Discrete Math for Computer Science Students

Ken Bogart
Dept. of Mathematics
Dartmouth College

Cliff Stein
Dept. of Industrial Engineering
and Operations Research
Columbia University

September 24, 2003

Contents

Chapter 1

Counting

1.1 Basic Counting

The Sum Principle

We begin with an example that illustrates a fundamental principle.

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

```
(1)  for i = 1 to n
(2)       for j = i + 1 to n
(3)            if (A[i] > A[j])
(4)                 exchange A[i] and A[j]
```

How many times is the comparison $A[i] > A[j]$ made in Line 3?

In Exercise 1.1-1, the segment of code from lines 2 through 4 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

$$(n-1) + (n-2) + \cdots + 1 + 0 \; . \tag{1.1}$$

This formula is not as important as the reasoning that lead us to it. In order to put the reasoning into a broadly applicable format, we will describe what we were doing in the language of sets. Think about the set S containing all comparisons the algorithm in Exercise 1.1-1 makes. We divided set S 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.

Using some set-theoretic terminology, we describe a general version of the process we used. Two sets are called *disjoint* when they have no elements in common. Each of the sets S_i 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 sets $\{S_1, \ldots, S_n\}$ is a family of *mutually disjoint sets*, meaning that it is a family (set) of sets, any 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.

Principle 1.1 (Sum Principle) *The size of a union of a family of mutually disjoint finite 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. We can describe the sum principle using an algebraic notation. Let $|S|$ denote the size of the set S. For example, $|\{a, b, c\}| = 3$ and $|\{a, b, a\}| = 2$.[1] Using this notation, we can state the sum principle as: if S_1, S_2, $\ldots S_n$ are disjoint sets, then

$$|S_1 \cup S_2 \cup \cdots \cup S_n| = |S_1| + |S_2| + \cdots + |S_n| \ . \tag{1.2}$$

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

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

When we can write a set S as a union of disjoint sets S_1, S_2, $\ldots$, S_k we say that we have *partitioned* S into the sets S_1, S_2, $\ldots$, S_k, and we say that the sets S_1, S_2, $\ldots$, S_k form a *partition* of S. Thus $\{\{1\}, \{3, 5\}, \{2, 4\}\}$ is a partition of the set $\{1, 2, 3, 4, 5\}$ and the set $\{1, 2, 3, 4, 5\}$ can be partitioned into the sets $\{1\}$, $\{3, 5\}$, $\{2, 4\}$. It is clumsy to say we are partitioning a set into sets, so instead we call the sets S_i into which we partition a set S the *blocks* of the partition. Thus the sets $\{1\}$, $\{3, 5\}$, $\{2, 4\}$ are the blocks of a partition of $\{1, 2, 3, 4, 5\}$. In this language, we can restate the sum principle as follows.

Principle 1.2 (Sum Principle) *If a finite set S has been partitioned into blocks, then the size of S is the sum of the sizes of the blocks.*

Abstraction

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 the axioms of set theory. In a course in discrete mathematics, this level of abstraction is unnecessary, so we will

[1] It may look strange to have $|\{a, b, a\}| = 2$, but an element either is or is not in a set. It cannot be in a set multiple times. (This situation leads to the idea of multisets that will be introduced later on in this section.) We gave this example to emphasize that the notation $\{a, b, a\}$ means the same thing as $\{a, b\}$. Why would someone even contemplate the notation $\{a, b, a\}$. Suppose we wrote $S = \{x | x \text{ is the first letter of Ann, Bob, or Alice}\}$. Explicitly following this description of S would lead us to first write down $\{a, b, a\}$ and the realize it equals $\{a, b\}$.

simply use the sum principle as the basis of computations when it is convenient to do so. If our goal were only to solve this one exercise, then our abstraction would have been almost a mindless exercise that complicated what was an "obvious" solution to Exercise 1.1-1. However the sum principle will prove to be useful in a wide variety of problems. Thus we observe the value of abstraction—when you can recognize the abstract elements of a problem, then abstraction often helps you solve subsequent problems as well.

Summing Consecutive Integers

Returning to the problem in Exercise 1.1-1, we need to compute the sum given in Equation 1.1. We may also write this sum 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 values we are summing are $n - 1, n - 2, \ldots, 1$, so we may write that

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

Notice that the first sum has n terms, one of which is zero, while the second sum has $n-1$ terms.

A clever trick, usually attributed to Gauss, gives us a shorter formula for this sum.

We write

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

The sum below the horizontal line has $n - 1$ terms each equal to n, and thus 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

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

This lovely trick gives us little or no real mathematical skill; learning how to think about things to discover answers ourselves is much more 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 at the end of this section and see a way that we could have discovered this formula for ourselves without any tricks.

The Product Principle

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

```
(1)  for i = 1 to r
(2)       for j = 1 to m
(3)            S = 0
(4)            for k = 1 to n
(5)                 S = S + A[i,k] * B[k,j]
(6)            C[i,j] = S
```

How many multiplications (expressed in terms of r, m, and n) does this code carry out in line 5?

Exercise 1.1-3 Consider the following longer piece of pseudocode that sorts a list of numbers and then counts "big gaps" in the list:

```
(1)  for i = 1 to n - 1
(2)       minval = A[i]
(3)       minindex = i
(4)       for j = i to n
(5)            if (A[j] < minval)
(6)                 minval = A[j]
(7)                 minindex = j
(8)       exchange A[i] and A[minindex]
(9)
(10) for i = 2 to n
(11)      if (A[i] < 2 * A[i - 1])
(12)           bigjump = bigjump +1
```

How many comparisons does the above code make in lines 5 and 11 ?

In Exercise 1.1-2, the program segment in lines 4 through 5, 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 in lines 2 through 5 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.

Why did we add in Exercise 1.1-1, but multiply here? We can answer this question using the abstract point of view we adopted in discussing Exercise 1.1-1. Our algorithm performs a certain set of multiplications. For any given i, the set of multiplications performed in lines 2 through 5 can be divided into the set S_1 of multiplications performed when $j = 1$, the set S_2 of multiplications performed when $j = 2$, and, in general, the set S_j of multiplications performed for any given j value. Each 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. Let T_i be the set of multiplications that our program segment carries out for a certain i value. The set T_i is the union of the sets S_j; restating this as an equation, we get

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

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

$$|T_i| = |\bigcup_{j=1}^{m} S_j| = \sum_{j=1}^{m} |S_j| = \sum_{j=1}^{m} n = mn \ . \tag{1.3}$$

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.

Principle 1.3 (Product Principle) *The size of a union of m disjoint sets, each of size n, is mn.*

We now complete our discussion of Exercise 1.1-2. Lines 2 through 5 are executed once for each value of i from 1 to r. Each time those lines are executed, they are 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 demonstrates that thinking about whether the sum or product principle is appropriate for a problem can help to decompose the problem into easily-solvable pieces. If you can decompose the problem into smaller pieces 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 in lines 1 through 8 with the number of comparisons in the second loop in lines 10 through 12 (what two disjoint sets are we talking about here?). Further, the first loop makes $n(n+1)/2 - 1$ comparisons[2], 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.

Two element subsets

Often, there are several ways to solve a problem. We originally solved Exercise 1.1-1 by using the sum principal, but it is also possible to solve it using the product principal. Solving a problem two ways not only increases our confidence that we have found the correct solution, but it also allows us to make new connections and can yield valuable insight.

Consider the set of comparisons made by the entire execution of the code in this exercise. When $i = 1$, j takes on every value from 2 to n. When $i = 2$, j takes on every value from 3 to n. Thus, 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 whether i or j is smaller.) Thus the number of comparisons we make is the same as the number of two element subsets of the set $\{1, 2, \ldots, n\}$[3]. In how many ways can we choose two elements from this set? If we choose a first and second element, 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 the set of all such choices is the union of n sets

[2]To see why this is true, ask yourself first where the $n(n+1)/2$ comes from, and then why we subtracted one.

[3]The relationship between the set of comparisons and the set of two-element subsets of $\{1, 2, \ldots, n\}$ is an example of a bijection, an idea which will be examined more in Section 1.2.

of size $n-1$, one set for each first element. Thus it might appear that, by the product principle, 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 2 first and 5 second to get the ordered pair $(2,5)$, or we could choose 5 first and 2 second to get the ordered pair $(5,2)$. Since each pair of distinct elements of $\{1,2,\ldots,n\}$ can be ordered 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

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

Important Concepts, Formulas, and Theorems

1. *Set.* A *set* is a collection of objects. In a set order is not important. Thus the set $\{A,B,C\}$ is the same as the set $\{A,C,B\}$. An element either is or is not in a set; it cannot be in a set more than once, even if we have a description of a set which names that element more than once.

2. *Disjoint.* Two sets are called *disjoint* when they have no elements in common.

3. *Mutually disjoint sets.* A set of sets $\{S_1,\ldots,S_n\}$ is a family of *mutually disjoint sets*, if each two of the sets S_i are disjoint.

4. *Size of a set.* Given a set S, the size of S, denoted $|S|$, is the number of distinct elements in S.

5. *Sum Principle.* The size of a union of a family of mutually disjoint sets is the sum of the sizes of the sets. In other words, 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

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

6. *Partition of a set.* A partition of a set S is a set of mutually disjoint subsets (sometimes called blocks) of S whose union is S.

7. *Sum of first $n-1$ numbers.*

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

8. *Product Principle.* The size of a union of m disjoint sets, each of size n, is mn.

9. *Two element subsets.* $\binom{n}{2}$ stands for the number of two element subsets of an n element set and equals $n(n-1)/2$. $\binom{n}{2}$ is read as "n choose 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 $A[j] < A[j-1]$? Describe as succinctly as you can those lists that require this number of comparisons.

2. Five schools are going to send their baseball teams to a tournament, in which each team must play each other team exactly once. How many games are required?

3. Use notation similar to that in Equations 1.2 and 1.3 to rewrite the solution to Exercise 1.1-3 more algebraically.

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

5. In how many ways can you draw two cards from a deck of 52 cards.

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

7. In how many ways may a ten person club select a president and a secretary-treasurer from among its members?

8. In how many ways may a ten person club select a two person executive committee from among its members?

9. In how many ways may a ten person club select a president and a two person executive advisory board from among its members (assuming that the president is not on the advisory board)?

10. By using the formula for $\binom{n}{2}$ is is straightforward to show that

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

    However this proof just uses blind substitution and simplification. Find a more conceptual explanation of why this formula is true.

11. If M is an m element set and N is an n-element set, how many ordered pairs are there whose first member is in M and whose second member is in N?

12. 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.)

13. Now suppose that you decide to disagree with your mother in Exercise 12 and say that the order of the scoops does matter. How many different possible two-scoop cones are there?

14. 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?

15. 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?

1.2 Counting Lists, Permutations, and Subsets.

Using the Sum and Product Principles

Exercise 1.2-1 A password for a certain computer system is supposed to be between 4 and 8 characters long and composed of lower and/or upper case letters. How many passwords are possible? What counting principles did you use? Estimate the percentage of the possible passwords that have exactly four characters.

A good way to attack a counting problem is to ask if we could use either the sum principle or the product principle to simplify or completely solve it. Here that question might lead us to think about the fact that a password can have 4, 5, 6, 7 or 8 characters. The set of all passwords is the union of those with 4, 5, 6, 7, and 8 letters so the sum principle might help us. To write the problem algebraically, let P_i be the set of i-letter passwords and P be the set of all possible passwords. Clearly,

$$P = P_4 \cup P_5 \cup P_6 \cup P_7 \cup P_8 \ .$$

The P_i are mutually disjoint, and thus we can apply the sum principal to obtain

$$|P| = \sum_{i=4}^{8} |P_i| \ .$$

We still need to compute $|P_i|$. 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, $|P_i|$, 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

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

Although this is a nasty formula to evaluate by hand, 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 thus just a bit less than

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

which is $100/52^4$, or approximately .000014. Thus to five decimal places, only .00001% of the passwords have four letters. It is therefore much easier guess a password that we know has four letters than it is to guess one that has between 4 and 8 letters—roughly 7 million times easier!

In our solution to Exercise 1.2-1, 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. We could have, but it would have been clumsy and repetitive. For this reason we will state a second version of the product principle that we can derive from the version for unions of sets by using the idea of mathematical induction that we study in Chapter 4.

Version 2 of the *product principle* states:

Principle 1.4 (Product Principle, Version 2) *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 those lists,*

then there are $i_1 i_2 \cdots i_m$ lists in S.

Let's apply this version of the product principle to compute the number of m-letter passwords. Since an m-letter password is just a list of m letters, and since there are 52 different first elements of the password and 52 choices for each other position of the password, we have that $i_1 = 52,\ i_2 = 52, \ldots, i_m = 52$. Thus, this version of the product principle tells us immediately that the number of passwords of length m is $i_1 i_2 \cdots i_m = 52^m$.

Lists and functions

We have left a term undefined in our discussion of version 2 of the product principle, namely the word "list." A *list* of 3 things chosen from a set T consists of a first member t_1 of T, a second member t_2 of T, and a third member t_3 of T. If we rewrite the list in a different order, we get a different list. A list of k things chosen from T consists of a first member of T through a kth member of T. We can use the word "function," which you probably recall from algebra or calculus, to be more precise.

Recall that a function from a set S (called the *domain* of the function) to a set T (called the *range* of the function) is a relationship between the elements of S and the elements of T that relates exactly one element of T to each element of S. We use a letter like f to stand for a function and use $f(x)$ to stand for the one and only one element of T that the function relates to the element x of S. You are probably used to thinking of functions in terms of formulas like $f(x) = x^2$. We need to use formulas like this in algebra and calculus because the functions that you study in algebra and calculus have infinite sets of numbers as their domains and ranges. In discrete mathematics, functions often have finite sets as their domains and ranges, and so it is possible to describe a function by saying exactly what it is. For example

$$f(1) = \text{Sam},\ f(2) = \text{Mary},\ f(3) = \text{Sarah}$$

is a function that describes a list of three people. This suggests a precise definition of a list of k elements from a set T: A *list of k elements* from a set T is a function from $\{1, 2, \ldots, k\}$ to T.

Exercise 1.2-2 Write down all the functions from the two-element set $\{1, 2\}$ to the two-element set $\{a, b\}$.

Exercise 1.2-3 How many functions are there from a two-element set to a three element set?

Exercise 1.2-4 How many functions are there from a three-element set to a two-element set?

In Exercise 1.2-2 one thing that is difficult is to choose a notation for writing the functions down. We will use f_1, f_2, etc., to stand for the various functions we find. To describe a function f_i from $\{1,2\}$ to $\{a,b\}$ we have to specify $f_i(1)$ and $f_i(2)$. We can write

$$\begin{array}{ll} f_1(1)=a & f_1(2)=b \\ f_2(1)=b & f_2(2)=a \\ f_3(1)=a & f_3(2)=a \\ f_4(1)=b & f_4(2)=b \end{array}$$

We have simply written down the functions as they occurred to us. How do we know we have all of them? The set of all functions from $\{1,2\}$ to $\{a,b\}$ is the union of the functions f_i that have $f_i(1)=a$ and those that have $f_i(1)=b$. The set of functions with $f_i(1)=a$ has two elements, one for each choice of $f_i(2)$. Therefore by the product principle the set of all functions from $\{1,2\}$ to $\{a,b\}$ has size $2\cdot 2=4$.

To compute the number of functions from a two element set (say $\{1,2\}$) to a three element set, we can again think of using f_i to stand for a typical function. Then the set of all functions is the union of three sets, one for each choice of $f_i(1)$. Each of these sets has three elements, one for each choice of $f_i(2)$. Thus by the product principle we have $3\cdot 3=9$ functions from a two element set to a three element set.

To compute the number of functions from a three element set (say $\{1,2,3\}$) to a two element set, we observe that the set of functions is a union of four sets, one for each choice of $f_i(1)$ and $f_i(2)$ (as we saw in our solution to Exercise 1.2-2). But each of these sets has two functions in it, one for each choice of $f_i(3)$. Then by the product principle, we have $4\cdot 2=8$ functions from a three element set to a two element set.

A function f is called *one-to-one* or an *injection* if whenever $x\neq y$, $f(x)\neq f(y)$. Notice that the two functions f_1 and f_2 we gave in our solution of Exercise 1.2-2 are one-to-one, but f_3 and f_4 are not.

A function f is called *onto* or a *surjection* if every element y in the range is $f(x)$ for some x in the domain. Notice that the functions f_1 and f_2 in our solution of Exercise 1.2-2 are onto functions but f_3 and f_4 are not.

Exercise 1.2-5 Using two-element sets or three-element sets as domains and ranges, find an example of a one-to-one function that is not onto.

Exercise 1.2-6 Using two-element sets or three-element sets as domains and ranges, find an example of an onto function that is not one-to-one.

Notice that the function given by $f(1)=c$, $f(2)=a$ is an example of a function from $\{1,2\}$ to $\{a,b,c\}$ that is one-to one but not onto.

Notice that the function given by $f(1)=a$, $f(2)=b$, $f(3)=a$ is an example of a function from $\{1,2,3\}$ to $\{a,b\}$ that is onto but not one to one.

The Bijection Principle

Exercise 1.2-7 The loop below is part of a program to determine the number of triangles formed by n points in the plane.

```
(1)  trianglecount = 0
(2)  for i = 1 to n
(3)       for j = i + 1 to n
(4)            for k = j + 1 to n
(5)                 if points i, j, and k are not collinear
(6)                      trianglecount = trianglecount +1
```

How many times does the above code check three points to see if they are collinear in line 5?

