Discrete Math in Computer Science

Ken Bogart
Dept. of Mathematics
Dartmouth College

Cliff Stein
Dept. of Computer Science
Dartmouth College

June 23, 2002

This is a working draft of a textbook for a discrete mathematics course. This course is designed to be taken by computer science students. The prerequisites are first semester calculus (Math 3) and the introductory computer science course (CS 5). The class is meant to be taken concurrently with or after the second computer science course, Data Structures and Computer Programming (CS 15). This class is a prerequite to Algorithms (CS 25) and it is recommended that it be taken before all CS courses other than 5 and 15.

Chapter 1

Counting

1.1 Basic Counting

About the course

In these notes, student activities alternate with explanations and extensions of the point of the activities. The best way to use these notes is to try to master the student activity before beginning the explanation that follows. The activities are largely meant to be done in groups in class; thus for activities done out of class we recommend trying to form a group of students to work together. The reason that the class and these notes are designed in this way is to help students develop their own habits of mathematical thought. There is considerable evidence that students who are actively discovering what they are learning remember it far longer and are more likely to be able to use it out of the context in which it was learned. Students are much more likely to ask questions until they understand a subject when they are working in a small group with peers rather than in a larger class with an instructor. There is also evidence that explaining ideas to someone else helps us organize these ideas in our own minds. However, different people learn differently. Also the amount of material in discrete mathematics that is desirable for computer science students to learn is much more than can be covered in an academic term if all learning is to be done through small group interaction. For these reasons about half of each section of these notes is devoted to student activities, and half to explanation and extension of the lessons of these activities.

Analyzing loops

1.1-1 The loop below is part of an implementation of selection sort to sort a list of items chosen from an ordered set (numbers, alphabet characters, words, etc.) into increasing order.

```
for i = 1 to n
     for j = i + 1 to n
          if (A(i) > A(j))
               exchange A(i) and A(j)
```

How many times is the comparison A(i) > A(j) made?

The Sum Principle

In Exercise 1.1-1, the segment of code

```
for j = i + 1 to n
     if (A(i) > A(j))
          exchange A(i) and A(j)
```

is executed n times, once for each value of i between 1 and n inclusive. The first time, it makes $n-1$ comparisons. The second time, it makes $n-2$ comparisons. The ith time, it makes $n-i$ comparisons. Thus the total number of comparisons is

$$(n-1)+(n-2)+\cdots+1+0.$$

The formula we have is not so important as the reasoning that lead us to it. In order to put the reasoning into a format that will allow us to apply it broadly, we will describe what we were doing in the language of sets. Think about the set of all comparisons the algorithm in Exercise 1.1-1 makes. We divided that set up into n pieces (i.e. smaller sets), the set S_1 of comparisons made when $i=1$, the set S_2 of comparisons made when $i=2$, and so on through the set S_n of comparisons made when $i=n$. We were able to figure out the number of comparisons in each of these pieces by observation, and added together the sizes of all the pieces in order to get the size of the set of all comparisons.

A little bit of set theoretic terminology will help us describe a general version of the process we used. Two sets are called *disjoint* when they have no elements in common. Each of the pieces we described above is disjoint from each of the others, because the comparisons we make for one value of i are different from those we make with another value of i. We say the set of pieces is a family of *mutually disjoint sets*, meaning that it is a family (set) of sets, each two of which are disjoint. With this language, we can state a general principle that explains what we were doing without making any specific reference to the problem we were solving.

The **sum principle** says:

> The size of a union of a family of mutually disjoint sets is the sum of the sizes of the sets.

Thus we were, in effect, using the sum principle to solve Exercise 1.1-1. There is an algebraic notation that we can use to describe the sum principle. For this purpose, we use $|S|$ to stand for the size of the set S. For example, $|\{a,b,c\}|=3$. Using this notation, we can state the sum principle as: if S_1, S_2, $\ldots$ S_n are disjoint sets, then

$$|S_1 \cup S_2 \cup \cdots \cup S_n| = |S_1| + |S_2| + \cdots + |S_n|.$$

To write this without the "dots" that indicate left-out material, we write

$$|\bigcup_{i=1}^{n} S_i| = \sum_{i=1}^{n} |S_i|.$$

The process of figuring out a general principle that "explains" why a certain computation makes sense is an example of the mathematical process of "abstraction." We won't try to give a precise definition of abstraction but rather point out examples of the process as we proceed. In a course in set theory, we would further abstract our work and derive the sum principle from certain principles that we would take as the axioms of set theory. In a course in discrete mathematics, this level of abstraction is unnecessary, so from now on we will simply use the sum principle as the basis of computations when it is convenient to do so. It may seem as though our abstraction was simply a mindless exercise that complicates what was an "obvious" solution to Exercise 1.1-1. If we were only working on this one exercise, that would be the case. However the principle we have derived will prove to be useful in a wide variety of problems. This is the value of abstraction. When you can recognize the abstract elements of a problem that helped you solve it in another problem, then abstraction often helps you solve that second problem as well.

There is a formula you may know for the sum

$$(n-1)+(n-2)+\cdots 1+0,$$

which we also write as

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

Now, if we don't like to deal with summing the values of $(n-i)$, we can observe that the set of values we are summing is $n-1, n-2, \ldots, 1$, so we may write that

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

There is a clever trick, usually attributed to Gauss, that gives us the formula for this sum.

We write

$$\begin{array}{cccccccccccc} & 1 & + & 2 & + & \cdots & + & n-2 & + & n-1 \\ + & n-1 & + & n-2 & + & \cdots & + & 2 & + & 1 \\ \hline & n & + & n & + & \cdots & + & n & + & n \end{array}$$

The sum below the horizontal line has $n-1$ terms each equal to n, and so it is $n(n-1)$. It is the sum of the two sums above the line, and since these sums are equal (being identical except for being in reverse order), the sum below the line must be twice either sum above, so either of the sums above must be $n(n-1)/2$. In other words, we may write

$$\sum_{i=1}^{n} n-i = \sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}.$$

While this is a lovely trick, learning tricks gives us little or no real mathematical skill; it is learning how to think about things to discover answers that is useful. After we analyze Exercise 1.1-2 and abstract the process we are using there, we will be able to come back to this problem and see a way that we could have discovered this formula for ourselves without any tricks.

The Product Principle

1.1-2 The loop below is part of a program in which the product of two matrices is computed. (You don't need to know what the product of two matrices is to answer this question.)

```
for i = 1 to r
     for j = 1 to m
          S = 0
          for k = 1 to n
               S = S + A[i, k] * B[k, j]
          C[i, j] = S
```

How many multiplications does this code carry out as a function of r, m, and n?

1.1-3 How many comparisons does the following code make?

```
for i = 1 to n - 1
     minval = A[i]
     minindex = i
     for j = i to n
          if (A[j] < A[i])
               minval = A[j]
               minindex = j
     exchange A[i] and A[minindex]

for i = 2 to n
     if (A[i] < 2 * A[i - 1])
          bigjump = bigjump + 1
```

In Exercise 1.1-2, the program segment

```
for k = 1 to n
     S = S + A[i, k] * B[k, j],
```

which we call the "inner loop," takes exactly n steps, and thus makes n multiplications, regardless of what the variables i and j are. The program segment

```
for j = 1 to m
     S = 0
     for k = 1 to n
          S = S + A[i, k] * B[k, j]
     C[i, j] = S
```

repeats the inner loop exactly m times, regardless of what i is. Thus this program segment makes n multiplications m times, so it makes nm multiplications.

A natural question to ask in light of our solution to Exercise 1.1-1 is why we added in Exercise 1.1-1 and multiplied here. Let's look at this problem from the abstract point of view we adopted in discussing Exercise 1.1-1. Our algorithm carries out a certain set of multiplications. For any given i, the set of multiplications carried out by the program segment we are analyzing can be divided into the set S_1 of multiplications carried out when $j = 1$, the set S_2 of multiplications carried out when $j = 2$, and, in general, the set S_j of multiplications carried out for any given j value. The set S_j consists of those multiplications the inner loop carries out for a particular value of j, and there are exactly n multiplications in this set. The set T_i of multiplications that our program segment carries out for a certain i value is the union of the sets S_j; stated as an equation,

$$T_i = \bigcup_{j=1}^{m} S_j.$$

Then, by the sum principle, the size of the set T_i is the sum of the sizes of the sets S_k, and a sum of m numbers, each equal to n is mn. Stated as an equation,

$$|T_i| = |\bigcup_{j=1}^{m} S_j| = \sum_{j=1}^{m} |S_j| = \sum_{j=1}^{m} n = mn.$$

Thus we are multiplying because multiplication is repeated addition!

From our solution we can extract a second principle that simply shortcuts the use of the sum principle. The **product principle** states:

> The size of a union of m disjoint sets, all of size n, is mn.

We still need to complete our discussion of Exercise 1.1-2. The program segment we just studied is used once for each value of i from 1 to r. Each time it is executed, it is executed with a different i value, so the set of multiplications in one execution is disjoint from the set of multiplications in any other execution. Thus the set of all multiplications our program carries out is a union of r disjoint sets T_i of mn multiplications each. Then by the product principle, the set of all multiplications has size rmn, so our program carries out rmn multiplications.

Exercise 1.1-3 is intended to show you how thinking about whether the sum or product principle is appropriate for a problem can help you decompose the problem into pieces you can solve. If you can decompose it and solve the smaller pieces, then you either add or multiply solutions to solve the larger problem. In this exercise, it is clear that the number of comparisons in the program fragment is the sum of the number of comparisons in the first loop with the number of comparisons in the second loop (what two disjoint sets are we talking about here?), that the first loop has $n(n+1)/2-1$ comparison, and that the second loop has $n-1$ comparisons, so the fragment makes $n(n+1)/2-1+n-1 = n(n+1)/2+n-2$ comparisons.

1.1-4 A password for a certain computer system is supposed to be between 4 and 8 characters long and composed of lower and upper case letters. How many passwords are possible? What counting principles did you use? Estimate the percentage of the possible passwords with four characters.

Here we use the sum principle to divide our problem into computing the number of passwords with four letters, the number with five letters, the number with six letters, the number with seven letters, and the number with 8 letters. For an i-letter password, there are 52 choices for the first letter, 52 choices for the second and so on. Thus by the product principle the number of passwords with i letters is 52^i. Therefore the total number of passwords is

$$52^4 + 52^5 + 52^6 + 52^7 + 52^8.$$

Of these, 52^4 have four letters, so the percentage with 54 letters is

$$\frac{100 \cdot 52^4}{52^4 + 52^5 + 52^6 + 52^7 + 52^8.}$$

Now this is a nasty formula to evaluate, but we can get a quite good estimate as follows. Notice that 52^8 is 52 times as big as 52^7, and even more dramatically larger than any other term in the sum in the denominator. Thus the ratio is approximately

$$\frac{100 \cdot 52^4}{52^8,}$$

which is $100/52^4$, or approximately .000014. In other words only .000014% of the passwords have four letters. This suggests how much easier it would be to guess a password if we knew it had four letters than if we just knew it had between 4 and 8 letters – it is roughly 7 millions times easier!

Notice how in our solution to Exercise 1.1-4 we casually referred to the use of the product principle in computing the number of passwords with i letters. We didn't write any set as a union of sets of equal size. Though we could have, it would be clumsy. For this reason we will state a second version of the product principle that is straightforward (but pedantic) to derive from the version for unions of sets.

Version 2 of the *product principle* states:

> If a set S of lists of length m has the properties that
>
> 1. There are i_1 different first elements of lists in S, and
>
> 2. For each $j > 1$ and each choice of the first $j-1$ elements of a list in S there are i_j choices of elements in position j of that list, then
>
> there are $i_1 i_2 \cdots i_k$ lists in S.

Since an i-letter password is just a list of i letters, and since there are 52 different first elements of the password and 52 choices for each other position of the password, this version of the product principle tells us immediately that the number of passwords of length i is 52^i.

With Version 2 of the product principle in hand, let us examine Exercise 1.1-1 again. Notice that for each two numbers i and j, we compare $A(i)$ and $A(j)$ exactly once in our loop. (The order in which we compare them depends on which one is smaller.) Thus the number of comparisons we make is the same as the number of two element subsets of the set $\{1, 2, \ldots, n\}$. In how many ways can we choose two elements from this set? There are n ways to choose a first element, and for each choice of the first element, there are $n-1$ ways to choose a second element. Thus it might appear that there are $n(n-1)$ ways to choose two elements from our set. However, what

we have chosen is an *ordered pair*, namely a pair of elements in which one comes first and the other comes second. For example, we could choose two first and five second to get the ordered pair $(2, 5)$, or we could choose five first and two second to get the ordered pair $(5, 2)$. Since each pair of distinct elements of $\{1, 2, \ldots, n\}$ can be listed in two ways, we get twice as many ordered pairs as two element sets. Thus, since the number of ordered pairs is $n(n-1)$, the number of two element subsets of $\{1, 2, \ldots, n\}$ is $n(n-1)/2$. This number comes up so often that it has its own name and notation. We call this number "n choose 2" and denote it by $\binom{n}{2}$. To summarize, $\binom{n}{2}$ stands for the number of two element subsets of an n element set and equals $n(n-1)/2$. Since one answer to Exercise 1.1-1 is $1 + 2 + \cdots + n - 1$ and a second answer to Exercise 1.1-1 is $\binom{n}{2}$, this shows that

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

Problems

1. The segment of code below is part of a program that uses insertion sort to sort a list A

   ```
   for i = 2 to n
        j=i
        while j ≥ 2 and A(j) < A(j − 1)
             exchange A(j) and A(j − 1)
             j − −
   ```

   What is the maximum number of times (considering all lists of n items you could be asked to sort) the program makes the comparison $A(i) < A(i-1)$? Describe as succinctly as you can those lists that require this number of comparisons.

2. In how many ways can you draw a first card and then a second card from a deck of 52 cards?

3. In how many ways may you draw a first, second, and third card from a deck of 52 cards?

4. Suppose that on day 1 you receive 1 penny, and, for $i > 1$, on day i you receive twice as many pennies as you did on day $i - 1$. How many pennies will you have on day 20? How many will you have on day n? Did you use the sum or product principal?

5. The "Pile High Deli" offers a "simple sandwich" consisting of your choice of one of five different kinds of bread with your choice of butter or mayonnaise or no spread, one of three different kinds of meat, and one of three different kinds of cheese, with the meat and cheese "piled high" on the bread. In how many ways may you choose a simple sandwich?

6. What is the number of ten digit (base ten) numbers? What is the number of ten digit numbers that have no two consecutive digits equal? What is the number that have at least one pair of consecutive digits equal?

7. We are making a list of participants in a panel discussion on allowing alcohol on campus. They will be sitting behind a table in the order in which we list them. There will be four administrators and four students. In how many ways may we list them if the administrators must sit together in a group and the students must sit together in a group? In how many ways may we list them if we must alternate students and administrators?

8. In the local ice cream shop, there are 10 different flavors. How many different two-scoop cones are there? (Following your mother's rule that it all goes to the same stomach, a cone with a vanilla scoop on top of a chocolate scoop is considered the same as a cone with a a chocolate scoop on top of a vanilla scoop.)

9. Now suppose that you decide to disagree with your mother in Exercise 8 and say that the order of the scoops does matter. How many different possible two-scoop cones are there?

10. In the local ice cream shop, you may get a sundae with two scoops of ice cream from 10 flavors (using your mother's rule from Exercise 8), any one of three flavors of topping, and any (or all or none) of whipped cream, nuts and a cherry. How many different sundaes are possible? (Note that the the way the scoops sit in the dish is not significant).

11. In the local ice cream shop, you may get a three-way sundae with three of the ten flavors of ice cream, any one of three flavors of topping, and any (or all or none) of whipped cream, nuts and a cherry. How many different sundaes are possible? Note that, according to your mother's rule, the way the scoops sit in the dish does not matter.

12. The idea of a function from the real numbers to the real numbers is quite familiar in calculus. A *function* f from a set S to a set T is a relationship between S and T that relates exactly one element of T to each element of S. We write $f(x)$ for the one and only one element of T that the function F relates to the element x of S. There are more functions from the real numbers to the real numbers than most of us can imagine. However in discrete mathematics we often work with functions from a finite set S with s elements to a finite set T with t elements. Then there are only a finite number of functions from S to T. How many functions are there from S to T in this case?

13. The word permutation is used in two different ways in mathematical circles.

    a A *k-element permutation* of a set N is usually defined as a list of k distinct elements of N. If N has n elements, how many k-element permutations does it have? Once you have your answer, find a way to express it as a quotient of factorials.

    b A *permutation* of an n-element set N is usually defined as a a one-to-one function from N onto N.[1] Show that the number of permutations of N is the same as the number of n-element permutations of N. What is this number? Try to give an intuitive explanation that (in part) reconciles the two uses of the word permutation.

    c Show that if S is a finite set, then a function f from S to S is one-to-one if and only if it is onto.

[1]The word function is defined in the previous exercise. A function f from S to T is called *one-to-one* if each member of T is associated with at most one member of S, and is called *onto* if each member of T is associated with at least one member of S.

1.2 Binomial Coefficients and Subsets

1.2-1 The loop below is part of a program to determine the number of triangles formed by n points in the plane.

```
for i = 1 to n
     for j = i + 1 to n
          for k = j + 1 to n
               if points i, j, and k are not collinear
                    add one to triangle count
```

How many times does the loop check three points to see if they are collinear?

The Correspondence Principle

In Exercise 1.2-1, we have a loop embedded in a loop that is embedded in another loop. Because the second loop began at the current i value, and the third loop began at the current j value, our code examines each triple of values i, j, k with $i < j < k$ exactly once. Thus one way in which we might have solved Exercise 1.2-1 would be to compute the number of such triples, which we will call *increasing triples*. As with the case of two-element subsets earlier, the number of such triples is the number of three-element subsets of an n-element set. This is the second time that we have proposed counting the elements of one set (in this case the set of increasing triples chosen from an n-element set) by saying that it is equal to the number of elements of some other set (in this case the set of three element subsets of an n-element set). When are we justified in making such an assertion that two sets have the same size? The **correspondence principle says**

> Two sets have the same size if and only if there is a one-to-one function[2] from one set onto the other.

Such a function is called a *one-to-one correspondence* or a *bijection*. What is the function that is behind our assertion that the number of increasing triples equals the number of three-element subsets? We define the function f to be the one that takes the increasing triple (i, j, k) to the subset $\{i, j, k\}$. Since the three elements of an increasing triple are different, the subset is a three element set, so we have a function from increasing triples to three element sets. Two different triples can't be the same set in two different orders, so different triples have to be associated with different sets. Thus f is one-to-one. Each set of three integers can be listed in increasing order, so it is the image under f of an increasing triple. Therefore f is onto. Thus we have a one to one correspondence between the set of increasing triples and the set of three element sets.

Counting Subsets of a Set

Now that we know that counting increasing triples is the same as counting three-element subsets, let us see if we can count three-element subsets in the way that we counted two-element subsets.

[2]Recall that a function from a set S to a set T is a relationship that associates a unique element of T to each element of S. If we use f to stand for the function, then for each s in S, we use $f(s)$ to stand for the element in T associated with s. A function is one-to-one if it associates different elements of T to different elements of S, and is onto if it associates at least one element of S to each element of T.

First, how many lists of three distinct numbers (between 1 and n) can we make? There are n choices for the first number, and $n-1$ choices for the second, so by the product principle there are $n(n-1)$ choices for the first two elements of the list. For each choice of the first two elements, there are $n-2$ ways to choose a third (distinct) number, so again by the product principle, there are $n(n-1)(n-2)$ ways to choose the list of numbers. This does not tell us the number of three element sets, though, because a given three element set can be listed in a number of ways. How many? Well, given the three numbers, there are three ways to choose the first number in the list, given the first there are two ways to choose the second, and given the first two there is only one way to choose the third element of the list. Thus by the product principle once again, there are $3 \cdot 2 \cdot 1 = 6$ ways to make the list.

Since there are $n(n-1)(n-2)$ lists of three distinct elements chosen from an n-element set, and each three-element subset appears in exactly 6 of these lists, there are $n(n-1)(n-2)/6$ three-element subsets of an n-element set.

If we would like to count the number of k-element subsets of an n element set, and $k > 0$, then we could first compute the number of lists of k distinct elements chosen from a k-element set which, by the product principle, will be $n(n-1)(n-2)\cdots(n-k+1)$, the first k terms of $n!$, and then divide that number by $k(k-1)\cdots 1$, the number of ways to list a set of k elements. This gives us

Theorem 1.2.1 *For integers n and k with $0 \leq k \leq n$, the number of k element subsets of an n element set is*

$$n!/k!(n-k)!$$

Proof: Essentially given above, except in the case that k is 0; however the only subset of our n-element set of size zero is the empty set, so we have exactly one such subset. This is exactly what the formula gives us as well. (Note that the cases $k = 0$ and $k = n$ both use the fact that $0! = 1$.[3]) ■

The number of k-element subsets of an n-element set is usually denoted by $\binom{n}{k}$ or $C(n,k)$, both of which are read as "n choose k." These numbers are called *binomial coefficients* for reasons that will become clear later.

Notice that it was the second version of the product principle, the version for counting lists, that we were using in computing the number of k-element subsets of an n-element set. As part of the computation we saw that the number of ways to make a list of k distinct elements chosen from an n-element set is

$$n(n-1)\cdots(n-k+1) = n!/(n-k)!,$$

the first k terms of $n!$. This expression arises frequently; we use the notation $n^{\underline{k}}$, (read "n to the k falling") for $n(n-1)\cdots(n-k+1)$. This notation is originally due to Knuth.

Pascal's Triangle

In Table 1 we see the values of the binomial coefficients $\binom{n}{k}$ for $n = 0$ to 6 and all relevant k values. Note that the table begins with a 1 for $n = 0$ and $k = 0$, which is as it should be because

[3]There are many reasons why 0! is defined to be one; making the formula for binomial coefficients work out is one of them.

the empty set, the set with no elements, has exactly one 0-element subset, namely itself. We have not put any value into the table for a value of k larger than n, because we haven't defined what we mean by the binomial coefficient $\binom{n}{k}$ in that case. (We could put zeros in the other places, signifying the fact that a set S has no subsets larger than S.)

Table 1.1: A table of binomial coefficients

$n\backslash k$	0	1	2	3	4	5	6
0	1						
1	1	1					
2	1	2	1				
3	1	3	3	1			
4	1	4	6	4	1		
5	1	5	10	10	5	1	
6	1	6	15	20	15	6	1

1.2-2 What general properties of binomial coefficients do you see in the table of values we created?

1.2-3 What do you think is the next row of the table of binomial coefficients?

1.2-4 How do you think you could prove you were right in the last two exercises?

There are a number of properties of binomial coefficients that are obvious from the table. The 1 at the beginning of each row reflects the fact that $\binom{n}{0}$ is always one, as it must be because there is just one subset of an n-element set with 0 elements, namely the empty set. The fact that each row ends with a 1 reflects the fact that an n-element set S has just one n-element subset, S itself. Each row seems to increase at first, and then decrease. Further the second half of each row is the reverse of the first half. The array of numbers called *Pascal's Triangle* emphasizes that symmetry by rearranging the rows of the table so that they line up at their centers. We show this array in Table 2. When we write down Pascal's triangle, we leave out the values of n and k.

Table 1.2: Pascal's Triangle

					1							
				1		1						
			1		2		1					
		1		3		3		1				
	1		4		6		4		1			
1		5		10		10		5		1		
1		6		15		20		15		6		1

You may have been taught a method for creating Pascal's triangle that does not involve computing binomial coefficients, but rather creates each row from the row above. In fact, notice that each entry in the table (except for the ones) is the sum of the entry directly above it to the left and the entry directly above it to the right. We call this the *Pascal Relationship*. If we need to compute a number of binomial coefficients, this can be an easier way to do so than the

multiplying and dividing formula given above. But do the two methods of computing Pascal's triangle always yield the same results? To verify this, it is handy to have an algebraic statement of the Pascal Relationship. To figure out this algebraic statement of the relationship, it is useful to observe how it plays out in Table 1, our original table of binomial coefficients. You can see that in Table 1, each entry is the sum of the one above it and the one above it and to the left. In algebraic terms, then, the Pascal Relationship says

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}, \tag{1.1}$$

whenever $n > 0$ and $0 < k < n$. It is possible to give a purely algebraic (and rather dreary) proof of this formula by plugging in our earlier formula for binomial coefficients into all three terms and verifying that we get an equality. A guiding principle of discrete mathematics is that when we have a formula that relates the numbers of elements of several sets, we should find an explanation that involves a relationship among the sets. We give such an explanation in the proof that follows.[4]

Theorem 1.2.2 *If n and k are integers with $n > 0$ and $0 < k < n$, then*

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}.$$

Proof: The formula says that the number of k-element subsets of an n-element set is the sum of two numbers. Since we've used the sum principle to explain other computations involving addition, it is natural to see if it applies here. To apply it, we need to represent the set of k-element subsets of an n-element set as a union of two other disjoint sets. Suppose our n-element set is $S = \{x_1, x_2, \ldots x_n\}$. Then we wish to take S_1, say, to be the $\binom{n}{k}$-element set of all k-element subsets of S and partition it into two disjoint sets of k-element subsets, S_2 and S_3, where the sizes of S_2 and S_3 are $\binom{n-1}{k-1}$ and $\binom{n-1}{k}$ respectively. We can do this as follows. Note that $\binom{n-1}{k}$ stands for the number of k element subsets of the first $n-1$ elements $x_1, x_2, \ldots, x_{n-1}$ of S. Thus we can let S_3 be the set of k-element subsets of S that don't contain x_n. Then the only possibility for S_2 is the set of k-element subsets of S that *do* contain x_n. How can we see that the number of elements of this set S_2 is $\binom{n-1}{k-1}$? By observing that removing x_n from each of the elements of S_2 gives a $(k-1)$-element subset of $S' = \{x_1, x_2, \ldots x_{n-1}\}$. Further each $(k-1)$-element subset of S' arises in this way from one and only one k-element subset of S containing x_n. Thus the number of elements of S_2 is the number of $(k-1)$-element subsets of S', which is $\binom{n-1}{k-1}$. Since S_2 and S_3 are two disjoint sets whose union is S, this shows that the number of element of S is $\binom{n-1}{k-1} + \binom{n-1}{k}$. ■

The Binomial Theorem

1.2-5 What is $(x+1)^4$? What is $(2+y)^4$? What is $(x+y)^4$?

The number of k-element subsets of an n-element set is called a *binomial coefficient* because of the role that these numbers play in the algebraic expansion of a binomial $x+y$. The **binomial**

[4] In this case the three sets are the set of k-element subsets of an n-element set, the set of $(k-1)$-element subsets of an $(n-1)$-element set, and the set of k-element set subsets of an $(n-1)$-element set.

theorem states that

$$(x+y)^n = x^n + \binom{n}{1}x^{n-1}y + \binom{n}{2}x^{n-2}y^2 + \cdots + \binom{n}{n-1}xy^{n-1} + \binom{n}{n}y^n \ ,$$

or in summation notation,

$$(x+y)^n = \sum_{i=0}^{n}\binom{n}{i}x^{n-i}y^i \ .$$

Unfortunately when we first see this theorem many of us don't really have the tools to see why it is true. Since we know that $\binom{n}{k}$ counts the number of k-element subsets of an n-element set, to really understand the theorem it makes sense to look for a way to associate a k-element set with each term of the expansion. Since there is a y^k associated with the term $\binom{n}{k}$, it is natural to look for a set that corresponds to k distinct y's. When we multiply together n copies of the binomial $x+y$, we choose y from some of the terms, and x from the remainder, multiply the choices together and add up all the products we get from all the ways of choosing the y's. The summands that give us $x^{n-k}y^k$ are those that involving choosing y from k of the binomials. The number of ways to choose the k binomials giving y from the n binomials is the number of sets of k binomials we may choose out of n, so the coefficient of $x^{n-k}y^k$ is $\binom{n}{k}$. Do you see how this proves the binomial theorem?

1.2-6 If I have k labels of one kind and $n-k$ labels of another, in how many ways may I apply these labels to n objects?

1.2-7 Show that if we have k_1 labels of one kind, k_2 labels of a second kind, and $k_3 = n-k_1-k_2$ labels of a third kind, then there are $\frac{n!}{k_1!k_2!k_3!}$ ways to apply these labels to n objects.

1.2-8 What is the coefficient of $x^{k_1}y^{k_2}z^{k_3}$ in $(x+y+z)^n$?

1.2-9 Can you come up with more than one way to solve Exercise 1.2-6 and Exercise 1.2-7?

Exercise 1.2-6 and Exercise 1.2-7 have straightforward solutions For Exercise 1.2-6, there are $\binom{n}{k}$ ways to choose the k objects that get the first label, and the other objects get the second label, so the answer is $\binom{n}{k}$. For Exercise 1.2-7, there are $\binom{n}{k_1}$ ways to choose the k_1 objects that get the first label, and then there are $\binom{n-k_1}{k_2}$ ways to choose the objects that get the second labels. After that, the remaining $k_3 = n-k_1-k_2$ objects get the third labels. The total number of labellings is thus, by the product principle, the product of the two binomial coefficients, which simplifies to the nice expression shown in the exercise. Of course, this solution begs an obvious question; namely why did we get that nice formula in the second exercise? A more elegant approach to Exercise 1.2-6 and Exercise 1.2-7 appears in the next section.

Exercise 1.2-8 shows why Exercise 1.2-7 is important. In expanding $(x+y+z)^n$, we think of writing down n copies of the trinomial $x+y+z$ side by side, and imagine choosing x from some number k_1 of them, choosing y from some number k_2, and z from some number k_3, multiplying all the chosen terms together, and adding up over all ways of picking the k_is and making our choices. Choosing x from a copy of the trinomial "labels" that copy with x, and the same for y and z, so the number of choices that yield $x^{k_1}y^{k_2}z^{k_3}$ is the number of ways to label n objects with k_1 labels of one kind, k_2 labels of a second kind, and k_3 labels of a third.

Problems

1. Find $\binom{12}{3}$ and $\binom{12}{9}$. What should you multiply $\binom{12}{3}$ by to get $\binom{12}{4}$?
2. Find the row of the Pascal triangle that corresponds to $n = 10$.
3. Prove Equation 1.1 by plugging in the formula for $\binom{n}{k}$.
4. Find the following
   a. $(x+1)^5$
   b. $(x+y)^5$
   c. $(x+2)^5$
   d. $(x-1)^5$
5. Carefully explain the proof of the binomial theorem for $(x+y)^4$. That is, explain what each of the binomial coefficients in the theorem stands for and what powers of x and y are associated with them in this case.
6. If I have ten distinct chairs to paint in how many ways may I paint three of them green, three of them blue, and four of them red? What does this have to do with labellings?
7. When n_1, n_2, $\ldots n_k$ are nonnegative integers that add to n, the number $\frac{n!}{n_1!,n_2!,\ldots,n_k!}$ is called a *multinomial coefficient* and is denoted by $\binom{n}{n_1,n_2,\ldots,n_k}$. A polynomial of the form $x_1 + x_2 + \cdots + x_k$ is called a multinomial. Explain the relationship between powers of a multinomial and multinomial coefficients.
8. In a Cartesian coordinate system, how many paths are there from the origin to the point with integer coordinates (m, n) if the paths are built up of exactly $m + n$ horizontal and vertical line segments each of length one?
9. What is the formula we get for the binomial theorem if, instead of analyzing the number of ways to choose k distinct y's, we analyze the number of ways to choose k distinct x's?
10. Explain the difference between choosing four disjoint three element sets from a twelve element set and labelling a twelve element set with three labels of type 1, three labels of type two, three labels of type 3, and three labels of type 4. What is the number of ways of choosing three disjoint four element subsets from a twelve element set? What is the number of ways of choosing four disjoint three element subsets from a twelve element set?
11. A 20 member club must have a President, Vice President, Secretary and Treasurer as well as a three person nominations committee. If the officers must be different people, and if no officer may be on the nominating committee, in how many ways could the officers and nominating committee be chosen? Answer the same question if officers may be on the nominating committee.
12. Give at least two proofs that
$$\binom{n}{k}\binom{k}{j} = \binom{n}{j}\binom{n-j}{k-j}.$$

13. You need not compute all of rows 7, 8, and 9 of Pascal's triangle to use it to compute $\binom{9}{6}$. Figure out which entries of Pascal's triangle not given in Table 2 you actually need, and compute them to get $\binom{9}{6}$.

14. Explain why
$$\sum_{i=0}^{n}(-1)^i\binom{n}{i} = 0$$

15. Apply calculus and the binomial theorem to show that
$$\binom{n}{1} + 2\binom{n}{2} + 3\binom{n}{3} + \cdots = n2^{n-1}.$$

16. True or False: $\binom{n}{k} = \binom{n-2}{k-2} + \binom{n-2}{k-1} + \binom{n-2}{k}$. If True, give a proof. If false, give a value of n and k that show the statement is false, find an analogous true statement, and prove it.

1.3 Equivalence Relations and Counting

Equivalence Relations and Equivalence Classes

In counting k-element subsets of an n-element set, we counted the number of lists of k distinct elements, getting $n^{\underline{k}} = n!/(n-k)!$ lists. Then we observed that two lists are equivalent as sets if I get one by rearranging (or "permuting") the other. This divides the lists up into classes, called *equivalence classes*, all of size $k!$. The product principle told us that if m is the number of such lists, then $mk! = n!/(n-k)!$ and we got our formula for m by dividing. In a way it seems as if the proof does not account for the symmetry of the expression $\frac{n!}{k!(n-k)!}$. The symmetry comes of course from the fact that choosing a k element subset is equivalent to choosing the $n-k$-element subset of elements we don't want. A principle that helps in learning and understanding mathematics is that if we have a mathematical result that shows a certain symmetry, it helps our understanding to find a proof that reflects this symmetry. We saw that the binomial coefficient $\binom{n}{k}$ also counts the number of ways to label n objects, say with the labels "in" and "out," so that we have k "ins" and therefore $n-k$ "outs." For each labelling, the k objects that get the label "in" are in our subset. Here is a new proof that the number of labellings is $n!/k!(n-k)!$ that explains the symmetry.

Suppose we have m ways to assign k labels of one type and $n-k$ labels of a second type to n elements. Let us think about making a list from such a labelling by listing first the objects with the label of type 1 and then the objects with the label of type 2. We can mix the k elements labeled 1 among themselves, and we can mix the $n-k$ labeled 2 among themselves, giving us $k!(n-k)!$ lists consisting of first the elements with label 1 and then the elements with label 2. Every list of our n objects arises from some labelling in this way. Therefore, by the product principle, $mk!(n-k)!$ is the number of lists we can form with n objects, namely $n!$. This gives us $mk!(n-k)! = n!$, and division gives us our original formula for m. With this idea in hand, we could now easily attack labellings with three (or more) labels, and explain why the product in the denominator of the formula for the number of labellings with three labels is what it is.

We can think of the process we described above as dividing the set of all lists of n elements into classes of lists that are mutually equivalent for the purposes of labeling with two labels. Two lists of the n objects are equivalent for defining labellings if we get one from the other by mixing the first k elements among themselves and mixing the last $n-k$ elements among themselves. Relating objects we want to count to sets of lists (so that each object corresponds to an set of equivalent lists) is a technique we can use to solve a wide variety of counting problems.

A relationship that divides a set up into mutually exclusive classes is called an **equivalence relation**.[5] Thus if

$$S = S_1 \cup S_2 \cup \ldots \cup S_m$$

and $S_i \cap S_j = \emptyset$ for all i and j, the relationship that says x and y are equivalent if and only if they lie in the same set S_i is an equivalence relation. The sets S_i are called *equivalence classes*, and the family $S_1, S_2, \ldots, S_m$ is called a **partition** of S. One partition of the set $S = \{a, b, c, d, e, f, g\}$ is

[5]The usual mathematical approach to equivalence relations, which we shall discuss in the exercises, is slightly different from the one given here. Typically, one sees an equivalence relation defined as a reflexive (everything is related to itself), symmetric (if x is related to y, then y is related to x), and transitive (if x is related to y and y is related to z, then x is related to z) relationship on a set X. Examples of such relationships are equality (on any set), similarity (on a set of triangles), and having the same birthday as (on a set of people). The two approaches are equivalent, and we haven't found a need for the details of the other approach in what we are doing in this course.

$\{a, c\}$, $\{d, g\}$, $\{b, e, f\}$. This partition corresponds to the following (boring) equivalence relation: a and c are equivalent, d and g are equivalent, and b, e, and f are equivalent.

1.3-1 On the set of integers between 0 and 12 inclusive, define two integers to be related if they have the same remainder on division by 3. Which numbers are related to 0? to 1? to 2? to 3? to 4?. Is this relationship an equivalence relation?

In Exercise 1.3-1, the numbers related to 0 are the set $\{0, 3, 6, 9, 12\}$, those related to 1 are $\{1, 4, 7, 10\}$, those related to 2 are $\{2, 5, 8, 11\}$, those related to 3 are $\{0, 3, 6, 9, 12\}$, those related to 4 are $\{1, 4, 7, 10\}$. From these computations it is clear that our relationship divides our set into three disjoint sets, and so it is an equivalence relation. A little more precisely, a number is related to one of 0, 3, 6, 9, or 12, if and only if it is in the set $\{0, 3, 6, 9, 12\}$, a number is related to 1, 4, 7, or 10 if and only if it is in the set $\{1, 4, 7, 10\}$ and a number is related to 2, 5, 8, or 11 if and only if it is in the set $\{2, 5, 8, 11\}$.

In Exercise 1.3-1 the equivalence classes had two different sizes. In the examples of counting labellings and subsets that we have seen so far, all the equivalence classes had the same size, and this was very important. The principle we have been using to count subsets and labellings is the following theorem. We will call this principle the **equivalence principle**.

Theorem 1.3.1 *If an equivalence relation on a s-element set S has m classes each of size t, then $m = s/t$.*

Proof: By the product principle, $s = mt$, and so $m = s/t$. ■

1.3-2 When four people sit down at a round table to play cards, two lists of their four names are equivalent as seating charts if each person has the same person to the right in both lists. (The person to the right of the person in position 4 of the list is the person in position 1). How many lists are in an equivalence class? How many equivalence classes are there?

