dark grey. Now we know we need at least three colors to color the graph, so it makes sense to see if we can finish off a coloring using just three colors. Vertex I must be colored differently from E and D, so if we use the same three colors, it must have the same color as B. Similarly, vertex A would have to be the same color as E if we use the same three colors. But now none of the colors can be used on vertex O, because it is adjacent to three vertices of different colors. Thus we need at least four colors rather than 3, and we show a proper four-coloring in Figure 6.30.

Figure 6.30: A proper coloring of the committee intersection graph.



Exercise 6.5-2 How many colors are needed to give a proper coloring of the complete graph K_n?

Exercise 6.5-3 How many colors are needed for a proper coloring of a cycle C_n on $n = 3, 4, 5,$ and 6 vertices?

In Exercise 6.5-2 we need n colors to properly color K_n, because each pair of vertices is adjacent and thus must have two different colors. In Exercise 6.5-3, if n is even, we can just alternate two colors as we go around the cycle. However if n is odd, using two colors would require that they alternate as we go around the cycle, and when we colored our last vertex, it would be the same color as the first. Thus we need at least three colors, and by alternating two of them as we go around the cycle until we get to the last vertex and color it the third color we get a proper coloring with three colors.

The *chromatic number* of a graph G, traditionally denoted $\chi(G)$, is the minimum number of colors needed to properly color G. Thus we have shown that the chromatic number of the complete graph K_n is n, the chromatic number of a cycle on an even number of vertices is two, and the chromatic number of a cycle on an odd number of vertices is three. We showed that the chromatic number of our committee graph is 4.

From Exercise 6.5-2, we see that if a graph G has a subgraph which is a complete graph on n vertices, then we need at least n colors to color those vertices, so we need at least n colors to color G. this is useful enough that we will state it as a lemma.

Lemma 6.22 *If a graph G contains a subgraph that is a complete graph on n vertices, then the chromatic number of G is at least n.*

Proof: Given above.■

Interval Graphs

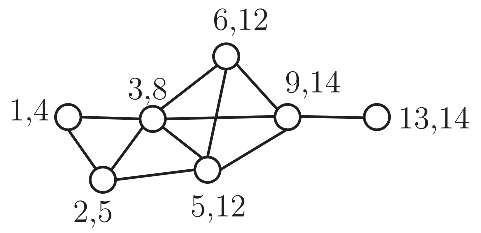
An interesting application of coloring arises in the design of optimizing compilers for computer languages. In addition to the usual RAM, a computer typically has some memory locations called registers which can be accessed at very high speeds. Thus values of variables which are going to be used again in the program are kept in registers if possible, so they will be quickly available when we need them. An optimizing compiler will attempt to decide the time interval in which a given variable may be used during a run of a program and arrange for that variable to be stored in a register for that entire interval of time. The time interval is not determined in absolute terms of seconds, but the relative endpoints of the intervals can be determined by when variables first appear and last appear as one steps through the computer code. This information is what is needed to set aside registers to use for the variables. We can think of coloring the variables by the registers as follows. We draw a graph in which the vertices are labeled with the variable names, and associated to each variable is the interval during which it is used. Two variables can use the same register if they are needed during non-overlapping time intervals. This is helpful, because registers are significantly more expensive than ordinary RAM, so they are limited in number. We can think of our graph on the variables as the intersection graph of the intervals. We want to color the graph properly with a minimum number of registers; hopefully this will be no more than the number of registers our computer has available. The problem of assigning variables to registers is called the *register assignment problem.*

An intersection graph of a set of intervals of real numbers is called an *interval graph.* The assignment of intervals to the vertices is called an *interval representation.* You will notice that so far in our discussion of coloring, we have not given an algorithm for properly coloring a graph efficiently. This is because the problem of whether a graph has a proper coloring with k colors, for any fixed k greater than 2 is another example of an **NP**-complete problem. However, for interval graphs, there is a very simple algorithm for properly coloring the graph in a minimum number of colors.

Exercise 6.5-4 Consider the closed intervals $[1, 4], [2, 5], [3, 8], [5, 12], [6, 12], [7, 14], [13, 14]$.
Draw the interval graph determined by these intervals and find its chromatic number.