In Exercise 1.2-7, we have a loop embedded in a loop that is embedded in another loop. Because the second loop, starting in line 3, begins with $j = i + 1$ and j increase up to n, and because the third loop, starting in line 4, begins with $k = j + 1$ and increases up to n, our code examines each triple of values i, j, k with $i < j < k$ exactly once. For example, if n is 4, then the triples (i, j, k) used by the algorithm, in order, are $(1, 2, 3)$, $(1, 2, 4)$, $(1, 3, 4)$, and $(2, 3, 4)$. Thus one way in which we might have solved Exercise 1.2-7 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? There is another fundamental principle that abstracts our concept of what it means for two sets to have the same size. Intuitively two sets have the same size if we can match up their elements in such a way that each element of one set corresponds to exactly one element of the other set. This description carries with it some of the same words that appeared in the definitions of functions, one-to-one, and onto. Thus it should be no surprise that one-to-one and onto functions are part of our abstract principle.

Principle 1.5 (Bijection Principle) *Two sets have the same size if and only if there is a one-to-one function from one set onto the other.*

Our principle is called the *bijection principle* because a one-to-one and onto function is called a *bijection*. Another name for a bijection is a *one-to-one correspondence*. A bijection from a set to itself is called a *permutation* of that set.

What is the bijection 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, or bijection, between the set of increasing triples and the set of three element sets.

k-element permutations of a set

Since counting increasing triples is equivalent to counting three-element subsets, we can count increasing triples by counting three-element subsets instead. We use a method similar to the one we used to compute the number of two-element subsets of a set. Recall that the first step was to compute the number of ordered pairs of distinct elements we could chose from the set $\{1, 2, \ldots, n\}$. So we now ask in how many ways may we choose an ordered triple of distinct elements from $\{1, 2, \ldots, n\}$, or more generally, in how many ways may we choose a list of k distinct elements from $\{1, 2, \ldots, n\}$. A list of k-distinct elements chosen from a set N is called a *k-element permutation* of N.[4]

How many 3-element permutations of $\{1, 2, \ldots, n\}$ can we make? Recall that a k-element permutation is a list of k distinct elements. There are n choices for the first number in the list. For each way of choosing the first element, there are $n-1$ choices for the second. For each choice of the first two elements, there are $n-2$ ways to choose a third (distinct) number, so by version 2 of the product principle, there are $n(n-1)(n-2)$ ways to choose the list of numbers. For example, if n is 4, the three-element permutations of $\{1, 2, 3, 4\}$ are

$$\begin{aligned} L \;=\; & \{123, 124, 132, 134, 142, 143, 213, 214, 231, 234, 241, 243, \\ & 312, 314, 321, 324, 341, 342, 412, 413, 421, 423, 431, 432\}. \end{aligned} \tag{1.4}$$

There are indeed $4 \cdot 3 \cdot 2 = 24$ lists in this set. Notice that we have listed the lists in the order that they would appear in a dictionary (assuming we treated numbers as we treat letters). This ordering of lists is called the *lexicographic ordering*.

A general pattern is emerging. To compute the number of k-element permutations of the set $\{1, 2, \ldots, n\}$, we recall that they are lists and note that we have n choices for the first element of the list, and regardless of which choice we make, we have $n-1$ choices for the second element of the list, and more generally, given the first $i-1$ elements of a list we have $n-(i-1) = n-i+1$ choices for the ith element of the list. Thus by version 2 of the product principle, we have $n(n-1)\cdots(n-k+1)$, which is the first k terms of $n!$, ways to choose a k-element permutation of $\{1, 2, \ldots, n\}$. There is a very handy notation for this product first suggested by Don Knuth. We use $n^{\underline{k}}$ to stand for $n(n-1)\cdots(n-k+1)$, and call it the *kth falling factorial power of n*. We can summarize our observations in a theorem.

Theorem 1.1 *The number k-element permutations of an n-element set is*

$$n^{\underline{k}} = n(n-1)\cdots(n-k+1) = n!/(n-k)! \; .$$

Counting subsets of a set

We now return to the question of counting the number of three element subsets of a $\{1, 2, \ldots, n\}$. We use $\binom{n}{3}$, which we read as "n choose 3" to stand for the number of three element subsets of

[4] In particular a k-element permutation of $\{1, 2, \ldots k\}$ is a list of k distinct elements of $\{1, 2, \ldots, k\}$, which, by our definition of a list is a function from $\{1, 2, \ldots, k\}$ to $\{1, 2, \ldots, k\}$. This function must be one-to-one since the elements of the list are distinct. Since there are k distinct elements of the list, every element of $\{1, 2, \ldots, k\}$ appears in the list, so the function is onto. Therefore it is a bijection. Thus our definition of a permutation of a set is consistent with our definition of a k-element permutation in the case where the set is $\{1, 2, \ldots, k\}$.

$\{1, 2, \ldots, n\}$, or more generally of any n-element set. We have just carried out the first step of computing $\binom{n}{3}$ by counting the number of three-element permutations of $\{1, 2, \ldots, n\}$.

Exercise 1.2-8 Let L be the set of all three-element permutations of $\{1, 2, 3, 4\}$, as in Equation 1.4. How many of the lists (permutations) in L are lists of the 3 element set $\{1, 3, 4\}$? What are these lists?

We see that this set appears in L as 6 different lists: 134, 143, 314, 341, 413, and 431. In general given three different numbers with which to create a list, 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 version 2 of the product principle once again, there are $3 \cdot 2 \cdot 1 = 6$ ways to make the list.

Since there are $n(n-1)(n-2)$ permutations of an n-element set, and each three-element subset appears in exactly 6 of these lists, the number of three-element permutations is six times the number of three element subsets. That is, $n(n-1)(n-2) = \binom{n}{3} \cdot 6$. Whenever we see that one number that counts something is the product of two other numbers that count something, we should expect that there is an argument using the product principle that explains why. Thus we should be able to see how to break the set of all 3-element permutations of $\{1, 2, \ldots, n\}$ into either 6 disjoint sets of size $\binom{n}{3}$ or into $\binom{n}{3}$ subsets of size six. Since we argued that each three element subset corresponds to six lists, we have described how to get a set of six lists from one three-element set. Two different subsets could never give us the same lists, so our sets of three-element lists are disjoint. In other words, we have divided the set of all three-element permutations into $\binom{n}{3}$ mutually sets of size six. In this way the product principle does explain why $n(n-1)(n-2) = \binom{n}{3} \cdot 6$. By division we get that we have

$$\binom{n}{3} = n(n-1)(n-2)/6$$

three-element subsets of $\{1, 2, \ldots, n\}$. For $n = 4$, the number is $4(3)(2)/6 = 4$. These sets are $\{1, 2, 3\}$, $\{1, 2, 4\}$, $\{1, 3, 4\}$, and $\{2, 3, 4\}$. It is straightforward to verify that each of these sets appears 6 times in L, as 6 different lists.

Essentially the same argument gives us the number of k-element subsets of $\{1, 2, \ldots, n\}$. We denote this number by $\binom{n}{k}$, and read it as "n choose k." Here is the argument: the set of all k-element permutations of $\{1, 2, \ldots, n\}$ can be partitioned into $\binom{n}{k}$ disjoint blocks[5], each block consisting of all k-element permutations of a k-element subset of $\{1, 2, \ldots, n\}$. But the number of k-element permutations of a k-element set is $k!$, either by version 2 of the product principle or by Theorem 1.1. Thus by version 1 of the product principle we get the equation

$$n^{\underline{k}} = \binom{n}{k} k!.$$

Division by $k!$ gives us our next theorem.

Theorem 1.2 *For integers n and k with $0 \leq k \leq n$, the number of k element subsets of an n element set is*

$$\frac{n^{\underline{k}}}{k!} = \frac{n!}{k!(n-k)!}$$

[5]using the language introduced for partitions of sets in Section 1.1

Proof: The proof is 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 $0! = 1$.[6]) The equality in the theorem comes from the definition of $n^{\underline{k}}$ ■

Another notation for the numbers $\binom{n}{k}$ is $C(n, k)$. Thus we have that

$$C(n,k) = \binom{n}{k} = \frac{n!}{k!(n-k)!}. \tag{1.5}$$

These numbers are called *binomial coefficients* for reasons that will become clear later.

Important Concepts, Formulas, and Theorems

1. *List.* A list of k items chosen from a set X is a function from $\{1, 2, \ldots k\}$ to X.
2. *Lists versus sets.* In a list, the order in which elements appear in the list matters, and an element may appear more than once. In a set, the order in which we write down the elements of the set does not matter, and an element can appear at most once.
3. *Product Principle, Version 2.* If a set S of lists of length m has the properties that
   (a) There are i_1 different first elements of lists in S, and
   (b) 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 those lists,

   then there are $i_1 i_2 \cdots i_m$ lists in S.
4. *Function.* 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. The same element of T may be related to different members of S.
5. *Onto, Surjection* A function f from a set S to a set T is *onto* if for each element $y \in T$, there is at least one $x \in S$ such that $f(x) = y$. An onto function is also called a *surjection.*
6. *One-to-one, Injection.* A function f from a set S to a set T is *one-to-one* if, for each $x \in S$ and $y \in S$ with $x \neq y$, $f(x) \neq f(y)$. A one-to-one function is also called an *injection.*
7. *Bijection, One-to-one correspondence.* A function from a set S to a set T is a *bijection* if it is both one-to-one and onto. A bijection is sometimes called a *one-to-one correspondence.*
8. *Permutation.* A one-to-one function from a set S to S is called a *permutation* of S.
9. *k-element permutation.* A *k-element permutation* of a set S is a list of k distinct elements of S.
10. *k-element subsets. n choose k. Binomial Coefficients.* For integers n and k with $0 \leq k \leq n$, the number of k element subsets of an n element set is $n!/k!(n-k)!$. 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.*

[6] There are many reasons why 0! is defined to be one; making the formula for $\binom{n}{k}$ work out is one of them.

11. The number of k-element permutations of an n-element set is

$$n^{\underline{k}} = n(n-1)\cdots(n-k+1) = n!/(n-k)!.$$

12. When we have a formula to count something and the formula expresses the result as a product, it is useful to try to understand whether and how we could use the product principle to prove the formula.

Problems

1. 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?

2. In how many ways can we pass out k distinct pieces of fruit to n children (with no restriction on how many pieces of fruit a child may get)?

3. Write down all the functions from the three-element set $\{1, 2, 3\}$ to the set $\{a, b\}$. Indicate which functions, if any, are one-to-one. Indicate which functions, if any, are onto.

4. Write down all the functions form the two element set $\{1, 2\}$ to the three element set $\{a, b, c\}$ Indicate which functions, if any, are one-to-one. Indicate which functions, if any, are onto.

5. 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?

6. Assuming $k \leq n$, in how many ways can we pass out k distinct pieces of fruit to n children if each child may get at most one? What is the number if $k > n$? Assume for both questions that we pass out all the fruit.

7. Assume $k \leq n$, in how many ways can we pass out k identical pieces of fruit to n children if each child may get at most one? What is the number if $k > n$? Assume for both questions that we pass out all the fruit.

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

9. 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?

10. (This problem is for students who are working on the relationship between k-element permutations and k-element subsets.) Write down all three element permutations of the five element set $\{1, 2, 3, 4, 5\}$ in lexicographic order. Underline those that correspond to the set

$\{1, 3, 5\}$. Draw a rectangle around those that correspond to the set $\{2, 4, 5\}$. How many three-element permutations of $\{1, 2, 3, 4, 5\}$ correspond to a given 3-element set? How many three-element subsets does the set $\{1, 2, 3, 4, 5\}$ have?

11. In how many ways may a class of twenty students choose a group of three students from among themselves to go to the professor and explain that the three-hour labs are actually taking ten hours?

12. We are choosing participants for a panel discussion allowing on allowing alcohol on campus. We have to choose four administrators from a group of ten administrators and four students from a group of twenty students. In how many ways may we do this?

13. 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 chosen from a group of ten administrators and four students chosen from a group of twenty students. In how many ways may we choose and 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 choose and list them if we must alternate students and administrators?

14. In the local ice cream shop, you may get a sundae with two scoops of ice cream from 10 flavors (in accordance with your mother's rules from Problem 12 in Section 1.1, the way the scoops sit in the dish does not matter), 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?

15. 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(in accordance with your mother's rules from Problem 12 in Section 1.1, the way the scoops sit in the dish does not matter) ?

16. A tennis club has $2n$ members. We want to pair up the members by twos for singles matches. In how many ways may we pair up all the members of the club? Suppose that in addition to specifying who plays whom, for each pairing we say who serves first. Now in how many ways may we specify our pairs?

17. A basketball team has 12 players. However, only five players play at any given time during a game. In how may ways may the coach choose the five players? To be more realistic, the five players playing a game normally consist of two guards, two forwards, and one center. If there are five guards, four forwards, and three centers on the team, in how many ways can the coach choose two guards, two forwards, and one center? What if one of the centers is equally skilled at playing forward?

18. Explain why a function from an n-element set to an n-element set is one-to-one if and only if it is onto.

19. The function g is called an *inverse* to the function f if the domain of g is the range of f, if $g(f(x)) = x$ for every x in the domain of f and if $f(g(y)) = y$ for each y in the range of f.

    (a) Explain why a function is a bijection if and only if it has an inverse function.

    (b) Explain why a function that has an inverse function has only one inverse function.

1.3 Binomial Coefficients

In this section, we will explore various properties of binomial coefficients.

Pascal's Triangle

Table 1 contains the values of the binomial coefficients $\binom{n}{k}$ for $n = 0$ to 6 and all relevant k values. The table begins with a 1 for $n = 0$ and $k = 0$, because 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. However, since there are no subsets of an n-element set that have size larger than n, it is natural to define $\binom{n}{k}$ to be zero when $k > n$, and so we define $\binom{n}{k}$ to be zero when $k > n$. Thus we could could fill in the empty places in the table with zeros. The table is easier to read if we don't fill in the empty spaces, so we just remember that they are zero.

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

Exercise 1.3-1 What general properties of binomial coefficients do you see in Table 1.1

Exercise 1.3-2 What is the next row of the table of binomial coefficients?

Several properties of binomial coefficients are apparent in Table 1.1. Each row begins with a 1, because $\binom{n}{0}$ is always 1, as it must be because there is just one subset of an n-element set with 0 elements, namely the empty set. Similarly, each row ends with a 1, because an n-element set S has just one n-element subset, namely S itself. Each row increases at first, and then decreases. 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.

You may know a method for creating Pascal's triangle that does not involve computing binomial coefficients, but rather creates each row from the row above. Each entry in Table 1.2, 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*, and it gives another way to compute binomial coefficients without doing the multiplying and dividing in equation 1.5. If we wish to compute many binomial coefficients, the Pascal relationship often yields a more efficient way to do so. Once the coefficients in a row have been computed, the coefficients in the next row can be computed using only one addition per entry.

Table 1.2: Pascal's Triangle

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

We now verify that the two methods for computing Pascal's triangle always yield the same result. In order to do so, we need an algebraic statement of the Pascal Relationship. In Table 1.1, 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

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}, \tag{1.6}$$

whenever $n > 0$ and $0 < k < n$. Notice that 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.

A proof using sets

From Theorem 1.2 and Equation 1.5, we know that the expression $\binom{n}{k}$ is the number of k-element subsets of an n element set. Each of the three terms in Equation 1.6 therefore represents the number of subsets of a particular size chosen from an appropriately sized set. In particular, 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 subsets of an $(n-1)$-element set. We should, therefore, be able to explain the relationship between these three quantities using the sum principle. This explanation will provide a proof, just as valid a proof as an algebraic derivation. Often, a proof using subsets will be less tedious, and will yield more insight into the problem at hand.

Before giving such a proof in Theorem 1.3 below, we give an example. Suppose $n = 5$, $k = 2$. Equation 1.6 says that

$$\binom{5}{2} = \binom{4}{1} + \binom{4}{2}. \tag{1.7}$$

Because the numbers are small, it is simple to verify this by using the formula for binomial coefficients, but let us instead consider subsets of a 5-element set. Equation 1.7 says that the number of 2 element subsets of a 5 element set is equal to the number of 1 element subsets of a 4 element set plus the number of 2 element subsets of a 4 element set. But to apply the sum principle, we would need to say something stronger. To apply the sum principle, we should be able to partition the set of 2 element subsets of a 5 element set into 2 disjoint sets, one of which

has the same size as the number of 1 element subsets of a 4 element set and one of which has the same size as the number of 2 element subsets of a 4 element set. Such a partition provides a proof of Equation 1.7. Consider now the set $S = \{A, B, C, D, E\}$. The set of two element subsets is

$$S_1 = \{\{A, B\}, \{AC\}, \{A, D\}, \{A, E\}, \{B, C\}, \{B, D\}, \{B, E\}, \{C, D\}, \{C, E\}, \{D, E\}\}.$$

We now partition S_1 into 2 blocks, S_2 and S_3. S_2 contains all sets in S_1 that do contain the element E, while S_3 contains all sets in S_1 that do not contain the element E. Thus,

$$S_2 = \{\{AE\}, \{BE\}, \{CE\}, \{DE\}\}$$

and

$$S_3 = \{\{AB\}, \{AC\}, \{AD\}, \{BC\}, \{BD\}, \{CD\}\}.$$

Each set in S_2 must contain E and then contains one other element from S. Since there are 4 other elements in S that we can choose along with E, $|S_2| = \binom{4}{1}$. Each set in S_3 contains 2 elements from the set $\{A, B, C, D\}$, and thus there are $\binom{4}{2}$ ways to choose such a subset. But $S_1 = S_2 \cup S_3$ and S_2 and S_3 are disjoint, and so, by the sum principle, Equation 1.7 must hold.