1.3-3 When making lists corresponding to attaching n distinct beads to the corners of a regular n-gon (or stringing them on a necklace), two lists of the n beads are equivalent if each bead is adjacent to exactly the same beads in both lists. (The first bead in the list is considered to be adjacent to the last.) How many lists are in an equivalence class? How many equivalence classes are there? Notice how this exercise models the process of installing n workstations in a ring network.

1.3-4 Sometimes when we think about choosing elements from a set, we want to be able to choose an element more than once. For example the set of letters of the word "roof" is $\{f, o, r\}$. However it is often more useful to think of the of the "multiset" of letters, which in this case is $\{f, o, o, r\}$. In general we specify a *multiset* chosen from a set S by saying how many times each of its elements occurs. Thus the "multiplicity" function for roof is given by $m(f) = 1$, $m(o) = 2$, $m(r) = 1$, and $m(\text{letter}) = 0$ for every other letter. If there were a way to visualize multisets as lists, we might be able to use it in order to compute the number of multisets. Here is one way. Given a multiset chosen from $\{1, 2, \ldots, n\}$, make a row of red and black checkers as follows. First put down $m(1)$ red checkers. Then put down one black checker. (Thus if $m(1) = 0$, we

start out with a black checker.) Now put down $m(2)$ red checkers; then another black checker, and so on until we put down a black checker and then $m(n)$ red checkers. The number of black checkers will be $n-1$, and the number of red checkers will be $k = m(1) + m(2) + \ldots m(n)$. Thus any way of stringing out $n-1$ black and k red checkers gives us a k-element multiset chosen from $\{1, 2, \ldots, n\}$. If our red and black checkers all happened to look different, how many such strings of checkers could we make? How many strings would a given string be equivalent to for the sake of defining a multiset? How many k-element multisets can we choose from an n-element set?

Equivalence class counting

We can think of Exercise 1.3-2 as a problem of counting equivalence classes by thinking of writing down a list of the four people who are going to sit at the table, starting at some fixed point at the table and going around to the right. Then each list is equivalent to three other lists that we get by shifting everyone around to the right. (Why not shift to the left also?) Now this divides the set of all lists up into sets of four lists each, and every list is in a set. The sets are disjoint, because if a list were in two different sets, it would be equivalent to everything in both sets, and so it would have more than three right shifts distinct from it and each other. Thus we have divided the set of all lists of the four names into equivalence classes each of size four, and so by Theorem 1.3.1 we have $4!/4 = 3! = 6$ seating arrangements.

Exercise 1.3-3 is similar in many ways to Exercise 1.3-2, but there is one significant difference. We can visualize the problem as one of dividing lists of n distinct beads up into equivalence classes, but turning a polygon or necklace over corresponds to reversing the order of the list. Any combination of rotations and reversals corresponds to natural geometric operations on the polygon, so an equivalence class consists of everything we can get from a given list by rotations and reversals. Note that if you rotate the list $x_1, x_2, \ldots, x_n$ through one place to get $x_2, x_3, \ldots, x_n, x_1$ and then reverse, you get $x_1, x_n, x_{n-1}, \ldots, x_3, x_2$, which is the same result we would get if we first reversed the list $x_1, x_2, \ldots, x_n$ and then rotated it through $n-1$ places. Thus a rotation followed by a reversal is the same as a reversal followed by some other rotation. This means that if we arbitrarily combine rotations and reversals, we won't get any lists other than the ones we get from rotating the original list in all possible ways and then reversing all our results. Thus since there are n results of rotations (including rotating through n, or 0, positions) and each one can be flipped, there are $2n$ lists per equivalence class. Since there are $n!$ lists, Theorem 1.3.1 says there are $(n-1)!/2$ bead arrangements.

In Exercise 1.3-4, if all the checkers were distinguishable from each other (say they were numbered, for example), there would be $n-1+k$ objects to list, so we would have $(n+k-1)!$ lists. Two lists would be equivalent if we mixed the red checkers among themselves and we mixed the black checkers among themselves. There are $k!$ ways to mix the red checkers among themselves, and $n-1$ ways to mix the black checkers among themselves. Thus there are $k!(n-1)!$ lists per equivalence class. Then by Theorem 1.3.1, there are

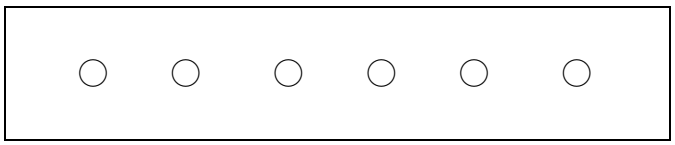
$$\frac{(n+k-1)!}{k!(n-1)!} = \binom{n+k-1}{k}$$

equivalence classes, so there are the same number of k-element multisets chosen from an n-element set.

Problems

1. In how many ways may n people be seated around a round table? (Remember, two seating arrangements around a round table are equivalent if everyone is in the same position relative to everyone else in both arrangements.)

2. In how many ways may we embroider n circles of different colors in a row (lengthwise, equally spaced, and centered halfway between the edges) on a scarf (as follows)?



3. Use binomial coefficients to determine in how many ways three identical red apples and two identical golden apples may be lined up in a line. Use equivalence class counting to determine the same number.

4. Use multisets to determine the number of ways to pass out k identical apples to n children?

5. In how many ways may n men and n women be seated around a table alternating gender? (Use equivalence class counting!!)

6. In how many ways may n red checkers and $n+1$ black checkers be arranged in a circle? (This number is a famous number called a *Catalan number*.

7. A standard notation for the number of partitions of an n element set into k classes is $S(n,k)$. $S(0,0)$ is 1, because technically the empty family of subsets of the empty set is a partition of the empty set, and $S(n,0)$ is 0 for $n>0$, because there are no partitions of a nonempty set into no parts. $S(1,1)$ is 1.

   a) Explain why $S(n,n)$ is 1 for all $n>0$. Explain why $S(n,1)$ is 1 for all $n>0$.

   b) Explain why, for $1<k<n$, $S(n,k)=S(n-1,k-1)+kS(n-1,k)$.

   c) Make a table like our first table of binomial coefficients that shows the values of $S(n,k)$ for values of n and k ranging from 1 to 6.

8. In how many ways may k distinct books be placed on n distinct bookshelves, if each shelf can hold all the books? (All that is important about the placement of books on a shelf is their left-to-right order.) Give as many different solutions as you can to this exercise. (Can you find three?)

9. You are given a square, which can be rotated 90 degrees at a time (i.e. the square has four orientations). You are also given two red checkers and two black checkers, and you will place each checker on one corner of the square. How many lists of four letters, two of which are R and two of which are B, are there? Once you choose a starting place on the square, each list represents placing checkers on the square in the natural way. Consider two lists to be equivalent if they represent the same arrangement of checkers at the corners of the square, that is, if one arrangement can be rotated to create the other one. Write down the equivalence classes of this equivalence relation. Why can't we apply Theorem 1.3.1 to compute the number of equivalence classes?

10. The terms "reflexive", "symmetric" and "transitive" were defined in Footnote 2. Which of these properties is satisfied by the relationship of "greater than?" Which of these properties is satisfied by the relationship of "is a brother of?" Which of these properties is satisfied by "is a sibling of?" (You are not considered to be your own brother or your own sibling). How about the relationship "is either a sibling of or is?"

    a Explain why an equivalence relation (as we have defined it) is a reflexive, symmetric, and transitive relationship.

    b Suppose we have a reflexive, symmetric, and transitive relationship defined on a set S. For each x is S, let $S_x = \{y | y \text{ is related to } x\}$. Show that two such sets S_x and S_y are either disjoint or identical. Explain why this means that our relationship is an equivalence relation (as defined in this section of the notes, not as defined in the footnote).

    c Parts b and c of this problem prove that a relationship is an equivalence relation if and only if it is symmetric, reflexive, and transitive. Explain why. (A short answer is most appropriate here.)

11. Consider the following C++ function to compute $\binom{n}{k}$.

```
int pascal(int n, int k)
{
  if (n < k)
    {

      cout << "error: n<k" << endl;
      exit(1);
    }

  if ( (k==0) || (n==k))
    return 1;


  return  pascal(n-1,k-1) + pascal(n-1,k);
}
```

    Enter this code and compile and run it (you will need to create a simple main program that calls it). Run it on larger and larger values of n and k, and observe the running time of the program. It should be surprisingly slow. (Try computing, for example, $\binom{30}{15}$.) Why is it so slow? Can you write a different function to compute $\binom{n}{k}$ that is *significantly faster*? Why is your new version faster? (Note: an exact analysis of this might be difficult at this point in the course, it will be easier later. However, you should be able to figure out roughly why this version is so much slower.)

12. Answer each of the following questions with either n^k, $n^{\underline{k}}$, $\binom{n}{k}$, or $\binom{n+k-1}{k}$.

    (a) In how many ways can k different candy bars be distributed to n people (with any person allowed to receive more than one bar)?

    (b) In how many ways can k different candy bars be distributed to n people (with nobody receiving more than one bar)?

(c) In how many ways can k identical candy bars distributed to n people (with any person allowed to receive more than one bar)?

(d) In how many ways can k identical candy bars distributed to n people (with nobody receiving more than one bar)?

(e) How many one-to-one functions f are there from $\{1, 2, \ldots, k\}$ to $\{1, 2, \ldots, n\}$?

(f) How many functions f are there from $\{1, 2, \ldots, k\}$ to $\{1, 2, \ldots, n\}$?

(g) In how many ways can one choose a k-element subset from an n-element set?

(h) How many k-element multisets can be formed from an n-element set?

(i) In how many ways can the top k ranking officials in the US government be chosen from a group of n people?

(j) In how many ways can k pieces of candy (not necessarily of different types) be chosen from among n different types?

(k) In how many ways can k children each choose one piece of candy (all of different types) from among n different types of candy?

Chapter 2

Cryptography and Number Theory

2.1 Cryptography and Modular Arithmetic

Introduction to Cryptography

For thousands of years people have searched for ways to send messages in secret. For example, there is a story that in ancient times a king desired to send a secret message to his general in battle. The king took a servant, shaved his head, and wrote the message on his head. He then waited for the servant's hair to grow back and then sent the servant to the general. The general then shaved the servant's head and read the message. If the enemy had captured the servant, they presumably would not have known to shave his head, and the message would have been safe.

Cryptography is the study of methods to send and receive secret messages; it is also concerned with methods used by the *adversary* to decode messages. In general, we have a *sender* who is trying to send a message to a *receiver*. There is also an *adversary*, who wants to steal the message. We are successful if the sender is able to communicate a message to the receiver without the adversary learning what that message was.

Cryptography has remained important over the centuries, used mainly for military and diplomatic communications. Recently, with the advent of the internet and electronic commerce, cryptography has become vital for the functioning of the global economy, and is something that is used by millions of people on a daily basis. Sensitive information, such as bank records, credit card reports, or private communication, is (and should be) *encrypted* – modified in such a way that it is only understandable to people who should be allowed to have access to it, and undecipherable to others.

In traditional cryptography, the sender and receiver agree in advance on a *secret code*, and then send messages using that code. For example, one of the oldest types of code is known as a Caesar cipher. In this code, the letters of the alphabet are shifted by some fixed amount. Typically, we call the original message the *plaintext* and the encoded text the *ciphertext*. An example of a Caesar cipher would be the following code

```
plaintext  A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
ciphertext E F G H I J K L M N O P Q R S T U V W X Y Z A B C D
```

Thus if we wanted to send the plaintext message

ONE IF BY LAND AND TWO IF BY SEA

we would send the ciphertext

SRI MJ FC PERH ERH XAS MJ FC WIE

The Ceaser ciphers are especially easy to implement on a computer using a scheme known as arithmetic mod 26. The symbolism

$$m \bmod n$$

means the remainder we get when we divide m by n. A bit more precisely, for integers m and n, $m \bmod n$ is the smallest nonnegative integer r such that

$$m = nq + r \tag{2.1}$$

for some integer q. [1]

2.1-1 Compute $10 \bmod 7$, $-10 \bmod 7$.

2.1-2 Using 0 for A, 1 for B, and so on, let the numbers from 0 to 25 stand for the letters of the alphabet. In this way, convert a message to a sequence of strings of numbers. For example SEA becomes 18 4 0. What does this word become if we shift every letter two places to the right? What if we shift every letter 13 places to the right? How can you use the idea of $m \bmod n$ to implement a Ceaser cipher?

2.1-3 Have someone use a Ceaser cipher to encode a message of a few words in your favorite natural language, without telling you how far they are shifting the letters of the alphabet. How fast can you figure out what the message is?

In Exercise 2.1-1, $10 \bmod 7$ is 3, while $-10 \bmod 7$ is 4. Note that $-3 \bmod 7$ is 4 also. Also, $-10 + 3 \bmod 7 = 0$, suggesting that -10 is essentially the same as -3 when we are considering integers mod 7. We will be able to make this idea more precise later on.

In Exercise 2.1-2, to shift every letter two places to the right, we replace every number n in our message by $(n + 2) \bmod 26$ and to shift 13 places to the right, we replace every number n in our message with $(n + 13) \bmod 26$. Similarly to implement a shift of s places, we replace each number n in our message by $(n + s) \bmod 26$. Since most computer languages give us simple ways to keep track of strings of numbers and a "mod function," it is easy to implement a Ceaser cipher on a computer.

[1] In an unfortunate historical evolution of terminology, the fact that for every nonnegative integer m and positive integer n, there exist unique nonnegative integers q and r such that $m = nq + r$ and $r < n$ is called "Euclid's algorithm." In modern language we would call this "Euclid's Theorem" instead. While it seems obvious that there is such a smallest nonnegative integer r and that there is exactly one such pair q, r with $r < n$, a technically complete study would derive these facts from the basic axioms of number theory, just as "obvious" facts of geometry are derived form the basic axioms of geometry. The reasons why mathematicians take the time to derive such obvious facts from basic axioms is so that everyone can understand exactly what we are assuming as the foundations of our subject; as the "rules of the game" in effect.

Even by hand, it is easy for the sender to encode the message, and for the receiver to decode the message. The disadvantage of this scheme is that it is also easy for the adversary to just try the 26 different possible Caesar ciphers and decode the message. (It is very likely that only one will decode into plain English.) Of course, there is no reason to use such a simple code; we can use any arbitrary permutation of the alphabet as the ciphertext, e.g.

```
plaintext  A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
ciphertext H D I E T J K L M X N Y O P F Q R U V W G Z A S B C
```

If we encode a short message with a code like this, it would be hard for the adversary to decode it. However, with a message of any reasonable length (greater than about 50 letters), an adversary with a knowledge of the statistics of the English language can easily crack the code.

There is no reason that we have to use simple mappings of letters to letters. For example, our coding algorithm can be to take three consecutive letters, reverse their order, interpret each as a base 26 integer (with A=0;B=1, etc.) multiply that number by 37, add 95 and then convert that number to base 8. We continue this processing with each block of three consecutive letters. We append the blocks, using either an 8 or a 9 to separate the blocks. When we are done, we reverse the number, and replace each digit 5 by two 5's. Here is an example of this method:

```
plaintext: ONEIFBYLANDTWOIFBYSEA

block and reverse: ENO  BFI   ALY     TDN   IOW     YBF    AES
base 26 integer:  3056  814   310    12935  5794   16255   122
*37 +95 base 8: 335017 73005 26455 1646742 642711 2226672 11001
appended      : 335017873005926455916467429642711822266728110 01
reverse, 5rep : 10011827662228117246924764619555546295500378710533
```

Now, we can verify that it is possible for the receiver, who knows the code, to decode this message, and that the casual reader of the message will have no hope of decoding it. So it seems that with a complicated enough code, we can have secure cryptography. Unfortunately, there are at least two flaws with this method. The first is that if the adversary learns, somehow, what the code is, then she can easily decode it. Second, if this coding scheme is repeated often enough, and if the adversary has enough time, money and computing power, this code could be broken. In the field of cryptography, there are definitely entities who have all these resources (such as a government, or a large corporation). The infamous German Enigma is an example of a much more complicated coding scheme, yet it was broken and this helped the Allies win World War II. (The reader might be interested in looking up more details on this.) In general, any scheme that uses a *codebook*, a secretly agreed upon (possibly complicated) code, suffers from these drawbacks.

Public-key Cryptosystems

A kind of system called a *public-key* cryptosystem overcomes the problems mentioned above. In a public key cryptosystem, the sender and receiver (often called *Alice* and *Bob* respectively) don't have to agree in advance on a secret code. In fact, they each publish part of their code in a public directory. Further, the adversary can intercept the message, and can have the public directory, and these will not be able to help him decode the message.

More precisely, Alice and Bob will each have two keys, a *public key* and a *secret key*. We will denote Alice's public and secret keys as KP_A and KS_A and Bob's as KP_B and KS_B. They each keep their secret keys to themselves, but can publish their public keys and make them available to anyone, including the adversary. While the key published is likely to be a symbol string of some sort, the key is used in some standardized way (we shall see examples soon) to create a function from the set $\mathcal{D}$ of possible messages onto itself. (In complicated cases, the key might be the actual function). We denote the functions associated with KS_A, KP_A, KS_B and KP_B by S_A, P_A, S_B, and P_B,respectively. We require that the public and secret keys are chosen to be inverses of each other, i.e for any $M \in \mathcal{D}$ we have that

$$M = S_A(P_A(M)) = P_A(S_A(M)), \text{ and} \tag{2.2}$$
$$M = S_B(P_B(M)) = P_B(S_B(M)). \tag{2.3}$$

We also assume that, for Alice, S_A and P_A are easily computable. However, it is essential that *for everyone except Alice, S_A is hard to compute, even if you know P_A.* At first glance, this may seem to be an impossible task, Alice creates a function P_A, that is public and easy to compute for everyone, yet this function has an inverse, S_A, that is hard to compute for everyone except Alice. It is not at all clear how to design such a function. The first such cryptosystem is the now-famous RSA cryptosystem, widely used in many contexts. To understand how such a cryptosystem is possible requires some knowledge of number theory and computational complexity. We will develop the necessary number theory in the next few sections.

Before doing so, let us just assume that we have such a function and see how we can make use of it. If Alice wants to send Bob a message M, she takes the following two steps:

1. Alice obtains Bob's public key P_B.
2. Alice applies Bob's public key to M to create ciphertext $C = P_B(M)$.

Alice then sends C to Bob. Bob can decode the message by using his secret key to compute $S_B(C)$ which is identical to $S_B(P_B(M))$, which by (2.3) is identical to M, the original message. The beauty of the scheme is that even if the adversary has C and knows P_B, she cannot decode the message without S_B. But S_B is a secret that only Bob has. And even though the adversary knows that S_B is the inverse of P_B, the adversary cannot easily compute this inverse.

Since it is difficult to describe an example of a public key cryptosystem that is hard to decode, we will give an example of one that is easy to decode. Imagine that our messages are numbers in the range 1 to 999. Then we can imagine that Bob's public key yields the function P_B given by $P_B(M) = rev(1000 - M)$, where $rev()$ is a function that reverses the digits of a number. So to encrypt the message 167, Alice would compute $1000 - 167 = 833$ and then reverse the digits and send Bob $C = 338$. In this case $S_B(C) = 1000 - rev(C)$, and Bob can easily decode. This is not a secure code, since if you know P_B, it is easy to figure out S_B. The challenge is to design a function P_B so that even if you know P_B and $C = P_B(M)$, it is exceptionally difficult to figure out what M is.

Congruence modulo n

The RSA encryption scheme is built upon the idea of arithmetic mod n, so we introduce this arithmetic now.

2.1-4 Compute 21 mod 9, 38 mod 9, $(21 \cdot 38) \bmod 9$, $(21 \bmod 9) \cdot (38 \bmod 9)$, $(21 + 38) \bmod 9$, $(21 \bmod 9) + (38 \bmod 9)$.

2.1-5 True or false: $i \bmod n = (i + 2n) \bmod n$; $i \bmod n = (i - 3n) \bmod n$

In Exercise 2.1-4, the point to notice is that

$$21 \cdot 38 \bmod 9 = (21 \bmod 9)(38 \bmod 9)$$

and

$$21 + 38 \bmod 9 = (21 \bmod 9) + (38 \bmod 9).$$

These equations are very suggestive, though the general equations that they first suggest aren't true! Some closely related equations are true as we shall soon see.

Exercise 2.1-5 is true in both cases; in particular getting i' from i by adding any multiple of n to i makes $i \bmod n = i' \bmod n$. This will help us understand the sense in which the equations suggested by Exercise 2.1-4 can be made true, so we will state and prove it as a lemma.

Lemma 2.1.1 *$i \bmod n = (i + kn) \bmod n$ for any integer k.*

Proof: If $i = nq + r$, with $0 \leq r < n$, then $i + kn = n(q + k) + r$, so by definition $r = i \bmod n$ and $r = (i + kn) \bmod n$. ■

Now we can go back to the equations of Exercise 2.1-4; the correct versions are stated below.

Lemma 2.1.2

$$\begin{aligned}(i + j) \bmod n &= [i + (j \bmod n)] \bmod n \\ &= [(i \bmod n) + j] \bmod n \\ &= [(i \bmod n) + (j \bmod n)] \bmod n\end{aligned}$$

$$\begin{aligned}(i \cdot j) \bmod n &= [i \cdot (j \bmod n)] \bmod n \\ &= [(i \bmod n) \cdot j] \bmod n \\ &= [(i \bmod n) \cdot (j \bmod n)] \bmod n\end{aligned}$$

Proof: We prove the first and last terms in the sequence of equations for plus are equal; the other equalities for plus follow appropriate substitutions or by similar computations, whichever you prefer. The proofs of the equalities for products are similar. We use the fact that $j = (j \bmod n) + nk$ for some integer k and that $i = (i \bmod n) + nm$ for some integer m. Then

$$\begin{aligned}(i + j) \bmod n &= [(i \bmod n) + nm + (j \bmod n) + nk)] \bmod n \\ &= [(i \bmod n) + (j \bmod n) + n(m + k)] \bmod n \\ &= [(i \bmod n) + (j \bmod n)] \bmod n\end{aligned}$$

■

The functions used in encryption and decryption in the RSA system use a special arithmetic on the numbers $0, 1, \ldots, n-1$. We will use the notation Z_n to represent the integers $0, 1, \ldots, n-1$ together with a redefinition of addition, which we denote by $+_n$, and a redefinition of multiplication, which we denote $\cdot_n$. The redefinitions are:

$$i +_n j = (i + j) \bmod n \tag{2.4}$$
$$i \cdot_n j = (i \cdot j) \bmod n \tag{2.5}$$

We call these new operations addition mod n and multiplication mod n. We must now verify that all the "usual" rules of arithmetic that normally apply to addition and multiplication still apply with $+_n$ and $\cdot_n$. In particular, we wish to verify the commutative, associative and distributive laws.

Theorem 2.1.3 *Addition and multiplication mod n satisfy the commutative and associative laws, and multiplication distributes over addition.*

Proof: Commutativitiy follows immediately from the definition and the commutativity of ordinary addition and multiplication. We prove the associative law for addition below, the other laws follow similarly.

$$\begin{aligned} a +_n (b +_n c) &= (a + (b +_n c)) \bmod n \\ &= (a + ((b + c) \bmod n)) \bmod n \\ &= (a + b + c) \bmod n \\ &= ((a + b) \bmod n + c) \bmod n \\ &= ((a +_n b) + c) \bmod n = (a +_n b) +_n c. \end{aligned}$$

■

Notice that $0 +_n i = i$, $1 \cdot_n i =$i, and $0 \cdot_n i = 0$, so we can use 0 and 1 in algebraic expressions mod n as we use them in ordinary algebraic expressions.

We conclude this section by observing that repeated applications of Lemma 2.1.2 are useful when computing sums or products in which the numbers are large. For example, suppose you had m integers $x_1, \ldots, x_m$ and you wanted to compute $(\sum_{j=1}^{m} x_i) \bmod m$. One natural way to do so would be to compute the sum, and take the result modulo m. However, it is possible that, on the computer that you are using, even though $(\sum_{j=1}^{m} x_i) \bmod m$ is a number that can be stored in an integer, and each x_i can be stored in an integer, $\sum_{j=1}^{m} x_i$ might be too large to be stored in an integer. (Recall that integers are typically stored as 4 or 8 bytes, and thus have a maximum value of roughly 2×10^9 or 9×10^{18}.) Lemma 2.1.2 tells us that if we are computing a result mod n, we may do all our calculations in Z_n using $+_n$ and $\cdot_n$, and thus never computing an integer that has significantly more digits than any of the numbers we are working with.

Cryptography Revisited

One natural way to use addition of a number a mod n in encryption is to first convert the message to a sequence of digits—say concatenating all the ASCII codes for all the symbols in the message—and then simply add a to the message mod n. Thus $P(M) = M +_n a$ and

$S(C) = C +_n (-a)$. If n happens to be larger than the message in numerical value, then it is simple for someone who knows a to decode the encrypted message. However an adversary who sees the encrypted message has no special knowledge and so unless a was ill chosen (for example having all or most of the digits be zero would be a silly choice) the adversary who knows what system you are using, even including the value of n, but does not know a, is essentially reduced to trying all possible a values. (In effect adding a appears to the adversary much like changing digits at random.) Because you use a only once, there is virtually no way for the adversary to collect any data that will aid in guessing a. Thus, if only you and your intended recipient know a, this kind of encryption is quite secure: guessing a is just as hard as guessing the message.

It is possible that once n has been chosen, you will find you have a message which translates to a larger number than n. Normally you would then break the message into segments, each with no more digits than n, and send the segments individually. It might seem that as long as you were not sending a large number of segments, it would still be quite difficult for your adversary to guess a by observing the encrypted information. However if your adversary knew you were adding a mod n, he or she could take two messages and subtract them in Z_n, thus getting the difference of two unencrypted messages. This difference could contain valuable information for your adversary.[2] Thus adding a mod n is not an encoding method you would want to use more than once.

Multiplication Mod n

2.1-6 One possibility for encryption is to take a message x and compute $a \cdot_n x$. You could then decrypt by doing division mod n. How well does this work? In particular, consider the following three cases. First, $n = 12$ and $a = 4$. Second $n = 12$ and $a = 3$. Third $n = 12$ and $a = 5$.

When we encoded a message by adding a in Z_n, we could decode the message simply by subtracting a in Z_n. By analogy, if we encode by multiplying by a in Z_n, we would expect to decode by dividing by a in Z_n. However division in Z_n can be problematic for some values of n. Let us do a trivial example. Suppose your value of n was 12 and the value of a was 4. You send the message 3 as $3 \cdot_{12} 4 = 0$. Thus you send the encoded message 0. Now your partner sees 0, and says the message might have been 0; after all, $0 \cdot_{12} 4 = 0$. On the other hand, $3 \cdot_{12} 4 = 0$, $6 \cdot_{12} 4 = 0$, and $9 \cdot_{12} 4 = 0$ as well. Thus your partner has four different choices for the original message, which is almost as bad as having to guess the original message itself!

It might appear that the fact that $3 \cdot_{12} 4 = 0$ was what presented special problems for division, so let's look at another example. Suppose that $m = 3$ and $n = 12$. Now we encode the message 6 by computing $6 \cdot_{12} 3 = 6$. Simple calculation shows that $2 \cdot_{12} 3 = 6$, $6 \cdot_{12} 3 = 6$, and $10 \cdot_{12} 3 = 6$. Thus in this case, there are three possible ways to decode the message.

Let's look at one more example that will provide some hope. Let $m = 5$ and $n = 12$, and let's encode the message 7 as $7 \cdot_{12} 5 = 11$. This time, simple checking of $5 \cdot_{12} 1$, $5 \cdot_{12} 2$, $5 \cdot_{12} 3$, and so on shows that 7 is the unique solution in Z_{12} to the equation $x \cdot_{12} 5 = 1$. Thus in this case we can correctly decode the message.

[2] If each segement of a message were equally likely to be any number beetween 0 and n, and if any second (or third, etc.) segemnt were equally likely to follow any first segement, then knowing the difference between two segments would yield no information about the two segments. However, because language is structured and most information is structured, these two conditions aare highly unlikely to hold, in which case your adversary could apply structural knowledge to deduce information about your two messages from their difference.

As we shall see in the next section, the kind of problem we had happens only when a and n have a common divisor that is greater than 1. Thus all our receiver needs to know, in our cases, is how to divide by a in Z_n, and she can decrypt our message. If you don't now know how to divide by a in Z_n, then you can begin to understand the idea of public key cryptography. The message is there for anyone who knows how to divide by a to find, but if nobody but our receiver can divide by a, we can tell everyone what a is and our messages will still be secret. As we shall soon see, dividing by a is not particularly difficult, so a better trick is needed for public key cryptography to work. [3]

Problems

1. What is 14 mod 9? What is -1 mod 9? What is -11 mod 9?

2. How many places has each letter been shifted in the Caesar cipher used to encode the message XNQQD RJXXFLJ.

3. What is $16 +_{23} 18$? What is $16 \cdot_{23} 18$?

4. A short message has been encoded by converting it to an integer by replacing each "a" by 1, each "b" by 2, and so on, and concatenating the integers. The result had six or fewer digits. An unknown number a was added to the message mod 913,647, giving 618,232. Without the knowledge of a, what can you say about the message? With the knowledge of a, what could you say about the message?

5. What would it mean to say there is an integer x equal to $\frac{1}{4}$ mod 9? If it is meaningful to say there is such an integer, what is it? Is there an integer equal to $\frac{1}{3}$ mod 9? If so, what is it?

6. By multiplying a number x times 487 in Z_{30031} we obtain 13008. If you know how to find the number x, do so. If not, explain why the problem seems difficult to do by hand.

7. Write down the addition table for $+_7$ addition. Why is the table symmetric? Why does every number appear in every row?

8. It is straightforward to solve any equation of the form

$$x +_n a = b$$

in Z_n, and to see that the result will be a unique value of x. On the other hand we saw that 0, 3, 6, and 9 are all solutions to the equation

$$4 \cdot_{12} x = 0.$$

a) Are there any equations of the form $a \cdot_{12} x = b$ that don't have any solutions at all in Z_{12}? If there are, then give one.

[3] In fact, since, when it is possible, division is not particularly difficult, your adversary might be able compute the quotient of two unencoded messages in Z_n much as we computed their difference above. Then your adversary could hope to extract information about your messages from this quotient. However since sometimes we can divide by a mod n, but we cannot divide by ax mod n for some values of x, we might be able to design a cryptosystem based on multiplication. We will not pursue this as other schemes have proved more fruitful.

b) Find out whether there are any numbers a such that each equation of the form $a \cdot_{12} x = b$ does have a solution. Alternatively, if each equation of the form $a \cdot_{12} x = b$ has a solution in Z_{12}, give a proof of this. (Note that having a solution is different from having a unique solution.)

9. Does every equation of the form $a \cdot_n x = b$ have a solution in Z_5? in Z_7? in Z_9? in Z_{11}?

10. Recall that if a prime number divides a product of two integers, then it divides one of the factors.

    a) Use this to show that as b runs though the integers from 0 to $p-1$, with p prime, the products $a \cdot_p b$ are all different (for each fixed choice of a between 1 and $p-1$).

    b) Explain why every integer greater than 0 and less than p has a unique multiplicative inverse in Z_p, if p is prime.

11. Modular arithmetic is used in generating pseudo-random numbers. One basic algorithm (still widely used) is *linear congruential random number generation*. The following piece of code generates a sequence of numbers that may appear random to the unaware user.

```
set seed to a random value
x = seed
Repeat
      x = (ax + b) mod n
      print x
Until x = seed
```

    Execute the loop by hand for $a = 3$, $b = 7$, $n = 11$ and $seed = 0$. How "random" are these random numbers?

12. Write down the $\cdot_7$ multiplication table for Z_7.

13. Prove the equalities for multiplication in Lemma 2.1.2

14. State and prove the associative law for $\cdot_n$ multiplication.

15. State and prove the distributive law(s) (Why is that "s" in parentheses anyhow? Do you need to prove more than one?) for $\cdot_n$ multiplication over $+_n$ addition.

2.2 Inverses and GCDs

Inverses mod p

In the last section we explored multiplication in Z_n. We saw in the special case with $n = 12$ and $a = 4$ that if we used multiplication by a in Z_n to encrypt a message, then our receiver would need to be able to solve, for x, the equation $a \cdot_n x = b$ in order to decode a received message b. In the end of section exercises there are exercises that show that with some values of n, equations of the form $a \cdot_n x = b$ have a unique solution, while for other values of n we could have equations with no solutions, or equations with more than one solution. Notice that if $n = mk$, then the equation $m \cdot_n x = 0$ will always have at least the two solutions, k and 0. Thus if we want the equation $a \cdot_n x = b$ to have a unique solution in Z_n for every $a \neq 0$ and every b in Z_n, then n must be a prime number.

If you experimented with the prime numbers 5, 7, and 11 in the last set of problems (and if you didn't, now would be a great time to do so), you probably recognized that what is relevant for solving the equation $a \cdot_n x = b$ is the question of whether a has a *multiplicative inverse* in Z_n, that is, whether there is another number a' such that $a' \cdot_n a = 1$. If a does have the inverse a', then the unique solution to the equation $a \cdot_n x = b$ is $a' \cdot_n b$. Once we realize the importance of inverses, we find it relatively easy to make computations that convince us of that every nonzero element in Z_5, Z_7, and Z_{11} does have a multiplicative inverse, so every equation of the form $a \cdot_n x = b$ (with $a \neq 0$) does have a unique solution. The evidence we have for $p = 5$, 7, and 11 suggests that whenever p is a prime then every nonzero element of Z_p has an inverse in Z_p, and therefore every equation of the form $a \cdot_p x = b$ (with $a \neq 0$) has a unique solution. This leads us to focus on trying to show that for each prime p, each element of Z_p has a multiplicative inverse.

Thus we are interested in showing that for each nonzero a in Z_p, the equation

$$a \cdot_p x = 1$$

has a solution. What does this equation mean, though? We can re-express it as

$$a \cdot_p x -_p 1 = 0,$$

or

$$(ax - 1) \bmod p = 0.$$

This means that $ax - 1$ is a multiple of p, so that there is some integer y such that $ax - 1 = py$, or

$$ax - py = 1. \tag{2.6}$$

It appears this is no help; we have converted the problem of solving (in Z_p) the equation $a \cdot_p x = 1$, an equation with just one variable x (that could only have $p - 1$ different values), to a problem of solving Equation 2.6, which has two variables. Further, in this second equation , y can take on any integer value. This seems to have made our work harder, not easier.

Fortunately, the greatest common divisor algorithm, first introduced by Euclid, helps us find x and y. We will discuss this algorithm later in this section.

Greatest Common Divisors (GCD)

2.2-1 Suppose m is not a prime, and a and x are integers such that $a \cdot_m x = 1$ in Z_m. What equation involving m in the integers does this give us?

2.2-2 Suppose that a and m are integers such that $ax - my = 1$, for some integers x and y. What does that tell us about being able to find a (multiplicative) inverse for a (mod m)?

2.2-3 If $ax - my = 1$ for integers x and y, can a and m have any common divisors other than 1 and -1?

In Exercise 2.2-1 we saw that if the equation

$$a \cdot_m x = 1$$

has a solution, then there is a number y such that

$$ax - my = 1.$$

In Exercise 2.2-2, we saw that if x and y are integers such that $ax - my = 1$, then $ax - 1$ is a multiple of m, and so x is a multiplicative inverse of a in Z_m. Thus we have actually proved the following:

Theorem 2.2.1 *A number a has a multiplicative inverse in Z_m if and only if there are integers x and y such that $ax - my = 1$.*

In Exercise 2.2-3, we saw that, given a and m, if we can find integers x and y such that $ax - my = 1$, then there are no common integer divisors to a and m except for 1 and -1. We say that "the greatest common divisor of a and m is 1." In general, the *greatest common divisor* of two numbers k and j is the largest number d that is a factor of both k and j.[4] We denote the greatest common divisor of k and j by $\gcd(k, j)$.

Euclid's Division Theorem

One of the important tools in understanding greatest common divisors is Euclid's division theorem, a result which we also used in the last section. While it appears obvious, as do many theorems in number theory, it follows from simpler principles of number theory, and the proof helps us understand how the greatest common divisor algorithm works. Thus we present a proof here.

Lemma 2.2.2 *(Euclid's division theorem). If k and n are positive integers, then there are non-negative integers q and r with $r < n$ such that $k = qn + r$.*

[4]There is one common factor of k and j for sure, namely 1. No common factor can be larger than the smaller of k and j in absolute value, and so there must be a largest common factor.

Proof: To prove this Lemma, assume instead, for purposes of contradiction, that it is false. Among all pairs (k, n) that make it false, choose the smallest k that makes it false. We cannot have $k < n$ because then the statement would be true with $q = 0$ and $r = k$, and we cannot have $k = n$ because then the statement is true with $q = 1$ and $r = 0$. This means $k - n$ is a positive number smaller than k. We assumed that k was the smallest value that made the lemma false, and so the lemma must be true for the pair $(k - n, n)$. Therefore, there must exist a q' and r' such that

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

Thus $k = (q' + 1)n + r'$, and by setting $q = q' + 1$ and $r = r'$, we can satisfy the lemma for the pair (k, n), contradicting the assumption that the statement is false. Thus the only possibility is that the statement is true. ■

We call proof technique used here *proof by smallest counterexample*. In this method,we assume, as in all proofs by contradiction, that the lemma is false. This implies that there must be a *counterexample* which does not satisfy the conditions of the lemma. In this case that counterexample consists of numbers k and n such that *no* q and r exists which satisfy $k = qn + r$. Further, if there are counterexamples, then there must be one that is smallest in some sense. (Here being smallest means having the smallest k.) We choose this smallest one, and then reason that if it exists, then every smaller example is true. If we can then use a smaller true example to show that our supposedly false example is true as well, we have created a contradiction. The only thing this can contradict is our assumption that the lemma was false. Therefore this assumption has to be invalid, and the lemma has to be true. As we will see later in the course, this method is actually closely related to a concept called *proof by induction* and to recursive algorithms. In essence, the proof of Lemma 2.2.2 describes a recursive program to find q and r in the Lemma above so that $r < n$. The connection with greatest common divisors appears in the following lemma.