We have drawn the graph of Exercise 6.5-4 in Figure 6.31. (We have not included the square braces to avoid cluttering the figure.) Because of the way we have drawn it, it is easy to see a

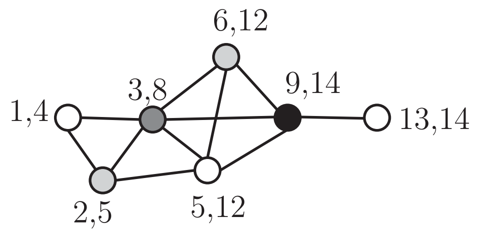
Figure 6.31: The graph of Exercise 6.5-4



subgraph that is a complete graph on four vertices, so we know by our lemma that the graph has chromatic number at least four. In fact, Figure 6.32 shows that the chromatic number is exactly four. This is no accident.

Figure 6.32: A proper coloring of the graph of Exercise 6.5-4 with four colors

.



Theorem 6.23 *In an interval graph G, the chromatic number is the size of the largest complete subgraph.*

Proof: List the intervals of an interval representation of the graph in order of their left endpoints. Then color them with the integers 1 through some number n by starting with 1 on the first interval in the list and for each succeeding interval, use the smallest color not used on any neighbor of the interval earlier in the list. This will clearly give a proper coloring. To see that the number of colors needed is the size of the largest complete subgraph, let n denote the largest color used, and choose an interval I colored with color n. Then, by our coloring algorithm, I must intersect with earlier intervals in the list colored 1 through $n-1$; otherwise we could have used a smaller color on I. All these intervals must contain the left endpoint of I, because they intersect I and come earlier in the list. Therefore they all have a point in common, so they form a complete graph on n vertices. Therefore the minimum number of colors needed is the size of a complete subgraph of G. But by Lemma 6.22, G can have no larger complete subgraph. Thus the chromatic number of G is the size of the largest complete subgraph of G. ■

Corollary 6.24 *An interval graph G may be properly colored using $\chi(G)$ consecutive integers as colors by listing the intervals of a representation in order of their left endpoints and going through* *the list, assigning the smallest color not used on an earlier adjacent interval to each interval in the list.*

Proof: This is the coloring algorithm we used in the proof of Theorem 6.23. ■

Notice that using the correspondence between numbers and grey-shades we used before, the coloring in Figure 6.32 is the one given by this algorithm. An algorithm that colors an (arbitrary) graph G with consecutive integers by listing its vertices in some order, coloring the first vertex in the list 1, and then coloring each vertex with the least number not used on any adjacent vertices earlier in the list is called a *greedy coloring algorithm.* We have just seen that the greedy coloring algorithm allows us to find the chromatic number of an interval graph. This algorithm takes time $O(n^2)$, because as we go through the list, we might consider every earlier entry when we are considering a given element of the list. It is good luck that we have a polynomial time algorithm, because even though we stated in Theorem 6.23 that the chromatic number is the size of the largest complete subgraph, determining whether the size of a largest complete subgraph in a general graph (as opposed to an interval graph) is k (where k may be a function of the number of vertices) is an **NP-complete** problem.

Of course we assumed that we were given an interval representation of our graph. Suppose we are given a graph that happens to be an interval graph, but we don't know an interval representation. Can we still color it quickly? It turns out that there is a polynomial time algorithm to determine whether a graph is an interval graph and find an interval representation. This theory is quite beautiful,[11] but it would take us too far afield to pursue it now.

Planarity

We began our discussion of coloring with the map coloring problem. This problem has a special aspect that we did not mention. A map is drawn on a piece of paper, or on a globe. Thus a map is drawn either on the plane or on the surface of a sphere. By thinking of the sphere as a completely elastic balloon, we can imagine puncturing it with a pin somewhere where nothing is drawn, and then stretching the pinhole until we have the surface of the balloon laid out flat on a table. This means we can think of all maps as drawn in the plane. What does this mean about the graphs we associated with the maps? Say, to be specific, that we are talking about the counties of England. Then in each county we take an important town, and build a road to the boundary of each county with which it shares more than a single boundary point. We can build these roads so that they don't cross each other, and the roads to a boundary line between two different counties join together at that boundary line. Then the towns we chose are the vertices of a graph representing the map, and the roads are the edges. Thus given a map drawn in the plane, we can draw a graph to represent it in such a way that the edges of the graph do not meet at any point except their endpoints.[12] A graph is called *planar* if it has a drawing in the plane such that edges do not meet except at their endpoints. Such a drawing is called a *planar drawing* of the graph. The famous four color problem asked whether all planar graphs have proper four colorings. In 1976, Apel and Haken, building on some of the early attempts at proving the theorem, used a computer to demonstrate that four colors are sufficient to color

[11]See, for example, the book *Algorithmic Graph Theory and the Perfect Graph Conjecture*, by Martin Golumbic, Academic Press, New York, 1980.