We now give a proof for general n and k.

Theorem 1.3 *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. As in our example, we will apply the sum principle. 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, the sum principle shows that the number of elements of S is $\binom{n-1}{k-1} + \binom{n-1}{k}$. ■

Notice that in our proof, we used a bijection that we did not explicitly describe. Namely, there is a bijection f between S_3 (the k-element sets of S that contain x_n) and the $(k-1)$-element subsets of S'. For any subset K in S_3, We let $f(K)$ be the set we obtain by removing x_n from K. It is immediate that this is a bijection, and so the bijection principle tells us that the size of S_3 is the size of the set of all subsets of S'.

The Binomial Theorem

Exercise 1.3-3 What is $(x+y)^3$? 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 Theorem** states that

Theorem 1.4 (Binomial Theorem) *For any integer $n \geq 0$*

$$(x+y)^n = \binom{n}{0}x^n + \binom{n}{1}x^{n-1}y + \binom{n}{2}x^{n-2}y^2 + \cdots + \binom{n}{n-1}xy^{n-1} + \binom{n}{n}y^n, \tag{1.8}$$

or in summation notation,

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

Unfortunately when most people first see this theorem, they do not have the tools to see easily why it is true. Armed with our new methodology of using subsets to prove algebraic identities, we can give a proof of this theorem.

Let us begin by considering the example $(x+y)^3$ which by the binomial theorem is

$$\begin{aligned}(x+y)^3 &= \binom{3}{0}x^3 + \binom{3}{1}x^2y + \binom{3}{2}xy^2 + \binom{3}{3}y^3 &(1.9)\\ &= x^3 + 3x^2y + 3xy^2 + x^3 \ . &(1.10)\end{aligned}$$

Suppose that we did not know the binomial theorem but still wanted to compute $(x+y)^3$. Then we would write out $(x+y)(x+y)(x+y)$ and perform the multiplication. Probably we would multiply the first two terms, obtaining $x^2+2xy+y^2$, and then multiply this expression by $x+y$. Notice that by applying distributive laws you get

$$(x+y)(x+y) = (x+y)x + (x+y)y = xx + xy + yx + y. \tag{1.11}$$

We could use the commutative law to put this into the usual form, but let us hold off for a moment so we can see a pattern evolve. To compute $(x+y)^3$, we can multiply the expression on the right hand side of Equation 1.11 by $x+y$ using the distributive laws to get

$$\begin{aligned}(xx+xy+yx+yy)(x+y) &= (xx+xy+yx+yy)x + (xx+xy+yx+yy)y &(1.12)\\ &= xxx + xyx + yxx + yxx + xsy + xyy + yxy + yyy &(1.13)\end{aligned}$$

Each of these 8 terms that we got from the distributive law may be thought of as a product of terms, one from the first binomial, one from the second binomial, and one from the third binomial. Multiplication is commutative, so many of these products are the same. In fact, we have one xxx or x^3 product, three products with two x's and one y, or x^2y, three products with one x and two y's, or xy^2 and one product which becomes y^3. Now look at Equation 1.9, which summarizes this process. There are $\binom{3}{0} = 1$ way to choose a product with 3 x's and 0 y's, $\binom{3}{1} = 3$ way to choose a product with 2 x's and 1 y, etc. Thus we can understand the binomial theorem

as counting the subsets of our binomial factors from which we choose a y-term to get a product with k y's in multiplying a string of n binomials.

Essentially the same explanation gives us a proof of the binomial theorem. Note that when we multiplied out three factors of $(x+y)$ using the distributive law but not collecting like terms, we had a sum of eight products. Each factor of $(x+y)$ doubles the number of summands. Thus when we apply the distributive law as many times as possible (without applying the commutative law and collecting like terms) to a product of n binomials all equal to (x+y), we get 2^n summands. Each summand is a product of a length n list of x's and y's. In each list, the ith entry comes from the ith binomial factor. A list that becomes $x^{n-k}y^k$ when we use the commutative law will have a y in k of its places and an x in the remaining places. The number of lists that have a y in k places is thus the number of ways to select k binomial factors to contribute a y to our list. But the number of ways to select k binomial factors from n binomial factors is simply $\binom{n}{k}$, and so that is the coefficient of $x^{n-k}y^k$. This proves the binomial theorem.

Applying the Binomial Theorem to the remaining questions in Exercise 1.3-3 gives us

$$\begin{aligned}(x+1)^4 &= x^4+4x^3+6x^2+4x+1\\ (2+y)^4 &= 16+32y+24y^2+8y^3+y^4 \text{ and}\\ (x+y)^4 &= x^4+4x^3y+6x^2y^2+4xy^3+y^4.\end{aligned}$$

Labeling and trinomial coefficients

Exercise 1.3-4 Suppose that I have k labels of one kind and $n-k$ labels of another. In how many different ways may I apply these labels to n objects?

Exercise 1.3-5 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.

Exercise 1.3-6 What is the coefficient of $x^{k_1}y^{k_2}z^{k_3}$ in $(x+y+z)^n$?

Exercise 1.3-4 and Exercise 1.3-5 can be thought of as immediate applications of binomial coefficients. For Exercise 1.3-4, 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 1.3-5, there are $\binom{n}{k_1}$ ways to choose the k_1 objects that get the first kind of label, and then there are $\binom{n-k_1}{k_2}$ ways to choose the objects that get the second kind of label. After that, the remaining $k_3 = n-k_1-k_2$ objects get the third kind of label. The total number of labellings is thus, by the product principle, the product of the two binomial coefficients, which simplifies as follows.

$$\begin{aligned}\binom{n}{k_1}\binom{n-k_1}{k_2} &= \frac{n!}{k_1!(n-k_1)!}\frac{(n-k_1)!}{k_2!(n-k_1-k_2)!}\\ &= \frac{n!}{k_1!k_2!(n-k_1-k_2)!}\\ &= \frac{n!}{k_1!k_2!k_3!}\ .\end{aligned}$$

A more elegant approach to Exercise 1.3-4, Exercise 1.3-5, and other related problems appears in the next section.

Exercise 1.3-6 shows how Exercise 1.3-5 applies to computing powers of trinomials. In expanding $(x+y+z)^n$, we think of writing down n copies of the trinomial $x+y+z$ side by side, and applying the distributive laws until we have a sum of terms each of which is a product of x's, y's and z's. How many such terms do we have with k_1 x's, k_2 y's and k_3 z's? Imagine choosing x from some number k_1 of the copies of the trinomial, choosing y from some number k_2, and z from the remaining k_3 copies, 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. Notice that this requires that $k_3 = n - k_1 - k_2$. By analogy with our notation for a binomial coefficient, we define the *trinomial coefficient* $\binom{n}{k_1,k_2,k_3}$ to be $\frac{n!}{k_1!k_2!k_3!}$ if $k_1+k_2+k_3 = n$ and 0 otherwise. Then $\binom{n}{k_1,k_2,k_3}$ is the coefficient of $x^{k_1}y^{k_2}z^{k_3}$in $(x+y+z)^n$. This is sometimes called the *trinomial theorem*.

Important Concepts, Formulas, and Theorems

1. *Pascal Relationship.* The Pascal Relationship says that

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

   whenever $n > 0$ and $0 < k < n$.

2. *Pascal's Triangle.* Pascal's Triangle is the triangular array of numbers we get by putting ones in row n and column 0 and in row n and column n of a table for every positive integer n and then filling the remainder of the table by letting the number in row n and column j be the sum of the numbers in row $n-1$ and columns $j-1$ and j whenever $0 < j < n$.

3. *Binomial Theorem.* The **Binomial Theorem** states that for any integer $n \geq 0$

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

   or in summation notation,

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

4. *Labeling.* The number of ways to apply k labels of one kind and $n-k$ labels of another kind to n objects is $\binom{n}{k}$.

5. *Trinomial coefficient.* We define the *trinomial coefficient* $\binom{n}{k_1,k_2,k_3}$ to be $\frac{n!}{k_1!k_2!k_3!}$ if $k_1+k_2+k_3 = n$ and 0 otherwise.

6. *Trinomial Theorem.* The coefficient of $x^iy^jz^k$ in $(x+y+z)^n$ is $\binom{n}{i,j,k}$.

Problems

1. Find $\binom{12}{3}$ and $\binom{12}{9}$. What can you say in general about $\binom{n}{k}$ and $\binom{n}{n-k}$?

2. Find the row of the Pascal triangle that corresponds to $n = 8$.

3. Find the following

   a. $(x+1)^5$
   b. $(x+y)^5$
   c. $(x+2)^5$
   d. $(x-1)^5$

4. 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.

5. 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?

6. 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. This relationship is called the Multinomial Theorem.

7. Give a bijection that proves your statement about $\binom{n}{k}$ and $\binom{n}{n-k}$ in Problem 1 of this section.

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. Prove Equation 1.6 by plugging in the formula for $\binom{n}{k}$.

13. Give two proofs that
$$\binom{n}{k} = \binom{n}{n-k}.$$

14. Give at least two proofs that
$$\binom{n}{k}\binom{k}{j} = \binom{n}{j}\binom{n-j}{k-j}.$$

15. Give at least two proofs that
$$\binom{n}{k}\binom{n-k}{j} = \binom{n}{j}\binom{n-j}{k}.$$

16. 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 $\binom{9}{6}$.

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

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

19. 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.

1.4 Equivalence Relations and Counting

Counting using Equivalence Classes

Consider again the example from Section 1.2 in which we wanted to count the number of 3 element subsets of a four element set. To do so, we first formed all possible lists of $k = 3$ distinct elements chosen from an $n = 4$ element set. (See Equation 1.4.) The number of lists of k distinct elements is $n^{\underline{k}} = n!/(n-k)!$. We then observed that two lists are equivalent as sets, if one can be obtained by rearranging (or "permuting") the other. This process divides the lists up into classes, called *equivalence classes*, each of size $k!$. Returning to our example in Section 1.2, we noted that one such equivalence class was

$$\{134, 143, 314, 341, 413, 431\} .$$

The other three are

$$\{234, 243, 324, 342, 423, 432\} ,$$

$$\{132, 123, 312, 321, 213, 231\} ,$$

and

$$\{124, 142, 214, 241, 412, 421\} .$$

The product principle told us that if q is the number of such equivalence class, if each equivalence class has $k!$ elements, and the entire set of lists has $n!/(n-k)!$ element, then we must have that

$$qk! = n!/(n-k)! .$$

Dividing, we solve for q and get an expression for the number of k element subsets of an n element set. In fact, this is how we proved Theorem 1.2.

A principle that helps in learning and understanding mathematics is that if we have a mathematical result that shows a certain symmetry, it often helps our understanding to find a proof that reflects this symmetry. We call this the *Symmetry Principle*. The proof above does not account for the symmetry of the $k!$ term and the $(n-k)!$ term in the expression $\frac{n!}{k!(n-k)!}$. This symmetry arises because choosing a k element subset is equivalent to choosing the $(n-k)$-element subset of elements we don't want. In Exercise 1.4-4, 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 blue and $n-k$ red labels to n elements. From each labeling, we can create a number of lists, using the convention of listing the k blue elements first and the remaining $n-k$ red elements last. For example, suppose we are considering the number of ways to label 3 elements blue (and 2 red) from a five element set $\{A, B, C, D, E\}$. Consider the particular labelling in which A, B, and D are labelled blue and C and E are labelled red. Which lists correspond to this labelling? They are

$$\begin{array}{cccccc} ABDCE & ABDEC & ADBCE & ADBEC & BADCE & BADEC \\ BDACE & BDAEC & DABCE & DABEC & DBACE & DBAEC \end{array}$$

that is, all lists in which A, B, and D precede C and E. Since there are 3! ways to arrange A, B, and D, and 2! ways to arrange C and E, by the product principal, there are $3!2! = 12$ lists in which A, B, and D precede C and E. For each of the q ways to construct a labelling, we could find a similar set of 12 lists that are associated with that labelling. Since *every* possible list of 5 elements will appear exactly once via this process, and since there are $5! = 120$ five-element lists overall, we must have by the product principle that

$$q \cdot 12 = 120, \tag{1.14}$$

or that $q = 10$. This agrees with our previous calculations of $\binom{5}{3} = 10$ for the number of ways to label 5 items so that 3 are blue and 2 are red.

Generalizing, we let q be the number of ways to label n objects with k blue labels and $n - k$ red labels. To create the lists associated with a labelling, we list the blue elements first and then the red elements. We can mix the k blue elements among themselves, and we can mix the $n - k$ red elements among themselves, giving us $k!(n - k)!$ lists consisting of first the elements with a blue label followed by the elements with a red label. Since we can choose to label any k elements blue, each of our lists of n distinct elements arises from some labelling in this way. Each such list arises from only one labelling, because two different labellings will have a different first k elements in any list that corresponds to the labelling. Each such list arises only once from a given labelling, because two different lists that correspond to the same labelling differ by a permutation of the first k places or the last $n - k$ places or both. Therefore, by the product principle, $qk!(n - k)!$ is the number of lists we can form with n distinct objects, and this must equal $n!$. This gives us

$$qk!(n - k)! = n!,$$

and division gives us our original formula for q. Recall that our proof of the formula we had in Exercise 1.4-5 did not explain why the product of three factorials appeared in the denominator, it simply proved the formula was correct. With this idea in hand, we could now explain *why* the product in the denominator of the formula in Exercise 1.4-5 for the number of labellings with three labels is what it is, and could generalize this formula to four or more labels.

Equivalence Relations

The process above divided the set of all $n!$ lists of n distinct elements into classes (another word for sets) of lists. In each class, all the lists are mutually equivalent, with respect to labeling with two labels. More precisely, 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. (This is another example of abstraction.)

A relationship that divides a set up into mutually exclusive classes is called an **equivalence relation**.[7] Thus, if

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

[7]The usual mathematical approach to equivalence relations, which we shall discuss in the exercises, is 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.

and $S_i \cap S_j = \emptyset$ for all i and j with $i \neq j$, then the relationship that says any two elements $x \in S$ and $y \in S$ 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, as we noted in Section 1.1 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 $\{a, c\}$, $\{d, g\}$, $\{b, e, f\}$. 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. A slightly less boring equivalence relation is that two letters are equivalent if typographically, their top and bottom are at the same height. This give the partition $\{a, c, e\}$, $\{b, d\}$, $\{f\}$, $\{g\}$.

Exercise 1.4-1 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 1.4-1, 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 $\{1, 4, 7, 10\}$. 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 $\{2, 5, 8, 11\}$. Therefore the relationship is an equivalence relation.

The quotient principle

In Exercise 1.4-1 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 **Quotient Principle**.

Theorem 1.5 (Quotient principle) *If an equivalence relation on a p-element set S has q classes each of size r, then $q = p/r$.*

Proof: By the product principle, $p = qr$, and so $q = p/r$. ■

Another statement of the quotient principle that uses the idea of a partition is

Principle 1.6 (Quotient principle.) *If we can partition a set of size p into q blocks of size r, then $q = p/r$.*

Returning to our example of 3 blue and 2 red labels, $s = 5! = 120$, $t = 12$ and so by Theorem 1.5,

$$m = \frac{s}{t} = \frac{120}{12} = 10 \ .$$

Equivalence class counting

We now give several examples of the use of Theorem 1.5.

Exercise 1.4-2 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[8]. (The person to the right of the person in position 4 of the list is the person in position 1). We will use Theorem 1.5 to count the number of possible ways to seat the players. We will take our set S to be the set of all 4-element permutations of the four people, i.e., the set of all lists of the four people.

(a) How many lists are equivalent to a given one?

(b) What are the lists equivalent to ABCD?

(c) Is the relationship of equivalence an equivalence relation?

(d) Use Theorem 1.5 to compute the number of equivalence classes, and hence, the number of possible ways to seat the players.

Exercise 1.4-3 We wish to count the number of ways to attach n distinct beads to the corners of a regular n-gon (or string them on a necklace). We say that 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 does this exercise differ from the previous exercise?
- How many lists are in an equivalence class?
- How many equivalence classes are there?

In Exercise 1.4-2, suppose we have named the places at the table north, east, south, and west. Given a list we get an equivalent one in two steps. First we observe that we have four choices of people to sit in the north position. Then there is one person who can sit to this person's right, one who can be next on the right, and one who can be the following on on the right, all determined by the original list. Thus there are exactly four lists equivalent to a given one, including that given one. The lists equivalent to ABCD are ABCD, BCDA, CDAB, and DABC. This shows that two lists are equivalent if and only if we can get one from the other by moving everyone the same number of places to the right around the table (or we can get one from the other moving everyone the same number of places to the left around the table). From this we can see we have an equivalence relation, because each list is in one of these sets of four equivalent lists, and if two lists are equivalent, they are right or left shifts of each other, and we've just observed that all right and left shifts of a given list are in the same class. This means our relationship divides the set of all lists of the four names into equivalence classes each of size four. There are a total of $4! = 24$ lists of four distinct names, and so by Theorem 1.5 we have $4!/4 = 3! = 6$ seating arrangements.