Lemma 2.2.3 *If k, q, n, and r are positive integers such that $k = nq + r$ then* $\gcd(k, n) = \gcd(n, r)$.

Proof: If d is a factor of k and n, then there are integers i_1 and i_2 so that $k = i_1 d$ and $n = i_2 d$. Thus d is also a factor of $r = k - nq = i_1 d - i_2 dq = (i_1 - i_2 q)d$. Similarly, if d is a factor of n and r, then it is a factor of $k = nq + r$. Thus the greatest common divisor of k and n is a factor of the greatest common divisor of k and r, and vice versa, so they must be equal.■

This reduces our problem of finding $\gcd(k, n)$ to the simpler (in a recursive sense) problem of finding $\gcd(n, r)$. (Notice the assumption in our lemma that q, n, and r are positive implies that $n < k$. Since Lemma 2.2.2 tells us that we may assume $r < n$, we have reduced the size of the larger of the two numbers whose greatest common divisor we want.)

The GCD Algorithm

2.2-4 Write a recursive algorithm to find $\gcd(k, n)$. Use it (by hand) to find the GCD of 252 and 189.

Our algorithm for Exercise 2.2-4 is based on Lemma 2.2.3 and the observation that if $k = nq$, for any q, then $n = \gcd(k, n)$. It is convenient to assume that $n \leq k$, so if this is not the case, we exchange k and n. We first write $k = nq + r$ in the usual way. If $r = 0$, then we return n as the

greatest common divisor. Otherwise,we apply our algorithm to find the greatest common divisor of n and r. Finally, we return the result as the greatest common divisor of k and n.

As an example, consider finding

$$\gcd(24, 14).$$

We can write

$$24 = 1(14) + 10.$$

In this case $k = 24$, $n = 14$, $q = 1$ and $r = 10$. Thus we can apply Lemma 2.2.3 and conclude that

$$\gcd(24, 14) = \gcd(14, 10).$$

We therefore continue our computation of $\gcd(14, 10)$, by writing $14 = 10 \cdot +4$, and have that

$$\gcd(14, 10) = \gcd(10, 4).$$

Now,

$$10 = 4 \cdot 2 + 2,$$

and so

$$\gcd(10, 4) = \gcd(4, 2).$$

Now

$$4 = 2 \cdot 2 + 0,$$

so that 2 is a divisor of four and is thus the greatest common divisor of 2 and 4. Since 2 is a divisor of 4, this last equation forms our "base case". Thus we have that

$$\gcd(24, 14) = \gcd(4, 2) = 2.$$

By analyzing our process in a bit more detail, we will be able to return not only the greatest common divisor, but also numbers x and z such that $\gcd(k, n) = kx + nz$. [5]

In the case that $k = nq$ and we want to return n as our greatest common divisor, we also want to return 0 for the value of x and 1 for the value of z. Suppose we are now in the case that that $k = nq + r$ with $0 < r < n$. Then we recursively compute $\gcd(n, r)$ and in the process get an x' and a z' such that $\gcd(n, r) = nx' + rz'$. Since $r = k - nq$, we get by substitution that

$$\gcd(n, r) = nx' + (k - nq)z' = kz' + n(x' - qz').$$

Thus when we return $\gcd(n, r)$ as $\gcd(k, n)$, we want to return the value of z' as x and and the value of $x' - qz'$ as z.

Finally, if we exchanged k and n to begin our process, we exchange the x and z values that were returned, and we have our greatest common divisor *and* our x and z with

$$\gcd(k, n) = kx + nz.$$

We will refer to the process we just described as "Euclid's extended GCD algorithm."

[5] We could fix things so that $\gcd(k, n) = kx - ny$ as we want for use above, but we have chosen to use the traditional equation with the plus sign to avoid confusing the reader who consults other texts. The negative of the z we get here will be the y we wanted in Section 2.2 .

2.2-5 Apply Euclid's extended GCD algorithm to find numbers x and z such that the GCD of 24 and 14 is $24x + 14z$.

For our discussion of Exercise 2.2-5 we give pseudocode for the extended GCD algorithm. While it is possible to express the algorithm more concisely by using recursion, we will give an iterative version that is longer but can make the computational process clearer. Instead of using the variables q, n, k, r, x and z, we will use six arrays, where $q(i)$ is the value of q computed on the ith iteration, and so forth. We will use the index zero for the input values, that is $k(0)$ and $n(0)$ will be the numbers whose gcd we want. Eventually $x(0)$ and $z(0)$ will become the x and z we want.

```
gcd(k, n)
// assume that k > n
i = 0; k(i) = k; n(i) = n
Repeat
      q(i) = ⌊k(i)/n(i)⌋
      r(i) = k(i) − q(i)n(i)
      k(i + 1) = n(i); n(i + 1) = r(i)
      i = i + 1
Until (r(i − 1) == 0) // we have found the value of the gcd, now we compute the x and z
i = i − 1
gcd = n(i)
x(i) = 0; z(i) = 1
i = i − 1
While (i ≥ 0)
      x(i) = z(i + 1)
      z(i) = x(i + 1) − q(i)z(i + 1)
      i = i − 1
Return gcd
```

We give the details of how this algorithm applies to $\gcd(24, 14)$.

i	$k(i)$	$n(i)$	$q(i)$	$r(i)$	$x(i)$	$z(i)$
0	24	14	1	10	3	−5
1	14	10	1	4	−2	3
2	10	4	2	2	1	−2
3	4	2	2	0	0	1
gcd = 2						

In a row, the $q(i)$ and $r(i)$ values are computed from the $k(i)$ and $n(i)$ values. Then the $n(i)$ and $r(i)$ are passed down to the next row as $k(i + 1)$ and $n(i + 1)$ respectively. This process continues until we finally reach a case where $k(i) = q(i)n(i)$ and we can answer $n(i)$ for the gcd. Once we have done this, we can then compute $x(i)$ and $z(i)$ going up. In the bottom row, we have that $x(i) = 0$ and $z(i) = 1$. Then, we compute $x(i)$ and $z(i)$ for a row by setting $x(i)$ to $z(i+1)$ and $z(i)$ to $x(i+1) - q(i)z(i+1)$. We note that in every row, we have the property that $k(i)x(i) + n(i)z(i) = \gcd(k, n)$.

Theorem 2.2.4 *Two positive integers k and n have greatest common divisor 1 if and only if there are integers x and z such that $kx + nz = 1$.*

Proof: We see, as earlier, that if there are integers x and z such that $kx + nz = 1$, then $\gcd(k, n) = 1$, because any common factor of k and n would have to be a factor of 1, and so 1 and -1 are the only possible common factors. (This proves the "if" part of the theorem.)

On the other hand, we just showed above that given positive integers k and n, there are integers x and z such that $\gcd(k, n) = kx + nz$. Therefore $\gcd(k, n) = 1$ only if there are integers x and z such that $kx + nz = 1$. ■

Corollary 2.2.5 $\gcd(a, m) = 1$ *if and only if there are integers x and y such that $ax - my = 1$.*

Proof: We use the theorem with $k = a$ and $n = m$, and then let $y = -z$. ■

Corollary 2.2.6 *For any positive integer m, an element a of Z_m has an inverse if and only if* $\gcd(a, m) = 1$.

In fact, we find the multiplicative inverse by finding the x and y in Corollary 2.2.5 by Euclid's extended GCD algorithm, and then x is the inverse.

Problems

1. If $a \cdot 133 - m \cdot 277 = 1$, does this guarantee that a has an inverse mod m? If so, what is it? If not, why not?

2. If $a \cdot 133 - m \cdot 277 = 1$, what can you say about all possible common divisors of a and m?

3. Bob and Alice want to choose a key they can use for cryptography, but all they have to communicate is a bugged phone line. Bob proposes that they each choose a secret number, a for Alice and b for Bob. They also choose, over the phone, a prime number p with more digits than any key they want to use, and one more number q. Bob will send Alice $bq \pmod p$, and Alice will send Bob $aq \pmod p$. Their key (which they will keep secret) will then be $abq \pmod p$. (Here we don't worry about the details of how they use their key, only with how they choose it.) As Bob explains, their wire tapper will know p, q, $aq \pmod p$, and $bq \pmod p$, but will not know a or b, so their key should be safe.

   Is this scheme safe, that is can the wiretapper compute $abq \bmod p$? If so, how does she do it.

   Alice says "You know, the scheme sounds good, but wouldn't it be more complicated for the wire tapper if I send you $q^a \pmod p$, you send me $q^b \pmod p$ and we use $q^{ab} \pmod p$ as our key?" In this case can you think of a way for the wire tapper to compute $q^{ab} \pmod p$. If so, how can you do it? If not, what is the stumbling block? (It is fine for the stumbling block to be that you don't know how to compute something, you don't need to prove that you can't compute it.)

4. Write pseudocode for a recursive version of the extended GCD algorithm.

5. Run Euclid's extended GCD algorithm to compute $\gcd(576, 486)$. Show all the steps.

6. Use Euclid's extended GCD algorithm to compute the multiplicative inverse of 16 modulo 103.

7. The Fibonacci numbers F are defined as follows:

$$F_i = \begin{cases} 1 & \text{if } i \text{ is 1 or 2} \\ F_{i-1} + F_{i-2} & \text{otherwise.} \end{cases}$$

   What happens when you run Euclid's extended GCD algorithm on F_i and F_{i-1}. (We are asking about the execution of the algorithm, not just the answer.)

8. Write (and run on several different inputs) a program to implement Euclid's extended GCD algorithm. Be sure to return x and z in addition to the GCD. About how many times does your program have to make a recursive call to itself? What does that say about how long we should expect it to run as we increase the size of the k and n whose GCD we are computing.

9. The least common multiple of two numbers x and y is the smallest number z such that z is an integer multiple of both x and y. Give a formula for the least common multiple that involves the GCD.

2.3 The RSA Cryptosystem

Exponentiation mod n

The idea behind RSA encryption is *exponentiation* in Z_n. We can define, for $a \in Z_n$, and a positive integer j:

$$a^j = \underbrace{a \cdot_n a \cdot_n \cdots \cdot_n a}_{j \text{ factors}}. \tag{2.7}$$

In other words a^j is the product in Z_n of j factors, each equal to a. We also define a^0 to be 1. We use the same notation (syntax) for exponentiation mod n as for normal exponentiation. It will be clear from the context which is meant.[6] *Notice that exponents are integers, and not members of Z_n.*

While this definition only applies to non-negative exponents, we can sometimes define exponentiation for negative exponents. Since we don't need negative exponents to discuss RSA encryption, we will not explore them at this point.

The Rules of Exponents

One reason we use the notation a^j is that the normal rules of manipulating exponents apply, i.e.,

$$a^i \cdot_p a^j = a^{i+j},$$

and

$$(a^i)^j = a^{ij}.$$

We explain the first of these rules in the proof of the lemma below, and leave the second to you in the exercises at the end of the section.[7]

Lemma 2.3.1 *For any $a \in Z_n$, and any nonnegative integers i and j,*

$$a^i \cdot_n a^j = a^{i+j} \tag{2.8}$$

and

$$(a^i)^j = a^{ij}. \tag{2.9}$$

Proof: Equation 2.8 follows immediately from Equation 2.7 because a product of i factors equal to a times a product of j factors equal to a is a product of $i + j$ factors all equal to a. The proof of Equation 2.9 is left for the exercises at the end of the section. ■

[6] In fact most sources you read will use $+$ rather than $+_n$ and $\cdot$ or juxtaposition rather than $\cdot_n$. You will find that you can also quickly determine from context which is meant.

[7] We remarked earlier that exponents are integers, not members of Z_n, so the sum and product in the rules of exponents are ordinary addition and multiplication. Throughout this chapter, sums, differences and products appearing in exponents will be sums, differences and products in the integers.

Fermat's Little Theorem

2.3-1 Compute the powers of 2 mod 7. What do you observe? Now compute the powers of 3 mod 7. What do you observe?

2.3-2 Compute the numbers $1 \cdot 2 \bmod 7$, $2 \cdot 2 \bmod 7$, $3 \cdot 2 \bmod 7$, $4 \cdot 2 \bmod 7$, $5 \cdot 2 \bmod 7$, $6 \cdot 2 \bmod 7$. Do you observe anything interesting? Now compute the numbers $1 \cdot 3 \bmod 7$, $2 \cdot 3 \bmod 7$, $3 \cdot 3 \bmod 7$, $4 \cdot 3 \bmod 7$, $5 \cdot 3 \bmod 7$, $6 \cdot 3 \bmod 7$. Do you observe anything interesting?

2.3-3 Suppose we choose an arbitrary nonzero number a between 1 and 6. Are the numbers $1 \cdot a \bmod 7$, $2 \cdot a \bmod 7$, $3 \cdot a \bmod 7$, $4 \cdot a \bmod 7$, $5 \cdot a \bmod 7$, $6 \cdot a \bmod 7$ all different? That is, are the elements $1 \cdot_7 a$, $2 \cdot_7 a$, $3 \cdot_7 a$, $4 \cdot_7 a$, $5 \cdot_7 a$, $6 \cdot_7 a$ all different? Why or why not?

In Exercise 2.3-1, we have that

$$\begin{aligned} 2^0 \bmod 7 &= 1 \\ 2^1 \bmod 7 &= 2 \\ 2^2 \bmod 7 &= 4 \\ 2^3 \bmod 7 &= 1 \\ 2^4 \bmod 7 &= 2 \\ 2^5 \bmod 7 &= 4 \\ 2^6 \bmod 7 &= 1 \\ 2^7 \bmod 7 &= 2 \,. \end{aligned}$$

Continuing we see that the powers of 2 will cycle through the three values $\{1, 2, 4\}$. Performing the same computation for 3, we have

$$\begin{aligned} 3^0 \bmod 7 &= 1 \\ 3^1 \bmod 7 &= 3 \\ 3^2 \bmod 7 &= 2 \\ 3^3 \bmod 7 &= 6 \\ 3^4 \bmod 7 &= 4 \\ 3^5 \bmod 7 &= 5 \\ 3^6 \bmod 7 &= 1 \\ 3^7 \bmod 7 &= 3 \,. \end{aligned}$$

In this case, we will cycle through the six values $\{1, 3, 2, 6, 4, 5\}$. In particular, we see that in Z_7, $2^6 = 1$ and $3^6 = 1$. Of course this immediately gives us that $(2 \cdot_7 2)^6 = 4^6 = 1$ and $(2 \cdot_7 3)^6 = 6^6 = 1$ in Z_7. What about 5^6? Notice that $3^5 = 5$ in Z_7 by the computations we made above. Using Equation 2.9 twice, this gives us

$$5^6 = (3^5)^6 = 3^{5 \cdot 6} = 3^{6 \cdot 5} = (3^6)^5 = 1^5 = 1$$

in Z_7. Thus the sixth power of each element of Z_7 is 1.

In Exercise 2.3-2 we see that

$$\begin{aligned} 1 \cdot 2 \bmod 7 &= 1 \cdot_7 2 = 2 \\ 2 \cdot 2 \bmod 7 &= 2 \cdot_7 2 = 4 \\ 3 \cdot 2 \bmod 7 &= 3 \cdot_7 2 = 6 \\ 4 \cdot 2 \bmod 7 &= 4 \cdot_7 2 = 1 \\ 5 \cdot 2 \bmod 7 &= 5 \cdot_7 2 = 3 \\ 6 \cdot 2 \bmod 7 &= 6 \cdot_7 2 = 5. \end{aligned}$$

Thus these numbers are a permutation of the set $\{1, 2, 3, 4, 5, 6\}$. Similarly,

$$\begin{aligned} 1 \cdot 3 \bmod 7 &= 1 \cdot_7 3 = 3 \\ 2 \cdot 3 \bmod 7 &= 2 \cdot_7 3 = 6 \\ 3 \cdot 3 \bmod 7 &= 3 \cdot_7 3 = 2 \\ 4 \cdot 3 \bmod 7 &= 4 \cdot_7 3 = 5 \\ 5 \cdot 3 \bmod 7 &= 5 \cdot_7 3 = 1 \\ 6 \cdot 3 \bmod 7 &= 6 \cdot_7 3 = 4. \end{aligned}$$

. Again we get a permutation of $\{1, 2, 3, 4, 5, 6\}$.

In Exercise 2.3-3 we are asked whether this is always the case. Notice that since 7 is a prime, any nonzero number between 1 and 6 has a mod 7 multiplicative inverse a'. Thus if i and j were integers in Z_7 with $i \cdot_7 a = j \cdot_7 a$, we multiply mod 7 on the right by a' to get

$$(i \cdot_7 a) \cdot_7 a') = (j \cdot_7 a) \cdot_7 a').$$

Then w and after using the associative law we get

$$i \cdot_7 (a \cdot_7 a') = j \cdot_7 (a \cdot_7 a'). \tag{2.10}$$

Since $a \cdot_7 a' = 1$, Equation 2.10 simply becomes $i = j$. In other words all the values ia for $i = 1, 2, 3, 4, 5, 6$ must be different. Since we have six different values, all between 1 and 6, we have that the values ia for $i = 1, 2, 3, 4, 5, 6$ are a permutation of $\{1, 2, 3, 4, 5, 6\}$.

As you can see, the only fact we used in our analysis of Exercise 2.3-3 is that if p is a prime, then any number between 1 and $p-1$ has a multiplicative inverse in Z_p. In other words, we have really proved the following lemma.

Lemma 2.3.2 *Let p be a prime number. For any fixed nonzero number a in Z_p, the numbers $(1 \cdot a) \bmod p$, $(2 \cdot a) \bmod p$, ..., $((p-1) \cdot a) \bmod p$, are a permutation of the set $\{1, 2, \cdots, p-1\}$.*

With this lemma in hand, we can prove a famous theorem that explains the phenomenon we saw at then end of the analysis of Exercise 2.3-1.

Theorem 2.3.3 *(Fermat's little theorem). Let p be a prime number. Then $a^{p-1} = 1$ in Z_p for each nnonzero a in Z_p.*

Proof: Since p is a prime, Lemma 2.3.2 tells us that the numbers $1 \cdot_p a$, $2 \cdot_p a$, $\ldots$, $(p-1) \cdot_p a$, are a permutation of the set $\{1, 2, \cdots, p-1\}$. But then

$$1 \cdot_p 2 \cdot_p \cdots \cdot_p (p-1) = (1 \cdot_p a) \cdot_p (2 \cdot_p a) \cdot_p \cdots \cdot_p ((p-1) \cdot_p a).$$

Using the commutative and associative laws for multiplication of integers, we get

$$1 \cdot_p 2 \cdot_p \cdots \cdot_p (p-1) = 1 \cdot_p 2 \cdot_p \cdots \cdot_p (p-1) \cdot_p a^{p-1}.$$

Now we multiply both sides of the equation by the multiplicative inverses in Z_p of 2, 3, $\ldots$, $p-1$ and the left hand side of our equation becomes 1 and the right hand side becomes a^{p-1} (in Z_p). But this is exactly the conclusion of our theorem. ■

Corollary 2.3.4 *(Fermat's little Theorem, version 2) For every positive integer a, and prime p,*

$$a^{p-1} \bmod p = 1.$$

Proof: This is a direct application of Lemma 2.1.2, because if we replace a by $a \bmod p$, then Theorem 2.3.3 applies.

The RSA Cryptosystem

Fermat's little theorem is at the heart of the RSA cryptosystem, a system that allows Alice to tell the world a way that they can encode a message to send to her so that she and only she can read it. In other words, even though she tells everyone how to encode the message, nobody except Alice and the person who sent the message has a significant chance of figuring out what the message is from looking at the encoded message. What Alice is giving out is called a "one-way function." This is a function f that has an inverse f^{-1}, but even though $y = f(x)$ is reasonably easy to compute, nobody but Alice (who has some extra information that she keeps secret) can compute $f^{-1}(y)$. Thus someone who has a message x to send Alice computes $f(x)$ and sends it to Alice, who uses her secret information to compute $f^{-1}(f(x)) = x$.

In the RSA cryptosystem Alice chooses two prime numbers p and q (which in practice each have at least a hundred digits) and computes the number $n = pq$. She also chooses a number $e \neq 1$ which need not have a large number of digits but is relatively prime to $(p-1)(q-1)$, so that it has an inverse d, and she computes $d = e^{-1} \bmod (p-1)(q-1)$. Alice publishes e and n.

To summarize what we just said, here is a pseudocode outline of what Alice does:

```
RSA key choice algorithm
Choose 2 large prime numbers p and q
n = pq
Choose e ≠ 1 so that e is relatively prime to (p − 1)(q − 1)
Compute d = e^{-1} mod (p − 1)(q − 1).
Publish e and n.
```

People who want to send a message x to Alice compute $y = x^e \bmod n$ and send that to her instead. (We assume x has fewer digits than n so that it is in Z_n. If not, the sender has to

break the message into blocks of size less than the number of digits of n and send each block individually.)

To decode the message, Alice will compute $z = y^d \bmod n$.

In order to show that RSA is correct, we must show that $z = x$, i.e., when Alice decodes, she gets back the original message. In order to show that RSA is secure, we must argue that an eavesdropper, who knows n, e, and y, but does not know p, q or e, can not easily compute x.

2.3-4 Why is $y^d \bmod p = x \bmod p$? Does this tell us what x is?

Plugging in the value of y, we have

$$y^d = x^{ed}. \tag{2.11}$$

But we chose e and d so that $e \cdot_m d = 1$, where $m = (p-1)(q-1)$. In other words,

$$ed \bmod (p-1)(q-1) = 1.$$

Therefore, for some integer r,

$$ed = r(p-1)(q-1) + 1 \ .$$

Plugging this into Equation (2.11), we obtain

$$\begin{aligned} x^{ed} &= x^{r(p-1)(q-1)+1} \\ &= x^{(r(q-1))(p-1)}x. \end{aligned}$$

But for any number z, $z^{p-1} \bmod p = 1$ by Fermat's little theorem. In this case our z is $x^{r(q-1)}$, and we have that

$$x^{(r(q-1))(p-1)} \bmod p = x^{(r(q-1))^{(p-1)}} \bmod p = 1$$

and hence $y^d \bmod p = x \bmod p$.

The same reasoning shows us that $y^d \bmod q = x \bmod q$. What is not immediately clear is whether these two facts tell us anything about $y^d \bmod pq$, which is what we said Alice would compute. However they actually do tell us exactly what we want to know.

Notice that by Lemma 2.1.2 we have proved that

$$(y^d - x) \bmod p = 0 \tag{2.12}$$

and

$$(y^d - x) \bmod q = 0. \tag{2.13}$$

2.3-5 Write down an equation using only integers, addition, subtraction and multiplication, but perhaps more symbols, that is equivalent to Equation 2.12.

2.3-6 Write down an equation using only integers, addition, subtraction and multiplication, but perhaps more symbols, that is equivalent to Equation 2.13.

2.3-7 If a number is a multiple of a prime p and a different prime q, then what else is it a multiple of? What does this tell us about y^d and x?

The statement that $y^d - x \bmod p = 0$ is equivalent to saying that $y^d - x = kp$ for some integer k. The statement that $y^d - x \bmod q = 0$ is equivalent to saying $y^d - x = mq$ for some integer m. If something is a multiple of the prime p and the prime q, then it is a multiple of pq. Thus $(y^d - x) \bmod pq = 0$. Now $y^d \bmod pq$ and x are both members of Z_{pq}. Lemma 2.1.2 tells us that $(y^d - x) \bmod pq = ((y^d \bmod pq - x) \bmod pq = 0$. Therefore $y^d \bmod pq - x = 0$ in Z_{pq}, or $y^d \bmod pq = x$ in Z_{pq}. In other words,

$$\begin{aligned} x &= y^d \bmod pq \\ &= y^d \bmod n \ , \end{aligned}$$

which means that Alice will in fact get the correct answer.

Theorem 2.3.5 *(Rivest, Shamir, and Adleman) The RSA procedure for encoding and decoding messages works correctly.*

Proof: Proved above.■

One might ask why, given that Alice published e and n, and messages are encrypted by computing $x^e \bmod n$, why can't any adversary who learns $x^e \bmod n$ just compute eth roots mod n and break the code. At present, nobody knows a quick scheme for computing eth roots mod n, for an arbitrary n. Someone who does not know p and q cannot duplicate Alice's work and discover x. Thus, as far as we know, modular exponentiation is an example of a one-way function.

The Chinese Remainder Theorem

The method we used to do the last step of the proof of Theorem 2.3.5 also proves a theorem known as the "Chinese Remainder Theorem."

2.3-8 For each number in $x \in Z_{15}$, write down $x \bmod 3$ and $x \bmod 5$. Is x uniquely determined by these values? Can you explain why this is true?

It turns out that each of the $3 \cdot 5 = 15$ pairs (i, j) of integers i, j with $0 \leq i \leq 2$ and $0 \leq j \leq 4$ occurs exactly once as x ranges through the fifteen integers from 0 to 14. Thus the function f given by $f(x) = (x \bmod 3, x \bmod 5)$ is a one-to-one function from a fifteen element set to a fifteen element set, so each x is uniquely determined by its pair of remainders.

The Chinese Remainder Theorem tells us that this observation always holds.

Theorem 2.3.6 *(Chinese Remainder Theorem) If m and n are relatively prime integers and $a \in Z_m$ and $b \in Z_n$, then the equations*

$$x \bmod m = a \tag{2.14}$$

$$x \bmod n = b \tag{2.15}$$

have one and only one solution for an integer x between 0 and $mn - 1$.

Proof: If we show that as x ranges over the integers from 0 to $mn - 1$, then the ordered pairs $(x \bmod m, x \bmod n)$ are all different, then we will have shown that the function given by $f(x) = (x \bmod m, x \bmod n)$ is a one to one function from an mn element set to an mn element set, so it is onto as well.[8] In other words, we will have shown that each pair of equations 2.14

[8] If the function weren't onto, then because the number of pairs is the same as the number of possible x-values, two x values would have to map to the same pair, so the function wouldn't be one-to-one after all.

and 2.15 has one and only one solution.

In order to show that f is one-to-one, we must show that if x and y are different numbers between 0 and $mn-1$, then $f(x)$ and $f(y)$ are different. To do so, assume instead that we have an x and y with $f(x) = f(y)$. Then $x \bmod m = y \bmod m$ and $x \bmod n = y \bmod n$, so that $(x-y) \bmod m = 0$ and $(x-y) \bmod n = 0$. That is, $x-y$ is a multiple of both m and n. Then as we show in Exercise 2.3-11, $x-y$ is a multiple of mn; that is $x-y = dmn$ for some integer d. Since we assumed x and y were different, this means x and y cannot both be between 0 and $mn-1$ because their difference is mn or more. This contradicts our hypothesis that x and y were different numbers between 0 and $mn-1$, so our assumption must be incorrect; that is f must be one-to-one. This completes the proof of the theorem. ■

Problems

1. Compute the powers of 4 in Z_7. Compute the powers of 4 in Z_{10}. What is the most striking similarity? What is the most striking difference?

2. Compute the numbers $1 \cdot_{11} 5$, $2 \cdot_{11} 5$, $3 \cdot_{11} 5$, ..., $10 \cdot_{11} 5$. Do you get a permutation of the set $\{1,2,3,4,5,6,7,8,9,10\}$? Would you get a permutation of the set $\{1,2,3,4,5,6,7,8,9,10\}$ if you used another nonzero member of of $Z_1 1$ in place of 5?

3. The numbers 29 and 43 are primes. What is $(29-1)(43-1)$? What is $199 \cdot 1111$ in Z_{1176}? What is $(23^{1111})^{199}$ in Z_{29}? In Z_{43}? In Z_{1247}?

4. The numbers 29 and 43 are primes. What is $(29-1)(43-1)$? What is $199 \cdot 1111$ in Z_{1176}? What is $(105^{1111})^{199}$ in Z_{29}? In Z_{43}? In Z_{1247}? How does this answer the second question in Exercise 2.3-4?

5. How many solutions with x between 0 and 34 are there to the system of equations

$$\begin{aligned} x \bmod 5 &= 4 \\ x \bmod 7 &= 5? \end{aligned}$$

   What are these solutions?

6. Compute each of the following. Show your work, and do not use a calculator or computer.

   (a) 15^{96} in Z_{97}

   (b) 67^{72} in Z_{73}

   (c) 67^{73} in Z_{73}

7. Show that in Z_p, with p prime, if $a^i = 1$, then $a^n = a^{n \bmod i}$.

8. Show that for any nonnegative integers i and j , $(a^i)^j = a^{ij}$ in Z_n.

9. Show that there are $p^2 - p$ elements with multiplicative inverses in Z_{p^2} when p is prime. If x has a multiplicative inverse, what is $x^{p^2-p} \bmod p^2$? Is the same statement true for an element without an inverse? (Working out an example might help here.) Can you find something that is true about x^{p^2-p} when x does not have an inverse?

10. How many elements have multiplicative inverses in Z_{pq} when p and q are primes?

11. In the paragraph preceding the proof of Theorem 2.3.5 we said that af a number is a multiple of the prime p and the prime q, then it is a multiple of pq. We will see how that is proved here.

    (a) What equation in the integers does Euclid's extended GCD algorithm solve for us when m and n are relatively prime?

    (b) Suppose that m and n are relatively prime and that k is a multiple of each one of them; that is, $k = bm$ and $k = cn$ for integers b and c. If you multiply both sides of the equation in part a by k, you get an equation expressing k as a sum of two products. By making appropriate substitutions in these terms, you can show that k is a multiple of mn. Do so. Does this justify the assertion we made in the paragraph preceding the proof of Theorem 2.3.5?

12. The relation of "congruence modulo n" is the relation $\equiv$ defined by $x \equiv y \pmod{n}$ if and only if $x \bmod n = y \bmod n$.

    (a) Show that congruence modulo n is an equivalence relation by showing that it defines a partition of the integers into equivalence classes.

    (b) Show that congruence modulo n is an equivalence relation by showing that it is reflexive, symmetric, and transitive.

    (c) Express the Chinese Remainder theorem in the notation of congruence modulo n.

13. Write and implement code to do RSA encryption and decryption. Use it to send a message to someone else in the class. (You may use smaller numbers for the sake of efficiency.)

2.4 Details of the RSA Cryptosystem

Practical Aspects of Exponentiation

Suppose you are going to raise a 100 digit number a to the 10^{120}th power modulo a 200 digit integer n. Note that the exponent is thus a 121 digit number. (Saying that we are doing arithmetic modulo n is the same as saying that we are doing arithmetic in Z_n.)

2.4-1 Propose an algorithm to compute $a^{10^{120}} \bmod n$, where a is a 100 digit number and n is a 200 digit number?

2.4-2 What can we say about how long this algorithm would take on a computer that can do one infinite precision arithmetic operation in constant time?

2.4-3 What can we say about how long this would take on a computer that can multiply integers in time proportional to the product of the number of digits in the two numbers, i.e. multiplying an x-digit number by a y-digit number takes roughly xy time.

Notice that if we form the sequence a, a^2, a^3, a^4, a^5, a^6, a^7,a^8, a^9, a^{10}, we are modeling the process of forming a^{10} by successively multiplying by a. If, on the other hand, we form the sequence a, a^2, a^4, a^8, a^{16}, a^{32}, a^{64}, a^{128}, a^{256}, a^{512}, a^{1024}, we are modeling the process of successive squaring, and in the same number of multiplications we are able to get a raised to a three digit number. Each time we square we double the exponent, so every ten steps or so we will add three to the number of digits of the exponent. Thus in a bit under 400 multiplications, we will get $a^{10^{120}}$. This suggests that our algorithm should be to square a some number of times until the result is almost $a^{10^{120}}$, and then multiply by some smaller powers of a until we get exactly what we want. More precisely, we square a and continue squaring the result until we get the largest $a^{2^{k_1}}$ such that 2^{k_1} is less than 10^{120}, then multiply $a^{2^{k_1}}$ by the largest $a^{2^{k_2}}$ such that $2^{k_1} + 2^{k_2}$ is less than 10^{120} and so on until we have

$$10^{120} = 2^{k_1} + 2^{k_2} + \cdots + 2^{k_r}$$

for some integer r. (Can you connect this with the binary representation of 10^{120}?) Then we get

$$a^{10^{120}} = a^{2^{k_1}} a^{2^{k_2}} \cdots a^{2^{k_r}}.$$

Notice that all these powers of a have been computed in the process of discovering k_1. Thus it makes sense to save them as you compute them.

To be more concrete, lets see how to compute a^{43}. $43 = 32 + 8 + 2 + 1$, and thus

$$a^{43} = a^{2^5} a^{2^3} a^{2^1} a^{2^0}.$$

So, we first compute $a^{2^0}, a^{2^1}, a^{2^2}, a^{2^3} a^{2^4}, a^{2^5}$, using 5 multiplications. Then we can compute a^{43} using 3 additional multiplications. You can see the obvious savings over doing 43 multiplications.

On a machine that could do infinite precision arithmetic in constant time, we would need about $\log_2(10^{120})$ steps to compute all the powers a^{2^i}, and perhaps equally many steps to do the multiplications of the appropriate powers. At the end we could take the result mod n. Thus the length of time it would take to do these computations would be more or less $2\log_2(10^{120}) =$

$240 \log_2 10$ times the time needed to do one operation. (Since $\log_2 10$ is about 3.33, it will take at most 800 times the amount of time for one operation to compute our power of a.)

You may not be used to thinking about how large the numbers get when you are doing computation. Computers hold fairly large numbers, (4-byte integers in the range roughly -2^{31} to 2^{31} are typical) and this suffices for most purposes. Because of the way computer hardware works, as long as numbers fit into one 4-byte integer, the time to do simple arithmetic operations doesn't depend on the value of the numbers involved. (A standard way to say this is to say that the time to do a simple arithmetic operation is constant.) However, when we talk about numbers that are much larger than 2^{31}, we have to take special care to implement our arithmetic operations correctly, and also we have to be aware that operations are slower.

It is accurate to assume that when multiplying large numbers, the time it takes is roughly proportional to the product of the number of digits in each. The constant of proportionality would depend on the implementation; with stratightforward implementations that don't make use of special hardware features the constant could be around 1 and would be unlikely to be as small as 0.01. If we computed our 100 digit number to the 10^{120}th power, we would be computing a number with more than 10^{120} digits. We clearly do **not** want to be doing computation on such numbers, as our computer cannot even store such a number!

Fortunately, since the number we are computing will ultimately be taken modulo some 200 digit number, we can make all our computations modulo that number (See Lemma 2.1.2.) By doing so, we ensure that the two numbers we are multiplying together have at most 200 digits, and so the time needed for the problem proposed in Exercise 2.4-1 would be a proportionality constant times 40,000 times $\log_2(10^{120})$ times the time needed for a basic operation plus the time needed to figure out which powers of a are multiplied together, which would be quite small in comparison.

Note that this algorithm, on 200 digit numbers, is as much as $40,000$ times slower than on simple integers. This is a noticeable effect and if you use or write an encryption program, you can see this effect when you run it. However, we can still typically do this calculation in less than a second, a small price to pay for secure communication.

How long does it take to use the RSA Algorithm?

There is a lot of arithmetic involved in encoding and decoding messages according to the RSA algorithm. How long will all this arithmetic take? Let's assume for now that Alice has already chosen p, q, e, and d, and so she knows n as well. When Bob wants to send Alice the message x, he sends $x^e \bmod n$. By our analyses in Exercise 2.4-2 and Exercise 2.4-3 we see that this amount of time is more or less proportional to $\log_2 e$, which is itself proportional to the number of digits of e, though the first constant of proportionality depends on the method our computer uses to multiply numbers. Since e has no more than 200 digits, this should not be too time consuming for Bob if he has a reasonable computer. (On the other hand, if he wants to send a message consisting of many segments of 200 digits each, he might want to use the RSA system to send a key for another simpler (secret key) system, and then use that simpler system for the message.

It takes Alice a similar amount of time to decode, as she has to take the message to the dth power, mod n.

We commented already that nobody knows a fast way to find x from $x^e \bmod n$. In fact, nobody knows that there isn't a fast way either, which means that it is possible that the RSA

cryptosystem could be broken some time in the future. We also don't know whether extracting eth roots mod n is in the class of NP-complete problems, an important family of problems with the property that a reasonably fast solution of any one of them will lead to a reasonably fast solution of any of them. We do know that extracting eth roots is no harder than these problems, but it may be easier.

However here someone is not restricted to extracting roots to discover x. Someone who knows n and knows that Alice is using the RSA system, could presumably factor n, discover p and q, use the extended GCD algorithm to compute d and then decode all of Alice's messages. However nobody knows how to factor integers quickly either. Again, we don't know if factoring is NP-complete, but we do know that it is no harder than the NP-complete problems. Thus here is a second possible way around the RSA system. However, enough people have worked on the factoring problem, that most people are confident that it is in fact difficult, in which case the RSA system is safe, as long as we use keys that are long enough.

How hard is factoring?

2.4-4 Factor 224,551. (The idea is to do this without resorting to computers, but if you give up by hand and calculator, using a computer is fine.)

With current technology, keys with roughly 100 digits are not that hard to crack. In other words, people can factor numbers that are roughly 100 digits long, using methods that are a little more sophisticated than the obvious approach of trying all possible divisors. However, when the numbers get long, say over 120 digits, they become very hard to factor. The record as of the year 2000 for factoring is a roughly 130-digit number. To factor this number, thousands of computers around the world were used, and it took several months. So given the current technology, RSA with a 200 digit key seems to be very secure.

Finding large primes

There is one more issue to consider in implementing the RSA system for Alice. We said that Alice chooses two primes of about a hundred digits each. But how does she choose them? It follows from some celebrated work on the density of prime numbers that if we were to choose a number m at random, and check about $\log_e(m)$ numbers around m for primality, we would expect that one of these numbers was prime. Thus if we have a reasonably quick way to check whether a number is prime, we shouldn't have to guess too many numbers, even with a hundred or so digits, before we find one we can show is prime.

However, we have just observed that nobody knows a quick way to find any or all factors of a number. The standard way of proving a number is prime is by showing that it and 1 are its only factors. For the same reasons that factoring is hard, the simple approach to primality testing, test all possible divisors, is much too slow. If we did not have a faster way to check whether a number is prime, the RSA system would be useless.