[12]We are temporarily ignoring a small geographic feature of counties that we will mention when we have the terminology to describe it

any planar graph. While we do not have time to indicate how their proof went, there is now a book on the subject that gives a careful history of the problem and an explanation of what the computer was asked to do and why, assuming that the computer was correctly programmed, that led to a proof.[13]

What we will do here is derive enough information about planar graphs to show that five colors suffice, giving the student some background on planarity relevant to the design of computer chips.

We start out with two problems that aren't quite realistic, but are suggestive of how planarity enters chip design.

Exercise 6.5-5 A circuit is to be laid out on a computer chip in a single layer. The design includes five terminals (think of them as points to which multiple electrical circuits may be connected) that need to be connected so that it is possible for a current to go from any one of them to any other without sending current to a third. The connections are made with a narrow layer of metal deposited on the surface of the chip, which we will think of as a wire on the surface of the chip. Thus if one connection crosses another one, current in one wire will flow through the other as well. Thus the chip must be designed so that no two wires cross. Do you think this is possible?

Exercise 6.5-6 As in the previous exercise, we are laying out a computer circuit. However we now have six terminals, labeled a, b, c, 1, 2, and 3, such that each of a, b, and c must be connected to each of 1, 2, and 3, but there must be no other connections. As before, the wires cannot touch each other, so we need to design this chip so that no two wires cross. Do you think this is possible?

The answer to both these exercises is that it is not possible to design such a chip. One can make compelling geometric arguments why it is not possible, but they require that we visualize simultaneously a large variety of configurations with one picture. We will instead develop a few equations and inequalities relating to planar graphs that will allow us to give convincing arguments that both these designs are impossible.

The Faces of a Planar Drawing

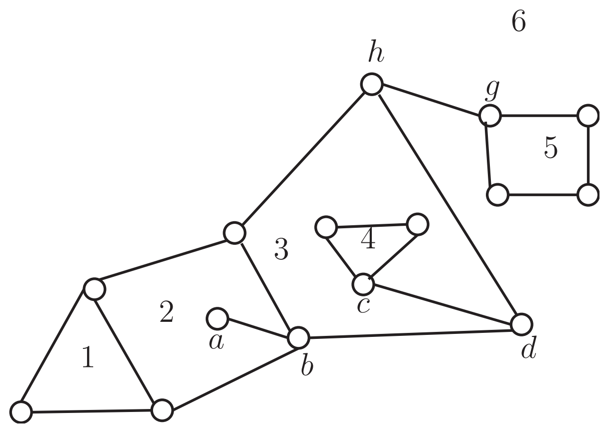
If we assume our graphs are finite, then it is easy to believe that we can draw any edge of a graph as a broken line segment (i.e. a bunch of line segments connected at their ends) rather than a smooth curve. In this way a cycle in our graph determines a polygon in our drawing. This polygon may have some of the graph drawn inside it and some of the graph drawn outside it. We say a subset of the plane is *geometrically connected* if between any two points of the region we can draw a curve.[14] (In our context, you may assume this curve is a broken line segment, but a careful study of geometric connectivity in general situations is less straightforward.) If we remove all the vertices and edges of the graph from the plane, we are likely to break it up into a number of connected sets.

Such a connected set is called a *face* of the drawing if it not a proper subset of any other connected set of the plane with the drawing removed. For example, in Figure 6.33 the faces are

[13]Robin Wilson, *Four Colors Suffice.* Princeton University Press, Princeton NJ 2003.

[14]The usual thing to say is that it is connected, but we want to distinguish this kind of connectivity form graphical connectivity. The fine point about counties that we didn't point out earlier is that they are geometrically connected. If they were not, the graph with a vertex for each county and an edge between two counties that share some boundary line would not necessarily be planar.