Exercise 1.4-3 is similar in many ways to Exercise 1.4-2, 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 now two lists are equivalent if each bead is adjacent to exactly the same beads in both of them. Suppose we number the vertices of our polygon as 1 through n clockwise. Given a list, we can count the equivalent lists as follows. We have n choices for which bead to put in position 1. Then either of the two beads adjacent to it in the given list can go in position 2. But now, only one bead can go in position 3, because the other bead adjacent to position 2 is already in position

[8]Think of the four places at the table as being called north, east, south, and west, or numbered 1-4. Then we get a list by starting with the person in the north position (position 1), then the person in the east position (position 2) and so on clockwise

1. We can continue in this way to fill in the rest of the list. For example, with $n = 4$, the lists ABCD, ADBC, DABC DCBA, CDBA, CBAD, BCDA,and BADC are all equivalent. Notice the first, third , fifth and seventh lists are obtained by shifting the beads around the polygon, as are the second, fourth, sixth and eighth. Also note that the fourth list is the reversal of the first, the fifth is the reversal of the second, and so on. Rotating a necklace in space corresponds to shifting the letters in the list. Flipping a necklace over in space corresponds to reversing the order of a list. There will always be $2n$ lists we can get by shifting and reversing shifts of a list. The lists equivalent to a given one consist of everything we can get from the given list by rotations and reversals. Thus the relationship of every bead being adjacent to the same beads divides the set of lists of beads into disjoint sets. These sets, which have size $2n$, are the equivalence classes of our equivalence relation. Since there are $n!$ lists, Theorem 1.5 says there are

$$\frac{n!}{2n} = \frac{(n-1)!}{2}$$

bead arrangements.

Multisets

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 $\{\{f, o, o, r\}\}$. We use the double brackets to distinguish a multiset from a set. We can specify a *multiset* chosen from a set S by saying how many times each of its elements occurs. If S is the set of English letters, 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. In a multiset, order is not important, that is the multiset $\{\{r, o, f, o\}\}$ is equivalent to the multiset $\{\{f, o, o, r\}\}$. We know that this is the case, because they each have the same multiplicity function. We would like to say that the size of $\{\{f, o, o, r\}\}$ is 4, so we define the*size* of a multiset to be the sum of the multiplicities of its elements.

Exercise 1.4-4 Explain how placing k identical books onto the n shelves of a bookcase can be thought of as giving us a k-element multiset of the shelves of the bookcase. Explain how distributing k identical apples to n children can be thought of as giving us a k-element multiset of the children.

In Exercise 1.4-4 we can think of the multiplicity of a bookshelf as the number of books it gets and the multiplicity of a child as the number of apples the child gets. In fact, this idea of distribution of identical objects to distinct recipients gives a great mental model for a multiset chosen from a set S. Namely, to determine a k-element multiset chosen from S form S, we "distribute" k identical objects to the elements of S and the number of objects an element x gets is the multiplicity of x.

Notice that it makes no sense to ask for the number of multisets we may choose from a set with n elements, because $\{\{A\}\}$, $\{\{A, A\}\}$, $\{\{A, A, A\}\}$, and so on are infinitely many multisets chosen from the set $\{A\}$. However it does make sense to ask for the number of k-element multisets we can choose from an n-element set. What strategy could we employ to figure out this number? To count k-element subsets, we first counted k-element permutations, and then divided by the number of different permutations of the same set. Here we need an analog of permutations that

allows repeats. A natural idea is to consider lists with repeats. After all, one way to describe a multiset is to list it, and there could be many different orders for listing a multiset. However the two element multiset $\{\{A, A\}\}$ can be listed in just one way, while the two element multiset $\{\{A, B\}\}$ can be listed in two ways. When we counted k-element subsets of an n-element set by using the quotient principle, it was essential that each k-element set corresponded to the same number (namely $k!$) of permutations (lists), because we were using the reasoning behind the quotient principle to do our counting here. So if we hope to use similar reasoning, we can't apply it to lists because different k-element multisets can correspond to different numbers of lists.

Suppose, however, we could count the number of ways to arrange k distinct books on the n shelves of a bookcase. We can still think of the multiplicity of a shelf as being the number of books on it. However, many different arrangements of distinct books will give us the same multiplicity function. In fact, any way of mixing the books up among themselves that does not change the number of books on each shelf will give us the same multiplicities. But the number of ways to mix the books up among themselves is the number of permutations of the books, namely $k!$. Thus it looks like we have an equivalence relation on the arrangements of distinct books on a bookshelf such that

1. Each equivalence class has $k!$ elements, and
2. There is a bijection between the equivalence classes and k-element multisets of the n shelves.

Thus if we can compute the number of ways to arrange k *distinct* books on the n shelves of a bookcase, we should be able to apply the quotient principle to compute the number of k-element multisets of an n-element set.

The bookcase arrangement problem.

Exercise 1.4-5 We have k books to arrange on the n shelves of a bookcase. The order in which the books appear on a shelf matters, and each shelf can hold all the books. We will assume that as the books are placed on the shelves they are moved as far to the left as they will go so that all that matters is the order in which the books appear and not the actual places where the books sit. When book i is placed on a shelf, it can go between two books already there or to the left or right of all the books on that shelf.

(a) Since the books are distinct, we may think of a first, second, third, etc. book. In how many ways may we place the first book on the shelves.

(b) Once the first book has been placed, in how many ways may the second book be placed?

(c) Once the first two books have been placed, in how many ways may the third book be placed?

(d) Once $i - 1$ books have been placed, book i can be placed on any of the shelves to the left of any of the books already there, but there are some additional ways in which it may be placed. In how many ways in total may book i be placed?

(e) In how many ways may k distinct books be place on n shelves in accordance with the constraints above?

In Exercise 1.4-5 there are n places where the first book can go, namely on the left side of any shelf. Then the next book can go in any of the n places on the far left side of any shelf, or it can go to the right of book one. Thus there are $n+1$ places where book 2 can go. At first, placing book three appears to be more complicated, because we could create two different patterns by placing the first two books. However book 3 could go to the far left of any shelf or to the immediate right of any of the books already there. (Notice that if book 2 and book 1 are on shelf 3 in that order, putting book 3 to the immediate right of book 2 means putting it between book 2 and book 1.) Thus in any case, there are n+2 ways to place book 3. Similarly, once $i-1$ books have been placed, there are $n+i-1$ places where we can place book i. It can go at the far left of any of the n shelves or to the immediate right of any of the $i-1$ books that we have already placed. Thus the number of ways to place k distinct books is

$$n(n+1)(n+2)\cdots(n+k-1) = \prod_{i=1}^{k}(n+i-1) = \prod_{j=0}^{k-1}(n+j) = \frac{(n+k-1)!}{(n-1)!}. \tag{1.15}$$

The Pi notation (also known as the product notation) we introduced for the product in Equation 1.15 is completely analogous to the Sigma notation we have learned to use for summation. The specific product that arose in Equation 1.15 is called a *rising factorial power*. It has a notation (also introduced by Don Knuth) analogous to that for the falling factorial notation. Namely, we write

$$n^{\overline{k}} = n(n+1)\cdots(n+k-1) = \prod_{i=1}^{k}(n+i-1).$$

This is the product of k successive numbers beginning with n.

Since the last expression in Equation 1.15 is quotient of two factorials it is natural to ask whether it is counting equivalence classes of an equivalence relation. If so, the set on which the relation is defined has size $(n+k-1)!$. Thus it might be all lists or permutations of $n+k-1$ distinct objects. The size of an equivalence class is $(n-1)!$ and so what makes two lists equivalent might be permuting $n-1$ of the objects among themselves. Can we find such an interpretation?

Exercise 1.4-6 In how many ways may we arrange k distinct books and $n-1$ identical blocks of wood in a straight line?

Exercise 1.4-7 How does Exercise 1.4-6 relate to arranging books on the shelves of a bookcase?

In Exercise 1.4-6, if we tape numbers to the wood so that so that the pieces of wood are distinguishable, there are $n+k-1$ arrangements of the books and wood. But since the pieces of wood are actually indistinguishable, $(n-1)!$ of these arrangements are equivalent. Thus by the quotient principle there are $(n+k-1)!/(n-1)!$ arrangements. Such an arrangement allows us to put the books on the shelves as follows: put all the books before the first piece of wood on shelf 1, all the books between the first and second on shelf 2, and so on until you put all the books after the last piece of wood on shelf n.

The number of k-element multisets of an n-element set

We now define two bookcase arrangements of k books on n shelves to be equivalent if we get one from the other by permuting the books among themselves. Thus if two arrangements put the

same number of books on each shelf they are put into the same class by this relationship. On the other hand, if two arrangements put a different number of books on at least one shelf, they are not equivalent, and therefore they are put into different classes by this relationship. Thus the classes into which this relationship divides the the arrangements are disjoint and partition the set of all arrangements. Each class has $k!$ arrangements in it. The set of all arrangements has $n^{\overline{k}}$ arrangements in it. This leads to the following theorem.

Theorem 1.6 *The number of k-element multisets chosen from an n-element set is*

$$\frac{n^{\overline{k}}}{k!} = \binom{n+k-1}{k}.$$

Proof: The relationship on bookcase arrangements that two relationships are equivalent if and only if we get one from the other by permuting the books is an equivalence relation. The set of all arrangements has $n^{\overline{k}}$ elements, and the number of elements in an equivalence class is $k!$. By the quotient principle, the number of equivalence classes is $\frac{n^{\overline{k}}}{k!}$. There is a bijection between equivalence classes of bookcase arrangements with k books and multisets with k elements. The second equality follows from the definition of binomial coefficients. ■

The right-hand side of the formula is a binomial coefficient, so it is natural to ask whether there is a way to interpret choosing a k-element *multi*set form an n-element set as choosing a k-element *sub*set of some different $n+k-1$-element set. This illustrates an important principle. When we have a quantity that turns our to be equal to a binomial coefficient, it helps our understanding to interpret it as counting the number of ways to choose a subset of an appropriate size from a set of an appropriate size. We explore this idea for multisets in Problem 8 in this section.

Important Concepts, Formulas, and Theorems

1. *Symmetry Principle.* If we have a mathematical result that shows a certain symmetry, it often helps our understanding to find a proof that reflects this symmetry.

2. *Partition.* Given a set S of items, a *partition* of S consists of m sets $S_1, S_2, \ldots, S_m$, sometimes called *blocks* so that $S_1 \cup S_2 \cup \cdots \cup S_m = S$ and for each i and j with $i \neq j$, $S_i \cap S_j = \emptyset$.

3. *Equivalence relation. Equivalence class.* A relationship that partitions a set up into mutually exclusive classes is called an **equivalence relation**. Thus if $S = S_1 \cup S_2 \cup \ldots \cup S_m$ is a partition of S, the relationship that says any two elements $x \in S$ and $y \in S$ 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*

4. *Quotient principle.* The **quotient principle** says that if we can partition a set of p objects up into q classes of size r, then $q = p/r$. Equivalently, if an equivalence relation on a set of size p has q equivalence classes of size r, then $q = p/r$. The quotient principle is frequently used for counting the number of equivalence classes of an equivalence relation. When we have a quantity that is a quotient of two others, it is often helpful to our understanding to find a way to use the quotient principle to explain why we have this quotient.

5. *Multiset.* A multiset is similar to a set except that each item can appear multiple times. We can specify a *multiset* chosen from a set S by saying how many times each of its elements occurs.

6. *Choosing k-element multisets.* The number of k-element multisets that can be chosen from an n-element set is
$$\frac{(n+k-1)!}{k!(n-1)!} = \binom{n+k-1}{k}.$$
This is sometimes called the formula for "combinations with repetitions."

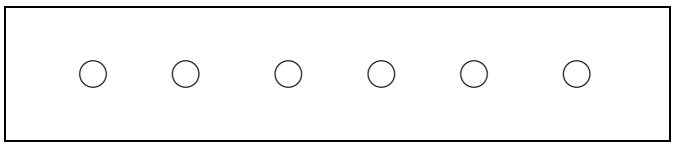
7. *Product notation, Pi notation* The notation $\prod_{i=1}^{n} a_i$ means the product of the quantities a_1 through a_i; that is, $a_1 a_2 \cdots a_n$.

8. When we have a quantity that turns out to be a binomial coefficient (or some other formula we recognize) it is often helpful to our understanding to try to interpret the quantity as the result of choosing a subset of a set (or doing whatever the formula that we recognize counts.)

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 top and bottom 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 (in particular, the quotient principle) 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 we pass out k identical apples to n children if each child must get at least one apple?

7. In how many ways may we place k distinct books on n shelves of a bookcase (all books pushed to the left as far as possible) if there must be at least one book on each shelf?

8. The formula for the number of multisets is $(n+k-1)!$ divided by a product of two other factorials. We seek an explanation using the quotient principle of why this counts multisets. The formula for the number of multisets is also a binomial coefficient, so it should have an interpretation involving choosing k items from $n+k-1$ items. The parts of the problem that follow lead us to these explanations.

(a) In how many ways may we place k red checkers and $n-1$ black checkers in a row?

(b) How can we relate the number of ways of placing k red checkers and $n-1$ black checkers in a row to the number of k-element multisets of an n-element set, say the set $\{1, 2, \ldots, n\}$ to be specific?

(c) How can we relate the choice of k items out of $n+k-1$ items to the placement of red and black checkers as in the previous parts of this problem?

9. How many solutions to the equation $x_1 + x_2 + \cdots x_n = k$ are there with each $x_i \geq 0$?

10. How many solutions to the equation $x_1 + x_2 + \cdots x_n = k$ are there with each $x_i > 0$?

11. 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*.)

12. 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.

13. 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 clockwise order. 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 1.5 to compute the number of equivalence classes?

14. 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.)

15. 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.)

16. 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 secretly. There is a story that, in ancient times, a king needed 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 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. 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, passwords, or private communication, is (and should be) *encrypted*—modified in such a way that, hopefully, it is only understandable to people who should be allowed to have access to it, and *undecipherable* to others.

Undecipherability by an adversary is, of course, a difficult goal. No code is completely undecipherable. If there is a printed "codebook," then the adversary can always steal the codebook, and no amount of mathematical sophistication can prevent this possibility. More likely, an adversary may have extremely large amounts of computing power and human resources to devote to trying to crack a code. Thus our notion of security is tied to computing power – a code is only as safe as the amount of computing power needed to break it. If we design codes that seem to need exceptionally large amounts of computing power to break, then we can be relatively confident in their security.

Private Key Cryptography

Traditional cryptography is known as *private key 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 codes 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 .

A Caeser cipher is 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. We will refer to the fact that $m \bmod n$ is always well defined as Euclid's division theorem. The proof appears in the next section.[1]

Theorem 2.1 (Euclid's division theorem) *For every integer m and positive integer n, there exist unique integers q and r such that $m = nq + r$ and $0 \leq r < n$. Furthermore, r is equal to $m \bmod n$.*

Exercise 2.1-1 Use Equation 2.1 to compute $10 \bmod 7$ and $-10 \bmod 7$. What are q and r in each case? Does $(-m) \bmod n = -(m \bmod n)$?

Exercise 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 (the numerical representation of) 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 Caeser cipher?

[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 from 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.

Exercise 2.1-3 Have someone use a Caeser 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 can you figure out what the message is? Is this something a computer could do quickly?

In Exercise 2.1-1, $10 = 7(1)+3$ and so 10 mod 7 is 3, while $-10 = 7(-2)+4$ and so -10 mod 7 is 4. These two calculations show that $(-m) \bmod n = -(m \bmod n)$ is not necessarily true. Note that -3 mod 7 is 4 also. Furthermore, $-10 + 3 \bmod 7 = 0$, suggesting that -10 is essentially the same as -3 when we are considering integers mod 7.

In Exercise 2.1-2, to shift each letter two places to the right, we replace each number n in our message by $(n+2)$ mod 26, so that SEA becomes 20 8 2. To shift 13 places to the right, we replace each number n in our message with $(n + 13)$ mod 26 so that SEA becomes 5 17 13. Similarly to implement a shift of s places, we replace each number n in our message by $(n + s)$ mod 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 Caeser cipher on a computer.

Exercise 2.1-3 considers the complexity of encoding, decoding and cracking a Ceasar cipher. 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. (These codes appear in many newspapers and puzzle books under the name cryptograms. Many people are able to solve these puzzles, which is compelling evidence of the lack of security in such a code.)