Miller and Rabin had a stroke of insight that has effectively solved this problem. We know, by Fermat's Little Theorem, that in Z_p with p prime, $x^{p-1} = 1$ for every x between 1 and $p-1$. What about x^{m-1}, in Z_m, when m is not prime?

Lemma 2.4.1 *Let m be a non-prime, and let x be a number in Z_n which has no multiplicative inverse. Then $x^{m-1} \not\equiv 1 \pmod{m}$.*

Proof: Assume, for the purpose of contradiction, that

$$x^{m-1} \equiv 1 \pmod{m} .$$

Then

$$x \cdot x^{m-2} \equiv 1 \pmod{m} .$$

But then x^{m-2} is the inverse of x, which contradicts the fact that x has no multiplicative inverse. ■

This distinction between primes and non-primes gives the idea for an algorithm. Suppose we have some number m, and are not sure whether it is prime or not. We can run the following algorithm:

```
PrimeTest(m)
choose a random number x, 2 ≤ x ≤ m − 1.
compute y = x^{m−1} mod m
if (y = 1)
     output ‘‘ m might be prime’’
else
     output ‘‘m is definitely not prime’’
```

Note the asymmetry here. If $y \neq 1$, then m is definitely not prime, and we are done. On the other hand, if $y = 1$, then the m might be prime, and we probably want to do some other calculations. In fact, we can repeat the algorithm Primetest(m) many times, with a different random number x each time. If on any of the t runs, the algorithm outputs "m is definitely not prime", then the number m is definitely not prime, as we have "proof" of an x for which $x^{m-1} \neq 1$. On the other hand, if on all t runs, the algorithm Primetest(m) outputs "m might be prime", then, with reasonable certainty, we can say that the number m is prime. This is actually an example of a *randomized algorithm* and we will be studying these in greater detail later in the course. For now, let's informally see how likely it is that we make a mistake.

We can see that the chance of making a mistake depends on, for a particular non-prime m, exactly how many numbers a have the property that $a^{m-1} = 1$. If the answer is that very few do, then our algorithm is very likely to give the correct answer. On the other hand, if the answer is most of them, then we are more likely to give an incorrect answer.

In Exercise 10 at the end of the section, you will show that the number of elements in Z_m without inverses is at least $\sqrt{m}$. In fact, even many numbers that do have inverses will also fail the test $x^{m-1} = 1$. For example, in Z_{12} only 1 passes the test while in Z_{15} only 1 and 14 pass the test. (Z_{12} really is not typical; can you explain why?)

In fact, what Miller and Rabin showed was how to modify the test slightly (in a way that we won't describe here) and show that for any non-prime m, at least half of the possible x will have fail the modified test and hence will show that m is composite. As we will see when we learn about probability, this implies that if we repeat the test t times, the probability of believing that a non-prime number is prime is actually 2^{-t}. So, if we repeat the test 10 times, we have only a

1 in a thousand chance of making a mistake, and if we repeat it 100 times, we have only a 1 in 2^{100} (a little less than one in a nonillion) chance of making a mistake!

Numbers we have chosen by this algorithm are sometimes called *pseudoprimes*. They are called this because they are very likely to be prime. In practice, pseudoprimes are used instead of primes in implementations of the RSA cryptosystem. The worst that can happen when a pseudoprime is not prime is that a message may be garbled; in this case we know that our pseudoprime is not really prime, and choose new pseudoprimes and ask our sender to send the message again. (Note that we do not change p and q with each use of the system; unless we were to receive a garbled message, we would have no reason to change them.)

Problems

1. What is 3^{1024} in Z_7? (This is an easy problem to do by hand.)
2. Suppose we have computed a^2, a^4, a^8, a^{16} and a^{32}. What is the most efficient way for us to compute a^{53}?
3. Find all numbers a different from 1 and -1 (which is the same as 8) such that $a^8 \bmod 9 = 1$.
4. Use a spreadsheet, programmable calculator or computer to find all numbers a different from 1 and -1 with $a^{32} \bmod 33 = 1$. (This problem is relatively easy to do with a spreadsheet that can compute mods and will let you "fill in" rows and columns with fommulas. However you do have to know how to use the spreadsheet in this way to make it easy!)
5. If a is a 100 digit number, is the number of digits of $a^{10^{120}}$ closer to 10^{120} or 10^{240}. Is it a lot closer?
6. Explain what our outline of the solution to Exercise 2.4-1 has to do with the binary representation of 10^{120}.
7. Give careful pseudocode to compute $a^x \bmod n$. Make your algorithm as efficient as possible.
8. Suppose we want to compute $a^{e_1 e_2 \cdots e_m} \bmod n$. Discuss whether it makes sense to reduce the exponents mod n as we compute their product. In particular, what rule of exponents would allow us to do this, and do you think this rule of exponents makes sense?
9. Number theorists use $\varphi(n)$ to stand for the number of elements of Z_n that have inverses. Suppose we want to compute $a^{e_1 2_2 \cdots e_m} \bmod n$. Would it make sense for us to reduce our exponents mod $\varphi(n)$ as we compute their product? Why?
10. Show that if m is not prime, then at least $\sqrt{m}$ elements of Z_m do not have multiplicative inverses.
11. Show that in Z_{p+1}, where p is prime, only one element passes the primality test $x^{m-1} = 1$. (In this case, $m = p + 1$.)
12. Suppose for RSA, $p = 11$, $q = 19$, and $e = 7$. What is the value of d? Show how to encrypt the message 100, and then show how to decrypt the resulting message.
13. Suppose for applying RSA, $p = 11$, $q = 23$, and $e = 13$. What is the value of d? Show how to encrypt the message 100 and then how to decrypt the resulting message.

14. A digital signature is a way to securely sign a document. That is, it is a way to put your "signature" on a document so that anyone reading it knows that it is you who have signed it, but no one else can "forge" your signature. Digital signatures are, in a way, the opposite of encryption, as if Alice wants to sign a message, she first applies her signature to it (think of this as encryption) and then the rest of the world can easily read it (think of this as decryption). Explain, in detail, how to achieve digital signatures, using ideas similar to those used for RSA. in particular, anyone who has the document and has your signature (and knows your public key) should be able to determine that you signed it.

Chapter 3

Reflections on Logic and Proof

3.1 Equivalence and Implication

Equivalence of statements

3.1-1 A group of students are working on a project that involves writing a merge sort program. Joe and Mary have each tried their hand at writing an algorithm for a function that takes two lists, List1 and List2 of lengths p and q and merges them into a third list. Part of Joe's algorithm is the following:

```
if ((i + j ≤ p + q) && (i ≤ p) && ((j ≥ q)||(List1(i) ≤ List2(j))))
          List3(k) = List1(i)
          i = i + 1
else
          List3(k) = List2(j)
          j = j + 1

k = k + 1
Return List3
```

The corresponding part of Mary's algorithm is

```
if (((i + j ≤ p + q) && (i ≤ p) && (j ≥ q)) || ((i + j ≤ p + q) && (i ≤ p) && (List1(i) ≤ List2(j))))
          List3(k) = List1(i)
          i = i + 1
else
          List3(k) = List2(j)
          j = j + 1

k = k + 1
Return List3
```

Do Joe and Mary's algorithms do the same thing?

Notice that Joe and Mary's algorithms are exactly the same except for the "If statement." (How convenient; they even used the same local variables!) In Joe's algorithm we put entry i of `List1` into position k of `List3` if

$i + j \leq p + q$ and $i \leq p$ and ($j \geq q$ or $List1(i) \leq List2(j)$)

while in Mary's algorithm we put entry i of `List1` into position k of `List3` if

($i+j \leq p+q$ and $i \leq p$ and $j \geq q$) or ($i+j \leq p+q$ and $i \leq p$ and $List1(i) \leq List2(j)$)

Joe and Mary's statements are both built up from the same constituent parts (namely comparison statements), so let's name these constituent parts and look at the statements in this form. Let's use

s to stand for $i + j \leq p + q$
t to stand for $i \leq p$
u to stand for $j \geq q$ and
v to stand for $List1(i) \leq List2(j)$

Then the condition in Mary's "If statement" becomes

s and t and (u or v)

while Joe's becomes

(s and t and u) or (s and t and v).

One immediate benefit from recasting the statements in this symbolic form is that we see that s and t always appear together as "s and t." This means we can simplify the form we need to analyze even more by substituting w for "s and t," making Mary's condition have the form

w and (u or v)

and Joe's have the form

(w and u) or (w and v).

While we can show, based on our knowledge of the structure of the English language that these two statements are saying the same thing, it is by no means easy to see. It is true that these two statements are saying the same thing, and one way to see it is to observe that in English, the word "and" distributes over the word "or," just as set intersection distributes over set union, and multiplication distributes over addition. To make this analogy easier to see, there is a standard notation that logicians have adopted for writing symbolic versions of compound statements. We shall use the symbol $\wedge$ to stand for "and" and $\vee$ to stand for "or." In this notation, Mary's condition becomes

$$w \wedge (u \vee v)$$

and Joe's becomes

$$(w \wedge u) \vee (w \wedge v).$$

It is one thing to see the analogy of notation, and another to understand why two statements with this symbolic form must mean the same thing. To deal with this, we must give a precise definition of "meaning the same thing," and develop a tool for analyzing when two statements satisfy this definition. We are going to consider symbolic compound statements that may be built up with the following kinds of symbols: symbols (s, t, etc.) standing for statements, the symbol $\wedge$, standing for "and," the symbol $\vee$, standing for "or," the symbol $\oplus$ standing for "exclusive or," and the symbol $\neg$, standing for "not." We will develop a theory for deciding when a compound statement is true based on the truth or falsity of its component statements. This theory will enable us to determine on the basis of the truth or falsity of, say, s, t, and u, whether a particular compound statement, say $(s \oplus t) \wedge (\neg u \vee (s \wedge t)) \wedge \neg(s \oplus (t \vee u))$, is true or false. Our technique is going to be based on truth tables, which you have almost certainly seen before. What may not have been clear before is what they are good for; we will now show you one thing they are good for.

In our sample compound statement $(s \oplus t) \wedge (\neg u \vee (s \wedge t)) \wedge \neg(s \oplus (t \vee u))$ we used parentheses to make it clear which operation to do first, with one exception, namely our use of the $\neg$ symbol. Notice we wrote $\neg u \vee (s \wedge t)$. In principle we could have meant $(\neg u) \vee (s \wedge t)$, which we did, or $\neg(u \vee (s \wedge t))$, which we did not mean. The principle we use here is simple; the symbol $\neg$ applies to the smallest number of following symbols that makes sense. This is the same principle we use with minus signs in algebraic expressions; if you did not wonder what we meant by $\neg u \vee (s \wedge t)$, this may be the reason why. With this one exception we will always use parentheses to make the order in which we are to perform operations clear; you should do the same.

The truth table for a logical connective states, in terms of the possible truth or falsity of the component parts, when the compound statement made by connecting those parts is true and when it is false. Here are the truth tables for the connectives we have mentioned so far.

s	t	$s \wedge t$
T	T	T
T	F	F
F	T	F
F	F	F

s	t	$s \vee t$
T	T	T
T	F	T
F	T	T
F	F	F

s	t	$s \oplus t$
T	T	F
T	F	T
F	T	T
F	F	F

s	$\neg s$
T	F
F	T

In effect these truth tables define what we mean by the words "and," "or," "exclusive or," and "not" in the context of symbolic compound statements. Since we want to apply them to English statements, let's see what a typical truth table says. The truth table for $\vee$—or—tells us that when s and t are both true, then so is "s or t." It tells us that when s is true and t is false, or s is false and t is true, then "s or t" is true. Finally it tells us that when s and t are both false, then so is "s or t." Is this how we use the word or in English? The answer is sometimes! The word "or" is used ambiguously in English. When a teacher says "Each question on the test will be short answer or multiple choice," the teacher is presumably not intending that a question could be both. Thus the word "or" is being used here in the sense of "exclusive or"—the "$\oplus$" in the truth tables above. When someone says "Let's see, this afternoon I could take a walk or I

could shop for some new gloves," she probably does not mean to preclude the possibility of doing both—perhaps even taking a walk downtown and then shopping for new gloves before walking back. Thus in English, we determine the way in which someone uses the word "or" from context. In mathematics and computer science we don't always have context and so we agree that we will say "exclusive or" or "xor" for short when that is what we mean, and otherwise we will mean the "or" whose truth table is given by $\vee$. In the case of "and" and "not" the truth tables are exactly what we would expect.

When we have a more complex compound statement, as we did in Joe and Mary's programs, we will still want to be able to describe exactly the situations in which it is true and the situations in which it is false. We will do this by working out a truth table for the compound statement from the truth tables of its symbolic statements and its connectives. We will have one row across the top of the table that says what the original symbolic statements are and how we may build up the compound statement. It also has one row for each possible combination of truth and falsity of the original symbolic statements (In this example, as is often the case, the original symbolic statements are just single variables.). Thus if we have two statements, we have, as above, four rows. If we have just one statement, then we have, as above, just two rows. If we have three statements, then we will have $2^3 = 8$ rows, and so on. To show you how we work out truth tables, here is the truth table for Joe's symbolic statements above. The columns to the left of the double line are the input variables, the columns to the right are the various sub-expressions that we need to compute. We put as many columns as we need in order to correctly compute the final result; as a general rule, each column should be easily computed from one or two previous columns.

w	u	v	$u \vee v$	$w \wedge (u \vee v)$
T	T	T	T	T
T	T	F	T	T
T	F	T	T	T
T	F	F	F	F
F	T	T	T	F
F	T	F	T	F
F	F	T	T	F
F	F	F	F	F

Here is the truth table for Mary's symbolic statement.

w	u	v	$w \wedge u$	$w \wedge v$	$(w \wedge u) \vee (w \wedge v)$
T	T	T	T	T	T
T	T	F	T	F	T
T	F	T	F	T	T
T	F	F	F	F	F
F	T	T	F	F	F
F	T	F	F	F	F
F	F	T	F	F	F
F	F	F	F	F	F

We see that Joe's symbolic statement and Mary's symbolic statement are true in exactly the same cases so that they must say the same thing. Therefore Mary and Joe's program segments

return exactly the same values. We say that two symbolic compound statements are *equivalent* if they are true in exactly the same cases. Thus they are equivalent if their truth tables have the same final column (assuming both tables assign truth values to the original symbolic statements in the same pattern).

3.1-2 De Morgan's Law says that $\neg(p \vee q)$ is equivalent to $\neg p \wedge \neg q$. Use a truth table to demonstrate the truth of DeMorgan's law.

3.1-3 Show that $p \oplus q$, the exclusive or of p and q, is equivalent to $(p \vee q) \wedge \neg(p \wedge q)$. Apply DeMorgan's law to $\neg\neg(p \vee q) \wedge \neg(p \wedge q)$ to find another symbolic statement equivalent to the exclusive or.

To verify DeMorgan's law, we create the following pair of truth tables that we have condensed into one "double truth table."

p	q	$p \vee q$	$\neg(p \vee q)$	$\neg p$	$\neg q$	$\neg p \wedge \neg q$
T	T	T	F	F	F	F
T	F	T	F	F	T	F
F	T	T	F	T	F	F
F	F	F	T	T	T	T

To show that $p \oplus q$ is equivalent to $(p \vee q) \wedge \neg(p \wedge q)$, we use another "double truth table."

p	q	$p \oplus q$	$p \vee q$	$p \wedge q$	$\neg(p \wedge q)$	$(p \vee q) \wedge \neg(p \wedge q)$
T	T	F	T	T	F	F
T	F	T	T	F	T	T
F	T	T	T	F	T	T
F	F	F	F	F	T	F

By DeMorgan's law, $p \oplus q$ is also equivlaent to $\neg(\neg(p \vee q) \vee (p \wedge q))$. It was certainly easier to use DeMorgan's law to show this equivalence than to use another double truth table.

Implication

There is another kind of compound statement that occurs frequently in mathematics and computer science. Think about the statement of Fermat's little Theorem:

> If p is a prime, then $a^{p-1} \bmod p = 1$ for every non-zero $a \in Z_p$.

This is a combination of two constituent statements,

> p is a prime

and

$a^{p-1} \bmod p = 1$ for every non-zero $a \in Z_p$.

We can restate this (a bit clumsily) as

p is a prime only if $a^{p-1} \bmod p = 1$ for every non-zero $a \in Z_p$.

or

p is a prime implies $a^{p-1} \bmod p = 1$ for every non-zero $a \in Z_p$.

Using s to stand for "p is a prime" and t to stand for "$a^{p-1} \bmod p = 1$ for every non-zero $a \in Z_p$," we symbolize any of these three symbolic statements as

$$s \Rightarrow t,$$

which most people read as "s implies t." When we translate from symbolic language to English, it is often clearer to say "If s then t."

There is a stronger version[1] (which we have not proved) of Fermat's Little Theorem, namely

The integer n is a prime if and only if $a^{n-1} \bmod n = 1$ for every non-zero a in Z_n.

How is this different from the statement of Fermat's little Theorem we gave earlier? If we now use s to stand for "The integer n is a prime" and t to stand for "$x^{n-1} = 1$ for every $x \in Z_n$," we are asserting that $s \Rightarrow t$ and $t \Rightarrow s$. We denote the statement "s if and only if t" by $s \Leftrightarrow t$. (Notice that we are adopting the convention that "s if t" is the same as $t \Rightarrow s$; in other words, "s if t" is the same as "if t then s.")

3.1-4 Use truth tables to explain the difference between $s \Rightarrow t$ and $s \Leftrightarrow t$.

In order to be able to analyze the truth and falsity of statements involving "implies" and "if and only if," we need to understand exactly how they are different. By constructing truth tables for these statements, we see that there is only one case in which they could have different truth values. These truth tables are:

s	t	$s \Rightarrow t$
T	T	T
T	F	F
F	T	T
F	F	T

s	t	$s \Leftrightarrow t$
T	T	T
T	F	F
F	T	F
F	F	T

.

Here is another place where English usage is sometimes inconsistent. Suppose a parent says "I will take the family to McDougalls for dinner if you get an A on this test," and even though the student gets a C, the parent still takes the family to McDougalls for dinner. While this is something we didn't expect, was the parent's statement still true? Some people would say yes;

[1] It is reasonable to ask if there is an even stronger version of Fermat's little theorem that number theorists have asked about, namely "The integer n is a prime if and only if $a^{n-1} \bmod n = 1$ for every non-zero a relatively prime to n in Z_n." This statement is false, however; to find counter-examples, the reader may wish to look up Charmichael numbers in a number theory book or CLR.

others would say no. Those who would say no mean, in effect, that in this context the parent's statement meant the same as "I will take the family to dinner at McDougalls if and only if you get an A on this test." In other words, to some people, in certain contexts, "If" and "If and only if" mean the same thing! Fortunately questions of child rearing aren't part of mathematics or computer science (at least not this kind of question!). In mathematics and computer science, we adopt the two truth tables just given as the meaning of the compound statement $s \Rightarrow t$ (or "if s then t" or "t if s") and the compound statement $s \Leftrightarrow t$ (or "s if and only if t.")

People often find they have difficulty using the truth table for $s \Rightarrow t$ because of this ambiguity in English. If this happens to you, think about the following example. I hold an ordinary playing card (with its back to you) and say "If this card is a heart, then it is a queen." In which of the following four circumstances would you say I lied:

1. the card is a heart and a queen
2. the card is a heart and a king
3. the card is a diamond and a queen
4. the card is a diamond and a king?

You would certainly say I lied in the case the card is the king of hearts, and you would certainly say I didn't lie if the card is the queen of hearts. Hopefully in this example, the inconsistency of English language seems out of place to you and you would not say I am a liar in either of the other cases. Now we apply the principle (called the *principle of the excluded middle*) that a statement is true exactly when it is not false to say that my statement is true in the three cases where you wouldn't say I lied.

Problems

1. Give truth tables for the following expressions:
   a. $(s \vee t) \wedge (\neg s \vee t) \wedge (s \vee \neg t)$
   b. $(s \Rightarrow t) \wedge (t \Rightarrow u)$
   c. $(s \vee t \vee u) \wedge (s \vee \neg t \vee u)$
2. Show that the statements $s \Rightarrow t$ and $\neg s \vee t$ are equivalent.
3. Give an example in English where "or" seems to you to mean exclusive or (or where you think it would for many people) and an example in English where "or" seems to you to mean inclusive or (or where you think it would for many people).
4. Give an example in English where "if ... then" seems to you to mean "if and only if" (or where you think it would to many people) and an example in English where it seems to you not to mean "if and only if" (or where you think it would not to many people).
5. Find a statement involving only $\wedge$, $\vee$ and $\neg$ (and s and t) equivalent to $s \Leftrightarrow t$. Does your statement have as few symbols as possible? If you think it doesn't, try to find one with fewer symbols.

6. Given the pattern of Trues and Falses desired in the final column of a truth table, explain how to create a statement using the symbols s, t, $\wedge$, $\vee$, and $\neg$ that has that pattern as its final column.

7. In the previous question, analyze which, if any, of the symbols you could do without. In general, are there some symbols you can do without?

8. We proved that $\wedge$ distributes over $\vee$ in the sense of giving two equivalent statements that represent the two "sides of the distributive law. For each question below, explain why your answer is true.

    a. Does $\vee$ distribute over $\wedge$?

    b. Does $\vee$ distribute over $\oplus$?

    c. Does $\wedge$ distribute over $\oplus$?

3.2 Variables and Quantifiers

Statements about universes

Statements we use in computer languages to control loops or conditionals are statements about variables. When we declare these variables, we give the computer information about their possible values. Clearly it is the use of variables in such statements that gives them enough expressive power to be useful in controlling a computer program. In English and in mathematics, we also make statements about variables, but it is not always so clear what words are being used as variables and what values these variables may take on. For example in English, we might say "If someone's umbrella is up, then it must be raining." In this case, the word someone is a variable, and presumably it "varies over" the people who happen to be in a given place at a given time. (Saying that a variable varies over a set means that the values we may assign to it lie in that set.) In mathematics, we might say "For every pair of positive integers m and n, there are nonnegative integers q and r with $0 \leq r < n$ such that $m = qn + r$." In this case m, n, q, and r are clearly our variables; our statement itself suggests that two of our variables range over the positive integers and two range over the nonnegative integers. We call the set of values of a variable that we are willing to consider in a statement the *universe* of that variable.

In the statement "m is an even integer" it is clear that m is a variable, but the universe for m might be the integers, just the even integers, or maybe the rational numbers, or many other sets. Note that if we choose the set of integers for the universe for m, then the statement is true for some integers and false for others. In the same way, when we control a while loop with a statement such as `i < j` there are some values of `i` and `j` for which the statement is true and some for which it is false. In statements like "m is an even integer" and `i < j` our variables are not constrained in any way whatever, and so are called *free variables*. For each possible value of a free variable, we have a new statement, which might be either true or false, determined by substituting the possible value for the variable, and the truth value is determined only after such a substitution.

3.2-1 For what values of m is the statement $m^2 > m$ a true statement and for what values is it a false statement? Since we have not specified a universe, your answer will depend on what universe you choose to use.

If you used the universe of positive integers, the statement is true for every value of m but 1; if you used the real numbers, the statement is true for every value of m except for those in the closed interval $[0, 1]$. There are really two points to make here. First, a statement about a variable can often be interpreted as a statement about more than one universe, and so to make it unambiguous, we have to make our universe clear. Second, a statement about a variable can be true for some values of a variable and false for others.

Quantifiers

In contrast, the statement

$$\text{For every integer } m,\ m^2 > m. \tag{3.1}$$

is true or false without reference to substituting any values in whatever. We could, of course, imagine substituting various values for m into the simpler statement $m^2 > m$ as you may have

in Exercise 3.2-1, and deciding for each of these values whether the statement $m^2 > m$ is true or false. When we do so, we see the statement $m^2 > m$ is true for $m = -3$, for example, or $m = 9$, but false for $m = 0$ or $m = 1$. Thus it is not the case that for every integer m, $m^2 > m$, so our statement 3.1 is false. It is false as a statement because it is an assertion that the simpler statement $m^2 > m$ holds for each integer value of m we substitute in. A phrase like "for every integer m" that converts a symbolic statement about potentially any member of our universe into a statement about the universe instead is called a quantifier. A quantifier that asserts a statement about a variable is true for every value of the variable in its universe is called a **universal quantifier.**

The previous example illustrates a very important point.

> If a statement asserts something for every value of a variable, then to show the statement is false, we need only give one value of the variable for which the assertion is untrue.

Another example of a quantifier is the phrase "There is an integer m" in the sentence

> There is an integer m such that $m^2 > m$.

This is once again a statement about our universe, and as such it is true—there are plenty of integers m we can substitute into the symbolic statement $m^2 > m$ to make it true. This is an example of an "existential quantifier." An **existential quantifier** asserts that a certain element of our universe exists. A second important point similar to the one we made above is:

> To show that a statement with an existential quantifier is *true*, then we need only exhibit one value of the variable being quantified that makes the statement true.

As the more complex statement

> For every pair of positive integers m and n, there are nonnegative integers q and r with $0 \leq r < n$ such that $m = qn + r$.

shows, statements of mathematical interest abound with quantifiers. So do statements of interest to computer science, as the following definition of the "big oh" notation you have used in earlier computer science courses shows.

> We say that $f(x) = O(g(x))$ if there are positive numbers c and n_0 such that $f(x) \leq cg(x)$ for every $x > n_0$.

3.2-2 Quantification is present in our everyday language as well. The sentences "Every child wants a pony" and "No child wants a toothache" are two different examples of quantified sentences. Give ten examples of everyday sentences that use quantifiers, but use different words to indicate the quantification.

3.2-3 Convert the sentence "No child wants a toothache" into a sentence of the form "It is not the case that..." Find an existential quantifier in your sentence.

3.2-4 what would you need to do in order to show that a statement about one variable with an existential quantifier is *false*? Correspondingly,what would you need to do in order to show that a statement about one variable with a universal quantifier is *true*?

To show that a statement about one variable with an existential quantifier is false, we have to show that every element of the universe makes the statement (such as $m^2 > m$) is false. Thus to show that the statement "There is an x in $[0, 1]$ with $x^2 > x$" is false, we have to show that every x in the interval makes the statement $x^2 > x$ false. Similarly, to show that a statement with a universal quantifier is true, we have to show that the statement being quantified is true for every member of our universe. We will take up the details of how to show a statement about a variable is true or false for every member of our universe in a later section of this chapter.

Standard notation for quantification

While there are many variants of language that describe quantification, they each describe one of two situations:

> A quantified statement about a variable x asserts either that the statement is true for all x in the universe or that there exists an x in the universe that makes the statement true.

Thus in order to talk about quantified statements, we shall now assume that they have one of these two forms. A standard shorthand for the phrase "for all" is $\forall$, and a standard shorthand for the phrase "there exists" is $\exists$. Thus using Z to stand for the universe of all integers, positive or negative, we can write

$$\forall n \in Z\ (n^2 \geq n)$$

as a shorthand for the statement "For all integers n, $n^2 \geq n$." It is perhaps more natural to read the notation as "For all n in Z, $n^2 \geq n$," which is how we recommend reading the symbolism. We similarly use

$$\exists n \in Z(n^2 \not> n)$$

to stand for "There exists an n in Z such that $n^2 \not> n$." Notice that in order to cast our symbolic form of an existence statement into grammatical English we have included the supplementary word "an" and the supplementary phrase "such that." People often leave out the "an" as they read an existence statement, but rarely leave out the "such that." Such supplementary language is not needed with $\forall$.

To have another example, let us rewrite the definition of the "Big Oh" notation with these symbols. To do so we use the letter R to stand for the universe of real numbers, and the symbol R^+ to stand for the universe of positive real numbers.

> $f = O(g)$ means that $\exists c \in R^+(\exists n_0 \in R^+(\forall x \in R(x > n_0 \Rightarrow f(x) \leq cg(x))))$

We would read this literally as

> f is big Oh of g means that there exists a c in R^+ such that there exists an n_0 in R^+ such that for all x in R, if $x > n_0$, then $f(x) \leq cg(x)$.

Clearly this has the same meaning (when we translate it into more idiomatic English) as

> f is big Oh of g means that there exist a c in R^+ and an n_0 in R^+ such that for all real numbers $x > n_0$, $f(x) \leq cg(x)$.

This is the definition of "big Oh" that we gave earlier with a bit more precision as to what c and r actually are.

3.2-5 How would you rewrite Euclid's division theorem using the shorthand notation we have introduced for quantifiers? We recommend using Z^+ to to stand for the positive integers and either N or $Z_{\geq 0}$ to stand for the nonnegative integers.

We can rewrite Euclid's division theorem as

$$\forall m \in Z^+(\forall n \in Z^+(\exists q \in N(\exists r \in N((r < n) \wedge (m = qn + r))))).$$

Statements about variables

In your study of mathematics so far you have likely dealt with statements about variables. In this section and the next two, we want to be able to talk about statements about variables. In order to talk about a property of statements about variables, we need a notation to stand for statements. In effect we are introducing variables that stand for statements about (other) variables! We typically use symbols like $p(n)$, $q(x)$, etc. to stand for statements about a variable n or x. Then the statement "For all x in U $p(x)$" can be written as $\forall x \in U(p(x))$ and the statement "There exists an n in U such that $q(n)$" can be written as $\exists n \in U(q(n))$. Sometimes we have statements about more than one variable; for example, our definition of "big Oh" notation had the form $\exists c(\exists n_0(\forall x(p(c, n_0, x))))$. Here $p(c, n_0, x)$ is $(x > n_0 \Rightarrow f(x) \leq cg(x))$. (We have left out mention of the universes for our variables here to emphasize the form of the statement. People also often omit mention of the universe for a variable when it is clear from context.)

3.2-6 What is the form of Euclid's division theorem?

3.2-7 In what follows, we use R to stand for the real numbers and R^+ to stand for the positive real numbers. For each of the following statements, say whether it is true or false and why.

a $\forall x \in R^+(x > 1)$

b $\exists x \in R^+(x > 1)$

c $\forall x \in R(\exists y \in R(y > x))$

d $\forall x \in R(\forall y \in R(y > x))$

e $\exists x \in R(x \geq 0 \wedge \forall y \in R^+(y > x))$

3.2-8 Rewrite the statements in parts a and b of the previous exercise so that the universe is all the real numbers, but the statements say the same thing in everyday English that they did before.

The form of Euclid's division theorem is $\forall m(\forall n(\exists q(\exists r p(m, n, q, r))))$

In Exercise 3.2-7 since .5 is not greater than 1, statement (a) is false. However since $2 > 1$, statement (b) is true. Statement c says that for each x there is a y bigger than x, which we know is true. Statement d says that every y in R is larger than every x in R, which is so false as to be nonsensical! (Technically, there is no such thing as degrees of falseness, so statement d is simply a false statement. However to a mathematician, statement d is nonsensical as well, but being nonsensical is not part of what we study in logic.) Statement e says that there is a nonnegative number x such that every positive y is larger than x, which is true because $x = 0$ fills the bill.

For Exercise 3.2-8, there are potentially many ways to rewrite the statements. Two particularly simple ways are $\forall x \in R(x > 0 \Rightarrow x > 1)$ and $\exists x \in R(x > 0 \wedge x > 1)$. Notice that we translated one of these statements with "implies" and one with "and." We actually left our two sets of parentheses in each translation, one around $x > 0$ and one around $x > 1$. It is conventional to do so when there is really only one interpretation of which statements the logical connective is joining.

We can summarize what we know about the meaning of quantified statements as follows.

> The statement $\exists x(p(x))$ about a variable x in a universe U is true if there is at least one value of x in U for which the statement $p(x)$ is true. It is false if there is no such x in the universe.

> The statement $\forall x(p(x))$ about a variable x in a universe U is true if $p(x)$ is true for each value of x in U and is false otherwise; in particular, it is false if the statement $p(x)$ is false for one (or more) value of x in U.

Negation of Quantified statements

There is an interesting connection between $\forall$ and $\exists$ which arises from the negation of statements. What would the statement

> It is not the case that for all integers n, $n^2 \geq 0$.

mean? From our knowledge of English we see that the statement is asserting there is some integer n such that $n^2 \ngeq 0$. In other words, it says there is some integer n such that $n^2 < 0$. Thus the negation of our "for all" statement is a "there exists" statement. We can make this idea more precise by recalling the notion of equivalence of statements. We have said that two symbolic statements are *equivalent* if they are true in exactly the same cases. In the case of the truth or falsity of a statement of the form $\neg\forall x(p(x))$, there is one case to consider for each assignment of truth and falsity to $p(x)$ that arises from assigning elements of our universe to x. The theorem below, which formalizes the example above in which $p(x)$ was the statement $x^2 \geq 0$, is proved by dividing these cases into two possibilities.

Theorem 3.2.1 *The statements* $\neg\forall x \in U(p(x))$ *and* $\exists x \in U(\neg p(x))$ *are equivalent.*

Proof: If there is a value of x in the universe U that makes the statement $p(x)$ false, then the statement $\forall x \in U(p(x))$ is false, and so the statement $\neg\forall x \in U(p(x))$ is true. For this value of x, the statement $\neg p(x)$ is true, and thus the statement $\exists x \in U(\neg p(x))$ is true.

On the other hand, if there is no value of x that makes the statement $p(x)$ false, then the statement $p(x)$ is true for each x in the universe U, and therefore $\forall x \in U(p(x))$ is true. Thus in this situation $\neg\forall x \in U(p(x))$ is false. Since there is no x that makes $p(x)$ true, the statement $\neg p(x)$ is false for every x, and therefore the statement $\exists x \in U \neg p(x)$ is false. Thus $\neg\forall x \in U(p(x))$ and $\exists x \in U(\neg p(x))$ are true in exactly the same cases, so they are equivalent. ■

Corollary 3.2.2 *The statements* $\neg\exists x \in U(q(x))$ *and* $\forall x \in U(\neg q(x))$ *are equivalent.*

Proof: The negations of the two statements in Theorem 3.2.1 are equivalent. To prove this corollary, substitute $\neg q(x)$ for $p(x)$ in these negations. ■

To deal with the negation of more complicated statements, we simply take them one quantifier at a time. For example, we had a statement of the definition of the big Oh notation, $f = O(g)$ in symbolic terms, namely,

$$f = O(g) \text{ if } \exists c \in R^+(\exists n_0 \in R^+(\forall x \in R(x > n_0 \Rightarrow f(x) \leq cg(x)))).$$

What does it mean to say that f is *not* $O(g)$? First we can write

$$f \neq O(g) \text{ if } \neg\exists c \in R^+(\exists n_0 \in R^+(\forall x \in R(x > n_0 \Rightarrow f(x) \leq cg(x)))).$$

After one application of Corollary 3.2.2 we get

$$f \neq O(g) \text{ if } \forall c \in R^+(\neg\exists n_0 \in R^+(\forall x \in R(x > n_0 \Rightarrow f(x) \leq cg(x)))).$$

After another application of Corollary 3.2.2 we obtain

$$f \neq O(g) \text{ if } \forall c \in R^+(\forall n_0 \in R^+(\neg\forall x \in R(x > n_0 \Rightarrow f(x) \leq cg(x)))).$$

Now we apply Theorem 3.2.1 and obtain

$$f \neq O(g) \text{ if } \forall c \in R^+(\forall n_0 \in R^+(\exists x \in R(\neg(x > n_0 \Rightarrow f(x) \leq cg(x)))))).$$

Now $\neg(p \Rightarrow q)$ is equivalent to $p \wedge \neg q$, so we can write

$$f \neq O(g) \text{ if } \forall c \in R^+(\forall n_0 \in R^+(\exists x \in R((x > n_0) \wedge (f(x) \not\leq cg(x)))))).$$

Thus f is *not* $O(g)$ if for every c in $R+$ and every n_0 in R^+, there is an x such that $x > n_0$ and $f(x) \not\leq cg(x)$. Hopefully this fits your intuition of what it means for f not to be $O(g)$.

In the exercises that follow, we use the "Big Theta" notation defined by

$$f = \Theta(g) \text{ means that } f = O(g) \textbf{ and } g = O(f).$$

3.2-9 Express $\neg(f = \Theta(g))$ in terms similar to those we used to describe $f \neq O(g)$.

3.2-10 Suppose the universe for a statement $p(n)$ is the integers from 1 to 10. Express the statement $\forall x(p(x))$ without any quantifiers. Express the negation in terms of $\neg p$ without any quantifiers. Discuss how negation of "for all" and "there exists" statements corresponds to DeMorgan's Law.

3.2-11 Are there any quantifiers in the statement "The sum of even integers is even?"

By DeMorgan's law, $\neg(f = \Theta(g))$ means $\neg(f = O(g)) \vee \neg(g = O(f))$. Thus $\neg(f = \Theta(g))$ means that either for every c and n_0 in R^+ there is an x in R with $x > n_0$ and $f(x) \not< cg(x)$ or for every c and n_0 in R^+ there is an x in R with $x > n_0$ and $g(x) < cf(x)$ (or both).

For Exercise 3.2-10 we see that $\forall x(p(x))$ is simply

$$p(1) \wedge p(2) \wedge p(3) \wedge p(4) \wedge p(5) \wedge p(6) \wedge p(7) \wedge p(8) \wedge p(9) \wedge p(10).$$

By DeMorgan's law the negation of this statement is

$$\neg p(1) \vee \neg p(2) \vee \neg p(3) \vee \neg p(4) \vee \neg p(5) \vee \neg p(6) \vee \neg p(7) \vee \neg p(8) \vee \neg p(9) \vee \neg p(10).$$

Thus the relationship that negation gives between "for all" and "there exists" statements is the extension of DeMorgan's law from a finite number of statements to potentially infinitely many statements about a potentially infinite universe.