Figure 6.33: A typical graph and its faces.



marked 1, a triangular face, 2, a quadrilateral face that has a line segment and point removed for the edge $\{a, b\}$ and the vertex z, 3, another quadrilateral that now has not only a line but a triangle removed from it as well, 4, a triangular face, 5, a quadrilateral face, and 6 a face whose boundary is a heptagon connected by a line segment to a quadrilateral. Face 6 is called the "outside face" of the drawing and is the only face with infinite area. Each planar drawing of a graph will have an *outside face*, that is a face of infinite area in which we can draw a circle that encloses the entire graph. (Remember, we are thinking of our graphs as finite at this point.) Each edge either lies between two faces or has the same face on both its sides.The edges $\{a, b\}$, $\{c, d\}$ and $\{g, h\}$ are the edges of the second type. Thus if an edge lies on a cycle, it must divide two faces; otherwise removing that edge would increase the number of connected components of the graph. Such an edge is called a *cut edge* and cannot lie between two distinct faces. It is straightforward to show that any edge that is not a cut edge lies on a cycle. But if an edge lies on only one face, it is a cut edge, because we can draw a broken line segment from one side of the edge to the other, and this broken line segment plus part of the edge forms a closed curve that encloses part of the graph. Thus removing the edge disconnects the enclosed part of the graph from the rest of the graph.

Exercise 6.5-7 Draw some planar graphs with at least three faces and experiment to see if you can find a numerical relationship among v, the number of vertices, e, the number of edges, and f the number of faces. Check your relationship on the graph in Figure 6.33.

Exercise 6.5-8 In a simple graph, every face has at least three edges. This means that the number of pairs of a face and an edge bordering that face is at least $3f$. Use the fact that an edge borders either one or two faces to get an inequality relating the number of edges and the number of faces in a simple planar graph.

Some playing with planar drawings usually convinces people fairly quickly of the following theorem known as *Euler's Formula.*

Theorem 6.25 (Euler) *In a planar drawing of a graph G with v vertices, e edges, and f faces,*

$$v - e + f = 2.$$

Proof: We induct on the number of cycles of G. If G has no cycles, it is a tree, and a tree has one face because all its edges are cut-edges. Then $v - e + f = v - (v - 1) + 1 = 2$. Now suppose G has $n > 0$ cycles. Choose an edge which is between two faces, so it is part of a cycle. Deleting that edge joins the two faces it was on together, so the new graph has $f' = f - 1$ faces. The new graph has the same number of vertices and one less edge. It also has fewer cycles than G, so we have $v - (e - 1) - (f - 1) = 2$ by the inductive hypothesis, and this gives us $v - e + f = 2$. ■

ForExercise 6.5-8 let's define an edge-face pair to be an edge and a face such that the edge borders the face. Then we said that the number of such pairs is at least $3f$ in a simple graph. Since each edge is in either one or two faces, the number of edge-face pairs is also no more than $2e$. This gives us

$$3f \leq \# \text{ of edge-face pairs} \leq 2e,$$

or $3f \leq 2e$, so that $f \leq \frac{2}{3}e$ in a planar drawing of a graph. We can combine this with Theorem 6.25 to get

$$2 = v - e + f \leq v - e + \frac{2}{3}e = v - e/3$$

which we can rewrite as

$$e \leq 3v - 6$$

in a planar graph.

Corollary 6.26 *In a simple planar graph,* $e \leq 3v - 6$.

Proof: Given above.■

In our discussion of Exercise 6.5-5 we said that we would see a simple proof that the circuit layout problem was impossible. Notice that the question in that exercise was really the question of whether the complete graph on 5 vertices, K_5, is planar. If it were, the inequality $e \leq 3v - 6$ would give us $10 \leq 3 \cdot 5 - 6 = 9$, which is impossible, so K_5 can't be planar. The inequality of Corollary 6.26 is not strong enough to solve Exercise 6.5-6. This exercise is really asking whether the so-called "complete bipartite graph on two parts of size 3," denoted by $K_{3,3}$, is planar. In order to show that it isn't, we need to refine the inequality of Corollary 6.26 to take into account the fact that in a simple bipartite graph there are no cycles of size 3, so there are no faces that are bordered by just 3 edges. You are asked to do that in Problem 13.

Exercise 6.5-9 Prove or give a counter-example: Every planar graph has at least one vertex of degree 5 or less.

Exercise 6.5-10 Prove that every planar graph has a proper coloring with six colors.

In Exercise 6.5-9 suppose that G is a planar graph in which each vertex has degree six or more. Then the sum of the degrees of the vertices is at least 6v, and also is twice the number of edges. Thus $2e \geq 6v$, or $e \geq 3v$, contrary to $e \leq 3v - 6$. This gives us yet another corollary to Euler's formula.