We do not 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      : 335017873005926455916467429642711822266728 11001
reverse, 5rep : 10011827662228117246924764619555546295500378710533
```

As Problem 18 shows, a receiver who knows the code can decode this message. Furthermore, a casual reader of the message, without knowledge of the encryption algorithm, would have no hope of decoding the message. 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, some entities have all these resources (such as a government, or a large corporation). The infamous German Enigma code 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; it helped a lot in breaking the code to have a stolen Enigma machine, though even with the stolen machine, it was not easy to break the code.) In general, any scheme that uses a *codebook*, a secretly agreed upon (possibly complicated) code, suffers from these drawbacks.

Public-key Cryptosystems

A *public-key* cryptosystem overcomes the problems associated with using a codebook. 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, an adversary with access to the encoded message and the public directory still cannot 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 so that the corresponding functions are inverses of each other, i.e for any message $M \in \mathcal{D}$ 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. In fact, when the idea for public key cryptography was proposed (by Diffie and Hellman[2]), no one knew of any such functions. The first complete public-key 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, since S_B is a secret that only Bob has. 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, at this point, 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 code is not secure, since if you know P_B, you can 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.

Arithmetic modulo n

The RSA encryption scheme is built upon the idea of arithmetic mod n, so we introduce this arithmetic now. Our goal is to understand how the basic arithmetic operations, addition, subtraction, multiplication, division, and exponentiation behave when all arithmetic is done mod n. As we shall see, some of the operations, such as addition, subtraction and multiplication, are straightforward to understand. Others, such as division and exponentiation, behave very differently than they do for normal arithmetic.

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

Exercise 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

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

[2]Whitfield Diffie and Martin Hellman. "New directions in cryptography" *IEEE Transactions on Information Theory*, **IT-22**(6) pp 644-654, 1976.

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, as adding multiples of n to i does not change the value of $i \bmod n$. In general, we have

Lemma 2.2 *$i \bmod n = (i + kn) \bmod n$ for any integer k.*

Proof: By Theorem 2.1, for unique integers q and r, with $0 \leq r < n$, we have

$$i = nq + r \ . \tag{2.4}$$

Adding kn to both sides of Equation 2.4, we obtain

$$i + kn = n(q + k) + r \ . \tag{2.5}$$

Applying the definition of $i \bmod n$ to Equation 2.4, we have that $r = i \bmod n$ and applying the same definition to Equation 2.5 we have that $r = (i + kn) \bmod n$. The lemma follows. ■

Now we can go back to the equations of Exercise 2.1-4; the correct versions are stated below. Informally, we are showing if we have a computation involving addition and multiplication, and we plan to take the end result mod n, then we are free to take any of the intermediate results mod n also.

Lemma 2.3

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

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

Proof: We prove the first and last terms in the sequence of equations for plus are equal; the other equalities for plus follow by similar computations. The proofs of the equalities for products are similar.

By Theorem 2.1, we have that for unique integers q_1 and q_2,

$$i = (i \bmod n) + q_1 n \text{ and } j = (j \bmod n) + q_2 n \ .$$

Then adding these two equations together mod n, and using Lemma 2.2, we obtain

$$\begin{aligned}(i + j) \bmod n &= [(i \bmod n) + q_1 n + (j \bmod n) + q_2 n)] \bmod n \\ &= [(i \bmod n) + (j \bmod n) + n(q_1 + q_2)] \bmod n \\ &= [(i \bmod n) + (j \bmod n)] \bmod n \ .\end{aligned}$$

■

We now introduce a convenient notation for performing modular arithmetic. 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:

$$i +_n j = (i + j) \bmod n \tag{2.6}$$
$$i \cdot_n j = (i \cdot j) \bmod n \tag{2.7}$$

We will use the expression "$x \in Z_n$" to mean that x is a variable that can take on any of the integral values between 0 and $n-1$. In addition, $x \in Z_n$ is a signal that if we do algebraic operations with x, we are will use $+_n$ and $\cdot_n$ rather than the usual addition and multiplication. In ordinary algebra it is traditional to use letters near the beginning of the alphabet to stand for constants; that is, numbers that are fixed throughout our problem and would be known in advance in any one instance of that problem. This allows us to describe the solution to many different variations of a problem all at once. Thus we might say "For all integers a and b, there is one and only one integer x that is a solution to the equation $a + x = b$, namely $x = b - a$." We adopt the same system for Z_n. When we say "Let a be a member of Z_n," we mean the same thing as "Let a be an integer between 0 and $n-1$," but we are also signaling that in equations involving a, we will use $+_n$ and $\cdot_n$.

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

Proof: Commutativity follows immediately from the definition and the commutativity of ordinary addition and multiplication. We prove the associative law for addition in the following equations; the other laws follow similarly.

$$\begin{aligned}
a +_n (b +_n c) &= (a + (b +_n c)) \bmod n && \text{(Equation 2.6)} \\
&= (a + ((b + c) \bmod n)) \bmod n && \text{(Equation 2.6)} \\
&= (a + (b + c)) \bmod n && \text{(Lemma 2.3)} \\
&= ((a + b) + c) \bmod n && \text{(Associative law for ordinary sums)} \\
&= ((a + b) \bmod n + c) \bmod n && \text{(Lemma 2.3)} \\
&= ((a +_n b) + c) \bmod n && \text{(Equation 2.6)} \\
&= (a +_n b) +_n c && \text{(Equation 2.6).}
\end{aligned}$$

■

Notice that $0 +_n i = i$, $1 \cdot_n i = i$, (these equations are called the *additive identity properties* and the *multiplicative identity* properties) and $0 \cdot_n i = 0$, so we can use 0 and 1 in algebraic expressions in Z_n (which we may also refer to as agebraic expressions mod n) as we use them in ordinary algebraic expressions. We use $a -_n b$ to stand for $a +_n (-b)$.

We conclude this section by observing that repeated applications of Lemma 2.3 and Theorem 2.4 are useful when computing sums or products mod n in which the numbers are large. For

example, suppose you had m integers $x_1, \ldots, x_m$ and you wanted to compute $(\sum_{j=1}^{m} x_i) \bmod n$. One natural way to do so would be to compute the sum, and take the result modulo n. However, it is possible that, on the computer that you are using, even though $(\sum_{j=1}^{m} x_i) \bmod n$ 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 2.3 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 using addition mod n

One natural way to use addition of a number a mod n in encryption is first to 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 $S(C) = C +_n (-a) = C -_n a$. 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 n but not a and 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. (In Problem 11 we ask you to explain why, even if your adversary didn't know n, but just believed you were adding some secret number a mod some other secret number n, she or he could use three encoded messages to find three differences in the integers, instead of Z_n, one of which was the difference of two messages.) This difference could contain valuable information for your adversary.[3] Thus adding a mod n is not an encoding method you would want to use more than once.

Cryptography using multiplication mod n

We will now explore whether multiplication is a good method for encryption. In particular, we could encrypt by multiplying a message (mod n) by a prechosen value a. We would then expect

[3] If each segement of a message were equally likely to be any number between 0 and n, and if any second (or third, etc.) segment 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 are highly unlikely to hold, in which case your adversary could apply structural knowledge to deduce information about your two messages from their difference.

to decrypt by "dividing" by a. What exactly does division mod a mean? Informally, we think of division as the "inverse" of multiplication, that is, if we take a number x, multiply by a and then divide by a, we should get back x. Clearly, with normal arithmetic, this is the case. However, with modular arithmetic, division is trickier.

Exercise 2.1-6 One possibility for encryption is to take a message x and compute $a \cdot_n x$, for some value a, that the sender and receiver both know. You could then decrypt by doing division by a in Z_n if you knew how to divide in Z_n. How well does this work? In particular, consider the following three cases. First, consider $n = 12$ and $a = 4$ and $x = 3$. Second, consider $n = 12$ and $a = 3$ and $x = 6$. Third, consider $n = 12$ and $a = 5$ and $x = 7$.

When we encoded a message by adding a in Z_n, we could decode the message simply by subtracting a in Z_n. However, this method had significant disadvantages, even if our adversary did not know n. If instead of encoding by adding a mod n, we encoded by multiplying by a mod n, we would foil the adversary's ability to subtract each two of three messages from each other to get a difference of two unencoded messages. (This doesn't give us a great secret key cryptosystem, even if both a and n are secret, but it does give us a better one.) 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, Exercise 2.1-6 shows that division in Z_n doesn't always make very much sense. Suppose your value of n was 12 and the value of a was 4. You send the message 3 as $4 \cdot_{12} 3 = 0$. Thus you send the encoded message 0. Now your partner sees 0, and says the message might have been 0; after all, $4 \cdot_{12} 0 = 0$. On the other hand, $4 \cdot_{12} 3 = 0$, $4 \cdot_{12} 6 = 0$, and $4 \cdot_{12} 9 = 0$ 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 special problems arose because the encoded message was 0, so the next question in Exercise 2.1-6 gives us an encoded message that is not 0. Suppose that $a = 3$ and $n = 12$. Now we encode the message 6 by computing $3 \cdot_{12} 6 = 6$. Straightforward calculation shows that $3 \cdot_{12} 2 = 6$, $3 \cdot_{12} 6 = 6$, and $3 \cdot_{12} 10 = 6$. Thus, the message 6 can be decoded in three possible ways, as 2, 6, or 10.

The final question in Exercise 2.1-6 provides some hope. Let $a = 5$ and $n = 12$. The message is 7 is encoded as $5 \cdot_{12} 7 = 11$. 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 $5 \cdot_{12} x = 11$. Thus in this case we can correctly decode the message.

As we shall see in the next section, the kinds of problems we had in Exercise 2.1-6 happen only when a and n have a common divisor that is greater than 1. Thus, when a and n have no common factors greater than one, all our receiver needs to know 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 and n are 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.

Important Concepts, Formulas, and Theorems

1. *Cryptography* is the study of methods to send and receive secret messages.

(a) The *sender* wants to send a message to a *receiver*.

(b) The *adversary* wants to steal the message.

(c) In *private key cryptography*, the sender and receiver agree in advance on a *secret code*, and then send messages using that code.

(d) In *public key cryptography*, the encoding method can be published. Each person has a *public key* used to encrypt messages and a *secret key* used to encrypt an encrypted message.

(e) The original message is called the *plaintext*.

(f) The encoded text is called the *ciphertext*.

(g) A *Caesar cipher* is one in which each letter of the alphabet is shifted by a fixed amount.

2. *Euclid's Division Theorem.* For every integer m and positive integer n, there exist unique integers q and r such that $m = nq + r$ and $0 \leq r < n$. By definition, r is equal to $m \bmod n$.

3. *Adding multiples of n does not change values mod n.* That is, $i \bmod n = (i + kn) \bmod n$ for any integer k.

4. *Mods (by n) can be taken anywhere in calculation, so long as we take the final result mod n.*

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

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

5. *Commutative, associative and distributive laws.* Addition and multiplication mod n satisfy the commutative and associative laws, and multiplication distributes over addition.

6. Z_n. We 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:

$$\begin{aligned}i +_n j &= (i + j) \bmod n \\ i \cdot_n j &= (i \cdot j) \bmod n\end{aligned}$$

We use the expression "$x \in Z_n$" to mean that x is a variable that can take on any of the integral values between 0 and $n-1$, and that in algebraic expressions involving x we will use $+_n$ and $\cdot_n$. We use the expression $a \in Z_n$ to mean that a is a constant between 0 and $n-1$, and in algebraic expressions involving a we will use $+_n$ and $\cdot_n$.

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, for x, 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 integral values of a and b, with $1 \leq a, b < 12$, for which the equation $a \cdot_{12} x = b$ does not have any solutions in Z_{12}? If there are, give one set of values for a and b. If there are not, explain how you know this.

   b) Are there any integers a, with $1 < a < 12$ such that for every integral value of b, $1 \leq b < 12$, the equation $a \cdot_{12} x = b$ has a solution? If so, give one and explain why it works. If not, explain how you know this.

9. Does every equation of the form $a \cdot_n x = b$, with $a, b \in Z_n$ 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. Explain why, if you were encoding messages x_1, x_2, and x_3 to obtain y_1, y_2 and y_3 by adding a mod n, your adversary would know that at least one of the differences $y_1 - y_2$, $y_1 - y_3$ or $y_2 - y_3$ taken in the integers, not in Z_n, would be the difference of two unencoded messages. (Note: we are not saying that your adversary would know which of the three was such a difference.)

12. 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.