Implicit Quantification

It is an elementary fact about numbers that the sum of even integers is even. Another way to say this is that if m and n are even, then $m + n$ is even. If $p(n)$ stands for the statement "n is even," then this last sentence translates to $p(m) \wedge p(n) \Rightarrow p(m+n)$. From the logical form of the statement, we see that our variables are free, so we could substitute various integers in for m and n to see whether the statement is true. But we said we were stating a fact about the integers, not a statement that we can check the truth or falsity of for various integers. What we meant to say is that for every pair of integers m and n, if m and n are even, then $m + n$ is even. In symbols, using $p(k)$ for "k is even," we have

$$\forall m \in Z(\forall n \in Z(p(m) \wedge p(n) \Rightarrow p(m + n))).$$

This way of representing the statement captures the meaning we originally intended. This is one of the reasons that mathematical statements and their proofs sometimes seem confusing–just as in English, sentences in mathematics have to be interpreted in context. Since mathematics has to be written in some natural language, and since context is used to remove ambiguity in natural language, so must context be used to remove ambiguity from mathematical statements made in natural language.

Proof of Quantified Statements

We said that "the sum of even integers is even" is an elementary fact about numbers. How do we know it is a fact? One answer is that we know it because our teachers told us so. (And presumably they knew it because their teachers told them so.) But someone had to figure it out in the first place, and so we ask, if we didn't already believe this is a fact, how would we prove it? A mathematician asked to give a proof that the sum of even numbers is even might write

> If m and n are even, then $m = 2i$ and $n = 2j$ so that
>
> $$m + n = 2i + 2j = 2(i + j)$$
>
> and thus $m + n$ is even.

Because mathematicians think and write in natural language, they will often rely on context to remove ambiguities. For example, there are no quantifiers in the proof above. However the sentence, while technically incomplete as a proof, captures the essence of why the sum of two even numbers is even. A typical complete (but more formal and wordy than usual) proof might go like this.

> Let m and n be integers. Suppose m and n are even. If m and n are even, then by definition there are integers i and j such that $m = 2i$ and $n = 2j$. Thus there are integers i and j such that $m = 2i$ and $n = 2j$. Then
>
> $$m + n = 2i + 2j = 2(i + j),$$
>
> so $m + n$ is an even integer. We have shown that if m and n are even, then $m + n$ is even. Therefore for every m and n, if m and n are even integers, then so is $m + n$.

We began our proof by assuming that m and n are integers. This gives us symbolic notation for talking about two integers. We then appealed to the definition of an even integer, namely than an integer h is even if there is another integer k so that $h = 2k$. (Note the use of a quantifier in the definition.) Then we used algebra to show that $m + n$ is also two times another number. Since this is the definition of $m + n$ being even, we concluded that $m + n$ is even. This let us say that if m and n are even, the $m + n$ is even. Then we asserted that for every pair of integers m and n, if m and n are even, then $m + n$ is even.

There are a number of principles of proof illustrated here. The next section will be devoted to a discussion principles we use in constructing proofs. For now, let us conclude with a remark about the limitations of logic. How did we know that we wanted to write the symbolic equation

$$m + n = 2i + 2j = 2(i + j)?$$

It was not logic that told us to do this, but intuition and experience.

Problems

1. For what positive integers x is the statement $(x - 2)^2 + 1 \leq 2$ true? For what integers is it true? For what real numbers is it true? If we expand the universe for which we are considering a statement about a variable, does this always increase the size of the statement's truth set?

2. Is the statement There is an integer greater than 2 such that $(x-2)^2+1 \leq 2$ true or false? How do you know?

3. Write the statement that the square of every real number is greater than or equal to zero as a quantified statement about the universe of real numbers. You may use R to stand for the universe of real numbers.

4. The definition of a prime number is that is is an integer greater than 1 whose only positive integer factors are itself and 1. Find two ways to write this definition so that all quantifiers are explicit. (It may be convenient to introduce a variable to stand for the number and perhaps for its factors.)

5. Write down the definition of a greatest common divisor of m and n in such a way all quantifiers are explicit and expressed explicitly as "for all" or "there exists." Write down Euclid's extended greatest common divisor theorem that relates the greatest common divisor of m and n algebraically to m and n. Again make sure all quantifiers are explicit and expressed explicitly as "for all" or "there exists."

6. What is the form of the definition of a greatest common divisor, using $s(x, y, z)$ to be the statement $x = yz$ and $t(x, y)$ to be the statement $x < y$?

7. Which of the following statements (in which Z^+ stands for the positive integers and Z stands for all integers) is true and which is false, and why?

a. $\forall z \in Z^+(z^2 + 6z + 10) > 20$.

b. $\forall z \in Z(z^2 - z \geq 0)$.

c. $\exists z \in Z^+(z - z^2 > 0)$.

d. $\exists z \in Z(z^2 - z = 6)$.

8. Are there any quantifiers in the statement "The product of odd integers is odd?"

9. Rewrite the statement "The product of odd numbers is odd," with all quantifiers (including any in the definition of odd numbers) explicitly stated as "for all" or "there exist."

10. Rewrite the following statement without any negations. It is not the case that there exists an integer n such that $n > 0$ and for all integers $m > n$, for every polynomial equation $p(x) = 0$ of degree m there are no real numbers for solutions.

11. Let $p(x)$ stand for "x is a prime," $q(x)$ for "x is even," and $r(x, y)$ stand for "$x = y$." Write down the statement "There is one and only one even prime," using these three symbolic statements and appropriate logical notation. (Use the set of integers for your universe.)

12. Each expression below represents a statement about the integers. Using $p(x)$ for "x is prime," $q(x, y)$ for "$x = y^2$," $r(x, y)$ for "$x \leq y$," $s(x, y, z)$ for "$z = xy$," and $t(x, y)$ for "$x = y$," determine which expressions represent true statements and which represent false statements.

    (a) $\forall x \in Z(\exists y \in Z(q(x, y) \vee p(x)))$

    (b) $\forall x \in Z(\forall y \in Z(s(x, x, y) \Leftrightarrow q(x, y)))$

    (c) $\forall y \in Z(\exists x \in Z(q(y, x)))$

    (d) $\exists z \in Z(\exists x \in Z(\exists y \in Z(p(x) \wedge p(y) \wedge \neg t(x, y))))$

13. Find a reason why $(\exists x \in U(p(x))) \wedge (\exists y \in U(q(y)))$ is not equivalent to $\exists z \in U(p(z) \vee q(z))$. Are the statements $(\exists x \in U(p(x))) \vee (\exists y \in U(q(y)))$ and $\exists z \in U(p(z) \vee q(z))$ equivalent?

14. Give an example (in English) of a statement that has the form $\forall x \in U(\exists y \in V(p(x, y)))$. (The statement can be a mathematical statement or a statement about "every day life," or whatever you prefer.) Now write in English the statement using the same $p(x, y)$ but of the form $\exists y \in V(\forall x \in U(p(x, y)))$. Comment on whether "for all" and "there exist" commute.

3.3 Inference

Direct Inference (Modus Ponens) and Proofs

We concluded our last section with a proof that the sum of even numbers is even. There were several crucial ingredients in that proof. One consisted of introducing symbols for members of the universe of integers numbers. In other words, rather than saying "suppose we have two integers," we introduced symbols for the two members of our universe we assumed we had. How did we know to use algebraic symbols? As usual, there is more than one possible answer to this question. In this case, our intuition was probably based on thinking about what an even number is, and realizing that the definition itself is essentially symbolic. (You may argue that an even number is just twice another number, and you would be right. Apparently there are no symbols are in that definition. But they really are there; they are the phrases "even number" and "another number." Since we all know algebra is easier with symbolic variables rather than words, we should recognize that it makes sense to use algebraic notation.) Thus this decision was based on experience, not logic.

Next we assumed the two integers we were talking about were even. We then used the definition of even numbers, and our previous parenthetic comment suggests that it was natural for us to use the definition symbolically. The definition tells us that if m is an even number, then there exists another integer i such that $m = 2i$. We combined this with the assumption that m is even to conclude that in fact there does exist an integer i such that $m = 2i$. This is an example of using the principle of *direct inference* (called *modus ponens* in Latin, though we won't worry about Latin names in this course.) The principle says that from p and $p \Rightarrow q$ we may conclude q. This principle is a cornerstone of logical arguments. If you think about the truth table for $p \Rightarrow q$, you can view this rule as describing one row of this truth table. There are quite a few such rules (called rules of inference) that people commonly use in proofs without explicitly stating them. We will try to tease out which rules we were using in the proof before beginning a formal study of rules of inference.

We then used algebra to show that because $m = 2i$ and $n = 2j$, there exists a k such that $m + n = 2k$ (our k was $i + j$). Next we used the definition of even number again. We then used a rule of inference which says that if, by assuming p, we may prove q, then the statement $p \Rightarrow q$ is true. We finally reached a grand conclusion that for all pairs of integers, if they are even, then their sum is even. Here is another rule of inference, one of the more difficult to describe. We introduced the variables m and n. We used only well-known consequences of the fact that they were in the universe of integers in our proof. Thus we felt justified in asserting that what we concluded about m and n is true for any pair of integers. We might say that we were treating m and n as generic members of our universe. Thus our rule of inference says that if we can prove a statement for a generic member of our universe, then we can conclude it is true for every member of our universe. Perhaps the reason this rule is hard to put into words is that it is not simply a description of a truth table, but is a principle that we use in order to prove universally quantified statements.

Rules of inference

We have seen the ingredients of a typical proof. What do we mean by a proof in general? A proof of a statement is a convincing argument that the statement is true. To be more precise

about it, we can agree that a proof consists of a sequence of statements, each of which is either a hypothesis (to be described in more detail in our description of rules of inference below), a generally accepted fact, or the result of one of the following rules of inference for compound statements.

1 From an example x that does not satisfy $p(x)$, we may conclude $\neg p(x)$.

2 From $p(x)$ and $q(x)$, we may conclude $p(x) \wedge q(x)$.

3 From either $p(x)$ or $q(x)$, we may conclude $p(x) \vee q(x)$.

4 From either $q(x)$ or $\neg p(x)$ we may conclude $p(x) \Rightarrow q(x)$.

5 From $p(x) \Rightarrow q(x)$ and $q(x) \Rightarrow p(x)$ we may conclude $p(x) \Leftrightarrow q(x)$.

6 From $p(x)$ and $p(x) \Rightarrow q(x)$ we may conclude $q(x)$.

7 From $p(x) \Rightarrow q(x)$ and $q(x) \Rightarrow r(x)$ we may conclude $p(x) \Rightarrow r(x)$.

8 If we can derive $q(x)$ from the hypothesis that x satisfies $p(x)$, then we may conclude $p(x) \Rightarrow q(x)$.

9 If we can derive $p(x)$ from the hypothesis that x is a (generic) member of our universe U, we may conclude $\forall x \in U(p(x))$.

10 From an example of an $x \in U$ satisfying $p(x)$ we may conclude $\exists x \in U(p(x))$.

The first four rules are in effect a description of the truth tables for compound statements. Rule 5 says what we must do in order to write a proof of an "if and only if" statement. Rule 6, exemplified in our discussion above, is the principle of logical reasoning we called *direct inference*. We may think of it as describing one row of the truth table for $p \Rightarrow q$. Rule 7 is the transitive law, one we could derive by truth table analysis. Rule 8, which is also exemplified above, may be regarded as yet another description of one row of the truth table of $p \Rightarrow q$. Rules 9 and 10 specify what we mean by the truth of a quantified statement.

A number of our rules of inference are redundant. However they are useful. For example, we could have written a portion of our proof that the sum of even numbers is even as follows.

> "Let m and n be integers. If m is even, then there is a k with $m = 2k$. If n is even, then there is a j with $n = 2j$. Thus if m is even and n is even, there are a k and j such that $m + n = 2k + 2j = 2(k + j)$. Thus if m is even and n is even, there is an integer $h = k + j$ such that $m + n = 2h$. Thus if m is even and n is even, $m + n$ is even."

This kind of argument could always be used to circumvent the use of rule 8, so rule 8 is not required as a rule of inference, but because it permits us to avoid such unnecessarily complicated "silliness" in our proofs, we choose to include it. Rule 7, the transitive law, has a similar role.

3.3-1 Prove that if m is even, then m^2 is even.

3.3-2 Show that "p implies q" is equivalent to "$\neg q$ implies $\neg p$."

3.3-3 Is "p implies q" equivalent to "q implies p?"

In Exercise 3.3-1 we can mimic the proof that the sum of even numbers is even. We let m be an even number, use the fact that it is $2i$ for some i, and observe that $m^2 = (2i)^2 = 2 \cdot (2i^2)$, which lets us conclude that m^2 is even.

Contrapositive rule of inference.

In Exercise 3.3-2 we saw that if we know that $\neg q \to \neg p$, then we can conclude that $p \to q$. To see what that is good for, consider the following example.

Prove that if n is a positive integer with $n^2 > 100$, then $n > 10$.

Proof: Suppose n is not greater than 10. (Now we use the rule of algebra for inequalities which says that if $x \leq y$ and $c \geq 0$, then $cx \leq cy$.) Then since $1 \leq n \leq 10$,

$$n \cdot n \leq n \cdot 10 \leq 10 \cdot 10 = 100.$$

Thus n^2 is not greater than 100. Therefore, if $n^2 > 100$, n must be greater than 10.■

We could give an intuitive explanation of the reasoning (and doing so would be a good exercise), but we don't need to because we can see we are just using the result of Exercise 2.

We adopt the result of Exercise 2 as a rule of inference, called the *contrapositive* rule of inference.

11 From $\neg q \Rightarrow \neg p$ we may conclude $p \Rightarrow q$.

In our proof of the Chinese Remainder Theorem, Theorem 2.3.6, we wanted to prove that for a certain function f that if x and y were different integers between 0 and $mn-1$, then $f(x) \neq f(y)$. To prove this we assumed that in fact $f(x) = f(y)$ and proved that x and y were not different integers between 0 and $mn - 1$. Had we known the principle of contrapositive inference, we could have concluded then and there that f was one-to-one. Instead, we used the more common principle of proof by contradiction, discussed in detail below, to complete our proof. If you look back at the proof, you will see that we might have been able to shorten it by a sentence by using contrapositive inference.

A quick look at the truth tables for $p \Rightarrow q$ and $q \Rightarrow p$ should convince us that these two statements are *not* equivalent. $q \Rightarrow p$ is called the converse of $p \Rightarrow q$. It is surprising how often people, even professional mathematicians, absent-mindedly prove the converse of a statement when they mean to prove the statement itself. Try not to join this crowd!

Proof by contradiction

Proof by contrapositive inference is an example of what we call *indirect inference*. We have actually seen another example indirect inference, the method of proof by contradiction. Recall that in our proof of Euclid's Division Theorem we began by assuming that the theorem was false. We then chose among the pairs of integers (m, n) such that $m \neq qn + r$ with $0 \leq r < n$ a pair with the smallest possible m. We then made some computations by which we proved that in this case there *are* a q and r with $0 \leq r < n$ such that $m = qn + r$. Thus we started out by assuming the theorem was false, and from that assumption we drew drew a contradiction to the

assumption. Since all our reasoning, except for the assumption that the theorem was false, used accepted rules of inference, the only source of that contradiction was our assumption. Thus our assumption had to be incorrect. This leads us to another rule of inference, called the principle of *proof by contradiction* or the principle of *reduction to absurdity*.

12 If from assuming p and $\neg q$, we can derive both r and $\neg r$ for some statement r, then we may conclude $p \Rightarrow q$.

There can be many variations of proof by contradiction. For example, we may assume p is true and q is false, and from this derive the contradiction that p is false, as in the following example.

Prove that if $x^2 + x - 2 = 0$, then $x \neq 0$.

Proof: Suppose that $x^2 + x - 2 = 0$. Assume that $x = 0$. Then $x^2 + x - 2 = 0 + 0 - 2 = -2$. This contradicts $x^2 + x - 2 = 0$. Thus (by the principle of proof by contradiction), if $x^2 + x - 2 = 0$, then $x \neq 0$. ■

Here the statement r was identical to p, namely $x^2 + x - 2 = 0$.

On the other hand, we may instead assume p is true and q is false, and derive a contradiction of a known fact, as in the following example.

Prove that if $x^2 + x - 2 = 0$, then $x \neq 0$.

Proof: Suppose that $x^2 + x - 2 = 0$. Assume that $x = 0$. Then $x^2 + x - 2 = 0 + 0 - 2 = -2$. Thus $0 = -2$, a contradiction. Thus (by the principle of proof by contradiction), if $x^2 + x - 2 = 0$, then $x \neq 0$. ■

Here the statement r is the known fact that $0 \neq -2$.

Sometimes the statement r that appears in the principle of proof by contradiction is simply a statement that arises naturally as we are trying to construct our proof, as in the following example.

Prove that if $x^2 + x - 2 = 0$, then $x \neq 0$.

Proof: Suppose that $x^2 + x - 2 = 0$. Then $x^2 + x = 2$. Assume that $x = 0$. Then $x^2 + x = 0 + 0 = 0$. But this is a contradiction (to our observation that $x^2 + x = 2$). Thus (by the principle of proof by contradiction), if $x^2 + x - 2 = 0$, then $x \neq 0$. ■

Here the statement r is "$x^2 + x = 2$."

Finally, if proof by contradiction seems to you not to be much different from proof by contraposition, you are right, as the example that follows shows.

Prove that if $x^2 + x - 2 = 0$, then $x \neq 0$.

Proof: Assume that $x = 0$. Then $x^2 + x - 2 = 0 + 0 - 2 = -2$, so that $x^2 + x - 2 \neq 0$. Thus (by the principle of proof by contraposition), if $x^2 + x - 2 = 0$, then $x \neq 0$. ■

The last four examples illustrate the rich possibilities that indirect proof provides us. Of course they also illustrate why indirect proof can be confusing. There is no set formula that we use in writing a proof by contradiction, so there is no rule we can memorize in order to formulate indirect proofs. Instead, we have to ask ourselves whether assuming the opposite of what we are trying to prove gives us insight into why the assumption makes no sense. If it does, we have the basis of an indirect proof, and the way in which we choose to write it is a matter of personal choice.

3.3-4 Without extracting square roots, prove that if n is a positive integer such that $n^2 < 9$, then $n < 3$. You may use rules of algebra for dealing with inequalities.

3.3-5 Prove that $\sqrt{5}$ is not rational.

To prove the statement in Exercise 3.3-4, we assume, for purposes of contradiction, that $n \geq 3$. Squaring both sides of this equation, we obtain

$$n^2 \geq 9 \ ,$$

which contradicts our hypothesis that $n^2 < 9$.

To prove the statement in Exercise 3.3-5, we assume, for the purposes of contradiction that 5 is rational. This means that it can be expressed as the fraction $\frac{m}{n}$, where m and n are integers and $\gcd(m, n) = 1$. Squaring both sides of the equation $\frac{m}{n} = \sqrt{5}$, we obtain

$$\frac{m^2}{n^2} = 5 \ .$$

Now the numerator must be divisible by 5, or else the fraction would not be divisible by 5. This means that m is divisible by 5, and m^2 is actually divisible by 25. But this means that the denominator must be divisible by 5. So the numerator and denominator are both divisible by 5, and hence $\gcd(m, n) \neq 1$, which is a contradiction.

Problems

1. Write down the converse and contrapositive of each of these statements.
   (a) If the hose is 60 feet long, then the hose will reach the tomatoes.
   (b) George goes for a walk only if Mary goes for a walk.
   (c) Pamela recites a poem if Andre asks for poem.
2. Construct a proof that if m is odd, then m^2 is odd.
3. Construct a proof that for all integers m and n, if m is even and n is odd, the $m + n$ is odd.
4. What do we really mean when we say "prove that if m is odd and n is odd then $m + n$ is even?" Prove this more precise statement.
5. Prove that for all integers m and n if m is odd and n is odd, then $m \cdot n$ is odd.
6. Is the statement $p \Rightarrow q$ equivalent to the statement $\neg p \Rightarrow \neg q$?

7. Construct a contrapositive proof that for all real numbers x if $x^2 - 2x \neq -1$, then $x \neq 1$.

8. Construct a proof by contradiction that for all real numbers x if $x^2 - 2x \neq -1$, then $x \neq 1$.

9. Prove that if $x^3 > 8$, then $x > 2$.

10. Prove that $\sqrt{3}$ is irrational.

11. Construct a proof that if m is an even integer m^2 is even, then m is even.

12. Prove or disprove the following statement. "For every positive integer n, if n is prime, then 12 and $n^3 - n^2 + n$ have a common factor."

13. Prove or disprove the following statement. "For all integers b, c, and d, if x is a rational number such that $x^2 + bx + c = d$, then x is an integer." (Hints: Are all the quantifiers given explicitly? It is ok to use the quadratic formula.)

14. Prove that there is no largest prime number.

15. Prove that if f, g and h are functions from R^+ to R^+ such that $f = O(g)$ and $g = O(h)$, then $f = O(h)$.

Chapter 4

Induction, Recursion and Recurrences

4.1 Mathematical Induction

Smallest Counter-Examples

We've seen one way of proving statements about infinite universes, namely consider a "generic" member of the universe and try to derive 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 (Lemma 2.2.2), which says that for each pair (k, n) of positive integers, there are nonnegative integers q and r such that $k = nq + r$ and $r < n$. "Among all pairs (k, n) that make it false, choose the smallest k that makes it false. We cannot have $k < n$ because then the statement would be true with $q = 0$, and we cannot have $k = n$ because then the statement is true with $q = 1$ and $r = 0$. This means $k - n$ 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 $k = (q + 1)n + r$, contradicting the assumption that the statement is false, so the only possibility is that the statement is true."

Focus on the sentences "This means $k - n$ 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 $k = (q + 1)n + r, \ldots$." To analyze these sentences, let $p(k, n)$ be the statement "there are nonnegative integers q and r with $0 \leq r < n$ such that $k = nq + r$" The sentences we quoted are a proof that $p(k - n, n) \Rightarrow p(k, n)$. It is this implication that makes the proof work. In outline our proof went like this: we assumed a counter-example with a smallest k existed, then we took advantage of the fact that $p(k', n)$ had to be true for every k' smaller than k, we chose $k' = k - n$, and used the implication $p(k - n, n) \Rightarrow p(k, n)$ to conclude the truth of $p(k, n)$, which we had assumed to be false. This is what gave us our contradiction in our proof by contradiction.

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

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

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?

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?

4.1-3 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,

$$1 + 2 + \cdots + i = \frac{i(i+1)}{2}. \tag{4.1}$$

Because $1 = 1 \cdot 2/2$, we know $n > 1$, so that $n - 1$ is one of the positive integers i for which the formula holds. Substituting $n - 1$ for i in Equation 4.1 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. For this purpose, we assume that the statement that "$p(n)$ is true for all nonnegative integers n" is false. When a forall statement is false, there must be some n for which it is false. Therefore, there is some smallest nonnegative integer n so that $2^{n+1} \not\geq n^2 + 2$. 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

$$\begin{aligned} 2^n &\geq n^2 - 2n + 1 + 2 \\ &= n^2 - 2n + 3. \end{aligned} \tag{4.2}$$

From this we want to draw a contradiction, presumably a contradiction to $2^{n+1} \ngeq n^2 + 2$.

To get the contradiction, we want to convert the left-hand side of Equation 4.2 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 get a contradiction, we want to see a way to convert the right hand side into something like $n^2 + 2$. Thus we write

$$\begin{aligned} 2^{n+1} &\geq 2n^2 - 4n + 6 \\ &= n^2 + 2 + n^2 - 4n + 4 \\ &= n^2 + 2 + (n-2)^2 \\ &\geq n^2 + 2\ , \end{aligned} \tag{4.3}$$

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

What implication did we have to prove above? Let us use $p(n)$ to stand for $2^{n+1} \geq n^2 + 2$. Then in Equations 4.2 and 4.3 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. This might lead you to ask whether it would 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 $p(n-1) \Rightarrow p(n)$ give us $p(1)$, $p(1)$ and $p(n-1) \Rightarrow p(n)$ give us $p(2)$, and so on so that we have $p(k)$ for every k without having to deal with any contradictions. This is an excellent question which we will address shortly.

Now let's consider a more complicated example in which $p(0)$ (and $p(1)$) are not true, but $p(n)$ is true for larger n. Notice that $2^{n+1} \ngtr n^2 + 3$ 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 $n \geq 2$. So let us let $p'(n)$ be the statement $2^{n+1} > n^2 + 3$. We can easily prove $p'(2)$: since $8 = 2^3 \geq 2^2 + 3 = 7$. 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 $p'(n-1) \Rightarrow p'(n)$. Thus we conclude from $p'(2)$ and $p'(2) \Rightarrow p'(3)$ that $p'(3)$ is true, and similarly for $p'(4)$, and so on.

The Principle of Mathematical Induction

It may seem clear that this procedure of repeatedly using the implication will prove $p(n)$ (or $p'(n)$) for all n (or all $n \geq 2$). That observation is the central idea of the Principle of Mathematical

Induction, described in what follows. In a theoretical discussion of how one constructs the integers from first principles, this principle (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. Thus it is not surprising that the principle is a fundamental one in discrete mathematics. The principle of mathematical induction is usually described in two forms. The one we have talked about so far is called the "weak form."

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

4.1-4 Use mathematical induction to show that

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

for each positive integer k

4.1-5 For what values of n is $2^n > n^2$? Use mathematical induction to show that your answer is correct.

For Exercise 4.1-4, we note that the formula holds when $k = 1$. Assume inductively that the forumla 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

$$\begin{aligned} 1 + 3 + \cdots + (2n - 3) + (2n - 1) &= n^2 - 2n + 1 + 2n - 1 \\ &= n^2 \ . \end{aligned} \tag{4.4}$$

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 4.1-4 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 4.4 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 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 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 talking about 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 4.1-5 we note that $2 > 1$, 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

$$2^n > 2(n^2 - 2n + 1) = n^2 + n^2 - 4n + 2 > n^2 + n^2 - n \cdot n = n^2,$$

since $n > 6$ implies that $-4n > -n \cdot n$. 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 $p(n - 1)$ is true. This gives us $2^{n-1} > (n - 1)^2$. Multiplying by 2 gives

$$2^n > 2(n^2 - 2n + 1) = n^2 + n^2 - 4n + 2 > n^2 + n^2 - n \cdot n = n^2,$$

since $n > 6$ 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 indcution from the context 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 gives us

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

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, there is no need to restrict ourselves to proving the implication $p(n-1) \Rightarrow p(n)$. In this case, we proved the implication $p(k) \Rightarrow p(k+1)$, and 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 is different 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 this case is that we have a statement $q(k)$ we want to prove, we suppose it is false so there is a smallest k for which $q(k)$ is false, meaning 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 the other 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. This is common in using this version of induction, so we do not treat the first of the two implications separately.) This leads us to 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 $n \geq b$.

4.1-6 Prove that every positive integer is a power of a prime number or the product of powers of prime numbers.

In this exercise 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 course. We typically will not distinguish between strong and weak induction, we just think of them both as induction. (In Exercise 4.1-14 and Exercise 4.1-15 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 proof by mathematical induction showing that a statement $p(n)$ is true for all integers $n \geq b$ consists of two steps. First we show that $p(b)$ is true. This is called "establishing a *base case*." 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 $p(b) \wedge p(b+1) \wedge \ldots \wedge p(n-1) \Rightarrow p(n)$. This is where the core of an inductive proof lies, and is usually where we need the most insight into what we are trying to prove. As we have discussed, this suffices to show that $p(n)$ is true for all $n \geq b$. You will often see inductive proofs in which the implication being proved is $p(n) \Rightarrow p(n+1)$. In light of our discussion of Exercise 4.1-5, it should be clear that this is simply another variation of writing an inductive proof.

It is important to realize that induction arises in many circumstances that do not fit the "pat" description we gave above. However, inductive proofs always involve two 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. For example, consider the problem of proving the following statement:

$$\sum_{i=0}^{n} \lfloor \frac{i}{2} \rfloor = \begin{cases} \frac{n^2}{4} & \text{if } n \text{ is even} \\ \frac{n^2-1}{4} & \text{if } n \text{ is odd} \end{cases} \tag{4.5}$$

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. 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.

We can even consider more involved examples. Consider proving the following by induction:

$$\forall n \geq 1 (n \geq 2^{\lfloor \log_2 n \rfloor}). \tag{4.6}$$

While there may be easier ways to prove this statement, we will outline how a proof by induction could go. We could easily prove $p(1)$. We could then prove that, if n is a power of 2, that $p(n/2) \Rightarrow p(n)$. Finally we could show that, for n not a power of 2 and $a < n$, that $p(n) \Rightarrow p(n+a)$. These three things together would "cover" all the integers and would suffice to prove (4.6).

Logically speaking, we could rework the examples above so that they fit 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 both the mathematics and 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.

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 priciple 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 $1^4 + 2^4 + \cdots + n^4 = O(n^5)$.

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 (4.5).

17. Prove (4.6) by induction.

4.2 Recursion, Recurrences and Induction

Recursion

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

4.2-2 Recall that in the Towers of Hanoi problem we have three pegs, and on one peg we have a stack of n disks, each smaller in diameter than the one below it. An allowable move consists of removing a disk 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 used in a recursive program to solve this problem.

For the Tower of Hanoi problem, you told the computer that to solve the problem with no disks you do nothing. To solve the problem of moving all disks to peg 2, you deal with the problem of moving $n-1$ disks to peg 3, then move disk n to peg 2, and then deal with the problem of moving the $n-1$ 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 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. 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. 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:

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

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. They are both 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

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.

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.

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

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

The main idea 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 will give us the inductive step of a proof by induction that the recurrence correctly describes the number of subsets of an n-element set for all n

How, exactly, does the inductive proof go? We know the initial condition of our recurrence properly describes the number of subsets of a zero-element set, namely 1 (remember that the empty set has itself and only itself as a subset). Now we suppose our recurrence properly describes the number of subsets of a k-element set for k less than n. Then since the number of subsets of an n-element set containing a specific element x is the same as the number of subsets without element x, the total number of number of subsets of an n-element set is just twice the number of subsets of an $(n-1)$-element set. Therefore, because the recurrence properly describes the number of subsets of an $(n-1)$-element subset, it properly describes the number of subsets of an n-element set. Therefore by the principle of mathematical induction, our recurrence describes the number of subsets of an n-element set for all integers $n \geq 0$. Notice how the proof by induction goes along exactly the same lines as our derivation of the recurrence. This is one of the beautiful facets of the relationship between recurrences and induction; usually what we do to figure out a recurrence is exactly what we would do to prove we are right!

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

$$\begin{aligned} T(n) &= r^nT(0) + a\sum_{i=0}^{n-1} r^i \\ &= r^nb + a\sum_{i=0}^{n-1} r^i. \end{aligned}$$

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

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

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 $\sum_{i=0}^{n-1} r^i$. 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 that $(1-x)(1+x) = 1-x^2$. You may or may not have seen in algebra that $(1-x)(1+x+x^2) = 1-x^3$ or $(1-x)(1+x+x^2+x^3) = 1-x^4$. These factorizations are easy to verify, and they suggest that $(1-r)(1+r+r^2+\cdots+r^{n-1}) = 1-r^n$, or

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

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

Theorem 4.2.1 *If* $T(n) = rT(n-1) + a$, $T(0) = b$, *and* $r \neq 1$ *then*

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

for all nonnegative integers n.

Proof: We will prove our formula by induction. Notice that the formula gives $T(0) = r^0 b + a\frac{1-r^0}{1-r}$ 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. ■

Corollary 4.2.2 *The formula for the sum of a geometric series with $r \neq 1$ is*

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

Proof: Take $b = 0$ and $a = 1$ in Theorem 4.2.1. Then the sum $1 + r + \cdots + r^{n-1}$ satisfies the recurrence given. ■

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 4.2.3 *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 4.2.4 *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 4.2.3, 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 $T(n) = f(n)T(n-1) + g(n)$ 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

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

This suggests our next theorem.

Theorem 4.2.5 *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

$$T(n) = r^n a + \sum_{i=1}^{n} r^{n-i} g(i). \tag{4.11}$$

Proof: Let's prove this by induction.

Since the sum in equation 4.11 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 4.11 for all nonnegative integers n. ■

The formula in Theorem 4.2.5 is a little less easy to use than that in Theorem 4.2.1, but for a number of commonly occurring functions g the sum $\sum_{i=1}^{n} r^{n-i} g(i)$ is reasonable to compute.

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

Using Equation 4.11, we can write

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

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

Problems

1. 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 Equation 4.7?
2. 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 Equation 4.7.
3. 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 Equation 4.7. Using iteration of the recurrence, we can write

$$\begin{aligned} M(n) &= M(n-1) + 2 \\ &= M(n-2) + 2 + 2 \\ &= M(n-3) + 2 + 2 + 2 \\ &\vdots \\ &= M(n-i) + \underbrace{2 + 2 + \cdots + 2}_{i \text{ times}} \\ &= M(n-i) + 2i \end{aligned}$$

Since we are given that $M(1)$ is 1, we take $i = n - 1$ to get

$$\begin{aligned} &= M(1) + 2(n-1) \\ &= 1 + 2(n-1) \\ &= 2n - 1 \end{aligned}$$

Now, we verify the correctness of our derivation through mathematical induction.

(a) Base Case : $n = 1$: $M(1) = 2 \cdot 1 - 1 = 1$. Thus, base case is true.

(b) Suppose inductively that $M(n-1) = 2(n-1) - 1$.
Now, for $M(n)$, we have

$$\begin{aligned} M(n) &= M(n-1) + 2 \\ &= 2n - 2 - 1 + 2 \\ &= 2n - 1 \end{aligned}$$

So, from $1, 2$ and the principle of mathematical induction, the statment is true for all n. We could have gotten the same result by using Theorem 4.2.1.

Here, the recurrence is of the form $M(n) = M(n-1) + 2$ while in (4.8), it is of the form $M(n) = 2M(n-1) + 1$. Thus, in Theorem 4.2.1, r would have been 1, and thus our geometric series sums to a multiple of the number of terms.

4. 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.

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

6. 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.

7. Consider the recurrence $T(n) = 3T(n-1) + 1$ with the initial condition that $T(0) = 2$. We know from Theorem 4.2.1 exactly what the solution is. 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.

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

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

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

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

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

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

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

15. Solve the recurrence $T(n) = rT(n-1) + n$ with $T(0) = 1$. (There is a sum here that you may not know; thinking about the derivative of $1 + x + x^2 + \cdots + x^n$ will help you figure it out.)

16. 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 Rooted Trees

The idea of a rooted tree

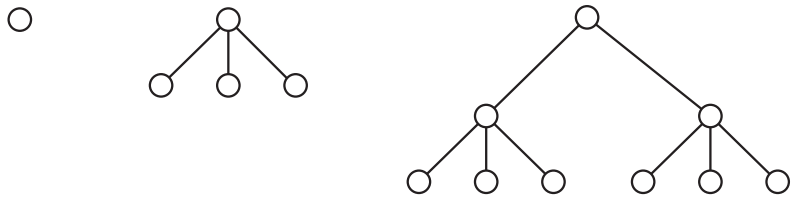
You have seen trees as data structures for solving a variety of different kinds of problems. Mathematically speaking, a tree is a geometric figure consisting of vertices and edges—but of special kind. We think of vertices as points in the plane and we think of edges as line segments. In this section we will study some of the properties of trees that are useful in computer science. Most, but not all, of the trees we use in computer science have a special vertex called a root. Such trees are usually called "rooted trees." We could certainly describe what we mean by a rooted tree in ordinary English. However it is particularly easy to give a recursive description of what we mean by a rooted tree. We say

A vertex P is a rooted tree with root P.

If T_1, T_2,...T_k are rooted trees with roots R_i, and P is a vertex, then drawing edges from P to each of the roots R_i gives a rooted tree T with root P. We call the trees T_1, T_2,...T_k *subtrees* of the root P our rooted tree T.

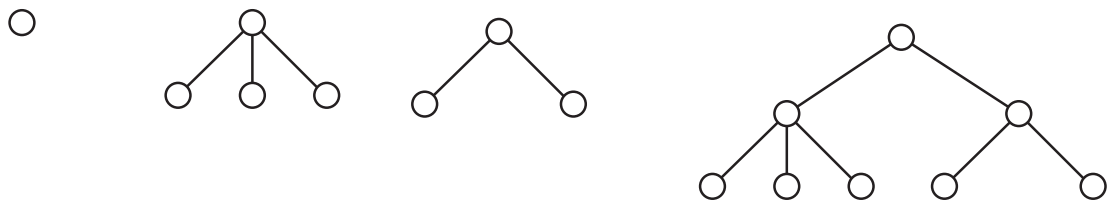
Any geometric figure we can construct by using these two rules is called a rooted tree. Just as we don't really define points and lines in geometry, we won't formally define the terms vertex and edge here. We do note that a standard synonym for vertex is *node*, and that edges represent relationships between vertices. In Figure 4.1 we show first a single vertex. We then imagine three trees which are copies of this single vertex and connect them to a new root. Finally we take two copies of the resulting tree and attach them to a new root.

Figure 4.1: The evolution of a rooted tree.



Of course we need not take multiple copies of the same tree. In Figure 4.2 we began as in Figure 4.1 but at the second stage, we created two different trees, one with two one-vertex trees attached to the root and one with three one-vertex trees attached to the root. Then at the third stage we attached the two trees from the second stage to a common root.

Figure 4.2: The evolution of a less symmetric rooted tree.



The number of vertices and edges in a tree

4.3-1 Draw 5 rooted trees with 7 vertices each. How many edges do they have? What do you think the relationship between the number of vertices and edges is in general. Prove that you are right.

In Exercise 4.3-1 you saw that each of your 7-vertex trees had six edges. This suggests a pattern, namely that a rooted tree with n vertices has $n-1$ edges. How would we go about proving that? Well, a rooted tree with one vertex has no edges, because in our recursive definition the only way to get a tree with one vertex is to take a single vertex called a root. (That sounds like a base step for an inductive proof, doesn't it?) If we have a rooted tree, it has a root vertex attached by edges to a certain number, say k, of trees. If these k trees have sizes n_1, n_2,..., n_k, then we may assume inductively that they have $n_1 - 1$, $n_2 - 1$,..., $n_k - 1$ edges respectively. (That should sound like the inductive hypothesis of an inductive proof.) If our original tree has n vertices, then these trees have a total of $n - 1 = \sum_{i=1}^{k} n_i$ vertices and

$$\sum_{i=1}^{k}(n_i - 1) = (\sum_{i=1}^{k} n_i) - k = n - 1 - k$$

edges. In addition to these vertices and edges, our original tree has 1 more vertex and k more edges (those that connect the subtrees to the root). Thus our original rooted tree has $(n-1-k)+k = n-1$ edges. Therefore by the (strong) principle of mathematical induction, a rooted tree on n vertices has $n-1$ edges.