Corollary 6.27 *Every planar graph has a vertex of degree 5 or less.*

Proof: Given above.■

The Five Color Theorem

We are now in a position to give a proof of the five color theorem, essentially Heawood's proof, which was based on his analysis of an incorrect proof given by Kempe to the four color theorem about ten years earlier in 1879. First we observe that in Exercise 6.5-10 we can use straightforward induction to show that any planar graph on n vertices can be properly colored in six colors. As a base step, the theorem is clearly true if the graph has six or fewer vertices. So now assume $n > 6$ and suppose that a graph with fewer than n vertices can be properly colored with six colors. Let x be a vertex of degree 5 or less. Deleting x gives us a planar graph on $n - 1$ vertices, so by the inductive hypothesis it can be properly colored with six colors. However only five or fewer of those colors can appear on vertices which were originally neighbors of x, because x had degree 5 or less. Thus we can replace x in the colored graph and there is at least one color not used on its neighbors. We use such a color on x and we have a proper coloring of G. Therefore, by the principle of mathematical induction, every planar graph on $n \geq 1$ vertices has a proper coloring with six colors.

To prove the five color theorem, we make a similar start. However, it is possible that after deleting x and using an inductive hypothesis to say that the resulting graph has a proper coloring with 5 colors, when we want to restore x into the graph, five distinct colors are already used on its neighbors. This is where the proof will become interesting.

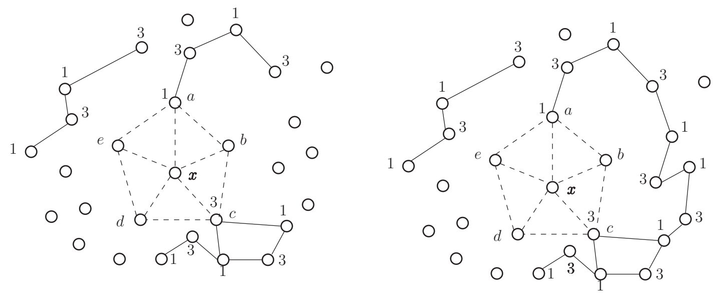
Theorem 6.28 *A planar graph G has a proper coloring with at most 5 colors.*

Proof: We may assume that every face except perhaps the outside face of our drawing is a triangle for two reasons. First, if we have a planar drawing with a face that is not a triangle, we can draw in additional edges going through that face until it has been divided into triangles, and the graph will remain planar. Second, if we can prove the theorem for graphs whose faces are all triangles, then we can obtain graphs with non-triangular faces by removing edges from graphs with triangular faces, and a proper coloring remains proper if we remove an edge from our graph. Although this appears to muddy the argument at this point, at a crucial point it makes it possible to give an argument that is clearer than it would otherwise be.

Our proof is by induction on the number of vertices of the graph. If G has five or fewer vertices then it is clearly properly colorable with five or fewer colors. Suppose G has n vertices and suppose inductively that every planar graph with fewer than n vertices is properly colorable with five colors. G has a vertex x of degree 5 or less. Let G' be the graph obtained by deleting x form G. By the inductive hypothesis, G' has a coloring with five or fewer colors. Fix such a coloring. Now if x has degree four or less, or if x has degree 5 but is adjacent to vertices colored with just four colors in G', then we may replace x in G' to get G and we have a color available to use on x to get a proper coloring of G.

Thus we may assume that x has degree 5, and that in G' five different colors appear on the vertices that are neighbors of x in G. Color all the vertices of G other than x as in G'. Let the five vertices adjacent to x be a, b, c, d, e in clockwise order, and assume they are colored with colors 1, 2, 3, 4, and 5. Further, by our assumption that all faces are triangles, we have that $\{a, b\}$, $\{b, c\}$4,{c,d},$\{d, e\}$, and $\{e, a\}$ are all edges, so that we have a pentagonal cycle surrounding x. Consider the graph $G_{1,3}$ of G which has the same vertex set as G but has only edges with endpoints colored 1 and 3. (Some possibilities are shown in Figure 6.34. We show only edges connecting vertices colored 1 and 3, as well as dashed lines for the edges from x to its neighbors and the edges between successive neighbors. There may be many more vertices and edges in G.)