Notice how this inductive proof mirrored the recursive definition of a rooted tree. Notice also that while it is an induction on the number n of vertices of the tree, we didn't really mention n in our inductive hypothesis. So long as we have the correct structure for an inductive proof, it is quite acceptable to leave the number on which we are inducting implicit instead of explicit in our proof.

We have just proved the following theorem.

Theorem 4.3.1 *A rooted tree with n vertices has $n-1$ edges.*

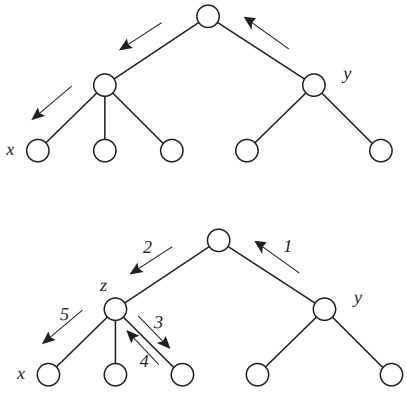
Paths in trees

A walk from vertex x to vertex y is an alternating sequence of vertices and edges starting at x and ending at y such that each edge connects the vertex that precedes it with the vertex that follows it. A path is a walk without repeated vertices. In Figure 4.3 we show a path from y to x by putting arrows beside its edges; we also show a walk that is not a path, using arrows with numbers to indicate its edges. In this walk, the vertex marked z is used twice (as is an edge).

4.3-2 Draw a rooted tree on 12 vertices. (Don't draw a "boring" one.) Choose two vertices. Can you find a path between them. Can you find more than one? Try another two vertices. What do you think is true in general? Can you prove it?

How **do** we prove that there is exactly one path between any two vertices of a rooted tree? It is probably no surprise that we use induction. In particular, in a rooted tree with one vertex the

Figure 4.3: A path and a walk in a rooted tree.



only path or walk is the "alternating sequence of vertices and edges" that has no edges, namely that single vertex. That single vertex is the unique path from the vertex to itself, so in a tree with one vertex we have a unique path between any pair of vertices. Now let T be a rooted tree with root vertex r, and let T_1, T_2, ..., T_k be the subtrees whose roots r_1, r_2, ..., r_k are joined to r by edges. Assume inductively that in each T_i there is a unique path between any pair of vertices. Then if we have a pair x and y of vertices in T, they are either in the same T_i (case 1), different subtrees T_i and T_j of the root (case 2), or one is the root and one is in a subtree T_i (case 3). In case 1, a walk from x to y that leaves tree T_i must enter it again, and so it must use vertex r twice. Thus the unique path between x and y in T_i is the unique path between x and y in T. In case 2 there is a unique path between x and r_i in their subtree T_i and a unique path between r_j and y in their subtree T_j. Thus in T we have a path from x to r_i which we may follow with the edge from r_i to r, then the vertex r, then the edge from r to r_j, and finally the path from r_j to y, which gives us a path from x to y.

Now we show that this is the unique path between x and y. When a path from x to y leaves T_i it must use the only edge that has one vertex in T_i and one vertex outside T_i, namely the edge from r_i to r. To get to this edge, the path must follow the unique path from x to r_i in T_i. The path continues along this edge to r. To get from r to y, the path must enter the subtree T_j, because y is in that subtree. When the path enters tree T_j, it must do so along the only edge that has one vertex in T_j and one vertex outside T_j, namely the edge from r to r_j. Before it enters T_j, the path cannot go to any other root r_k with $k \neq j$, thus entering tree T_k, because to get to y it would eventually have to leave the tree T_k and return to r in order to enter the tree T_j. But a walk that uses r twice would not be a path. Therefore the vertex following r in the path must be r_j, and there is only one way to complete the path by going from r_j to y in T_j. Therefore the path from x to y in T is unique.

Finally in case 3 suppose x is in the subtree T_i and y is the root of T. By our inductive hypothesis there is a unique in T_i from x to the root r_i of T_i. Following this path with the edge from r_i to r and the vertex r gives us a path from x to y since $y = r$. As before when a path leaves T_i it must go from r_i to r along the unique edge connecting them. Since r is the last vertex twice, the path could not have any more edges or vertices or r would have to appear in the path a second time. However since there is only one path from x to r_i, there is only one way for the path to reach r_i. This means that the the path we described is the only possible one.

Thus by the principle of mathematical induction, there is a unique path between any two vertices in a rooted tree.

We have just proved the following.

Theorem 4.3.2 *For each vertex x and y in a rooted tree, there is a unique path joining x to y.*

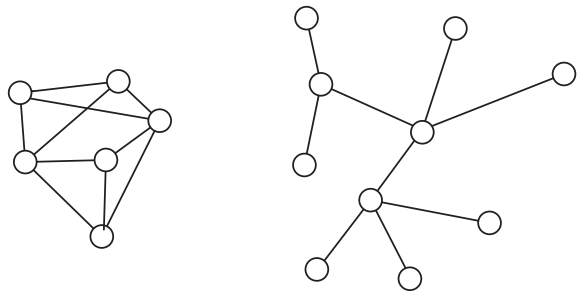
Notice that in our inductive proof, we never mentioned of an integer n. There were integers there implicitly—the number of vertices in the whole tree and the number in each subtree are relevant—but it was unnecessary to mention them. Instead we simply used the fact that we were decomposing a structure (the tree) into smaller structures of the same type (the subtrees) as our inductive step and the fact that we could prove our result for the smallest substructures we could get to by decomposition (in this case, the single vertex trees) as our base case. It is clear that you could rewrite our proof to mention integers, and doing so would make it clear that mentioning them adds nothing to the proof. When we use mathematical induction in this way, using smaller structures in place of smaller integers, people say we are using *structural induction*.

In a rooted tree we refer to the nodes connected to the root as its *children* and we call the root their *parent*. We call the members of a subtree of r *descendents* of r, and we call r their *ancestor*. We use this language recursively throughout the tree. Thus in Figure 4.3 vertex x is a child of vertex z, but vertex x is not an ancestor of vertex y.

Graphs and trees

Our use of the phrase "rooted tree" suggests that there must be a concept of a unrooted tree or simply a tree. In fact there is such a concept; it is a special case of the idea of a graph. A *graph* consists of a set called a vertex set and another set called an edge set. While we don't define exactly what we mean by vertices or edges, we do require that each edge is associated with two vertices that we call its endpoints. Walks and paths are defined in the same way they are for rooted trees. We draw pictures of graphs much as we would draw pictures of rooted trees (we just don't have a special vertex that we draw at the "top" of the drawing). In Figure 4.4 we show a typical drawing of a graph, and a typical drawing of a tree. What is a tree, though? A **tree** is a graph with property that there is a unique path between any two of its vertices. In Figure 4.4 we show an example of a graph that is not a tree (on the left) and a tree.

Figure 4.4: A graph and a tree.

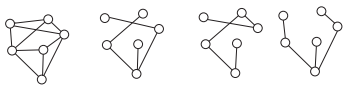


Notice that the tree has ten vertices and nine edges. We can prove by induction that a tree on n vertices has $n - 1$ edges. The inductive proof could be modeled on the proof we used for

rooted trees, but it can also be done more simply, because deleting an edge and no vertices from a tree always gives us two trees (one has to think about the definition a bit to see this). We assume inductively that each of these trees has one fewer edge than vertex, so between them they have two fewer edges than vertices, so putting one edge back in completes the inductive step.

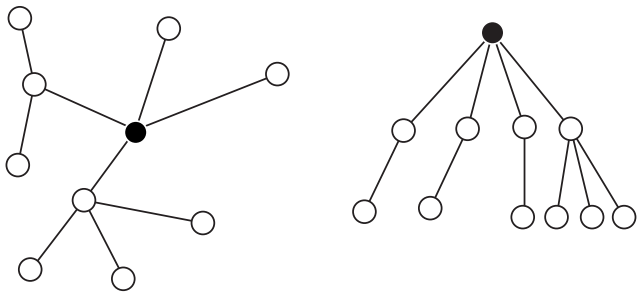
In computer science applications trees often arise in connection with graphs. For example the graph in Figure 4.5 might represent a communications network. If we want to send a message from the central vertex to each other vertex, we could choose a set of edges to send the message along. Each of the three trees in Figure 4.5 uses a minimum number of edges to send our message, so that if we have to pay for each edge we use, we pay a minimum amount to send our message. However if there is a time delay along each link (in military applications, for example, we might need to use a courier in times of radio silence), then the third tree might be preferred for sending messages from the center vertex to all other vertices. The three trees in Figure 4.5 are called spanning trees of our graph. A **spanning tree** of a graph is a tree that has the same vertex set as the graph and an edge set that is a subset of the edge set of the graph. As you might expect, there are many different criteria we might use for choosing a spanning tree of a graph.

Figure 4.5: Some spanning trees of a graph.



If it seems to you that there is not that much difference between trees and rooted trees, you are right. We can conclude from what we have already shown that if you ignore the root of a rooted tree, you get a tree. (We showed that there is a unique path between any two vertices.) In fact, given a tree, we may choose any vertex and declare it to be a root, and we have a rooted tree. We illustrate the process in Figure 4.6. How would we prove this? One guess—induction!

Figure 4.6: Choosing a vertex in a tree to be a root gives us a rooted tree.



4.3-3 Show that deleting an edge from a tree gives us two trees.

4.3-4 Show that selecting a vertex in a tree gives a rooted tree according to our definition of a rooted tree.

For Exercise 4.3-3, suppose we have a tree which we call T. Remember by definition a tree is a graph with a unique path between any pair of vertices. Choose an edge with endpoints x and y in T. Every vertex that had a path to x that did not use the edge (x, y) still has a path to x and every vertex that had a path to y that did not use the edge (x, y) still has a path to y. If a vertex v has no path to either x or y after we have removed the edge, then putting the edge from x to y back in cannot create a path from v to either x or y. Therefore every vertex has a path to either x or y after the edge is deleted, so removing the edge has given us two graphs each that have a path between each pair of their points. Those paths must be unique because they were unique before we deleted the edge. Thus we get two trees.

To do Exercise 4.3-4, we begin by noting that a single vertex is both a tree and a rooted tree according to our two definitions. Now suppose that choosing a root in a tree with $n-1$ or fewer vertices gives a tree, and let T stand for a tree on n vertices with $n > 1$. Choose a vertex y and an edge (x, y). Delete the edge. In the trees T_x and T_y that result, choose the vertices x and y to be roots, which, by our inductive hypothesis gives us rooted trees. Suppose the subtrees of the root y in the tree T_y are T'_1, T'_2, ..., T'_k. Let T' be the tree with root vertex y and subtrees T_x, T'_1, T'_2, ..., T'_k The by our recursive definition, T' is a rooted tree. Therefore, by the principle of mathematical induction, choosing a vertex to be root in any tree gives a rooted tree. Notice how little reference we actually made to the integer n. We could delete these references and simply make our inductive hypothesis for smaller trees and we would have a proof by structural induction.

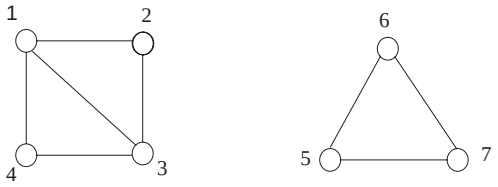
This completes a proof of the following theorem.

Theorem 4.3.3 *Any rooted tree is a tree, and choosing a vertex in a tree to call a root creates a rooted tree.*

Connected and Disconnected graphs

The graph in Figure 4.7 looks like two graphs. It is called a disconnected graph. In contrast, the two graphs in Figure 4.4 are called connected graphs.

Figure 4.7: A disconnected graph.



A graph is called connected if for each x and y in the graph, there is a walk from x to y. The relation of being connected is an equivalence relation, because the relation partitions the vertex set into disjoint sets as follows: For each x, let

$$S_x = \{y|\ x \text{ and } y \text{ are connected by a walk.}\}$$

Then the sets S_x and S_y are either identical or disjoint, and each x is in S_x. Thus the distinct sets S_x form a partition of the vertex set of the graph. The sets S_x are called the *connected*

components of the graph. A tree is connected because there is a path between any two of its vertices.

In Figure 4.7 the walks from 1 to 2 to 3 and back to 1 and from 1 to 2 to 3 to 4 and back to 1 are called cycles. A **cycle** is a walk whose first and last vertex are the same but has no other vertices or edges that are repeated.

4.3-5 Find the other cycles in Figure 4.7.

4.3-6 Can a tree have any cycles?

4.3-7 Is a connected graph with no cycles a tree?

The other cycles in Figure 4.7 are the walk from 1 to 3 to 4 and back to 1 and the walk from 5 to 6 to 7 and back to 5. Trees have no cycles and a graph is a tree if and only if it is connected and has no cycles.

Theorem 4.3.4 *A graph G is a tree if and only if it is connected and has no cycles.*

Proof: Suppose G is a tree. Then it is connected because between any two vertices there is a path. If G had a cycle, then we would use it to give two paths between any two vertices on the cycle, contradicting the definition of a tree. Therefore G has no cycles.

Suppose now that G is connected and has no cycles. Then there is a walk between any two vertices, so leaving out portions of the walk between successive repeated vertices, and continuing to do so until there are no repeated vertices will give give a path. Suppose now there are two distinct paths that start at the same vertex and end at the same vertex. Among all such pairs of paths, choose a pair with as few common vertices as possible. There must be at least two common vertices, the starting vertex s and the ending vertex t. If there were a third vertex x on both paths, then the portions of the two paths that go from s to x would two distinct paths with fewer common vertices or else would be identical, and the portions of the two paths that go from x to t would either be two distinct paths with fewer common vertices or would be identical. In any case, having such a vertex contradicts the assumption that we had a pair of distinct paths with as few common vertices as possible. Thus there is no such x. Therefore going from s to t on one path and then from t to s on the other gives a cycle, a contradiction to our hypothesis that G has no cycles. Thus the supposition of two distinct paths that start at the same and end at the same vertex is impossible, so G is a tree. ■

Problems

1. 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 4.7), that is fine, but the way in which you assign labels to the vertices is unimportant.

2. A vertex of a rooted tree is called a *leaf node* if it has no children. Draw all rooted trees on 6 vertices with four leaf nodes.

3. Draw all different trees on six vertices. Two trees are not considered different if simply by redrawing one we get the other. In other words, if you can label the vertices so that when you write down for each edge the labels of the pair of vertices they connect, you get exactly the same collections of pairs, then they are the same. In "fancier" terminology, graphs with the labelling property we just described are *isomorphic*.

4. Find a tree 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.)

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 binary tree called the right subtree of the root. Then a single vertex is a binary tree with an empty right subtree and an empty left subtree. A rooted tree with two nodes 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.

6. 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.

7. Are there any full binary trees (see Exercise 4.3-6) on an even number of vertices? Prove that what you say is correct.

8. 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 Exercise 4.3-6) and all its leaves (see Exercise 4.3-6) have the same depth. How many nodes does a complete binary tree of depth 1 have? Depth 2? Depth d? (Proof required for depth d.)

9. 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 4.3-8, what is the minimum height of *any* binary tree on n nodes? (Please prove this.)

10. A binary tree is complete if it is full and all its leaves have the same depth (see Exercise 4.3-6 and Exercise 4.3-8). A vertex that is not a leaf vertex is called an *internal* vertex. What is the relationship between the number I of internal nodes and the number L of leaf nodes in a complete binary tree. A full binary tree? (Proof please.)

11. The *internal path length* of a binary tree is the sum, taken over all internal (see Exercise 4.3-10) 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 $e = i + 2n$.

12. Show that a graph is connected if and only if it has a spanning tree.

13. Show that a graph on n vertices is a tree if and only if it is connected and has $n-1$ edges.

14. A graph with no cycles is called a *forest*. Show that if a forest has v vertices, e edges, and c connected components, then $v = e + c$.

15. Show that a graph on n vertices is a tree if and only if it has $n-1$ edges and has no cycles.

4.4 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, one of the possible ranges of values you could end up with would more than half the size of the original problem.) If you continue in this fashion, always cutting the problem size in half, you will be able to get the problem down to size one fairly quickly, and then you will know what the number is. Of course it would be easier to cut the problem 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.

4.4-1 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, get a recurrence of for $T(n)$.

For Exercise 4.4-1 we get:

$$T(n) = \begin{cases} T(n/2) + 1 & \text{if } n \geq 2 \\ 1 & \text{if } n = 1 \end{cases} \tag{4.12}$$

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

$$T(n) \leq \begin{cases} T(\lceil n/2 \rceil) + C_1 & \text{if } n \geq 2 \\ C_2 & \text{if } n = 1, \end{cases} \tag{4.13}$$

where C_1 and C_2 are constants.

It turns out that the solution to (4.12) and (4.13) are roughly the same, in a sense that will hopefully become clear later in the notes. This is almost always the case; we will come back to

this issue. 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:

$$T(n) = \begin{cases} 2T(n/2) + n & \text{if } n > 1 \\ 1 & \text{if } n = 1 \end{cases} \tag{4.14}$$

Recurrences such as this one can be understood via the idea of a recursion tree, which we introduce below. This concept will allow 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.

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

$$T(n) = 2T(n/2) + n \tag{4.15}$$

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

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

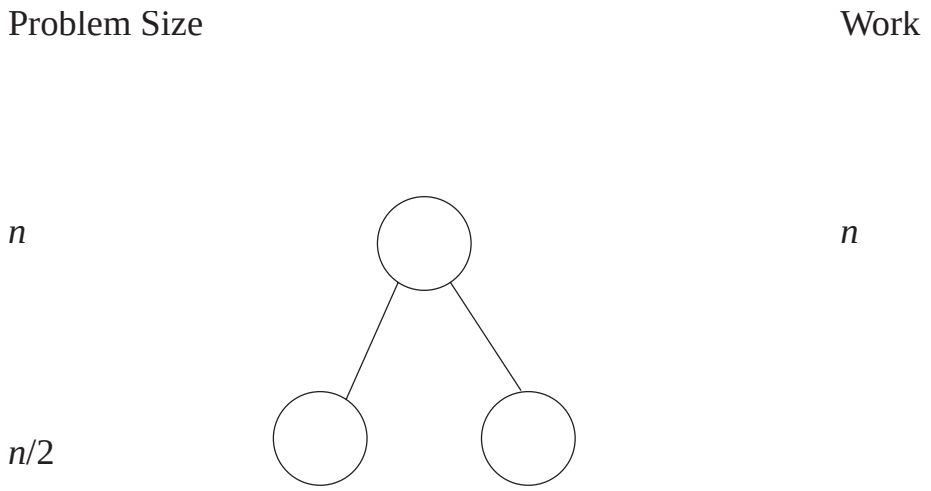
as "in order to solve a problem of size n we must solve 1 problems of size $n/4$ and do n^2 units of additional work."

We can also interpret the recurrence

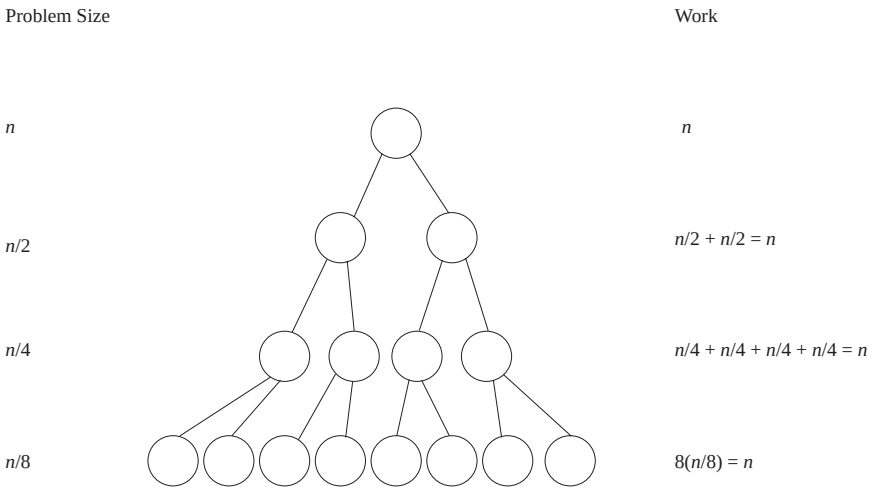
$$T(n) = 3T(n-1) + n$$

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.

We will now draw the beginning of the recursion diagram for (4.15). For now, assume n is a power of 2. A recursion tree diagram has three parts. 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. So to begin the recursion tree for (4.15), we take a problem of size n, split it into 2 problems of size $n/2$ and note on the right that we do n units of work. The fact that we break the problem into 2 problems is denoted by drawing two children of the root node (level 0). This gives us the following picture.



We then continue to draw the tree in this manner. Adding a few more levels, we have:



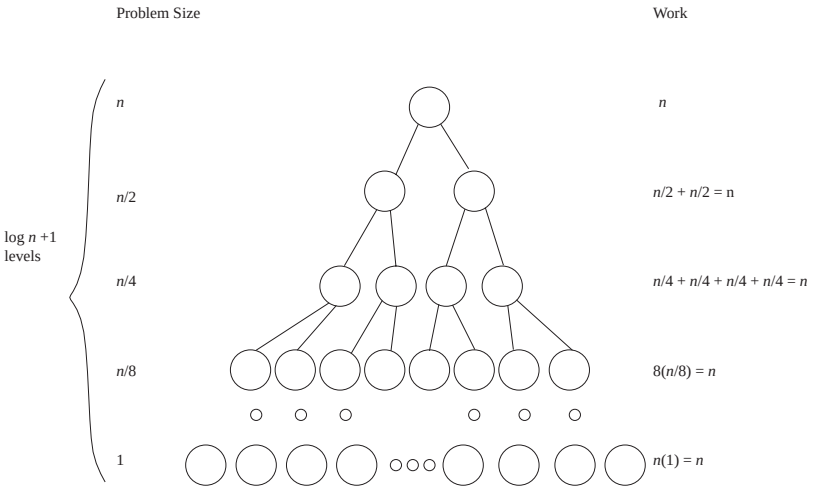
At level zero (the top level), n units of work are done. We see that at each succeeding level, we halve the problem size and double the number of subproblems. We also see that at level 1, each of the two subproblems requires $n/2$ units of additional work, and so a total of n units of additional work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

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 i, three things

- number of subproblems,
- size of each subproblem,
- total work done.

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.[1] We can thus visualize the whole tree as follows:



The bottom level is different from the other levels. In the other levels, the work is described by the recursive part of the recurrence, which in this case is $T(n) = 2T(n/2) + n$. 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 $T(1)$. For this particular recurrence, and for many others, it turns out that if you compute the amount of work on the bottom level as if you were computing the amount of additional work required after you split a problem of size one into 2 problems (which, of course, makes no sense) it will be the same value as if you compute it via the base case. We emphasize that the correct value always comes from the base case, it is just a useful coincidence that it sometimes also comes from the recursive part of the recurrence.

The important thing is that we now know exactly how many levels there are, and 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 (4.15) 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 (4.14) is $T(n) = n(\log n + 1)$.

[1] 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.

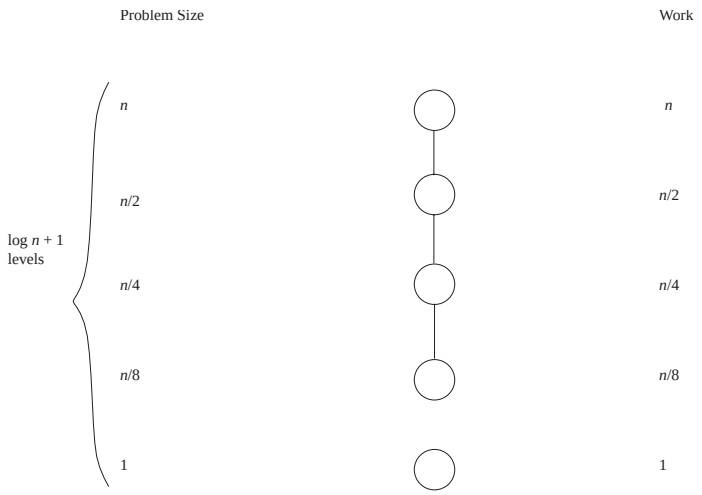
More generally, we can consider a recurrence that it identical to (4.14), except that $T(1) = a$, for some constant a. In this case, $T(n) = an + n \log n$, because an units of work are done at level 1 and n additional units of work are done at each of the remaining $\log n$ levels. It is still true 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. Since one unit of time will vary from computer to computer, and since some kinds of work might take longer than other kinds, it is the big-θ behavior of $T(n)$ that is really relevant. Thus although recursion trees can give us the exact solutions (such as $T(n) = an + n \log n$ above) to recurrences, we will often just analyze 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.

Let's look at one more recurrence.

$$T(n) = \begin{cases} T(n/2) + n & \text{if } n > 1 \\ 1 & \text{if } n = 1 \end{cases} \tag{4.16}$$

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.

Let's draw the tree for this problem:



We see here that the problem sizes are the same as in the previous tree. The rest, however, is different. The number of subproblems does not double, rather 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 4.2.4, the value of a geometric series in which the large 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 (4.16); 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 $T(n) = \Theta(n)$. We have not actually *found* a solution. 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. What is often important to us in applications is not the exact form of the solution, but a big-O upper bound, or, better, a Big-Θ bound on the solution.

4.4-2 Solve 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.

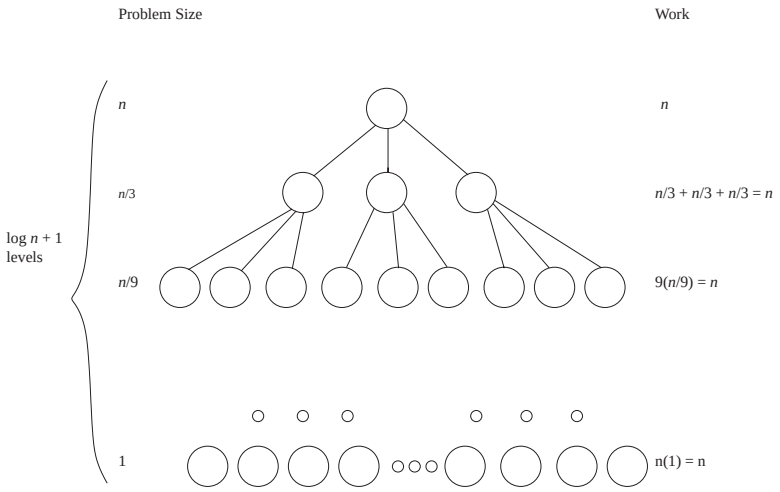
4.4-3 Solve the recurrence

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

using a recursion tree. Assume that n is a power of 2.

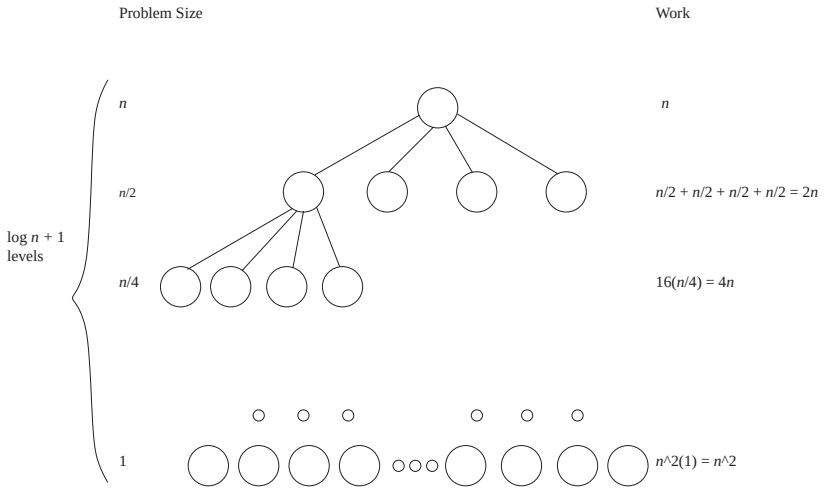
4.4-4 Can you give a general big-O formula for solutions to recurrences of the form $T(n) = aT(n/2) + n$ when n is a power of 2? You may have different answers for different values of a.

The recurrence in Exercise 4.4-2 is similar to the mergesort recurrence. The difference is that at each step we divide into 3 problems of size $n/3$. Thus we get the following picture:



The only difference is that the number of levels, instead of being $\log_2 n + 1$ is now $\log_3 n + 1$, so the total work is still $\Theta(n \log n)$ units.

Now let's look at the recursion tree for Exercise 4.4-3. Now, we have 4 children of size $n/2$, and we get the following:



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. Summing over the levels, we get

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

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

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

Three Different Behaviors

Now let's compare the trees 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 sizes of each subproblem is $1/2$ the size of of the parent problem. To see the differences, in the first case, on every level, there is the same amount of work. In the second case, the amount of work decreases, with the most work being at level 0. In fact, it decreases geometrically, so by Theorem 4.2.4 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 4.2.4 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 4.4-4, a general Big-O form for $T(n) = aT(n/2) + n$, we can conclude the following (and could replace the Big-O by a Big-Θ):

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

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

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

The sum is a geometric series, so the sum will be at most a constant times the largest term (see Lemma 4.2.3). Since $a > 2$, the largest term in this case is clearly the last one, namely $n(a/2)^{\log_2 n}$, and applying rules of exponents and logarithms, we get that n times the largest term is

$$n\left(\frac{a}{2}\right)^{\log_2 n} = \frac{n \cdot a^{\log_2 n}}{2^{\log_2 n}} = \frac{n \cdot 2^{\log_2 a \log_2 n}}{2^{\log_2 n}} = \frac{n \cdot n^{\log_2 a}}{n} = n^{\log_2 a}$$

Thus the total work done is $\Theta(n^{\log_2 a})$.

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 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 4.16.

4. 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 evaluate the following recurrences (do not use the master theorem). For all of these, assume that $T(1) = 1$.

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

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

   (c) $T(n) = T(\lfloor\sqrt{n}\rfloor) + 1$

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

5. If $S(n) = aS(n-1) + g(n)$ and $g(n) < c^n$ with $0 \leq c < a$, how fast does $S(n)$ grow (in big-O terms)? $S(n) = a^i S(n-1) + \sum_{j=0}^{i-1} a^j g(n-j) = a^n S(0) + \sum_{j=0}^{n-1} a^j g(n-j) < a^n S(0) + \sum_{j=0}^{n-1} a^j c^{n-j} = a^n S(0) + c^n \sum_{j=0}^{n-1} (\frac{a}{c})^j = a^n S(0) + O((\frac{a}{c})^n) = O(a^n)$

4.5 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 $a < 2$, $a = 2$, and $a > 2$. 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 4.5.1 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 4.5.1 *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 $\log_b a < c$, $T(n) = O(n^c)$,*
2. *if $\log_b a = c$, $T(n) = O(n^c \log n)$,*
3. *if $\log_b a > c$, $T(n) = O(n^{\log_b a})$.*

Proof: In this proof, we will set $d = 1$, so that the bottom level of the tree is equally well computed by the recursive step as by the base case. The proof easily extends for the case when $d \neq 1$.

Let's think about the recursion tree for this recurrence. There will be $\log_b n$ 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 of them, the total number of units of work on level i is

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

Recall from above 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. These three cases depend on whether $(\frac{a}{b^c})$ is 1, less than 1, or greater than 1. Now observe that

$$\begin{aligned} & (\tfrac{a}{b^c}) = 1 \\ \Leftrightarrow \quad & a = b^c \\ \Leftrightarrow \quad & \log_b a = c \log_b b \\ \Leftrightarrow \quad & \log_b a = c \end{aligned}$$

Thus we see where our three cases come from. Now we proceed to show the bound on $T(n)$ in the different cases. In the following paragraphs, we will use the easily proved facts that for any x, y and z, each greater than 1, $x^{\log_y z} = z^{\log_y x}$ and that $\log_x y = \Theta(\log_2 y)$.

In case 1, (part 1 in the statement of the theorem) we have that

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

Since this is n^c times a geometric series with a ratio of less than 1 Theorem 4.2.4 tells us that

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

4.5-1 Prove Case 2 of the Master Theorem.

4.5-2 Prove Case 3 of the Master Theorem.

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

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

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 Lemma 4.2.3,the sum is $\Theta\left(n^c \left(\frac{a}{b^c}\right)^{\log_b n}\right)$. But

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

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

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 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.

4.5-3 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?

4.5-4 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 2? (Note that $\lceil n/b \rceil$ denotes the *ceiling* of n/b, the smallest integer greater than or equal to n/b.)

To compute $t(7)$ in Exercise 4.5-3 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 $1 < x/2^i$. We can now prove by induction on i that $t(x)$ is uniquely determined by the recurrence relation.

In Exercise 4.5-4 there is one and only one solution. Why? Clearly $T(1)$ is determined by the recurrence. Now assume inductively that $n > 1$ and that $T(n)$ 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 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

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 4.5.2 *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 smallest integer power of b greater than or equal to x. Then $t(x) = \Theta(T(m(x)))$

Proof: If we analyze recursion trees for the two recurrences, we can see that they 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

We have also pointed out that a more realistic master would have 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, one of size 50 and one of size 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. The condition that $b > 2$ can be replaced by $b > 1$, but the the base case for the recurrence will depend on b. Since we may remove either floors or ceilings, that means that we may deal with recurrences of the form $T(n) = a'T(\lceil n/b \rceil) + (a - a')T(\lfloor n/b \rfloor) + n^c$

Theorem 4.5.3 *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 recursion trees for the two recurrences. It is straightforward (though dealing with the notation is difficult) to show that for a given value of n, the recursion tree for computing $T(n)$ has at most two more levels than the recursion tree 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 tree 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 4.5.2 and Theorem 4.5.3 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.

A version of the Master Theorem for more general recurrences.

We showed that in our 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 master theorem to a much wider variety of functions.

Theorem 4.5.4 *Theorems 4.5.3 and 4.5.2 apply to recurrences in which the x^c term is replaced by a function f in which $f(x) = \Theta(x^c)$.*

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 one of the solutions above and below with a multiple of the other solution. The details are similar to those in the original proofs. ■

4.5-5 If $f(x) = x\sqrt{x+1}$, what can you say about the Big-Θ behavior of solutions to

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

and the solutions to

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

where n is restricted to be a power of 3?

Since $f(x) = x\sqrt{x+1} \geq x\sqrt{x} = x^{3/2}$, we have that $x^{3/2} = O(f(x))$. Since

$$\sqrt{x+1} \leq \sqrt{x+x} = \sqrt{2x} = \sqrt{2}\sqrt{x}$$

for $x > 1$, we have $f(x) = x\sqrt{x+1} \leq \sqrt{2}x\sqrt{x} = \sqrt{2}x^{3/2} = O(x^{3/2})$. Thus the big-$\Theta$ behavior of the solutions to the two recurrences will be the same.

Extending the Master Theorem

As Exercise 4.5-5 suggests, Theorem 4.5.4 opens up a whole range of interesting recurrences to analyze. 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 4.5-5.

We now state a second version of the Master Theorem. A still stronger version of the theorem may be found in CLR, but the version here captures much of the interesting behavior of recurrences that arise from the analysis of algorithms.

Theorem 4.5.5 *Let a and b be positive real numbers with $a \geq 1$ and $b > 1$. Let $T(n)$ be defined by*

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

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* $T(n) = \Theta(n^{\log_b a})$

The same results apply with ceilings replaced by floors.

Proof: Since we have assumed that $f(n) = \Theta(n^c)$, we know by Theorem 4.5.4 that we may restrict our domain to exact powers of b. We mimic the original proof of the Master theorem, substituting the appropriate $\Theta(n^c)$ for $f(n)$ in computing the work done at each level. But this means 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. ■

4.5-6 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 4.5-5 $x\sqrt{x+1} = \Theta(x^{3/2})$. 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, $T(n) = \Theta(n^{\log_2 3})$.

Appendix: Proofs of Theorems

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

Theorem 4.5.6 *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 smallest integer power of* b *greater 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), we see that

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

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 will be a^e times $T(x/b^e)$ plus x^c or n^c times a geometric series $G(x) = \sum_{i=0}^{e} \Big(\frac{a}{b^c}\Big)^i$. 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 $\Theta\Big(\frac{a}{b^c}\Big)^e$, 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 smallest integer power of b greater than or equal to x, then $0 \leq e(m) - e(x) < 1$. Then for any real number r we have $r^0 \leq r^{e(m)-e(x)} < r$, or $r^{e(x)} \leq r^{e(m)} \leq r \cdot r^{e(x)}$. Thus we have $r^{e(x)} = \Theta(r^{e(m)})$ for every real number r, including $r = a$ and $r = \frac{a}{b^c}$. Finally, $x^c \leq m^c \leq b^c x^c$, and so $x^c = \Theta(m^c)$; Therfore, every term of $t(x)$ is Θ of the corresponding term of $T(m)$. Further, there are only a finite 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 4.5.7 *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*

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

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

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

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 it is straightforward to prove by induction 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

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

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

$$\begin{aligned} \sum_{i=0}^{j-1} a^i \Big(\frac{n}{b^i}\Big)^c &\le \sum_{i=0}^{j-1} a^i n_i^c \\ &\le n^c + \sum_{i=1}^{j-3} a^i \Big(\frac{n}{b^{i-1}}\Big)^c + a^{j-2}(b^2)^c + a^{j-1}b^c + a^j 1^c \ , \end{aligned}$$

or

$$\begin{aligned} \sum_{i=0}^{j-1} a^i \Big(\frac{n}{b^i}\Big)^c &\le \sum_{i=0}^{j-1} a^i n_i^c \\ &\le n^c + b\sum_{i=1}^{j-3} a^i \Big(\frac{n}{b^i}\Big)^c + a^{j-2}\left(\frac{b^j}{b^{j-2}}\right)^c + a^{j-1}\left(\frac{b^j}{b^{j-1}}\right)^c + a^j\left(\frac{b^j}{b^j}\right)^c \ . \end{aligned}$$

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 is 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 $n < b^j < nb^3$, so $b^j = \Theta(n^c)$, 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 $t(n)$. ■

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

$$T(x) = \begin{cases} 7xT(x-1) + 1 & \text{if } x \ge 1 \\ 1 & \text{if } 0 \le x < 1 \end{cases} .$$

3. Show that for each real number $x \geq 1$ there is one and only one value of $T(x)$ given by the recurrence
$$T(x) = \begin{cases} 3xT(x/2) + x^2 & \text{if } x \geq 2 \\ 1 & \text{if } 1 \leq x < 2 \end{cases}.$$

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

5. Give a big-Θ bound on the solution to the recurrence
$$T(n) = \begin{cases} 3T(n/2) + \sqrt{n+3} & \text{if } n > 1 \\ d & \text{if } n = 1. \end{cases}$$

6. Give a big-Θ bound on the solution to the recurrence
$$T(n) = \begin{cases} 3T(n/2) + \sqrt{n^3+3} & \text{if } n > 1 \\ d & \text{if } n = 1. \end{cases}$$

7. Give a big-Θ bound on the solution to the recurrence
$$T(n) = \begin{cases} 3T(n/2) + \sqrt{n^4+3} & \text{if } n > 1 \\ d & \text{if } n = 1. \end{cases}$$

8. Give a big-Θ bound on the solution to the recurrence
$$T(n) = \begin{cases} 2T(n/2) + \sqrt{n^2+3} & \text{if } n > 1 \\ d & \text{if } n = 1. \end{cases}$$

4.6 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

$$T(n) \leq 2T(n/2) + cn. \tag{4.17}$$

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 situations 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 inequalties or recurrence equations.

In Recurrence 4.17 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 4.17. (For example for any $c' < c$, the unique solution to

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

satisfies Inequality 4.17 for any constant k.) This idea of infinitely many solutions for a recurrence inequality is analogous to the idea that $x - 3 = 0$ has one solution, but $x - 3 \leq 0$ has infinitely many solutions. Later in this section we shall see how to show that all the solutions to Recurrence 4.17 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.

4.6-1 Carefully prove by induction that for any function T defined on the nonnegative 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 recurrences 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 4.6-1. Here we wish to show that $T(n) = O(n \log n)$. Using the definition of Big-O, we can see that we wish to show that $T(n) \leq kn \log n$ 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 $T(n) \leq 2T(n/2) + cn$ 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 $n > n_0$, $T(n) \leq kn \log n$. We cannot expect to have the inequality $T(n) \leq kn \log n$ hold for $n = 1$, because $\log 1 = 0$. To have $T(2) \leq k \cdot 2 \log 2 = k \cdot 2$, we must choose $k \geq \frac{T(2)}{2}$. 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 $T(m) \leq km \log m$. Then $n/2 < n$, and since n is a power of 2 greater than 2, we have that $n/2 \geq 2$, so $n/2 \log n/2 \geq 2$. By the inductive hypothesis, $T(n/2) \leq k(n/2) \log n/2$. But then

$$\begin{aligned} T(n) \leq 2T(n/2) + cn &\leq 2k\frac{n}{2}\log\frac{n}{2} + cn && (4.18)\\ &= kn \log\frac{n}{2} + cn && (4.19)\\ &= kn \log n - kn \log 2 + cn && (4.20)\\ &= kn \log n - kn + cn. && (4.21) \end{aligned}$$

This shows that we need to make another assumption about k, namely that $-kn + cn \leq 0$, or $k \geq c$. 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 1), $T(n) \leq kn \log n$, 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 you to be a proof, we will summarize our observations below 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 be prepared to read discussions like the one above as proofs.

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

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

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 $n > 4$, so that $n/2$ is an m with $2 \leq m < n$. Thus

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

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 two things to note about this proof. First without the preceding discussion, the choice of k seems arbitrary. Second, 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.

Further Wrinkles in Induction Proofs

4.6-2 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 $T(n) = O(n)$.

4.6-3 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 4.6-2 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

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

In this case, as long as $c - \frac{2k}{3} \leq 0$, the inductive step of the proof will be correct. Again, the elements of an inductive proof are in the preceeding discussion. However we will now present something that looks more like an inductive proof.

We choose k to be the maximum of $T(1)$ and $3c/2$. 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 4.6-3. 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 k so that the base case holds, we can bound $T(n)$ inductively by assuming that $T(n) \leq km^2$ 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 problem 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

$$\begin{aligned} T(n) &\leq 4T(n/2) + cn \\ &\leq 4\left(k_1\left(\frac{n}{2}\right)^2 - k_2\left(\frac{n}{2}\right)\right) + cn \\ &= 4\left(\frac{k_1n^2}{4} - k_2\left(\frac{n}{2}\right)\right) + cn \\ &= k_1n^2 - 2k_2n + cn \\ &= k_1n^2 - k_2n + (c - k_2)n. \end{aligned}$$

Now we have to make $(c - k_2)n \leq 0$ for the last line to be at most $k_1n^2 - k_2n$, and so we just choose $k_2 \geq c$ (and greater than whatever we need in order to make a base case work). Since $T(n) \leq k_1n^2 - k_2n$ for some constants k_1 and k_2, then $T(n) = O(n^2)$.

At first glance, this approch seems paradoxical: why is it easier to prove a stronger statement than it is to prove a weaker one? This phenomenom happens often in mathematics: a stronger

statement is often easier to prove than a weaker one. It happens particularly often when using induction. 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 these values, $p(1) \ldots p(n-1)$ 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.

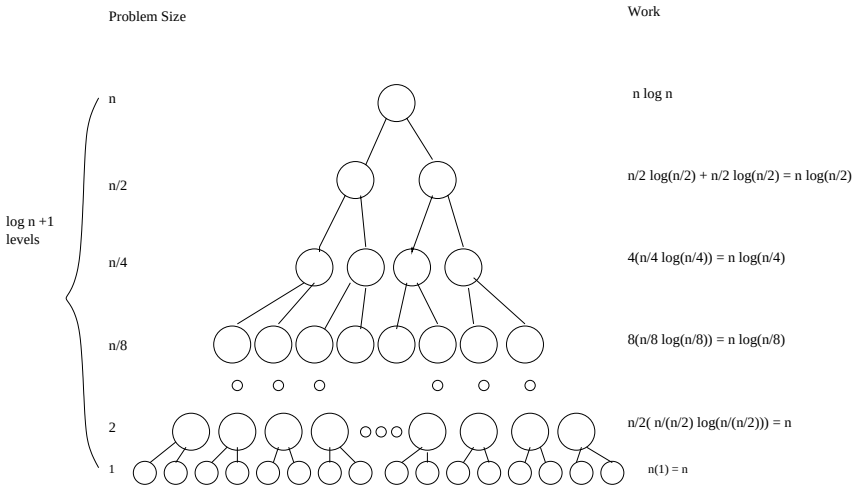
Dealing with functions other than n^c

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 has 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 as follows.



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 $i \geq 2$, and for level 1 it is n, the number of subproblems, times 1. Thus if we sum the work per level we get

$$\begin{aligned} \sum_{i=0}^{\log_2 n - 1} n \log\left(\frac{n}{2^i}\right) + n &= n \sum_{i=0}^{\log_2 n - 1} \log\left(\frac{n}{2^i}\right) + n \\ &= n \sum_{i=0}^{\log_2 n - 1} (\log n - \log 2^i) + O(n) \end{aligned}$$

$$\begin{aligned} &= n\left(\sum_{i=0}^{\log_2 n-1} \log n - \sum_{i=0}^{\log_2 n-1} i\right) + n \\ &= n\left((\log_2 n)(\log_2 n) - \frac{(\log_2 n)(\log_2 n - 1)}{2}\right) + n \\ &= O(n \log^2 n)\ . \end{aligned}$$

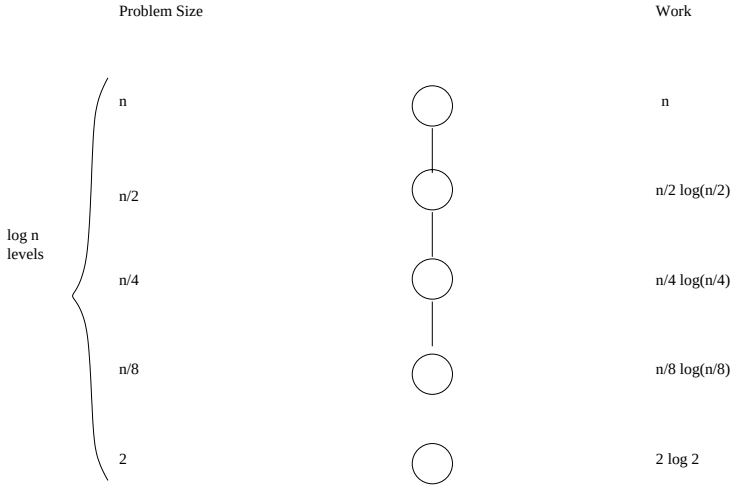
A bit of mental arithmetic in the second last line of our equations shows that the $\log^2 n$ will not cancel out, so our solution is in fact $\Theta(n \log^2 n)$.

4.6-4 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



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

$$\begin{aligned} \sum_{i=0}^{\log_2 n-1} \frac{n}{2^i} \log\left(\frac{n}{2^i}\right) + 1 &= n\left(\sum_{i=0}^{\log_2 n-1} \frac{1}{2^i}(\log n - \log 2^i)\right) + 1 \\ &= n\left(\sum_{i=0}^{\log_2 n-1} \left(\frac{1}{2}\right)^i (\log(n) - i)\right) + 1 \\ &\le n\left(\log n \sum_{i=0}^{\log_2 n-1} \left(\frac{1}{2}\right)^i\right) + 1 \\ &\le n(\log n)(2) + 1 \\ &= O(n \log n). \end{aligned}$$

In fact we have $T(n) = \Theta(n \log n)$, because 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 $n \log n$.

Removing Ceilings and Using Powers of b.

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 these two theorems 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

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

4.6-5 Show that if $f(x) = x^2 \log x$, then $f(bx) = \Theta(f(x))$.

4.6-6 If $f(x) = 3^x$ and $b = 2$, is $f(bx) = \Theta(f(x))$? Is $f(b(x)) = O(f(x))$?

Notice that if $f(x) = x^2 \log x$, then

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

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 $f(bx) = \Theta(x)$.

Theorem 4.6.1 *Theorems 4.5.2 and 4.5.3 apply to recurrences in which the x^c term is replaced by an increasing function 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 $f(x) = \Theta(x)$ applied to an increasing function gives us enough information to again bound one the to one kind of recurrence above and below with a multiple of the other solution. The details are similar to those in the original proofs so we omit them. ■

In fact there are versions of Theorems 4.5.2 and 4.5.3 for recurrence inequalities also. The proofs involve a similar analysis of iterated recurrences or recursion trees, and so we omit them.

Theorem 4.6.2 *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 $T(x)$ to the recurrence

$$T(n) \leq \begin{cases} aT(n/b) + f(n) & \text{if } n > 10 \\ d & \text{if } n = 1 \end{cases},$$

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

Theorem 4.6.3 *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 $t(x) = O(h(x))$.

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 yougot in part (a), prove, by induction, that your answer is correct.

2. Is the big-O upper bound in the previous exercise 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 upper bound?

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

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

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 the previous exercise by 4? What if you replace the $\log_3$ by $\log_b$? (Use recursion trees.)

7. 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 $O(f(x))$.

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

9. Find the best big-O upper bound you can to any solution to the recurrence defined on nonnegative integer powers of two by

$$T(n) \leq 2T(\lceil n/2 \rceil + 1) + cn.$$

Prove, by induction, that your answer is correct.

4.7 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, it makes sense to talk about one being smaller than another. 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 $\{3, 1, 8, 6, 4, 11, 7\}$, 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 $i = \lceil n/2 \rceil$.

4.7-1 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? Does this approach extend to finding the ith smallest? What is the running time?

4.7-2 Give the fastest algorithm you can to find the median ($i = \lceil n/2 \rceil$).

In Exercise 4.7-1, 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 $O(n)$ time. However, if we extend this to finding the ith smallest, the algorithm will take $O(in)$ time. Thus for finding the median, it 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. [2]

All these approaches, when applied to the median, take at least some multiple of $(n \log n)$ units of time. [3] As you know, the best sorting algorithms take $O(n \log n)$ time also. As you may not know, there is actually a sense in which 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 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;

```
Select(A, i, n)
(selects the ith smallest element in set A, where n = |A|)
if (n = 1)
     return the one item in A
```

[2] 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.

[3] 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.

```
else
     p = MagicMedian(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 (i ≤ |L|)
          Return Select(L, i, |L|)
     else
          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 which of these two sets the ith smallest element in A will be in. 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. [4]

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. We can express the running time by the following recurrence:

$$T(n) \leq T(n/2) + cn \tag{4.22}$$

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. First, let's consider what recurrences have only 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 solution $T(n) = O(n)$. 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.) Interpreting this as an abstract algorithm, and letting $b' = 1/b$, this says that as long as we can solve a problem of size n by first solving (recursively) a problem of size $b'n$, for any $b' < 1$, and then 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. 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 we will be able to solve the selection problem in linear time.

Now suppose instead of a MagicMedian box, we have a much weaker Magic Box, one which only guarantees that it will return some number in the middle half. 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. We will call this box a MagicMiddle box, and can use it in the following algorithm:

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

```
Select1(A,i,n)
(selects the ith smallest element in set A, where n = |A| )
if (n = 1)
     return the one item in A
else
     p = MagicMiddle(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 (i ≤ |L|)
          Return Select1(L, i, |L|)
     else
          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 $|L|$.

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 seems like a much simpler problem, and in fact it is. Thus our knowledge of which recurrences solve to $O(n)$ led us towards a more plausible algorithm.

Unfortunately, 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 select a subset of the numbers and then *recursively* find the median of that subset.

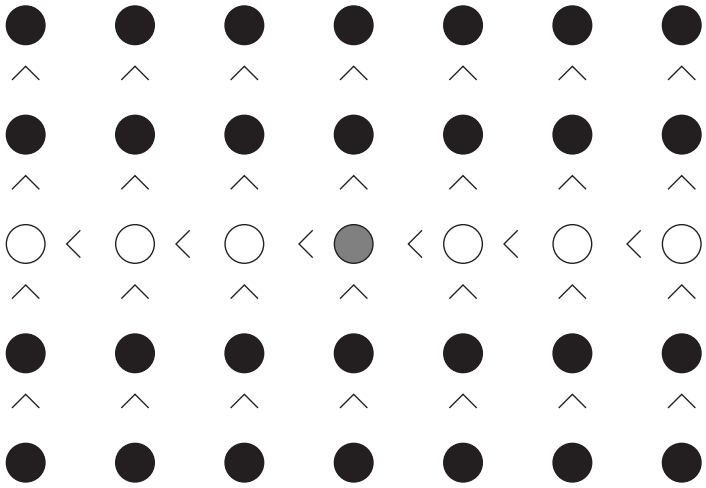
More precisely consider the following algorithm (in which we assume that $|A|$ is a multiple of 5.)

```
MagicMiddle(A)
Let n = |A|
if (n < 60)
     return the median of A
else
     Break A into k = n/5 groups of size 5, G_1, ..., G_k
     for i = 1 to k
          find m_i, the median of G_i
     Let M = {m_1, ..., m_k}
     return Select1 (M, ⌈k/2⌉, k)
```

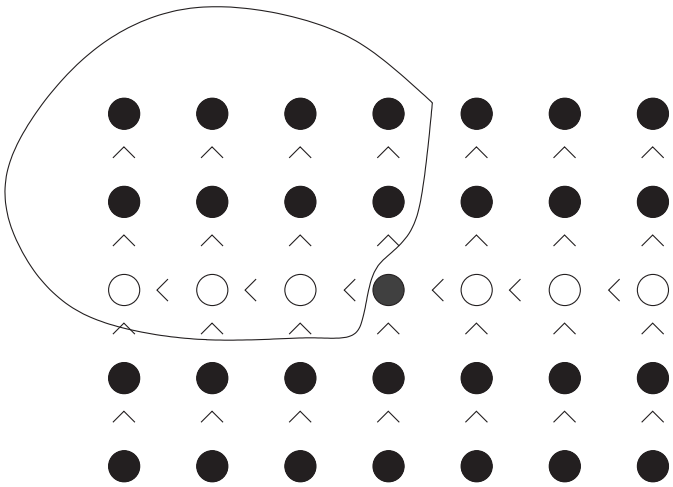
In this algorithm, we break A into $n/5$ sets of size 5, and then find the median of each set. We then (using Select1 recursively) find the median of medians and return this as our p.

Lemma 4.7.1 *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 following picture:



In this picture, 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. Thus we can circle a set of elements that is guaranteed to be smaller than the median of medians.



In one fewer than half the columns, we have circled 3 elements and in one column we have circled 2 elements. Therefore, we have circled at least[5]

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

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

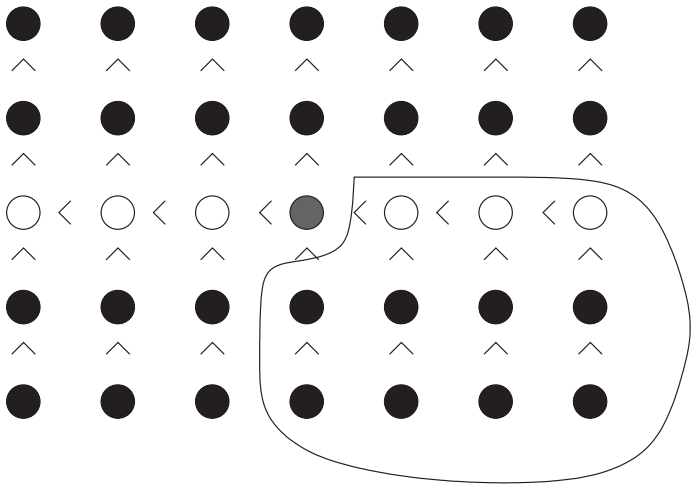
[5]We say "at least" because our argument applies exactly when n is even, but underestimates the number of circled elements when n is odd.

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. [6] 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 $|H| \leq 3n/4$.

We can make the same argument about larger elements. A set of elements that is guaranteed to be larger than the median of medians is circled in the picture below:



By the same argument as above, 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 $n < 60$, and sorting the set to find the element in position i if $n < 60$. Since 60 is a constant, this takes a constant amount of time.

4.7-3 Let $T(n)$ be the running time of Select1 on n items. What is the running time of Magic Middle?

4.7-4 What is a recurrence for the running time of Select1?

4.7-5 Can you prove by induction that each solution to the recurrence for Select1 is $O(n)$?

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 set. There are $n/5$ sets and we spend $O(1)$ time per set, so the total time is $O(n)$. Next we recursively call Select1 to find the median of medians;

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

this takes $T(n/5)$ time. Finally, we do the partition of A into those elements less than or equal to the "magic middle" and those that are not. This too takes $O(n)$ time. Thus the total running time is $T(n/5) + O(n)$.

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 to cover sets of size less than 60, we get the following recurrence for the running time of Select1:

$$T(n) \leq \begin{cases} T(3n/4) + T(n/5) + O(n) & \text{if } n \geq 60 \\ O(1) & \text{if } n < 60 \end{cases} \tag{4.23}$$

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

This kind of recurrence is actually an instance of a more general class which we will now explore.

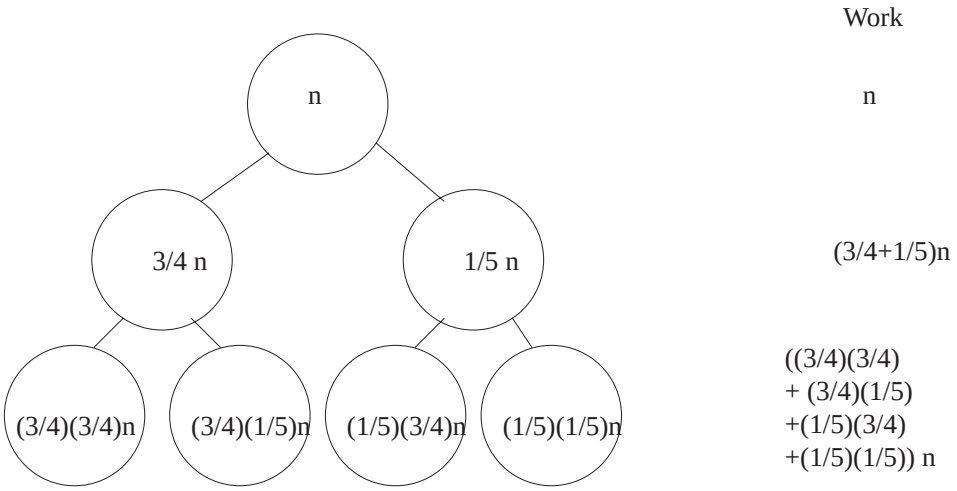
4.7-6 We already know that every solution of $T(n) = T(n/2) + O(n)$ satisfies $T(n) = O(n)$. Use the master theorem to find Big-O bounds to all solutions of $T(n) = T(cn) + O(n)$ for any constant $c < 1$.

4.7-7 Use the master theorem to find Big-O bounds to all solutions of $T(n) = 2T(cn) + O(n)$ for any constant $c < 1/2$.

4.7-8 Suppose you have a recurrence of the form $T(n) = T(an) + T(bn) + O(n)$ for some constants a and b. What conditions on a and b guarantee that all solutions to this recurrence all have $T(n) = O(n)$?

Using the master theorem, for Exercise 4.7-6, we get $T(n) = O(n)$, since $\log_{1/c} 1 < 1$. We also get $O(n)$ for Exercise 4.7-7, since $\log_{1/c} 2 < 1$ for $c < 1/2$. You might now guess that as long as $a + b < 1$, the recurrence $T(n) = T(an) + T(bn) + O(n)$ has solution $O(n)$. We will now see why this is the case.

First let's return to the recurrence we had, $T(n) = T(3/4n) + T(n/5) + O(n)$, and let's try to draw a recursion tree. This doesn't quite fit our model, as the two subproblems are unequal, but we will try to draw it 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, $(3/4 + 1/5)^2 n$ work. At level 3 we have, by the binomial theorem, $(3/4 + 1/5)^3 n$ work. And thus 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, we get that 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 has equal size, the "bottom" of the tree is more complicated. Different branches of the tree will reach 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 $\log_{20/19} n$ levels, and in fact, the upper bound 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 $T(n) = T(an)+T(bn)+O(n)$, we can study the simpler recurrence $T(n) = T((a+b)n)+O(n)$ instead. This simplifies things 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 size of the two recursive calls $3/4n$ and $n/5$ summed to less than 1. As long as that is the case for an algorithm, the algorithm will work in $O(n)$ time.

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 $T(n) \leq T(3n/4) + T(n/5) + cn$ for $n \geq 40$ rather than $n \geq 60$. (You will need to analyze the case where $\lceil n/5 \rceil$ is even and the case where it is odd separately.)

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 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 *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.

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.

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?

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?

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 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 two keys have hashed to the same location? We say two keys *collide* if the 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

[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.

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.

Theorem 5.1.1 *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 divide 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 the 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 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 5.1.1 which tells us to subtract this probability from 1 to get the probability of collision.

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

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

[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.

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 5.1.1 the probability of a collision is one minus the probability that all the keys hash to different locations. 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 5.1.1 as a theorem to emphasize the importance of this technique.

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

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 5.1.2 *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

$$p = \frac{1}{|S|} \ . \tag{5.1}$$

E is a subset of S with E elements and therefore

$$P(E) = \sum_{x_i \in E} p(x_i) = |E|p \ . \tag{5.2}$$

Combining equations 5.1 and 5.2 gives that $P(E) = |E|p = E(1/|S|) = |E|/|S|$. ■

5.1-4 What is the probability of an odd number of heads in three tosses of a coin? Use Theorem 5.1.2.

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 5.1.1 we see that the probability must be 1/2.

A word of caution is appropriate here. Theorem 5.1.2 applies only to probabilities that come from the equiprobable weighting function. The next example shows that it does not apply in general.

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 5.1.2.

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.

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 5.1.2 unless we are using the uniform probability measure.

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

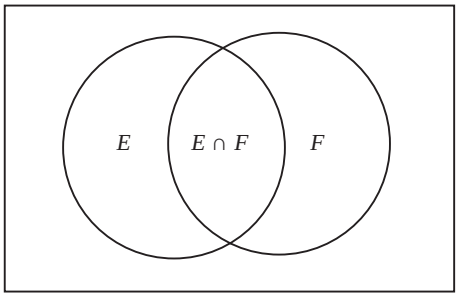
The probability of a union of events

5.2-1 If you roll two dice, what is the probability of an even sum or a sum of 8 or more?

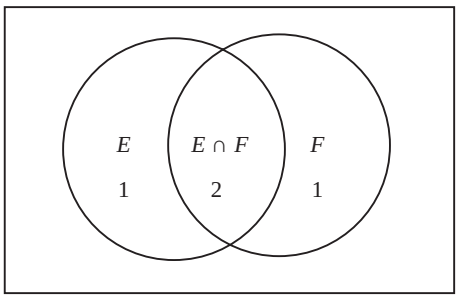
5.2-2 In Exercise 5.2-1, let E be the event "even sum" and let F be the even "8 or more" We found the probability of the union of the events E and F. Why isn't it the case that $P(E \cup F) = P(E) + P(F)$? What weights appear twice in the sum $P(E) + P(F)$? Find a formula for $P(E \cup F)$ in terms of 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?

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. In a Venn diagram, the rectangle represents the sample space, and the circles represent the events.



If we were to shade both E and FV, we would wind up shading the region $E \cap F$ twice. 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 weight of each element of $E \cap F$ twice. Thus to get a sum that includes the 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

$$P(E \cup F) = P(E) + P(F) - P(E \cap F) \tag{5.3}$$

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

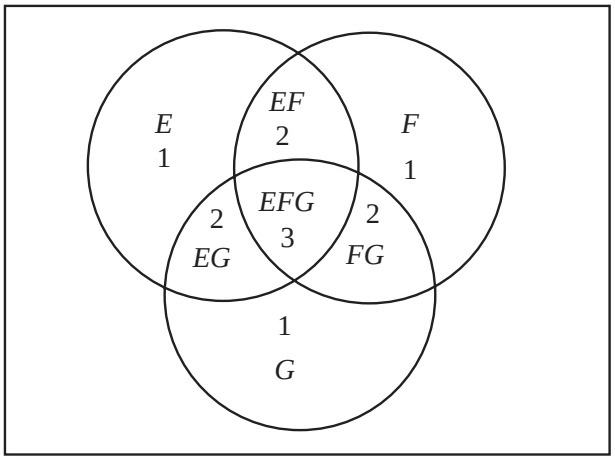
$$\frac{1}{36} + \frac{2}{36} + \frac{3}{36} + \frac{4}{36} + \frac{5}{36} = \frac{15}{36}.$$

The probability of an even sum of 8 or more is 9/36 from a similar sum, so the probability of a sum that is even or is 8 or more is

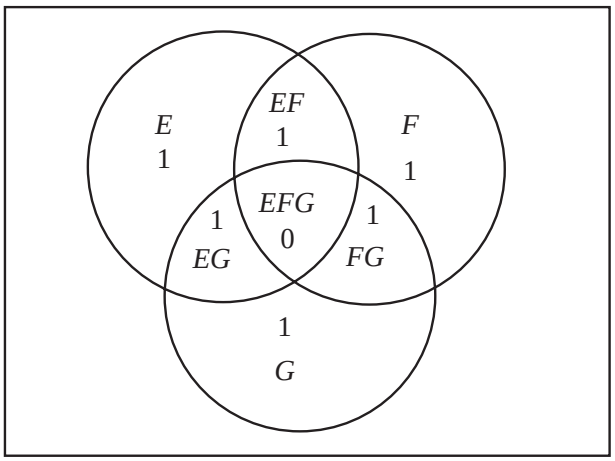
$$\frac{1}{2} + \frac{15}{36} - \frac{9}{36} = \frac{2}{3}.$$

(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



Thus we 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 the job, and more, 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:



We then see that 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 probabilities

From the last two exercises, it is natural to guess the formula

$$P(\bigcup_{i=1}^{n} E_i) = \sum_{i=1}^{n} P(E_i) - \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} P(E_i \cap E_j) + \sum_{i=1}^{n-2} \sum_{j=i+1}^{n-1} \sum_{k=j+1}^{n} P(E_i \cap E_j \cap E_k) - \dots . \quad (5.4)$$

All the sum signs in this notation suggest that we need some new notation to describe sums. Notice that what we were summing over in the second sum was all two-element sets $\{i, j\}$, and what we were summing over in the third sum was all three element sets $\{i, j, k\}$. The dots indicate that we continue summing, next summing the term $P(E_{i_1} \cap E_{i_2} \cap E_{i_3} \cap E_{i_4})$ over four element sets $\{i_1, i_2, i_3, i_4\}$, then a similar term over five element sets, and so on, so that every possible set of indices that we may choose from $N = \{1, 2, \dots n\}$ corresponds to one term of that sum. If a set of indices is odd in size, its term appears with a plus sign, and if it is even in size, its term appears with a minus sign.

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.

The first step is to introduce a notation for the terms we are summing up. Given a set $I = \{i_1, i_2, \dots i_k\}$ of indices, we define

$$\bigcap_{i:i \in I} E_i$$

to mean the intersection of all the sets E_i with i in I. In the case that $I = \{i_1, i_2, \dots i_k\}$, this can be written

$$\bigcap_{i:i \in I} E_i = E_{i_1} \cap E_{i_2} \cap \dots \cap E_{i_k} \ . \quad (5.5)$$

The second step is to introduce a notation that we can use to tell ourselves that the sum is to include one term for each subset J of our index set I, that the term of the sum corresponding to J is supposed to be $P(\bigcap_{i:i\in J} E_i)$, and that this term is to appear with a plus sign if the size of J is odd and with a minus sign if the size of J is even. We can take care of this last condition easily. Using $|J|$ to stand for the size of J, we multiply the term by $(-1)^{|J|+1}$ each time it appears in the sum. We write

$$\sum_{J:J\subseteq\{1,\ldots,n\},J\neq\emptyset} (-1)^{|J|+1} P\left(\bigcap_{i:i\in J} E_i\right)$$

to stand for the sum over all nonempty subsets J of the set $\{1,2,\ldots,n\}$ of the term

$$(-1)^{|J|+1} P(\bigcap_{i:i\in J} E_i) \ .$$

The set $\{1,2,\ldots,n\}$ will arise frequently, and we introduce the notation $[n]$ as shorthand for this set. In this notation the formula we believe to be true based on the exercises we solved is

$$P\left(\bigcup_{i=1}^{n} E_i\right) = \sum_{J:J\subseteq[n],J\neq\emptyset} (-1)^{|J|+1} P\left(\bigcap_{i:i\in J} E_i\right). \tag{5.6}$$

5.2-4 What is $\sum_{i:i\in[3]} 2^i$?

5.2-5 What is $\sum_{J:J\subseteq[3]} |J|$?

5.2-6 What is $\sum_{J:J\subseteq[n],|J|=7} (-1)^{|J|}$?

5.2-7 What is $\sum_{J:J\subseteq[7]} (-1)^{|J|}$

To answer Exercise 5.2-4, we just plug in the notation defined in Equation 5.5, and obtain

$$\sum_{i:i\in\{1,2,3\}} 2^i = 2^1 + 2^2 + 2^3 = 14 \ .$$

To answer Exercise 5.2-5, we see that we have to include a term for all subsets of the set $\{1,2,3\}$. Thus we obtain

$$\begin{aligned}\sum_{J:J\subseteq\{1,2,3\}} |J| &= |\{1\}| + |\{2\}| + |\{3\}| + |\{1,2\}| + |\{1,3\}| + |\{2,3\}| + |\{1,2,3\}| \\ &= 1+1+1+2+2+2+3 = 12 \ .\end{aligned}$$

Alternatively, we can observe that there are exactly $\binom{3}{i}$ subsets of size i, and so

$$\sum_{J:J\subseteq\{1,2,3\}} |J| = \sum_{i=1}^{3} \binom{3}{i} i = \binom{3}{1}1 + \binom{3}{2}2 + \binom{3}{3}3 = 12 \ .$$

For Exercise 5.2-5, the expression $\{J : J \subseteq [n], |J| = 7\}$, means that we are enumerating over all subsets of $\{1, 2, \ldots, n\}$ of size 7. There are $\binom{n}{7}$ such subsets and therefore

$$\sum_{J:J\subseteq[n],|J|=7} (-1)^{|J|} = \binom{n}{7}(-1)^7 = -\binom{n}{7}.$$

For Exercise 5.2-6, we have $\sum_{J:J\subseteq[7]} -1^{|J|} = \sum_{i=0}^{7}(-1)^i\binom{7}{i} = 0$. Why does the sum equal 0? By the binomial theorem, it is $(1-1)^7$ which is 0, or you can see in this case that $\binom{7}{k}$ and $\binom{7}{7-k}$ appear with opposite signs.

With this understanding of the notation in hand, we can now prove the principle of inclusion and Exclusion for Probability, which says that the formula we guessed in Equation 5.6 is correct. We will give two completely different proofs, 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 5.2.1 *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*

$$P\left(\bigcup_{i=1}^{n} E_i\right) = \sum_{J:J\subseteq[n],J\neq\emptyset} (-1)^{|J|+1} P\left(\bigcap_{i:i\in J} E_i\right). \tag{5.7}$$

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 $\bigcap_{i:i\in J} E_i$ if and only if $J \subseteq K$. Why is this? If there is an j in J that is not in K, then $x \notin E_j$ and thus $x \notin \bigcap_{i:i\in J} E_i$.

Recall that for any J, we define $P\left(\bigcap_{i:i\in J} E_i\right)$ to be the sum of the probability weights $p(y)$ for $y \in \bigcap_{i:i\in J} E_i$. This means that the coefficient of $p(x)$ on the right hand side of equation 5.7 is

$$\begin{aligned}
\sum_{J:J\subseteq K,J\neq\emptyset} -1^{|J|+1}p(x) &= p(x) + \sum_{J:J\subseteq K} (-1)^{|J|+1}p(x) \\
&= p(x) - p(x)\sum_{j=0}^{k}\binom{k}{j}(-1)^j \\
&= p(x) - p(x)(1-1)^k \\
&= p(x) - p(x)\cdot 0 \\
&= p(x),
\end{aligned}$$

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 the right hand side is the sum of $p(x)$ over each x in $\bigcup_{i=1}^{n} E_i$, and 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 5.3. 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 $F_1, F_2, \ldots, F_{n-1}$

$$P\left(\bigcup_{i=1}^{n-1} F_i\right) = \sum_{J:J\subseteq[n-1],J\neq\emptyset} (-1)^{|J|+1} P\left(\bigcap_{i:i\in J} F_i\right) \tag{5.8}$$

If in Equation 5.3 we let $E = E_1 \cup \ldots \cup E_{n-1}$ and $F = E_m$, we may apply Equation 5.3 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

$$P\left(\bigcup_{i=1}^{n} E_i\right) = P\left(\bigcup_{i=1}^{n-1} E_i\right) + P\left(E_n\right) - P\left(\bigcup_{i=1}^{n-1} \left(E_i \cap E_n\right)\right).$$

Now we use the inductive hypothesis (Equation 5.8) in two places to get

$$\begin{aligned} P\left(\bigcup_{i=1}^{n} E_i\right) &= \sum_{J:J\subseteq[n-1],J\neq\emptyset} (-1)^{|J|+1} P\left(\bigcap_{i:i\in J} E_i\right) \\ &+ P(E_n) \\ &- \sum_{J:J\subseteq[n-1],J\neq\emptyset} (-1)^{|J|+1} P\left(\bigcap_{i:i\in J} \left(E_i \cap E_n\right)\right). \end{aligned} \tag{5.10}$$

But when $J \subseteq [n-1]$

$$\bigcap_{i:i\in J} \left(E_i \cap E_n\right) = \bigcap_{i:i\in J\cup\{n\}} E_i$$

and $-(-1)^{|J|+1} = (-1)^{|J\cup\{n\}|+1}$. Substituting these into Equation 5.10 gives

$$\begin{aligned} P\left(\bigcup_{i=1}^{n} E_i\right) &= \sum_{J:J\subseteq[n-1],J\neq\emptyset} (-1)^{|J|+1} P\left(\bigcap_{i:i\in J} E_i\right) \\ &+ P(E_n) \\ &+ \sum_{J:J\subseteq[n-1],J\neq\emptyset} (-1)^{|J\cup\{n\}|+1} P\left(\bigcap_{i:i\in J\cup\{n\}} E_i\right). \end{aligned}$$

The first summation on the right hand side sums $(-1)^{|J|+1} P\left(\bigcap_{i:i\in J} E_i\right)$ over all subsets J of $\{1, 2, \ldots, n\}$ that *do not* contain n, while the $P(E_n)$ and the second summation work together to sum $(-1)^{|J'|+1} P(\bigcap_{i:i\in J'} E_i)$ over all subsets J' of $\{1, 2, \ldots, n\}$ that *do* contain n. Therefore,

$$P\left(\bigcup_{i=1}^{n} E_i\right) = \sum_{J:J\subseteq[n],J\neq\emptyset} (-1)^{|J|+1} P\left(\bigcap_{i:i\in J} E_i\right).$$

Thus by the principle of mathematical induction, this formula holds for all integers $n > 0$. ■

5.2-8 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? What is the probability that this happens? What is the probability that Sam gets his backpack? What is the probability that Judy and Sam both get their own backpacks? For any particular two element set of students, what is the probability that these two students get their own backpacks? 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.