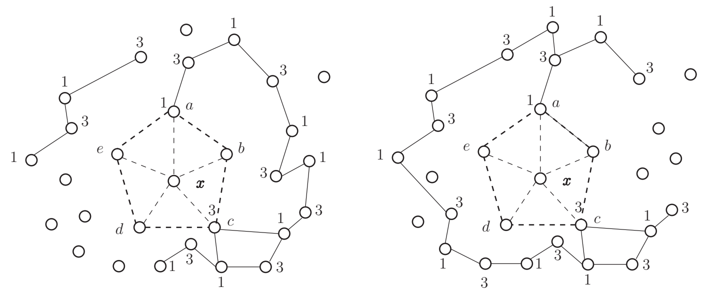
Figure 6.34: Some possibilities for the graph $G_{1,3}$.



The graph $G_{1,3}$ will have a number of connected components. If a and c are not in the same component, then we may exchange the colors on the vertices of the component containing a without affecting the color on c. In this way we obtain a coloring of G with only four colors, 3,2,3,4,5 on the vertices a, b, c, d, e. We may then use the fifth color (in this case 1) on vertex x and we have properly colored G with five colors.

Otherwise, as in the second part of Figure 6.34, since a and c are in the same component of $G_{1,3}$, there is a path from a to c consisting entirely of vertices colored 1 and 3. Now temporarily color x with a new color, say color 6. Then in G we have a cycle C of vertices colored 1, 3, and 6. This cycle has an inside and an outside. Part of the graph can be on the inside of C, and part can be on the outside. In Figure 6.35 we show two cases for how the cycle could occur, one in

Figure 6.35: Possible cycles in the graph $G_{1,3}$.



which vertex b is inside the cycle C and one in which it is outside C. (Notice also that in both cases, we have more than one choice for the cycle because there are two ways in which we could use the quadrilateral at the bottom of the figure.)

In G we also have the cycle with vertex sequence a, b, c, d, e which is colored with five different colors. This cycle and the cycle C can intersect only in the vertices a and c. Thus these two cycles divide the plane into four regions: the one inside both cycles, the one outside both cycles, and the two regions inside one cycle but not the other. If b is inside C, then the area inside both cycles is bounded by the cycle $a\{a,b\}b\{b,c\}c\{c,x\}x\{x,a\}a$. Therefore e and d are not inside the cycle

C. If one of d and e is inside C, then both are (because the edge between them cannot cross the cycle) and the boundary of the region inside both cycles is $a\{a,e\}e\{e,d\}d\{d,c\}c\{c,x\}x\{x,a\}a$. In this case b cannot be inside C. Therefore one of b and d is inside the cycle c and one is outside it. Therefore if we look at the graph $G_{2,4}$ with the same vertex set as G and just the edges connecting vertices colored 2 and 4, the connected component containing b and the connected component containing d must be different, because otherwise a path of vertices colored 2 and 4 would have to cross the cycle C colored with colors 1, 3, and 6. Therefore in G' we may exchange the colors 2 and 4 in the component containing d, and we now have only colors 1, 2, 3, and 5 used on vertices a, b, c, d, and e. Therefore we may use this coloring of G' as the coloring for the vertices of G different from x and we may change the color on x from 6 to 4, and we have a proper five coloring of G. Therefore by the principle of mathematical induction, every finite planar graph has a proper coloring with 5 colors. ■

Kempe's argument that seemed to prove the four color theorem was similar to this, though where we had five distinct colors on the neighbors of x and sought to remove one of them, he had four distinct colors on the five neighbors of x and sought to remove one of them. He had a more complicated argument involving two cycles in place of our cycle C, and he missed one of the ways in which these two cycles can interact.

Important Concepts, Formulas, and Theorems

1. *Graph Coloring.* An assignment of labels to the vertices of a graph, that is a function from the vertices to some set, is called a *coloring* of the graph. The set of possible labels (the range of the coloring function) is often referred to as a set of *colors.*

2. *Proper Coloring.* A coloring of a graph is called a *proper coloring* if it assigns different colors to adjacent vertices.

3. *Intersection Graph.* We call a graph an *intersection graph* if its vertices correspond to sets and it has an edge between two vertices if and only if the corresponding sets intersect.

4. *Chromatic Number.* The *chromatic number* of a graph G, traditionally denoted $\chi(G)$, is the minimum number of colors needed to properly color G.