In Exercise 5.2-8, 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. For any particular two element subset, such as Judy and Sam, there are 3! ways that these two people may get their backpacks back, so the probability is 3!/5! that each person in particular set of people gets his or her own backpack back. 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

$$P(E_1 \cup E_2 \cup E_3 \cup E_4 \cup E_5) = \sum_{J:J\subseteq[5],J\neq\emptyset} (-1)^{|J|+1} P\left(\bigcap_{i:i\in J} E_i\right) .$$

As we argued above, for a set of cardinality $|J|$, the probability that all $|J|$ people get their backpacks back is $(5-|J|)!/5!$. Therefore, we can rewrite the righthand side of the above equation as

$$\sum_{J:J\subseteq[5],J\neq\emptyset} (-1)^{|J|+1}(5-|J|)!/5! .$$

Now there are exactly $\binom{5}{j}$ subsets of {1,2,3,4,5} with cardinality j, so we can rewrite the above expression as

$$\begin{aligned}
\sum_{J:J\subseteq[5],J\neq\emptyset} (-1)^{|J|+1}\frac{(5-|J|)!}{5!} &= \sum_{j=1}^{5}(-1)^{j-1}\binom{5}{j}\frac{(5-j)!}{5!} \\
&= \sum_{j=1}^{5}(-1)^{j-1}\frac{5!}{j!(5-j)!}\frac{(5-j)!}{5!} \\
&= \sum_{j=1}^{5}(-1)^{j-1}\frac{1}{j!} \\
&= 1-\frac{1}{2}+\frac{1}{3!}-\frac{1}{4!}+\frac{1}{5!}.
\end{aligned}$$

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 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

$$\sum_{i=2}^{n}(-1)^{i}\frac{1}{i!} = \frac{1}{2} - \frac{1}{3!} + \cdots + \frac{(-1)^{n}}{n!} \tag{5.11}$$

Those who have had power series in calculus may recall theppower 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 5.11 as 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

5.2-9 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 one?

5.2-10 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?

5.2-11 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?

5.2-12 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 5.2-11.

In Exercise 5.2-11 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

$$|\bigcup_{i=1}^{k} E_i| = \sum_{J:J\subseteq[k],J\neq\emptyset} (-1)^{|J|+1} |\bigcap_{i:i\in J} E_i|. \tag{5.12}$$

In fact, this formula is proved by induction in virtually the same way. It is called *the principle of inclusion and exclusion for sets*. Applying this formula to the number of functions from N that map nothing to at least one element of K gives us

$$|\bigcup_{i=1}^{k} E_i| = \sum_{J:J\subseteq[k],J\neq\emptyset} (-1)^{|J|+1}(k-|J|)^n = \sum_{j=1}^{k}(-1)^{j-1}\binom{k}{j}(k-j)^n.$$

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.

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
$$\sum_{i : i \in [4]} i^2 \ ?$$

8. What is
$$\sum_{J : J \subseteq [4]} |J|^2 \ ?$$

9. What is
$$\sum_{J : J \subseteq [4]} (-1)^{|J|} |J|^2 \ ?$$

10. What is
$$\sum_{J : J \subseteq [4],\ J \neq [4]} (-1)^{|J|} |J|^2 \ ?$$

11. What is
$$\sum_{J : J \subseteq [4] \text{ and } |J|=3} |J|^2 \ ?$$

12. What is
$$\sum_{J : J \subseteq [4] \text{ and } |J|=3} \ \sum_{j : j \in J} j ?$$

13. 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?

14. 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?

15. From the formula for the number of onto functions, find a formula for $S(n, k)$ which is defined in Problem 7 of Section 1.3. These numbers are called *Stirling numbers (of the second kind).*

16. 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?

17. Suppose we have a collection of n 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 $f_a(S)$ (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, $f_a(\emptyset) = n$. In a typical application, formulas for $f_a(S)$ for other sets S are not difficult to figure out. Define $f_e(S)$ to be the number of objects in our collection that have exactly the properties in S. Show that

$$f_e(\emptyset) = \sum_{I:\emptyset \subseteq I \subseteq P} (-1)^{|I|} f_a(I).$$

    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 13 in this section?

18. In Exercise 5-17, what is the probability that an object has none of the properties, assuming all objects to be equally likely? How would this apply the Exercise 5-14?

19. 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?

20. A group of 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?

21. In Exercise 5-20 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.

22. If we hash n keys into a hash table with k locations, what is the probability that a given location (say slot 1) gets three or more keys? What is the probability that at least one location gets three or more keys? What is the probability that no location gets three or more keys? For extra credit write a function that computes this last probability and use it to compute the probability that no location gets 3 or more keys with ten keys and a hash table of size 20 and with 50 keys and a hash table of size 100.

23. In how many ways may we place n distinct books on k shelves so that shelf one gets at least m books? (See Exercise **??**-8.) In how many ways may we place n distinct books on k shelves so that no shelf gets more than m books?

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.

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

$$\frac{1}{36} + \frac{1}{9} + \frac{1}{4} = \frac{14}{36} = \frac{7}{18} \tag{5.13}$$

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

$$P(E|F) = \frac{P(E \cap F)}{P(F)}. \tag{5.14}$$

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. We then made this assumption into part of our definition of what probability means by introducing the definition of conditional probability.[4]

In the example above, we can let E be the event that there is greater 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 5.13, $\frac{7}{18}$. Dividing, we get the probability of $P(E|F)$, which is $\frac{1}{9}/\frac{7}{18} = \frac{2}{7}$.

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.

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."

5.3-4 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. What is the probability that someone who has positive test results in fact has the disease?

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,

[4]For those who like to think in terms of axioms of probability, we simply added the axiom that the relative probabilities of the outcomes in the event F don't change when F happens.

$P(F) = 1/6$ 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.

For Exercise 5.3-4, we are given that

$$P(\text{disease}) = .001 \ , \tag{5.15}$$

$$P(\text{positive test result}|\text{disease}) = .99 \ , \tag{5.16}$$

$$P(\text{positive test result}|\text{no disease}) = .02 \ . \tag{5.17}$$

We wish to compute

$$P(\text{disease}|\text{positive test result}) \ .$$

We use Equation 5.14 to write that

$$P(\text{disease}|\text{positive test result}) = \frac{P(\text{disease} \cap \text{positive test result})}{P(\text{positive test result})} \ . \tag{5.18}$$

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 5.14 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 5.16 and 5.15 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 5.18, we observe that since each person either has the disease or doesn't, we can write

$$P(\text{positive test result}) = P(\text{positive test result} \cap \text{disease}) + P(\text{positive test result} \cap \text{no disease}) \ . \tag{5.19}$$

We have already computed $P(\text{positive test result} \cap \text{disease})$, and we can compute $P(\text{positive test result} \cap \text{no disease})$ 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 5.15 and 5.17, we get that $P(\text{positive test result} \cap \text{no disease}) = (.02)(1 - .001) = .01998$ We now have the two components of the righthand side of Equation 5.19 and thus $P(\text{positive test result}) = .00099 + .01998 = .02097$. Finally, we have all the pieces in Equation 5.18, and conclude that

$$P(\text{disease}|\text{positive test result}) = \frac{P(\text{disease} \cap \text{positive test result})}{P(\text{positive test result})} = \frac{.00099}{.02097} = .0472 \ .$$

Thus, given a test with this error rate, a positive result only gives you roughly a 5% chance of having the disease! This is another instance where probability shows something counterintuitive.

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 5.3.1) gives another test for independence.

Theorem 5.3.1 *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 if "if and only if" is meant.) Using the definition of $P(E|F)$ in Equation 5.14, 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 5.3.2 *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 did not capture this! 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. Note that

$$P(H \text{ first})P(H \text{ second}) = \frac{1}{4} = P(\text{H first and H second}).$$

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.

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 $i = j$?

5.3-6 Show that if we flip a coin twice, the events E: "heads on the first flip," F: "Heads on the second flip," and H: "Heads on both flips" are each independent of the other two. Is

$$P(E \cap F \cap H) = P(E)P(F)P(H)?$$

Would it seem reasonable to say that these events are mutually independent?

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

$$P(E_1 \cap E_2 \cap \cdots E_{i-1} \cap E_i) = P(E_1 \cap E_2 \cap \cdots E_{i-1}) P(E_i). \tag{5.20}$$

Theorem 5.3.3 *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 5.20. ■

How does this reflect coin flipping? Here our sample space consists of sequences of 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. Suppose we have a three-stage process consisting of flipping a coin, then flipping it again, and then computing the total number of heads. In this process, the outcome of the third stage is 2 if and only if the outcome of stage 1 is H and the outcome of stage 2 is H. Thus probability that the third outcome is 2, given that the first two are H is 1. Since the probability of two heads is 1/4 the third outcome is not independent of the first two. How do independent trials reflect hashing a list of keys? Here 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 is independent of the event that the first $i-1$ keys hash to some number numbers q_1, q_2, q_{i-1}. Thus our model of hashing is an independent trials process.

5.3-7 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?

5.3-8 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 5.3-7 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 5.3-8, 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. Therefore we are not drawing from the same sample space. Further, the probability of getting a particular card on the ith draw increases as we draw more and more cards, unless we have already drawn this card, in which case the probability of getting this card drops to zero.

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 it would be interesting to 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.*

5.3-9 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 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 5.3.4.

Theorem 5.3.4 *The probability of having exactly k successes in a sequence of 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 5.3.4 are called **binomial probabilities**, or the **binomial probability distribution**.

5.3-10 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?

5.3-11 Recall the primality testing algorithm from Chapter **??**. 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 corresponds to 8, 9, or 10 successes in ten trials, in Exercise 5.3-10 we have

$$P(\text{80 or better}) = \binom{10}{8}(.8)^8(.2)^2 + \binom{10}{9}(.8)^9(.2)^1 + (.8)^{10}.$$

Some work with a calculator gives us that this sum is approximately .678.

In Exercise 5.3-11, 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 1 in a million, and the chance of certifying the non-prime as non-prime is 1 minus this. Therefore it is almost sure that a non-prime number will 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 instead of using H and T we put the probabilities of x times the probability of success and y times the probability of failure three independent trials in place of H and T 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

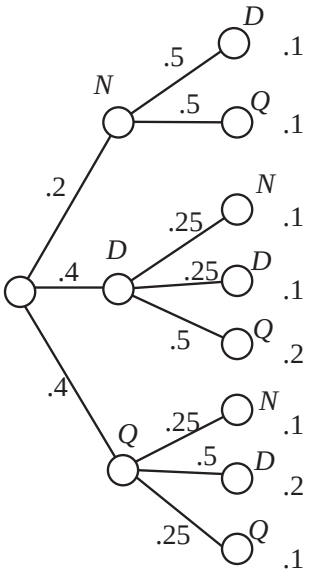
$$(px + (1-p)y)^n = \sum_{k=0}^{k}\binom{n}{k}p^k(1-p)^{n-k}x^ky^{n-k}.$$

Taking the coefficient of x^ky^{n-k} from this sum, we get exactly the result of Theorem 5.3.4. 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.

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 digram. We will explain what we mean by giving a tree diagram

Figure 5.1: A tree diagram illustrating a two-stage process.



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.

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.

5.3-12 How can we recognize from a probability tree whether it is the probability tree of an independent trials process?

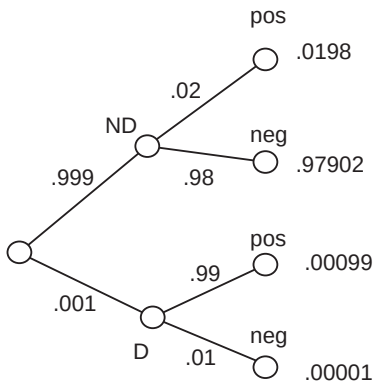
5.3-13 Draw a tree diagram for Exercise 5.3-4 in which stage 1 is whether or not the person has the disease and stage two is the outcome of the test for the disease. Which branches of the tree correspond to the event of having a positive test? Describe the computation we made in Exercise 5.3-4 in terms of probabilities of paths from the root to leaves of the tree.

5.3-14 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 do you expect her grade to be on a 100 question True-False test to be if she guesses at each 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.)

A tree for an independent trials process has the property that at each non-leaf node, the (labeled) tree consisting of that node and all its children is identical to the labeled tree consisting of the root and all its children. If we have such a tree, then it automatically satisfies the definition of an independent trials process.

In Figure 5.3 we show the tree diagram for Exercise 5.3-4. We use D to stand for having

Figure 5.2: A tree diagram illustrating Exercise 5.3-4.



the disease and ND to stand for not having the disease. The computation we made consisted of dividing the probability of the path root-D-pos by the sum of the probabilities of the paths root-ND-pos and root-D-pos.

In Exercise 5.3-14, 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? Then as we see in Figure 5.3 she actually has probability .9 of getting a right answer if she guesses at each question she does not know the answer to. We can also write this out algebraically. 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,

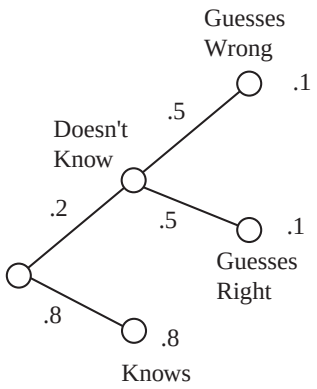
$$\begin{aligned} P(R) &= P(R \cap K) + P(R \cap \overline{K}) \\ &= P(R|K)P(K) + P(R|\overline{K})P(\overline{K}) \\ &= 1 \cdot .8 + .5 \cdot .2 = .9 \end{aligned}$$

Thus we would expect her to get a grade of 90%.

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?

Figure 5.3: The probability of getting a right answer is .9.



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. What is our sample space for hashing a list of n items into m slots? What probability weights do we use for this sample space. We can think of a member of this sample space as a sequence of n hashes. When we think of it in this way, is hashing an independent trials process?

6. 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?

7. What is the probability of exactly eight heads in ten flips of a coin? Of eight or more heads?

8. 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?

9. 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.

10. 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?

11. 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?

12. 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?

13. 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 $i + j + k = n$.)

14. 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?

15. Suppose E and F are events with $E \cap F = \emptyset$. Describe when E and F are independent and explain why.

16. 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.

5.4-1 Give several random variables that might be of interest to a doctor whose sample space is her patients.

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 having probability 1/2 of heads means that if we average the number of heads over all possible outcomes, the average should be half the number of flips. Thus we would say we expect the number of heads to be half the number of flips. We will explore this more formally below.

Expected Value

Notice that we have been talking the value we *expect* a random variable to have. We haven't yet defined this term, 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

$$\frac{0+1+1+1+2+2+2+3}{8} = 1.5.$$

5.4-3 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

$$\frac{0\binom{10}{0} + 1\binom{10}{1} + 2\binom{10}{2} + \cdots + 9\binom{10}{9} + 10\binom{10}{10}}{1024} = \frac{\sum_{i=0}^{10} i\binom{10}{i}}{1024}. \tag{5.21}$$

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 5.21. 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 5.21 as

$$0\frac{\binom{10}{0}}{1024} + 1\frac{\binom{10}{1}}{1024} + 2\frac{\binom{10}{2}}{1024} + \cdots + 9\frac{\binom{10}{9}}{1024} + 10\frac{\binom{10}{10}}{1024} = \sum_{i=0}^{10} i\frac{\binom{10}{i}}{1024}. \tag{5.22}$$

In Equation 5.22 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 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.

5.4-4 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 5.4-4 we asked for a proof of a fundamental lemma

Lemma 5.4.1 *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 we can take the items in the sample space, group them together into the events F_i and get the definition of expectation, as follows:

$$\begin{aligned} \sum_{s:s\in S} X(s)P(s) &= \sum_{i=1}^{k} \sum_{s:s\in F_i} X(s)P(s) \\ &= \sum_{i=1}^{k} \sum_{s:s\in F_i} x_i P(s) \end{aligned}$$

$$\begin{aligned} &= \sum_{i=1}^{k} x_i \sum_{s:s\in F_i} P(s) \\ &= \sum_{i=1}^{k} x_i P(F_i) \\ &= \sum_{i=1}^{k} x_i P(X = x_i) = E(X). \end{aligned}$$

■

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_k and add up their probabilities; this 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 5.4.2 *Suppose X and Y are random variables on the (finite) sample space S. Then*

$$E(X + Y) = E(X) + E(Y).$$

Proof: From Lemma 5.4.1 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 5.4.3 *Suppose X is a random variable on a sample space S. Then for any number c, $E(cX) = cE(X)$.*

Proof: Left as an exercise.■

Theorems 5.4.2 and 5.4.3 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, because in Theorems 5.4.2 and 5.4.3, it is not required that X and Y be independent.

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

$$X = X_1 + X_2 + \cdots X_n, \tag{5.23}$$

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 5.23 and apply Lemma 5.4.2 repeatedly 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.

5.4-5 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 5.4-5, 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 5.4.2 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 5.4.4 *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 5.4.2 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

5.4-6 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

$$\sum_{i=0}^{\infty}(1-p)^ip = p\sum_{i=0}^{\infty}(1-p)^i = p\frac{1}{1-(1-p)} = \frac{p}{p} = 1.$$

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 is called a *geometric* distribution because of the geometric series we used in proving the probabilities sum to 1.

Theorem 5.4.5 *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

[5] 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 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.

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 fourth to the fifth line we used the fact that

$$\sum_{j=0}^{\infty} j x^j = \frac{x}{(1-x)^2}. \tag{5.24}$$

You may have proved a finite version of this equation for homework; 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.

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. How many times do you expect to have to role a die until you see a six on the top face?
6. 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 $E(c)$.
7. 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 it to be the percentage of the material being tested that the test-taker knows. How can we do this?

8. Do Exercise 5-7 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.

9. In as many ways as you can, prove that

$$\sum_{i=0}^{n} i\binom{n}{i} = 2^{n-1}n.$$

10. Prove Theorem 5.4.3.

11. 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?

12. 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.

13. 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 $(XY)(s) = X(s)Y(s)$.

14. 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 5-13 for a definition of XY.

15. 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 5.24.

16. 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.

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?

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 5.5.1 *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.*

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. Thus the expected number of empty slots is, by Theorem 5.4.2, $k(1 - \frac{1}{k})^n$. Thus we have proved another nice theorem about hashing.

Theorem 5.5.2 *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 exercises 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. Is something wrong? No, but we simply have to accept that our expectations about expectation just don't always hold true. When we want to make a statement about expected values, we must use either our definitions or theorems to back it up.

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 5.4.2 and 5.4.3,

$$E(\text{collisions}) = n - E(\text{occupied locations}) = n - k + E(\text{empty locations}) \tag{5.25}$$

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 5.5.3 *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 5.5.2 that the expected number of empty locations is $k(1-\frac{1}{k})^n$. Substituting this into Equation 5.25 gives our formula. ■

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 $i-1$ full slots for the first time and the first time that there are i full 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 what the expected number of steps until the first success is. Thus we can apply Lemma 5.4.5 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 5.4.5 we get

$$\begin{aligned} E(X) &= \sum_{i=1}^{k} E(X_i) \\ &= \sum_{i=1}^{k} \frac{k}{k-i+1} \\ &= k \sum_{i=1}^{k} \frac{1}{k-i+1} \\ &= k \sum_{i=1}^{k} \frac{1}{i}, \end{aligned}$$

where the last line follows just by switching the order of the summation. 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 we shall see why in the exercises) that $\sum_{i=1}^{k} \frac{1}{i} = \Theta(\log k)$, and more precisely

$$\frac{1}{2} + \ln k \leq H_k \leq 1 + \ln k. \tag{5.26}$$

In fact, as n gets large, $H_n - \ln n$ approaches a limit called *Euler's constant*; Euler's constant is about .58. Equation 5.26 gives us that $E(X) = O(k \log k)$.

Theorem 5.5.4 *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 5.5.5 *For all $x > 0$, $(1+\frac{1}{x})^x \leq e$.*

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,

$$x! = \left(\frac{x}{e}\right)^x \sqrt{2\pi x}(1+\Theta(1/n)),$$

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

$$x! = \left(\frac{x}{e}\right)^x \sqrt{2\pi x}.$$

(All our statements can be corrected for the more accurate bound.) Using Stirling's formula, we can get a bound on $\binom{n}{t}$,

Lemma 5.5.6 *For $n > t > 0$,*

$$\binom{n}{t} \leq \frac{n^n}{t^t(n-t)^{n-t}}.$$

Proof:

$$\begin{aligned}
\binom{n}{t} &= \frac{n!}{t!(n-t)!} && (5.27)\\
&= \frac{(n/e)^n\sqrt{2\pi n}}{(t/e)^t\sqrt{2\pi t}((n-t)/e)^{n-t}\sqrt{2\pi(n-t)}} && (5.28)\\
&= \frac{n^n\sqrt{n}}{t^t(n-t)^{n-t}\sqrt{2\pi}\sqrt{t(n-t)}} && (5.29)
\end{aligned}$$

Now if $1 < t < n-1$, we have $t(n-t) \geq n$, so that $\sqrt{t(n-t)} \geq \sqrt{n}$. Further $\sqrt{2\pi} > 1$. We can use these to facts to upper bound the quantity marked 5.29 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 $n \leq n^n/(n-1)^{n-1}$ 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

$$P(H_{it}) = \binom{n}{t} \left(\frac{1}{n}\right)^t \left(1 - \frac{1}{n}\right)^{n-t}. \tag{5.30}$$

Now we relate this known quantity to the probability of the event M_t that the maximum list size is t.

Lemma 5.5.7 *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*

$$P(M_t) \leq nP(H_{1t})$$

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},

$$P(M_{it}) \leq P(H_{it}). \tag{5.31}$$

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,

$$P(M_t) \leq P(M_{1t}) + P(M_{2t}) + \cdots + P(M_{nt}). \tag{5.32}$$

(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 5.32 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 5.30 for $P(H_{1t})$ and then apply Lemma 5.5.6 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 5.5.5 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 5.5.8 *$P(M_t)$, 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 5.5.7 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 5.5.8, 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 5.5.8 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

$$\sum_{t=0}^{n} P(M_t)t = \sum_{t=0}^{5\log n/\log\log n} P(M_t)t + \sum_{t=5\log n/\log\log n}^{n} P(M_t)t \tag{5.33}$$

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

$$\sum_{t=0}^{5\log n/\log\log n} P(M_t)t \leq \sum_{t=0}^{5\log n/\log\log n} P(M_t)5\log n/\log\log n \tag{5.34}$$

$$= 4\log n/\log\log n \sum_{t=0}^{5\log n/\log\log n} P(M_t) \quad (5.35)$$

$$\leq 5\log n/\log\log n \quad (5.36)$$

(Note that we are not using Lemma 5.5.8 here; only the fact that the probabilities of disjoint events cannot add to more than 1.) For the rightmost sum in Equation 5.33, we want to first compute an upper bound on $P(M_t)$ for $t = (5\log n/\log\log n)$. Using Lemma 5.5.8, 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 5.5.8 decreases as t grows, and $t \leq n$, we can bound the right sum by

$$\sum_{t=4\log n/\log\log n}^{n} P(M_t)t \leq \sum_{t=4\log n/\log\log n}^{n} \frac{1}{n^2}n \leq \sum_{t=4\log n/\log\log n}^{n} \frac{1}{n} \leq 1. \quad (5.37)$$

Combining Equations 5.36 and 5.37 with 5.33 we get the desired result.

Theorem 5.5.9 *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 $ne^t/t^t = n^{-2}$, 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.

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. In Exercise 5.5-1 we said our model of hashing n items into a table of size k is that of an independent trials process with k equally likely outcomes in each trial. In an earlier section we said our model of hashing has a sample space of all possible hash functions, thought of as sequences that tell us for, each location, where it hashes to. We regarded all sequences (and thus all hash functions) as equally likely. Are these two models of hashing in conflict? Why or why not? No vague answers allowed.

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 5.25 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 $\lim_{n\to\infty}(1-\frac{1}{n})^n = \frac{1}{e}$.

14. What is the expected number of empty slots when we hash $2k$ items into a hash table with k slots?

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, and will introduce the idea of using recurrences to find bounds on expected running times of algorithms whose running time depends on their inputs rather than just the size of their inputs. 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].

When Running Times Depend on more than Size of Inputs

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?

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_i, 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 average 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.

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, 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

$$E(X|F) = \sum_{x:x\in F} X(x)\frac{P(x)}{P(F)}. \tag{5.38}$$

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 5.38 as

$$E(X|F) = \sum_{i=1}^{k} x_i P(X = x_i|F).$$

Theorem 5.6.1 *Let X be a random variable defined on a sample space and let F_1, F_2, ... 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

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?

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, if we let $T(n)$ be the expected running time on a list of length n, then 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 5.6.1, we see 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 1/3 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 5.4.3 since in this context $\Theta(n)$ behaves as if it were a constant[6], since n does not depend on R. By Lemma 5.4.5, 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 5.6.1, 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 4.7. Recall that in this algorithm, we first picked an an element p in the middle half, 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:

```
RandomSelect(A,i,n)
(selects the ith 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
```

[6]What we mean here is that $T \geq Rc_1 n$ for some constant c_1 and $T \leq Rc_2 n$ for some other constant c_2. Then we apply Theorem 5.4.3 to both these inequalities, using the fact that if $X > Y$, then $E(X) > E(Y)$ as well.

```
Let L be the set of elements less than or equal to p
If H is empty
     put p in H
if (i ≤ |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 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.

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

$$T(n) \leq \begin{cases} \frac{1}{2}T(\frac{3}{4}n) + \frac{1}{2}T(n) + bn & \text{if } n > 1 \\ d & \text{if } n = 1 \end{cases}.$$

We may rewrite the recursive part of the recurrence as

$$\frac{1}{2}T(n) \leq \frac{1}{2}T\left(\frac{3}{4}n\right) + bn,$$

or

$$T(n) \leq T\left(\frac{3}{4}n\right) + 2bn = T\left(\frac{3}{4}n\right) + b'n.$$

5.6-6 Why does every solution to the recurrence

$$T(n) \leq T\left(\frac{3}{4}n\right) + b'n.$$

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 5.6.2 *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 divide 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 ℓ = |L|
     If H is empty
          put p in H
     A_1 = QuickSort(H,h)
     A_2 = QuickSort(L,ℓ)
     return the concatenation of A_1 and A_2
```

There is an analysis of quicksort similar to the detailed analysis RecursiveSelect at the end of the section, and this is an exercise at the end of the section. Instead, based on the preceeding 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.

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 ℓ = |L|
     Until (|H| ≥ n/4) and (|L| ≥ n/4)
     A_1 = QuickSort(H,h)
     A_2 = QuickSort(L,ℓ)
     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 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 of our convention that $T(n)$ now 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

$$T(n) \leq \begin{cases} T(a_n n) + T((1 - a_n))n + bn & \text{if } n > 1 \\ d & \text{if } n = 1 \end{cases},$$

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($|H| \geq n/4$ and $|L| \geq n/4$, 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 is less than $n/4$. Then since with probability 1/2 the element p is between $n/4$ and $3n/4$, we can write

$$T(n) \leq \frac{1}{2}T(n) + \frac{1}{2}(T(a_n n) + T((1 - a_n)n) + bn),$$

which simplifies to

$$T(n) \leq T(a_n n) + T((1-a_n)n) + 2bn,$$

or

$$T(n) \leq T(a_n n) + T((1-a_n)n) + b'n.$$

Again by Exercise 5.6-7 the running time of this algorithm is $O(n \log n)$.

Further, it is easy 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 5.6.3 *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.

5.6-8 Explain why, if we pick the kth element as the random element in RandomSelect ($k \neq n$), 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 5.6.1 and the solution to Exercise 5.6-8, we can write that

$$T(n) \leq \begin{cases} \sum_{k=1}^{n-1} P(X=k)(T(\max\{k, n-k\}) + bn) + P(X=n)(T(\max\{1, n-1\} + bn) & \text{if } n > 1 \\ d & \text{if } n = 1 \end{cases}.$$

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

$$T(n) \leq \begin{cases} \frac{2}{n}\left(\sum_{k=n/2}^{n-1} T(k)\right) + \frac{1}{n}T(n-1) + bn & \text{if } n > 1 \\ d & \text{if } n = 1 \end{cases}. \tag{5.39}$$

If n is odd, the lower limit of the sum is a half-integer, so the possible integer values of the dummy variable k runs 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.

5.6-9 Show that every solution to the recurrence in Equation 5.39 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 5.6.2.

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).

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, exchanging $A(i)$ with

L is $A(i)$ 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 $T(n) \leq T(\frac{2}{3}n) + bn$ 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

$$T(n) \leq \frac{2}{n-1} \sum_{k=n/2}^{n-1} + bn.$$

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 5.6.1.

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 median algorithm. More precisely, we use can 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 Selection 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? CLIFF, WILL YOU READ THE LAST PARAGRAPH OF THE SOLUTION CAREFULLY? IS THERE A BETTER WAY TO ANSWER THE QUESTION? THANKS!

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 Exercise 5.6-13 for a definition) is the expected depth of a leaf node. Arguing as in Exercise 5.6-13 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 Exercise 5.6-13 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 the expected depth of one of its children. Argue as in Exercise 5.6-13 and the first proof of Theorem 5.6.2 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 = −∞
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

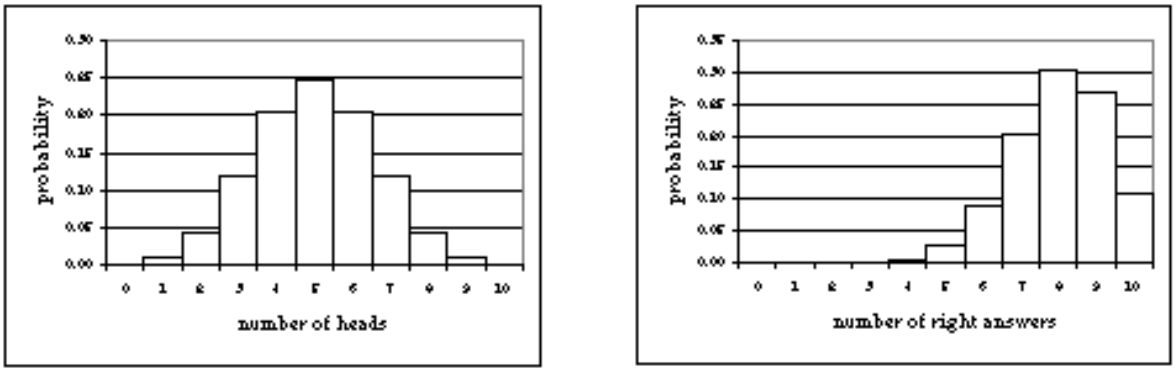
Distributions of random variables

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 as a histogram. In Figure 5.4 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.

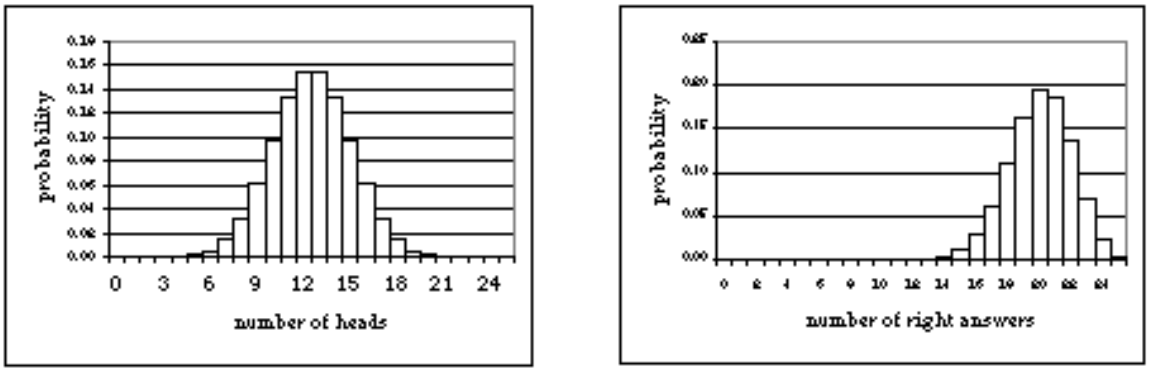
Figure 5.4: Two histograms.



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 is 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 5.5 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 18 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 5.5: Histograms of 25 trials



In Figure 5.6 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 5.6: One hundred independent trials

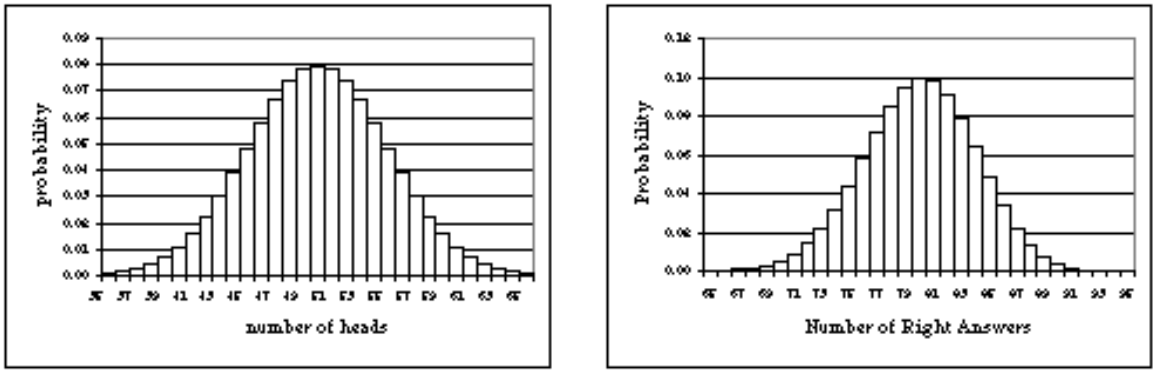
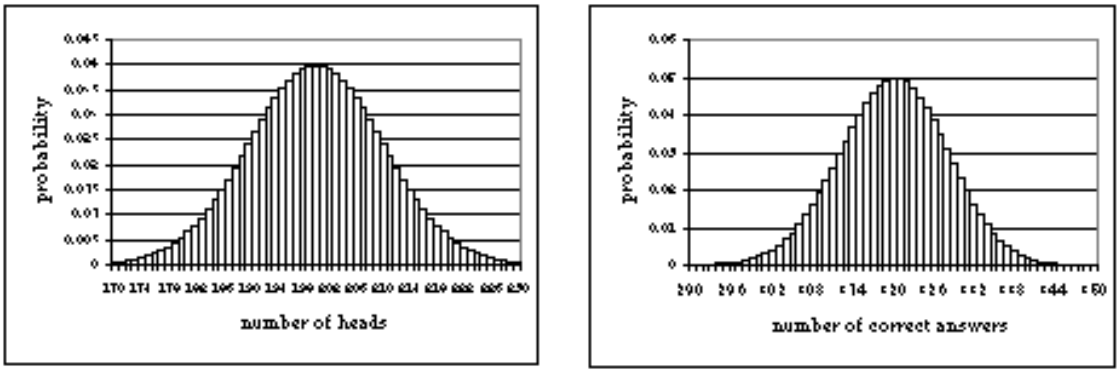


Figure 5.7: Four hundred independent trials



we might predict that we would need only about 60 values to see essentially all the results in four hundred trials. As Figure 5.7 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

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 $E(Y)$. Doe 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 Exercise 5.7-6) and corollary showing that the expected value of a constant is a constant.

Lemma 5.7.1 *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 5.7.2 $E(E(X)) = E(X)$.

Proof: When we think of $E(X)$ as a random variable, it has a constant value, μ. By Lemma 5.7.1 $E(E(x)) = E(\mu) = \mu = E(x)$. ■

Returning to Exercise 5.7-1, we can use linearity of expectation and Corollary 5.7.2 to show that

$$E(X - E(X)) = E(X) - E(E(X)) = E(X) - E(X) = 0. \tag{5.40}$$

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 go 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

$$V(X) = E(X - E(X))^2 = \sum_{s:s \in S} P(s)(X(s) - E(X))^2. \tag{5.41}$$

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.

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?

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?

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 5.7.3 *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 (5.42) to (5.43), we use the fact that X and Y are independent; the rest of the equations follow from definitions and algebra.

$$\begin{aligned} E(X)E(Y) &= \sum_{i=1}^{k} x_i P(X = x_i) \sum_{j=1}^{m} y_j P(Y = y_j) \\ &= \sum_{i=1}^{k} \sum_{j=1}^{m} x_i y_j P(X = x_i) P(y = y_j) \end{aligned}$$

$$= \sum_{z:\, z \text{is a value of } XY} z \sum_{(i,j):x_i y_j = z} P(X = x_i)P(Y = y_j) \quad (5.42)$$

$$= \sum_{z:\, z \text{is a value of } XY} z \sum_{(i,j):x_i y_j = z} P((X = x_i) \wedge (Y = y_j)) \quad (5.43)$$

$$= \sum_{z:\, z \text{is a value of } XY} zP(XY = z)$$

$$= E(XY)$$

■

Theorem 5.7.4 *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 Exercise 5.7-7.) Thus we can apply Lemma 5.7.3 to the middle term to obtain

$$= V(X) + 2E(X - E(X))E(Y - E(Y)) + V(Y).$$

Now we apply Equation 5.40 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 $10/4 = 2.5$.

5.7-5 Find the variance for 100 flips of a coin and 400 flips of a coin.

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 random variable is between a and b is an appropriately chosen multiple of $\int_a^b e^{-cx^2}\,dx$ when the number of random variables we are adding is sufficiently large.[7] The distribution given by that 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!

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?

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 $n = 10^6/25 = 40,000$.

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?

[7] Still more precisely, if we scale the sum of our random variables by letting $Z = \frac{X_1+X_2+....X_n-n\mu}{\sigma\sqrt{n}}$, then the probability that $a \le Z \le b$ is $\int_a^b \frac{1}{\sqrt{2\pi}} e^{-x^2/2}\,dx$.

2. In Exercise 5.7-1 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 Exercise 5.7-1?

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 withdraaw, 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 Exercise 5.7-1 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. Show that if X and Y are independent and b and c are constant, then $X - b$ and $Y - c$ are independent.

8. 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?

9. 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?

10. What are the variance and standard deviation for the sum of the tops of n dice that we roll?

11. 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%?

12. 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.")

13. Given a random variable X, how does the variance of cX relate to that of X?

14. 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 Exercise 5.7-9) of the "number of successes" random variable for n independent trials is less than or equal to $n/4$?

15. 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*.

16. Use Exercise 5.7-14 show that in n independent trials with probability p of success,
    $$P\left(\left|\frac{\#\ \text{ of successes} - np}{n}\right| \geq r\right) \leq \frac{1}{4nr^2}$$

17. 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.

18. 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?

19. Another way to bound the deviance from the expectation is known as Markov's inequality. This inequality says that is 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.