5. *Complete Subgraphs and Chromatic Numbers.* If a graph G contains a subgraph that is a complete graph on n vertices, then the chromatic number of G is at least n.

6. *Interval Graph.* An intersection graph of a set of intervals of real numbers is called an *interval graph.* The assignment of intervals to the vertices is called an *interval representation.*

7. *Chromatic Number of an Interval Graph.* In an interval graph G, the chromatic number is the size of the largest complete subgraph.

8. *Algorithm to Compute the Chromatic number and a proper coloring of an Interval Graph.* An interval graph G may be properly colored using $\chi(G)$ consecutive integers as colors by listing the intervals of a representation in order of their left endpoints and going through the list, assigning the smallest color not used on an earlier adjacent interval to each interval in the list.

9. *Planar Graph and Planar Drawing.* A graph is called *planar* if it has a drawing in the plane such that edges do not meet except at their endpoints. Such a drawing is called a *planar drawing* of the graph.

10. *Face of a Planar Drawing.* A geometrically connected connected subset of the plane with the vertices and edges of a planar graph taken away is called a *face* of the drawing if it not a proper subset of any other connected set of the plane with the drawing removed.

11. *Cut Edge.* An edge whose removal from a graph increases the number of connected components is called a *cut edge* of the graph. A cut edge of a planar graph lies on only one face of a planar drawing.

12. *Euler's Formula.* Euler's formula states that in a planar drawing of a graph with v vertices, e edges and f faces, $v - e + f = 2$. As a consequence, in a planar graph, $e \leq 3v - 6$.

Problems

1. What is the minimum number of colors needed to properly color a path on n vertices if $n > 1$?

2. What is the minimum number of colors needed to properly color a bipartite graph with parts X and Y.

3. If a graph has chromatic number two, is it bipartite? Why or why not?

4. Prove that the chromatic number of a graph G is the maximum of the chromatic numbers of its components.

5. A *wheel* on n vertices consists of a cycle on $n - 1$ vertices together with one more vertex, normally drawn inside the cycle, which is connected to every vertex of the cycle. What is the chromatic number of a wheel on 5 vertices? What is the chromatic number of a wheel on an odd number of vertices?

6. A *wheel* on n vertices consists of a cycle on $n - 1$ vertices together with one more vertex, normally drawn inside the cycle, which is connected to every vertex of the cycle. What is the chromatic number of a wheel on 6 vertices? What is the chromatic number of a wheel on an even number of vertices?

7. The usual symbol for the maximum degree of any vertex in a graph is Δ. Show that the chromatic number of a graph is no more than $\Delta + 1$. (In fact Brooks proved that if G is not complete or an odd cycle, then $\chi(G) \leq \Delta$. Though there are now many proofs of this fact, none are easy!)

8. Can an interval graph contain a cycle with four vertices and no other edges between vertices of the cycle?

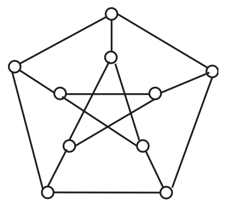
9. The Petersen graph is in Figure 6.36. What is its chromatic number?

10. Let G consist of a five cycle and a complete graph on four vertices, with all vertices of the five-cycle joined to all vertices of the complete graph. What is the chromatic number of G?

11. In how many ways can we properly color a tree on n vertices with t colors?

12. In how many ways may we properly color a complete graph on n vertices with t colors?

Figure 6.36: The Petersen Graph.



13. Show that in a simple planar graph with no triangles, $e \leq 2v - 4$.

14. Show that in a simple bipartite planar graph, $e \leq 2v - 4$, and use that fact to prove that $K_{3,3}$ is not planar.

15. Show that in a planar graph with no triangles there is a vertex of degree three or less.

16. Show that if a planar graph has fewer than twelve vertices, then it has at least one vertex of degree 4.

17. The Petersen Graph is in Figure 6.36. What is the size of the smallest cycle in the Petersen Graph? Is the Petersen Graph planar?

18. Prove the following Theorem of Welsh and Powell. If a graph G has degree sequence $d_1 \geq d_2 \geq \cdots \geq d_n$, then $\chi(G) \leq 1 + max_i[min(d_i, i-1)]$. (That is the maximum over all i of the minimum of d_i and $i-1$.)

19. What upper bounds do Problem 18 and Problem 7 and the Brooks bound in Problem 7 give you for the chromatic number in Problem 10. Which comes closest to the right value? How close?

Index