```
(1)  set seed to a random value
(2)  x = seed
(3)  Repeat
(4)        x = (ax + b) mod n
(5)        print x
(6)  Until x = seed
```

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

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

14. Prove the equalities for multiplication in Lemma 2.3.

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

16. State and prove the distributive law for $\cdot_n$ multiplication over $+_n$ addition.

17. Write pseudocode to take m integers $x_1, x_2, \ldots, x_m$, and an integer n, and return $\Pi_i^m x_i$ mod n. Be careful about *overflow*; in this context, being careful about overflow means that at no point should you ever compute a value that is greater than n^2.

18. Write pseudocode to decode a message that has been encoded using the algorithm

    - 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.

    Continue this processing with each block of three consecutive letters. Append the blocks, using either an 8 or a 9 to separate the blocks. Finally, reverse the number, and replace each digit 5 by two 5's.

2.2 Inverses and GCDs

Solutions to Equations and Inverses mod n

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 $4 \cdot_n x = b$ in order to decode a received message b. We saw that if the encrypted message was 0, then there were four possible values for x. More generally, Exercise 2.2-6 and some of the problems in the last section show that for certain values of n, a, and b, equations of the form $a \cdot_n x = b$ have a unique solution, while for other values of n, a, and b, the equation could have no solutions, or more than one solution.

To decide whether an equation of the form $a \cdot_n x = b$ has a unique solution in Z_n, it helps know whether a has a *multiplicative inverse* in Z_n, that is, whether there is another number a' such that $a' \cdot_n a = 1$. For example, in Z_9, the inverse of 2 is 5 because $2 \cdot_9 5 = 1$. On the other hand, 3 does not have an inverse in Z_9, because the equation $3 \cdot_9 x = 1$ does not have a solution. (This can be verified by checking the 9 possible values for x.) If a does have an inverse a', then we can find a solution to the equation

$$a \cdot_n x = b \ .$$

To do so, we multiply both sides of the equation by a', obtaining

$$a' \cdot_n (a \cdot_n x) = a' \cdot_n b.$$

By the associative law, this gives us

$$(a' \cdot_n a) \cdot_n x = a' \cdot_n b.$$

But $a' \cdot_n a = 1$ by definition so we have that

$$x = a' \cdot_n b \ .$$

Since this computation is valid for any x that satisfies the equation, we conclude that the only x that satisfies the equation is $a' \cdot_n b$. We summarize this discussion in the following lemma.

Lemma 2.5 *Suppose a has a multiplicative inverse a' in Z_n. Then for any $b \in Z_n$, the equation*

$$a \cdot_n x = b$$

has the unique solution

$$x = a' \cdot_n b \ .$$

Note that this lemma holds for *any* value of $b \in Z_n$.

This lemma tells us that whether or not a number has an inverse mod n is important for the solution of modular equations. We therefore wish to understand exactly when a member of Z_n has an inverse.

Inverses mod n

We will consider some of the examples related to Problem 9 of the last section.

Exercise 2.2-1 Determine whether every element a of Z_n has an inverse for n= 5, 6, 7, 8, and 9.

Exercise 2.2-2 If an element of Z_n has a multiplicative inverse, can it have two different multiplicative inverses?

For Z_5, we can determine by multiplying each pair of nonzero members of Z_5 that the following table gives multiplicative inverses for each element a of Z_5. For example, the products $2 \cdot_5 1 = 2$, $2 \cdot_5 2 = 4$, $2 \cdot_5 3 = 1$, and $2 \cdot_5 4 = 3$ tell us that 3 is the unique multiplicative inverse for 2 in Z_5. This is the reason we put 3 below 2 in the table. One can make the same kinds of computations with 3 or 4 instead of 2 on the left side of the products to get the rest of the table.

a	1	2	3	4
a'	1	3	2	4

For Z_7, we have similarly the table

a	1	2	3	4	5	6
a'	1	4	5	2	3	6

.

For Z_9, we have already said that $3 \cdot_9 x = 1$ does not have a solution, so by Lemma 2.5, 3 does not have an inverse. (Notice how we are using the Lemma. The Lemma says that if 3 had an inverse, then the equation $3 \cdot_9 x = 1$ *would* have a solution, and this would contradict the fact that $3 \cdot_9 x = 1$ does not have a solution. Thus assuming that 3 had an inverse would lead us to a contradiction. Therefore 3 has no multiplicative inverse.)

This computation is a special case of the following corollary to Lemma 2.5.

Corollary 2.6 *Suppose there is a b in Z_n such that the equation*

$$a \cdot_n x = b$$

does not have a solution. Then a does not have a multiplicative inverse in Z_n.

Proof: Suppose that $a \cdot_n x = b$ has no solution. Suppose further that a does have a multiplicative inverse a' in Z_n. Then by Lemma 2.5, $x = a'b$ is a solution to the equation $a \cdot_n x = b$. This contradicts the hypothesis given in the corollary that the equation does not have a solution. Thus some supposition we made above must be incorrect. One of the assumptions, namely that $a \cdot_n x = b$ has no solution was the hypothesis given to us in the statement of the corollary. The only other supposition we made was that a has an inverse a' in Z_n. Thus this supposition must be incorrect as it led to the contradiction. Therefore, it must be case that a does not have a multiplicative inverse in Z_n. ■

Our proof of the corollary is a classical example of the use of contradiction in a proof. The principle of *proof by contradiction* is the following.

Principle 2.1 (Proof by contradiction) *If by assuming a statement we want to prove is false, we are lead to a contradiction, then the statement we are trying to prove must be true.*

We can actually give more information than Exercise 1 asks for. You can check that the table below shows an X for the elements of Z_9 that do not have inverses and gives an inverse for each element that has one

a	1	2	3	4	5	6	7	8
a'	1	5	X	7	2	X	4	8

.

In Z_6, 1 has an inverse, namely 1, but the equations

$$2 \cdot_6 1 = 2, \quad 2 \cdot_6 2 = 4, \quad 2 \cdot_6 3 = 0, \quad 2 \cdot_6 4 = 2, \quad 2 \cdot_6 5 = 4$$

tell us that 2 does not have an inverse. Less directly, but with less work, we see that the equation $2 \cdot_6 x = 3$ has no solution because $2x$ will always be even, so $2x \bmod 6$ will always be even. Then Corollary 2.6 tells us that 2 has no inverse. Once again, we give a table that shows exactly which elements of Z_6 have inverses.

a	1	2	3	4	5
a'	1	X	X	X	5

A similar set of equations shows that 2 does not have an inverse in Z_8. The following table shows which elements of Z_8 have inverses.

a	1	2	3	4	5	6	7
a'	1	X	3	X	5	X	7

.

We see that every nonzero element in Z_5 and Z_7 does have a multiplicative inverse, but in Z_6, Z_8, and Z_9, some elements do not have a multiplicative inverse. Notice that 5 and 7 are prime, while 6, 8, and 9 are not. Further notice that the elements in Z_n that do not have a multiplicative inverse are exactly those that share a common factor with n.

We showed that 2 has exactly one inverse in Z_5 by checking each multiple of 2 in Z_5 and showing that exactly one multiple of 2 equals 1. In fact, for any element that has an inverse in Z_5, Z_6, Z_7, Z_8, and Z_9, you can check in the same way that it has exactly one inverse. We explain why in a theorem.

Theorem 2.7 *If an element of Z_n has a multiplicative inverse, then it has exactly one inverse.*

Proof: Suppose that an element a of Z_n has an inverse a'. Suppose that a^* is also an inverse of a. Then a' is a solution to $a \cdot_n x = 1$ and a^* is a solution to $a \cdot_n x = 1$. But by Lemma 2.5, the equation $a \cdot_n x = 1$ has a unique solution. Therefore $a' = a^*$. ■

Just as we use a^{-1} to denote the inverse of a in the real numbers, we use a^{-1} to denote the unique inverse of a in Z_n when a has an inverse. Now we can say precisely what we mean by division in Z_n. We will define what we mean by *dividing* a member of Z_n by a if and only if a has an inverse $a^{-1} \bmod n$; in this case dividing b by $a \bmod n$ is defined to be same as multiplying b by $a^{-1} \bmod n$. We were led to our discussion of inverses because of their role in solving equations. We observed that in our examples, an element of Z_n that has an inverse mod n has no factors greater than 1 in common with n. This is a statement about a and n as integers with ordinary multiplication rather than multiplication mod n. Thus to prove that a has an inverse mod n if and only if a and n have no common factors other than 1 and -1, we have to convert the equation $a \cdot_n x = 1$ into an equation involving ordinary multiplication.

Converting Modular Equations to Normal Equations

We can re-express the equation

$$a \cdot_n x = 1$$

as

$$ax \bmod n = 1.$$

But the definition of $ax \bmod n$ is that it is the remainder r we get when we write $ax = qn + r$, with $0 \leq r < n$. This means that $ax \bmod n = 1$ if and only if there is an integer q with $ax = qn + 1$, or

$$ax - qn = 1. \tag{2.8}$$

Thus we have shown

Lemma 2.8 *The equation*

$$a \cdot_n x = 1$$

has a solution in Z_n if and only if there exist integers x and y such that

$$ax + ny = 1.$$

Proof: We simply take $y = -q$. ■

We make the change from $-q$ to y for two reasons. First, if you read a number theory book, you are more likely to see the equation with y in this context. Second, to solve this equation, we must find both x and y, and so using a letter near the end of the alphabet in place of $-q$ emphasizes that this is a variable for which we need to solve.

It appears that we have made our work harder, not easier, as we have converted the problem of solving (in Z_n) the equation $a \cdot_n x = 1$, an equation with just one variable x (that could only have $n-1$ different values), to a problem of solving Equation 2.8, which has two variables, x and y. Further, in this second equation, x and y can take on any integer values, even negative values.

However, this equation will prove to be exactly what we need to prove that a has an inverse mod n if and only if a and n have no common factors larger than one.

Greatest Common Divisors (GCD)

Exercise 2.2-3 Suppose that a and n are integers such that $ax+ny = 1$, for some integers x and y. What does that tell us about being able to find a (multiplicative) inverse for $a \pmod n$? In this situation, if a has an inverse in Z_n, what is it?

Exercise 2.2-4 If $ax+ny = 1$ for integers x and y, can a and n have any common divisors other than 1 and -1?

In Exercise 2.2-3, since by Lemma 2.8, the equation $a \cdot_n x = 1$ has a solution in Z_n if and only if there exist integers x and y such that $ax + ny = 1$, we can can conclude that

Theorem 2.9 *A number a has a multiplicative inverse in Z_n if and only if there are integers x and y such that $ax + ny = 1$.*

We answer the rest of Exercise 2.2-3 with a corollary.

Corollary 2.10 *If $a \in Z_n$ and x and y are integers such that $ax+ny = 1$, then the multiplicative inverse of a in Z_n is $x \bmod n$.*

Proof: Since $n \cdot_n y = 0$ in Z_n, we have $a \cdot_n x = 1$ in Z_n and therefore x is the inverse of a in Z_n. ■

Now let's consider Exercise 2.2-4. If a and n have a common divisor k, then there must exist integers s and q such that

$$a = sk$$

and

$$n = qk \ .$$

Substituting these into $ax + ny = 1$, we obtain

$$\begin{aligned} 1 &= ax + ny \\ &= skx + qky \\ &= k(sx + qy). \end{aligned}$$

But then k is a divisor of 1. Since the only integer divisors of 1 are ± 1, we must have $k = \pm 1$. Therefore a and n can have no common divisors other than 1 and -1.

In general, the **greatest common divisor** of two numbers j and k is the largest number d that is a factor of both j and k.[4] We denote the greatest common divisor of j and k by $\gcd(j, k)$.

We can now restate Exercise 2.2-4 as follows:

Lemma 2.11 *Given a and n, if there exist integers x and y such that $ax + ny = 1$ then $\gcd(a, n) = 1$.*

If we combine Theorem 2.9 and Lemma 2.11, we see that that if a has a multiplicative inverse mod n, then $\gcd(a, n) = 1$. It is natural to ask whether the statement that "if $\gcd a, n = 1$, then a has a multiplicative inverse" is true as well.[5] If so, this would give us a way to test whether a has a multiplicative inverse mod n by computing the greatest common divisor of a and n. For this purpose we would need an algorithm to find $\gcd(a, n)$. It turns out that there is such an algorithm, and a byproduct of the algorithm is a proof of our conjectured converse statement! When two integers j and k have $\gcd(j, k) = 1$, we say that j and k are *relatively prime*.

Euclid's Division Theorem

One of the important tools in understanding greatest common divisors is Euclid's Division Theorem, a result which has already been important to us in defining what we mean by $m \bmod n$. 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

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

[5] Notice that this statement is *not* equivalent to the statement in the lemma. This statement is what is called the "converse" of the lemma; we will explain the idea of converse statements more in Chapter 3.

algorithm works. Thus we restate it and present a proof here. Our proof uses the method of proof by contradiction, which you first saw in Corollary 2.6. Notice that we are assuming m is nonnegative which we didn't assume in our earlier statement of Euclid's Division Theorem, Theorem 2.1. In Problem 16 we will explore how we can remove this additional assumption.

Theorem 2.12 (Euclid's Division Theorem, restricted version) *For every nonnegative integer m and positive integer n, there exist unique integers q and r such that $m = nq + r$ and $0 \leq r < n$. By definition, r is equal to $m \bmod n$.*

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

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

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

We call the proof technique used here *proof by smallest counterexample*. In this method, we assume, as in all proofs by contradiction, that the theorem is false. This implies that there must be a *counterexample* which does not satisfy the conditions of the theorem. In this case that counterexample consists of numbers m and n such that *no* integers q and r exist which satisfy $m = qn + r$. Further, if there are counterexamples, then there must be one that is smallest in some sense. (Here being smallest means having the smallest m.) We choose such a 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 theorem was false. Therefore this assumption has to be invalid, and the theorem has to be true. As we will see in Chapter **??**, this method is closely related to a proof method called *proof by induction* and to recursive algorithms. In essence, the proof of Theorem 2.1 describes a recursive program to find q and r in the theorem above so that $0 \leq r < n$.

Exercise 2.2-5 Suppose that $k = jq + r$ as in Euclid's Division Theorem. Is there a relationship between $\gcd(j, k)$ and $\gcd(r, j)$?

In this exercise, if $r = 0$, then $\gcd(r, j)$ is j, because any number is a divisor of zero. But this is the GCD of k and j as well since in this case $k = jq$. The answer to the remainder of Exercise 2.2-5 appears in the following lemma.

Lemma 2.13 *If j, k, q, and r are positive integers such that $k = jq + r$ then*

$$\gcd(j, k) = \gcd(r, j). \tag{2.9}$$

Proof: In order to prove that both sides of Equation 2.9 are equal, we will show that they have exactly the same set of factors. That is, we will first show that if d is a factor of the left-hand side, then it is a factor of the right-hand side. Second, we will show that if d is a factor of the right-hand side, then it is a factor of the left-hand side.

If d is a factor of $\gcd(j,k)$ then it is a factor of both j and k. There must be integers i_1 and i_2 so that $k = i_1 d$ and $j = i_2 d$. Thus d is also a factor of

$$\begin{aligned} r &= k - jq \\ &= i_1 d - i_2 dq \\ &= (i_1 - i_2 q)d \ . \end{aligned}$$

Since d is a factor of j (by supposition) and r (by the equation above), it must be a factor of $\gcd(r,j)$.

Similarly, if d is a factor of $\gcd(r,j)$, it is a factor of j and r, and we can write $j = i_3 d$ and $r = i_4 d$. Therefore,

$$\begin{aligned} k &= jq + r \\ &= i_3 dq + i_4 d \\ &= (i_3 q + i_4)d \ , \end{aligned}$$

and d is a factor of k and therefore of $\gcd(j,k)$.

Since $\gcd(j,k)$ has the same factors as $\gcd(r,j)$ they must be equal. ■

While we did not need to assume $r < j$ in order to prove the lemma, Theorem 2.1 tells us we may assume $r < j$. The assumption in the lemma that j, q and r are positive implies that $j < k$. Thus this lemma reduces our problem of finding $\gcd(j,k)$ to the simpler (in a recursive sense) problem of finding $\gcd(r,j)$.

The GCD Algorithm

Exercise 2.2-6 Using Lemma 2.13, write a recursive algorithm to find $\gcd(j,k)$, given that $j \leq k$. Use it (by hand) to find the GCD of 24 and 14 and the GCD of 252 and 189.

Our algorithm for Exercise 2.2-6 is based on Lemma 2.13 and the observation that if $k = jq$, for any q, then $j = \gcd(j,k)$. We first write $k = jq + r$ in the usual way. If $r = 0$, then we return j as the greatest common divisor. Otherwise, we apply our algorithm to find the greatest common divisor of j and r. Finally, we return the result as the greatest common divisor of j and k.

To find

$$\gcd(14, 24)$$

we write

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

In this case $k = 24$, $j = 14$, $q = 1$ and $r = 10$. Thus we can apply Lemma 2.13 and conclude that

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

We therefore continue our computation of $\gcd(10, 14)$, by writing $14 = 10 \cdot 1 + 4$, and have that

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

Now,

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

and so

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

Now

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

so that now $k = 4$, $j = 2$, $q = 2$, and $r = 0$. In this case our algorithm tells us that our current value of j is the GCD of the original j and k. This step is the base case of our recursive algorithm. Thus we have that

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

While the numbers are larger, it turns out to be even easier to find the GCD of 252 and 189. We write

$$252 = 189 \cdot 1 + 63,$$

so that $\gcd(189, 252) = \gcd(63, 189)$, and

$$189 = 63 \cdot 3 + 0.$$

This tells us that $\gcd(189, 252) = \gcd(189, 63) = 63$.

Extended GCD algorithm

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 y such that $\gcd(j, k) = jx + ky$. This will solve the problem we have been working on, because it will prove that if $\gcd(a, n) = 1$, then there are integers x and y such that $ax + ny = 1$. Further it will tell us how to find x, and therefore the multiplicative inverse of a.

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

$$\gcd(r, j) = (k - jq)x' + jy' = kx' + j(y' - qx').$$

Thus when we return $\gcd(r, j)$ as $\gcd(j, k)$, we want to return the value of x' as y and and the value of $y' - qx'$ as x.

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

Exercise 2.2-7 Apply Euclid's extended GCD algorithm to find numbers x and y such that the GCD of 14 and 24 is $14x + 24y$.

For our discussion of Exercise 2.2-7 we give pseudocode for the extended GCD algorithm. While we expressed the algorithm more concisely earlier 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, j, k, r, x and y, 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 $j[0]$ and $k[0]$ will be the numbers whose gcd we wish to compute. Eventually $x[0]$ and $y[0]$ will become the x and y we want.

(In Line 0 we are using the notation $\lfloor x \rfloor$ to stand for the floor of x, the largest integer less than or equal to x.)

```
gcd(j, k)
// assume that j < k
(1)  i = 0; k[i] = k; j[i] = j
(2)  Repeat
(3)        q[i] = ⌊k[i]/j[i]⌋
(4)        r[i] = k[i] − q[i]j[i]
(5)        k[i + 1] = j[i]; j[i + 1] = r[i]
(6)        i = i + 1
(7)  Until  (r[i − 1] = 0)
// we have found the value of the gcd, now we compute the x and y
(8)  i = i − 1
(9)  gcd = j[i]
(10) y[i] = 0; x[i] = 1
(11) i = i − 1
(12) While (i ≥ 0)
(13)       y[i] = x[i + 1]
(14)       x[i] = y[i + 1] − q[i]x[i + 1]
(15)       i = i − 1
(16) Return gcd
(17) Return x
(18) Return y
```

We show the details of how this algorithm applies to $\gcd(24, 14)$ in Table 2.1. In a row, the $q[i]$ and $r[i]$ values are computed from the $j[i]$ and $k[i]$ values. Then the $j[i]$ and $r[i]$ are passed down to the next row as $k[i+1]$ and $j[i+1]$ respectively. This process continues until we finally reach a case where $k[i] = q[i]j[i]$ and we can answer $j[i]$ for the gcd. We can then begin computing $x[i]$ and $y[i]$. In the row with $i = 3$, we have that $x[i] = 0$ and $y[i] = 1$. Then, as i decreases, we compute $x[i]$ and $y[i]$ for a row by setting $y[i]$ to $x[i+1]$ and $x[i]$ to $y[i+1] - q[i]x[i+1]$. We note that in every row, we have the property that $j[i]x[i] + k[i]y[i] = \gcd(j, k)$.

We summarize Euclid's extended GCD algorithm in the following theorem:

Theorem 2.14 *Given two integers j and k, Euclid's extended GCD algorithm computes* $\gcd(j, k)$ *and two integers x and y such that* $\gcd(j, k) = jx + ky$.

We now use Eculid's extended GCD algorithm to extend Lemma 2.11.

$\underline{i}$	$\underline{j[i]}$	$\underline{k[i]}$	$\underline{q[i]}$	$\underline{r[i]}$	$\underline{x[i]}$	$\underline{y[i]}$
0	14	24	1	10		
1	10	14	1	4		
2	4	10	2	2		
3	2	4	2	0	1	0
2	4	10	2	2	-2	1
1	10	14	1	4	3	-2
0	14	24	1	10	-5	3
$\gcd = 2$						
$x = -5$						
$y = 3$						

Table 2.1: The computation of $\gcd(14, 24)$ by algorithm $\gcd(j, k)$.

Theorem 2.15 *Two positive integers j and k have greatest common divisor 1 (and thus are relatively prime) if and only if there are integers x and y such that $jx + ky$=1.*

Proof: The statement that if there are integers x and y such that $jx+ky = 1$, then $\gcd(j, k) = 1$ is proved in Lemma 2.11. In other words, $\gcd(j, k) = 1$ if there are integers x and y such that $jx + ky = 1$.

On the other hand, we just showed, by Euclid's extended GCD algorithm, that given positive integers j and k, there are integers x and y such that $\gcd(j, k) = jx+ky$. Therefore, $\gcd(j, k) = 1$ only if there are integers x and y such that $jx + ky = 1$. ■

Combining Lemma 2.8 and Theorem 2.15, we obtain:

Corollary 2.16 *For any positive integer n, an element a of Z_n has a multiplicative inverse if and only if* $\gcd(a, n) = 1$.

Using the fact that if n is prime, $\gcd(a, n) = 1$ for all non-zero $a \in Z_n$, we obtain

Corollary 2.17 *For any prime p, every non-zero element a of Z_p has an inverse.*

Computing Inverses

Not only does Euclid's extended GCD algorithm tell us if an inverse exists, but, just as we saw in Exercise 2.2-3 it computes it for us. Combining Exercise 2.2-3 with Theorem 2.15, we get

Corollary 2.18 *If an element a of Z_n has an inverse, we can compute it by running Euclid's extended GCD algorithm to determine integers x and y so that $ax + ny = 1$; then the inverse of a is $x \bmod n$.*

For completeness, we now give pseudocode which determines whether an element a in Z_n has an inverse and computes the inverse if it exists:

```
inverse(a, n)
(1)  Run procedure gcd(a, n) to obtain gcd(a, n), x and y
(2)  if gcd(a, n) = 1
(3)       return x
(4)  else
(5)       print ‘‘no inverse exists’’
```

The correctness of the algorithm follows immediately from the fact that $\gcd(a, n) = ax + ny$, so if $\gcd(a, n) = 1$, $ax \bmod n$ must be equal to 1.

Important Concepts, Formulas, and Theorems

1. *Multiplicative inverse.* a' is a multiplicative inverse of a in Z_n if $a \cdot_n a' = 1$. If a has a multiplicative inverse, then it has a unique multiplicative inverse which we denote by a^{-1}.

2. *An important way to solve modular equations.* Suppose a has a multiplicative inverse mod n, and this inverse is a^{-1}. Then for any $b \in Z_n$, the unique solution to the equation

$$a \cdot_n x = b$$

   is

$$x = a^{-1} \cdot_n b \ .$$

3. *Converting modular to regular equations.* The equation

$$a \cdot_n x = 1$$

   has a solution in Z_n if and only if there exist integers x and y such that

$$ax + ny = 1 \ .$$

4. *When do inverses exist in Z_n?* A number a has a multiplicative inverse in Z_n if and only if there are integers x and y such that $ax + ny = 1$.

5. *Greatest common divisor (GCD).* The *greatest common divisor* of two numbers j and k is the largest number d that is a factor of both j and k.

6. *Relatively prime.* When two numbers, j and k have $\gcd(j, k) = 1$, we say that j and k are relatively prime.

7. *Connecting inverses to GCD.* Given a and n, if there exist integers x and y such that $ax + ny = 1$ then $\gcd(a, n) = 1$.

8. *GCD recursion lemma.* If j, k, q, and r are positive integers such that $k = jq + r$ then $\gcd(j, k) = \gcd(r, j)$.

9. *Euclid's GCD algorithm.* Given two numbers j and k, this algorithm returns $\gcd(j, k)$.

10. *Euclid's extended GCD algorithm.* Given two numbers j and k, this algorithm returns $\gcd(j, k)$, and two integers x and y such that $\gcd(j, k) = jx + ky$.

11. *Relating GCD of 1 to Euclid's extended GCD algorithm.* Two positive integers j and k have greatest common divisor 1 if and only if there are integers x and y such that $jx + ky$=1. One of the integers x and y could be negative.

12. *Restatement for Z_n.* $\gcd(a, n) = 1$ if and only if there are integers x and y such that $ax + ny = 1$.

13. *Condition for multiplicative inverse in Z_n* For any positive integer n, an element a of Z_n has an inverse if and only if $\gcd(a, n) = 1$.

14. *Multiplicative inverses in Z_p, p prime* For any prime p, every non-zero element a of Z_p has a multiplicative inverse.

15. *A way to solve some modular equations $a \cdot_n x = b$.* Use Euclid's extended GCD algorithm to compute a^{-1} (if it exists), and multiply both sides of the equation by a^{-1}. (If a has no inverse, the equation might or might not have a solution.)

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 $a \cdot 133 - 2m \cdot 277 = 1$, does this guarantee that a has an inverse mod m? If so, what is it? If not, why not?

3. Determine whether every nonzero element of Z_n has a multiplicative inverse for $n = 10$ and $n = 11$.

4. How many elements a are there such that $a \cdot_{31} 22 = 1$? How many elements a are there such that $a \cdot_{10} 2 = 1$?

5. Given an element b in Z_n, what can you say in general about the possible number of elements a such that $a \cdot_n b = 1$ in Z_n?

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

7. Compute the GCD of 210 and 126 by using Euclid's GCD algorithm.

8. If $k = jq + r$ as in Euclid's Division Theorem, is there a relationship between $\gcd(q, k)$ and $\gcd(r, q)$. If so, what is it?

9. 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 mod p, and Alice will send Bob aq mod p. Their key (which they will keep secret) will then be abq mod 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 mod p, and bq mod 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 mod 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 $q^a \bmod p$, you send me $q^b \pmod p$ and we use $q^{ab} \bmod p$ as our key?" In this case can you think of a way for the wire tapper to compute $q^{ab} \bmod p$? 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.)

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

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

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

13. Solve the equation $16 \cdot_{103} x = 21$ in Z_{103}.

14. Which elements of Z_{35} do not have multiplicative inverses in Z_{35}?

15. If $k = jq + r$ as in Euclid's Division Theorem, is there a relationship between $\gcd(j, k)$ and $\gcd(r, k)$. If so, what is it?

16. Notice that if m is negative, then $-m$ is positive, so that by Theorem 2.12 $-m = qn + r$, where $0 \leq r < n$. This gives us $m = -qn - r$. If $r = 0$, then we have written $m = q'n + r'$, where $0 \leq r' \leq n$ and $q' = -q$. However if $r > 0$, we cannot take $r' = -r$ and have $0 \leq r' < n$. Notice, though, that since since we have already finished the case $r = 0$ we may assume that $0 \leq n - r < n$. This suggests that if we were to take r' to be $n - r$, we might be able to find a q' so that $m = q'n + r'$ with $0 \leq r' \leq n$, which would let us conclude that Euclid's Division Theorem is valid for negative values m as well as nonnegative values m. Find a q' that works and explain how you have extended Euclid's Division Theorem from the version in Theorem 2.12 to the version in Theorem 2.1.

17. The Fibonacci numbers F_i 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.)

18. Write (and run on several different inputs) a program to implement Euclid's extended GCD algorithm. Be sure to return x and y 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 j and k whose GCD we are computing?

19. The least common multiple of two positive integers x and y is the smallest positive integer 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.

20. Write pseudocode that given integers a, b and n in Z_n, either computes an x such that $a \cdot_n x = b$ or concludes that no such x exists.

21. Give an example of an equation of the form $a \cdot_n x = b$ that has a solution even though a and n are not relatively prime, or show that no such equation exists.

22. Either find an equation of the form $a \cdot_n x = b$ in Z_n that has a unique solution even though a and n are not relatively prime, or prove that no such equation exists. In other words, you are either to prove the statement that if $a \cdot_n x = b$ has a unique solution in Z_n, then a and n are relatively prime or to find a counter example.

2.3 The RSA Cryptosystem

Exponentiation mod n

In the previous sections, we have considered encryption using modular addition and multiplication, and have seen the shortcomings of both. In this section, we will consider using exponentiation for encryption, and will show that it provides a much greater level of security.

The idea behind RSA encryption is *exponentiation* in Z_n. By Lemma 2.3, if $a \in Z_n$,

$$a^j \bmod n = \underbrace{a \cdot_n a \cdot_n \cdots \cdot_n a}_{j \text{ factors}}. \tag{2.10}$$

In other words $a^j \bmod n$ is the product in Z_n of j factors, each equal to a.

The Rules of Exponents

Lemma 2.3 and the rules of exponents for the integers tell us that

Lemma 2.19 *For any $a \in Z_n$, and any nonnegative integers i and j,*

$$(a^i \bmod n) \cdot_n (a^j \bmod n) = a^{i+j} \bmod n \tag{2.11}$$

and

$$(a^i \bmod n)^j \bmod n = a^{ij} \bmod n. \tag{2.12}$$

Exercise 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?

Exercise 2.3-2 Compute the sixth powers of the nonzero elements of Z_7. What do you observe?

Exercise 2.3-3 Compute the numbers $1 \cdot_7 2$, $2 \cdot_7 2$, $3 \cdot_7 2$, $4 \cdot_7 2$, $5 \cdot_7 2$, and $6 \cdot_7 2$. What do you observe? Now compute the numbers $1 \cdot_7 3$, $2 \cdot_7 3$, $3 \cdot_7 3$, $4 \cdot_7 3$, $5 \cdot_7 3$, and $6 \cdot_7 3$. What do you observe?

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

In Exercise 2.3-1, we have that

$$\begin{aligned}
2^0 \bmod 7 &= 1 \\
2^1 \bmod 7 &= 2 \\
2^2 \bmod 7 &= 4 \\
2^3 \bmod 7 &= 1 \\
2^4 \bmod 7 &= 2 \\
2^5 \bmod 7 &= 4 \\
2^6 \bmod 7 &= 1 \\
2^7 \bmod 7 &= 2 \\
2^8 \bmod 7 &= 4.
\end{aligned}$$

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

$$\begin{aligned} 3^0 \bmod 7 &= 1 \\ 3^1 \bmod 7 &= 3 \\ 3^2 \bmod 7 &= 2 \\ 3^3 \bmod 7 &= 6 \\ 3^4 \bmod 7 &= 4 \\ 3^5 \bmod 7 &= 5 \\ 3^6 \bmod 7 &= 1 \\ 3^7 \bmod 7 &= 3 \\ 3^8 \bmod 7 &= 2. \end{aligned}$$

In this case, we will cycle through the list of six values $1, 3, 2, 6, 4, 5$ again and again.

Now observe that in Z_7, $2^6 = 1$ and $3^6 = 1$. This suggests an answer to Exercise 2.3-2. Is it the case that $a^6 \bmod 7 = 1$ for all $a \in Z_7$? We can compute that $1^6 \bmod 7 = 1$, and

$$\begin{aligned} 4^6 \bmod 7 &= (2 \cdot_7 2)^6 \bmod 7 \\ &= (2^6 \cdot_7 2^6) \bmod 7 \\ &= (1 \cdot_7 1) \bmod 7 \\ &= 1. \end{aligned}$$

What about 5^6? Notice that $3^5 = 5$ in Z_7 by the computations we made above. Using Equation 2.12 twice, this gives us

$$\begin{aligned} 5^6 \bmod 7 &= (3^5)^6 \bmod 7 \\ &= 3^{5 \cdot 6} \bmod 7 \\ &= 3^{6 \cdot 5} \bmod 7 \\ &= (3^6)^5 = 1^5 = 1 \end{aligned}$$

in Z_7. Finally, since $-1 \bmod 7 = 6$, Lemma 2.3 tells us that $6^6 \bmod 7 = (-1)^6 \bmod 7 = 1$. Thus the sixth power of each element of Z_7 is 1.

In Exercise 2.3-3 we see that

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

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

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

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

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

In Exercise 2.3-4 we are asked whether this is always the case. Notice that since 7 is a prime, by Corollary 2.17, each nonzero number between 1 and 6 has a mod 7 multiplicative inverse a^{-1}. 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^{-1} to get

$$(i \cdot_7 a) \cdot_7 a^{-1} = (j \cdot_7 a) \cdot_7 a^{-1}.$$

After using the associative law we get

$$i \cdot_7 (a \cdot_7 a^{-1}) = j \cdot_7 (a \cdot_7 a^{-1}). \tag{2.13}$$

Since $a \cdot_7 a^{-1} = 1$, Equation 2.13 simply becomes $i = j$. Thus, we have shown that the only way for $i \cdot_7 a$ to equal $j \cdot_7 a$ is for i to equal j. Therefore, all the values $i \cdot_7 a$ 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 2.3-4 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 2.20 *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 in Exercise 2.3-2.

Fermat's Little Theorem

Theorem 2.21 *(Fermat's Little Theorem). Let p be a prime number. Then $a^{p-1} \bmod p = 1$ in Z_p for each nonzero a in Z_p.*

Proof: Since p is a prime, Lemma 2.20 tells us that the numbers $1 \cdot_p a$, $2 \cdot_p a$, ..., $(p-1) \cdot_p a$, 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 in Z_p and Equation 2.10, we get

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

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

Corollary 2.22 *(Fermat's Little Theorem, version 2) For every positive integer a, and prime p, if a is not a multiple of p,*

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

Proof: This is a direct application of Lemma 2.3, because if we replace a by $a \bmod p$, then Theorem 2.21 applies.

The RSA Cryptosystem

Fermat's Little Theorem is at the heart of the RSA cryptosystem, a system that allows Bob to tell the world a way that they can encode a message to send to him so that he and only he can read it. In other words, even though he tells everyone how to encode the message, nobody except Bob has a significant chance of figuring out what the message is from looking at the encoded message. What Bob 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 Bob (who has some extra information that he keeps secret) can compute $f^{-1}(y)$. Thus when Alice wants to send a message x to Bob, she computes $f(x)$ and sends it to Bob, who uses his secret information to compute $f^{-1}(f(x)) = x$.

In the RSA cryptosystem Bob chooses two prime numbers p and q (which in practice each have at least a hundred digits) and computes the number $n = pq$. He 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 in $Z_{(p-1)(q-1)}$, and he computes $d = e^{-1} \bmod (p-1)(q-1)$. Bob publishes e and n. The number e is called his *public key*. The number d is called his *private key*.

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

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

People who want to send a message x to Bob compute $y = x^e \bmod n$ and send that to him 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, Bob will compute $z = y^d \bmod n$.

We summarize this process in pseudocode below:

```
Alice-send-message-to-Bob(x)
Alice does:
(1)       Read the public directory for Bob's keys e and n.
(2)       Compute y = x^e mod n
```

(3) Send y to Bob
Bob does:
(4) Receive y from Alice
(5) Compute $z = y^d \bmod n$, using secret key d
(6) Read z

Each step in these algorithms can be computed using methods from this Chapter. In Section 2.4, we will deal with computational issues in more detail.

In order to show that the RSA cryptosystem works, that is, that it allows us to encode and then correctly decode messages, we must show that $z = x$. In other words, we must show that, when Bob decodes, he gets back the original message. In order to show that the RSA cryptosystem is secure, we must argue that an eavesdropper, who knows n, e, and y, but does not know p, q or d, can not easily compute x.

Exercise 2.3-5 To show that the RSA cryptosystem works, we will first show a simpler fact. Why is

$$y^d \bmod p = x \bmod p?$$

Does this tell us what x is?

Plugging in the value of y, we have

$$y^d \bmod p = x^{ed} \bmod p. \tag{2.14}$$

But, in Line 4 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 k,

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

Plugging this into Equation (2.14), we obtain

$$\begin{aligned} x^{ed} \bmod p &= x^{k(p-1)(q-1)+1} \bmod p \\ &= x^{(k(q-1))(p-1)} x \bmod p. \end{aligned} \tag{2.15}$$

But for any number a which is not a multiple of p, $a^{p-1} \bmod p = 1$ by Fermat's Little Theorem (Theorem 2.22). We could simplify equation 2.15 by applying Fermat's Little Theorem to $x^{k(q-1)}$, as you will see below. However we can only do this when $x^{k(q-1)}$ is not a multiple of p. This gives us two cases, the case in which $x^{k(q-1)}$ is not a multiple of p (we'll call this case 1) and the case in which $x^{k(q-1)}$ is a multiple of p (we'll call this case 2). In case 1, we apply Equation 2.12 and Fermat's Little Theorem with a equal to $x^{k(q-1)}$, and we have that

$$\begin{aligned} x^{(k(q-1))(p-1)} \bmod p &= \left(x^{k(q-1)}\right)^{(p-1)} \bmod p \\ &= 1. \end{aligned} \tag{2.16}$$

Combining equations 2.14, 2.15 and 2.17, we have that

$$y^d \bmod p = x^{k(q-1)(p-1)} x \bmod p = 1 \cdot x \bmod p = x \bmod p,$$

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

We still have to deal with case 2, the case in which $x^{k(q-1)}$ is a multiple of p. In this case x is a multiple of p as well since x is an integer and p is prime. Thus $x \bmod p = 0$. Combining Equations 2.14 and 2.15 with Lemma 2.3, we get

$$y^d \bmod p = (x^{k(q-1)(p-1)} \bmod p)(x \bmod p) = 0 = x \bmod p.$$

Hence in this case as well, we have $y^d \bmod p = x \bmod p$.

While this will turn out to be useful information, it does not tell us what x is, however, because x may or may not equal $x \bmod p$.

The same reasoning shows us that $y^d \bmod q = x \bmod q$. What remains is to show what these two facts tell us about $y^d \bmod pq = y \bmod n$, which is what Bob computes.

Notice that by Lemma 2.3 we have proved that

$$(y^d - x) \bmod p = 0 \tag{2.17}$$

and

$$(y^d - x) \bmod q = 0. \tag{2.18}$$

Exercise 2.3-6 Write down an equation using only integers and addition, subtraction and multiplication in the integers, but perhaps more letters, that is equivalent to Equation 2.17, which says that $(y^d - x) \bmod p = 0$. (Do not use mods.)

Exercise 2.3-7 Write down an equation using only integers and addition, subtraction and multiplication in the integers, but perhaps more letters, that is equivalent to Equation 2.18, which says that $(y^d - x) \bmod q = 0$. (Do not use mods.)

Exercise 2.3-8 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 $y^d - x = ip$ for some integer i. The statement that $y^d - x \bmod q = 0$ is equivalent to saying $y^d - x = jq$ for some integer j. If something is a multiple of the prime p and the prime q, then it is a multiple of pq. Thus $(y^d - x) \bmod pq = 0$. Lemma 2.3 tells us that $(y^d - x) \bmod pq = (y^d \bmod pq - x) \bmod pq = 0$. But x and $y^d \bmod pq$ are both integers between 0 and $pq - 1$, so their difference is between $-(pq - 1)$ and $pq - 1$. The only integer between these two values that is 0 mod pq is zero itself. Thus $(y^d \bmod pq) - x = 0$. In other words,

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

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

Theorem 2.23 *(Rivest, Shamir, and Adleman) The RSA procedure for encoding and decoding messages works correctly.*

Proof: Proved above.■

One might ask, given that Bob 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 Bob'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 2.23 also proves a theorem known as the "Chinese Remainder Theorem."

Exercise 2.3-9 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?

x	$x \bmod 3$	$x \bmod 5$
0	0	0
1	1	1
2	2	2
3	0	3
4	1	4
5	2	0
6	0	1
7	1	2
8	2	3
9	0	4
10	1	0
11	2	1
12	0	2
13	1	3
14	2	4

Table 2.2: The values of $x \bmod 3$ and $x \bmod 5$ for each x between zero and 14.

As we see from Table 2.2, 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 2.24 *(Chinese Remainder Theorem) If m and n are relatively prime integers and $a \in Z_m$ and $b \in Z_n$, then the equations*

$$x \bmod m = a \tag{2.19}$$

$$x \bmod n = b \tag{2.20}$$

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.[6] In other words, we will have shown that each pair of equations 2.19 and 2.20 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 Problem 11 in the problems at the end of this section, $x-y$ 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. ■

Important Concepts, Formulas, and Theorems

1. *Exponentiation in Z_n.* For $a \in Z_n$, and a positive integer j:

$$a^j \bmod n = \underbrace{a \cdot_n a \cdot_n \cdots \cdot_n a}_{j \text{ factors}} .$$

2. *Rules of exponents.* For any $a \in Z_n$, and any nonnegative integers i and j,

$$(a^i \bmod n) \cdot_n (a^j \bmod n) = a^{i+j} \bmod n$$

and

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

3. *Multiplication by a fixed nonzero a in Z_p is a permutation.* 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 $\{1, 2, \cdots, p-1\}$.

4. *Fermat's Little Theorem.* Let p be a prime number. Then $a^{p-1} \bmod p = 1$ for each nonzero a in Z_p.

5. *Fermat's Little Theorem, version 2.* For every positive integer a and prime p, if a is not a multiple of p, then

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

6. *RSA cryptosystem.* (The first implementation of a public-key cryptosystem) In the RSA cryptosystem Bob chooses two prime numbers p and q (which in practice each have at least a hundred digits) and computes the number $n = pq$. He 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 he computes $d = e^{-1} \bmod (p-1)(q-1)$. Bob publishes e and n. To send a message x to Bob, Alice sends $y = x^e \bmod n$. Bob decodes by computing $y^d \bmod n$.

[6] 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.

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

$$\begin{aligned} x \bmod m &= a \\ x \bmod n &= b \end{aligned}$$

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

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 $1 \cdot_{11} 5$, $2 \cdot_{11} 5$, $3 \cdot_{11} 5$, $\ldots$, $10 \cdot_{11} 5$. 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_{11} in place of 5?

3. Compute the fourth power mod 5 of each element of Z_5. What do you observe? What general principle explains this observation?

4. 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}?

5. 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 2.3-5?

6. 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?

7. Compute each of the following. Show or explain 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}

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

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 in Z_p^2, 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 (interesting) 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 2.23 we said that if 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 2.23?

12. The relation of "congruence modulo n" is the relation $\equiv$ defined by $x \equiv y \bmod n$ 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 than are usually used in implementing the RSA algorithm for the sake of efficiency. In other words, you may choose your numbers so that your computer can multiply them without overflow.)

14. For some non-zero $a \in Z_p$, where p is prime, consider the set

$$S = \{a^0 \bmod p,\ a^1 \bmod p,\ a^2 \bmod p,\ \ldots,\ a^{p-2} \bmod p,\ a^{p-1} \bmod p\},$$

    and let $s = |S|$. Show that s is always a factor of $p - 1$.

15. Show that if $x^{n-1} \bmod n = 1$ for all integers x that are not multiples of n, then n is prime. (The slightly weaker statement that $x^{n-1} \bmod n = 1$ for all x relatively prime to n, does not imply that n is prime. There is a famous family of numbers called Carmichael numbers that are counterexamples.[7])

[7] See, for example, Cormen, Leiserson, Rivest, and Stein, cited earlier.

2.4 Details of the RSA Cryptosystem

In this section, we deal with some issues related to implementing the RSA cryptosystem: exponentiating large numbers, finding primes, and factoring.

Practical Aspects of Exponentiation mod n

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 a 121 digit number.

Exercise 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.

Exercise 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?

Exercise 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}, a^11 we are modeling the process of forming a^{11} 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 four 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, let's see how to compute a^{43}. We may write $43 = 32 + 8 + 2 + 1$, and thus

$$a^{43} = a^{2^5} a^{2^3} a^{2^1} a^{2^0} \ . \tag{2.21}$$

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}, via equation 2.21, using 3 additional multiplications. This saves a large number of 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 $a^{10^{120}}$.)

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.

Since $2^{10} = 1024$, we have that 2^{31} is twice as big as $2^{30} = (2^{10})^3 = (1024)^3$ and so is somewhat more than two billion, or $2 \cdot 10^9$. In particular, it is less than 10^{10}. Since 10^{120} is a one followed by 120 zeros, raising a positive integer other than one to the 10^{120}th power takes us completely out of the realm of the numbers that we are used to making exact computations with. For example, $10^{(10^{120})}$ has 119 more zeros following the 1 in the exponent than does 10^{10}.

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. 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 2.3.) 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?

Encoding and decoding messages according to the RSA algorithm requires many calculations. How long will all this arithmetic take? Let's assume for now that Bob has already chosen p, q, e, and d, and so he knows n as well. When Alice wants to send Bob the message x, she 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 Alice if she has a reasonable computer. (On the other hand, if she wants to send a message consisting of many segments of 200 digits each, she 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 Bob a similar amount of time to decode, as he 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 Bob 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 Bob'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?

Exercise 2.4-4 Factor 225,413. (The idea is to try 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 155-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 Bob. We said that Bob chooses two primes of about a hundred digits each. But how does he 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 mentioned 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.

In August of 2002, Agrawal, Kayal and Saxena announced an algorithm for testing whether an integer n is prime which they can show takes no more than the twelveth power of the number of digits of n to determine whether n is prime, and in practice seems to take significantly less time. While the algorithm requires more than the background we are able to provide in this book, its description and the proof that it works in the specified time uses only results that one might find in an undergraduate abstract algebra course and an undergraduate number theory course! The central theme of the algorithm is the use of a variation of Fermat's Little Theorem.

In 1976 Miller[8] was able to use Fermat's Little Theorem to show that if a conjecture called the "Extended Reiman Hypothesis" was true, then an algorithm he developed would determine whether a number n was prime in a time bounded above by a polynomial in the number of digits of n. In 1980 Rabin[9] modified Miller's method to get one that would determine in polynomial time whether a number was prime without the extra hypothesis, but with a probability of error that could be made as small a positive number as one might desire, but not zero. We describe the general idea behind all of these advances in the context of what people now call the Miller-Rabin primality test. As of the writing of this book, variations on this kind of algorithm are used to provide primes for cryptography.

We know, by Fermat's Little Theorem, that in Z_p with p prime, $x^{p-1} \bmod p = 1$ for every x between 1 and $p-1$. What about x^{m-1}, in Z_m, when m is not prime?

Exercise 2.4-5 Suppose x is a member of Z_m that has no multiplicative inverse. Is it possible that $x^{n-1} \bmod n = 1$?

We answer the question of the exercise in our next lemma.

Lemma 2.25 *Let m be a non-prime, and let x be a number in Z_m which has no multiplicative inverse. Then $x^{m-1} \bmod m \neq 1$.*

Proof: Assume, for the purpose of contradiction, that

$$x^{m-1} \bmod m = 1.$$

Then

$$x \cdot x^{m-2} \bmod m = 1.$$

But then $x^{m-2} \bmod m$ is the inverse of x in Z_m, which contradicts the fact that x has no multiplicative inverse. Thus it must be the case that $x^{m-1} \bmod m \neq 1$. ■

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:

[8] G.L. Miller. "Riemann's Hypothesis and tests for primality," J. Computer and Systems Science **13**, 1976, pp 300-317.

[9] M. O. Rabin. "Probabilistic algorithm for testing primality." Journal of Number Theory, **12**, 1980. pp 128-138.

```
(1)  PrimeTest(m)
(2)  choose a random number x, 2 ≤ x ≤ m − 1.
(3)  compute y = x^{m−1} mod m
(4)  if (y = 1)
(5)       output `` m might be prime''
(6)  else
(7)       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) for t 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 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*; 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 12 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? See Problem 13 at the end of this section for a hint.)

In fact, the Miller-Rabin algorithm modifies the test slightly (in a way that we won't describe here[10]) so that for any non-prime m, at least half of the possible values we could choose for x will 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, and assert that an x which passes these t tests is prime, the probability of being wrong is actually 2^{-t}. So, if we repeat the test 10 times, we have only about a 1 in a thousand chance of making a mistake, and if we repeat it 100 times, we have only about 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.)

A number theory theorem called the Prime Number Theorem tells us that if we check about $\log_e n$ numbers near n we can expect one of them to be prime. A d digit number is at least 10^{d-1}

[10]See, for example, Cormen, Leiserson, Rivest and Stein, Introduction to Algorithms, McGraw Hill/MIT Press, 2002

and less than 10^d, so its natural logarithm is between $(d-1)\log_e 10$ and $d\log_e 10$. if we want to find a d digit prime, we can take any d digit number and test about $d\log_e 10$ numbers near it for primality, and it is reasonable for us to expect that one of them will turn out to be prime. The number $\log_e 10$ is 2.3 to two decimal places. Thus it does not take a really large amount of time to find two prime numbers with a hundred (or so) digits each.

Important Concepts, Formulas, and Theorems

1. *Exponentiation.* To perform exponentiation mod n efficiently, we use repeated squaring, and take mods after each arithmetic operation.

2. *Security of RSA.* The security of RSA rests on the fact that no one has developed an efficient algorithm for factoring, or for finding x, given $x^e \bmod n$.

3. *Fermat's Little Theorem does not hold for composites.* Let m b e a non-prime, and let x be a number in Z_n which has no multiplicative inverse. Then $x^{m-1} \bmod m \neq 1$.

4. *Testing numbers for primality.* The randomized Miller-Rabin algorithm will tell you almost surely if a given number is prime.

5. *Finding prime numbers.* By applying the randomized Miller-Rabin to $d\ln 10$ (which is about $2.3d$) numbers with d digits, you can expect to find at least one that is prime.

Problems

1. What is 3^{1024} in Z_7? (This is a straightforward 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. A gigabyte is one billion bytes; a terabyte is one trillion bytes. A byte is eight bits, each a zero or a 1. Since $2^{10} = 1024$, which is about 1000, we can store about three digits (any number between 0 and 999) in ten bits. About how many decimal digits could we store in a five gigabytes of memory? About how many decimal digits could we store in five terabytes of memory? How does this compare to the number 10^{120}? To do this problem it is reasonable to continue to assume that 1024 is about 1000.

4. Find all numbers a different from 1 and -1 (which is the same as 8) such that $a^8 \bmod 9 = 1$.

5. Use a spreadsheet, programmable calculator or computer to find all numbers a different from 1 and $-1 \bmod 33 = 32$ with $a^{32} \bmod 33 = 1$. (This problem is relatively straightforward to do with a spreadsheet that can compute mods and will let you "fill in" rows and columns with formulas. However you do have to know how to use the spreadsheet in this way to make it strightforward!)

6. How many digits does the 10^{120}th power of 10^{100} have?

7. 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? Does the answer depend on what a actually is rather than the number of digits it has?

8. Explain what our outline of the solution to Exercise 2.4-1 has to do with the binary representation of 10^{120}.

9. Give careful pseudocode to compute $a^x \bmod n$. Make your algorithm as efficient as possible. You may use right shift n in your algorithm.

10. 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?

11. Number theorists use $\varphi(n)$ to stand for the number of elements of Z_n that have inverses. Suppose we want to compute $a^{e_1 e_2 \cdots e_m} \bmod n$. Would it make sense for us to reduce our exponents mod $\varphi(n)$ as we compute their product? Why?

12. Show that if m is not prime, then at least $\sqrt{m}$ elements of Z_m do not have multiplicative inverses.

13. Show that in Z_{p+1}, where p is prime, only one element passes the primality test $x^{m-1} = 1 \pmod m$. (In this case, $m = p + 1$.)

14. 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.

15. 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.

    Applying the extended GCD algorithm, or just by experimenting we see that $d = 17$. $n = pq = 253$. Then

$$\begin{aligned} 100^{13} \bmod 253 &= 10^{26} \bmod 253 = (-12)^8 \cdot 100 \bmod 253 \\ &= (100(12^4 \bmod 253)(12^4 \bmod 253)) \bmod 253 = 133. \end{aligned}$$

    To reverse the process,

$$133^{17} = (((133^4 \bmod 253)^4 \bmod 253) \cdot 133) \bmod 253 = 210 \cdot 133 \bmod 253 = 100.$$

16. 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. The document itself may be public; it is your signature that we are trying to protect. Digital signatures are, in a way, the opposite of encryption, as if Bob wants to sign a message, he first applies his 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 of the document (and knows your public key) should be able to determine that you signed it.

Chapter 3

Reflections on Logic and Proof

In this chapter, we cover some basic principles of logic and describe some methods for constructing proofs. This chapter is not meant to be a complete enumeration of all possible proof techniques. The philosophy of this book is that most people learn more about proofs by reading, watching, and attempting proofs than by an extended study of the logical rules behind proofs. On the other hand, now that we have some examples of proofs, it will help you read and do proofs if we reflect on their structure and to discuss what constitutes a proof. To do so so we first develop a language that will allow us to talk about proofs, and then we use this language to describe the logical structure of a proof.

3.1 Equivalence and Implication

Equivalence of statements

Exercise 3.1-1 A group of students are working on a project that involves writing a merge sort program. Joe and Mary have each written an algorithm for a function that takes two lists, `List1` and `List2`, of lengths p and q and merges them into a third list, `List3`. Part of Mary's algorithm is the following:

```
(1)  if ((i + j ≤ p + q)  &&  (i ≤ p)  &&  ((j ≥ q)||(List1[i] ≤ List2[j])))
(2)            List3[k] = List1[i]
(3)            i = i + 1
(4)  else
(5)            List3[k] = List2[j]
(6)            j = j + 1

(7)  k = k + 1
(8)  Return List3
```

The corresponding part of Joe's algorithm is

```
(1)  if (((i + j ≤ p + q) && (i ≤ p) && (j ≥ q))
       ||  ((i + j ≤ p + q) && (i ≤ p) && (List1[i] ≤ List2[j])))
```

```
(2)            List3[k] = List1[i]
(3)            i = i + 1
(4)  else
(5)            List3[k] = List2[j]
(6)            j = j + 1

(7)  k = k + 1
(8)  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 in line 1. (How convenient; they even used the same local variables!) 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 `List1`$[i] \leq$ `List2`$[j]$),

while 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 ($i+j \leq p+q$ 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 we can name these constituent parts and rewrite the statements. We 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`$[j]$

The condition in Mary's if statement on Line 1 of her code becomes

s and t and (u or v)

while Joe's if statement on Line 1 of his code becomes

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

By recasting the statements in this symbolic form, we see that s and t always appear together as "s and t." We can thus simplify their expressions by substituting w for "s and t." Mary's condition now has the form

w and (u or v)

and Joe's has the form

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

Although we can argue, based on our knowledge of the structure of the English language, that Joe's statement and Mary's statement are saying the same thing, it will help us understand logic if we formalize the idea of "saying the same thing." If you look closely at Joe's and Mary's statements, you can see that we are saying that, the word "and" distributes over the word "or," just as set intersection distributes over set union, and multiplication distributes over addition. In order to analyze when statements mean the same thing, and explain more precisely what we mean when we say something like "and" distributes over "or," logicians have adopted a standard notation 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

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

We now have a nice notation (which makes our compound statements look a lot like the two sides of the distributive law for intersection of sets over union), but we have not yet explained why two statements with this symbolic form mean the same thing. We must therefore 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 from the following notation:

- symbols (s, t, etc.) standing for statements (these will be called *variables*),
- 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."

Truth tables

We will develop a theory for deciding when a compound statement is true based on the truth or falsity of its component statements. Using this theory, we will determine, for a particular setting of variables, 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 uses truth tables, which you have probably seen before. We will see how truth tables are the proper tool to determine whether two statements are equivalent.

As with arithmetic, the order of operations in a logical statement is important. 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. The symbol $\neg$ always has the highest priority, which means that when we wrote $\neg u \vee (s \wedge t)$, we meant $(\neg u) \vee (s \wedge t)$, rather than $\neg(u \vee (s \wedge t))$. The principle we use here is simple; the symbol $\neg$

applies to the smallest number of possible following symbols needed for it to make sense. This is the same principle we use with minus signs in algebraic expressions. 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 operators $\wedge$, $\vee$, $\oplus$ and $\neg$ are called *logical connectives*. 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. The the truth tables for the connectives we have mentioned so far are in Figure 3.1

Figure 3.1: The truth tables for the basic logical connectives.

AND

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

OR

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

XOR

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

NOT

s	$s \oplus t$
T	F
F	T

These truth tables define the words "and," "or," "exclusive or" ("xor" for short), and "not" in the context of symbolic compound statements. For example, 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.

We have been thinking of s and t as variables that stand for statements. The purpose of a truth table is to define when a compound statement is true or false in terms of when its component statements are true and false. Since we focus on just the truth and falsity of our statements when we are giving truth tables, we can also think of s and t as variables that can take on the values "true" (T) and "false" (F). We refer to these values as the *truth values* of s and t. Then a truth table gives us the truth values of a compound statement in terms of the truth values of the component parts of the compound statement. The statements $s \wedge t$, $s \vee t$ and $s \oplus t$ each have two component parts, s and t. Because there are two values we can assign to s, and for each value we assign to s there are two values we can assign to t, by the product principle, there are $2 \cdot 2 = 4$ ways to assign truth values to s and t. Thus we have four rows in our truth table, one for each way of assigning truth values to s and t.

For a more complex compound statement, such as the one in Line 1 in Joe and Mary's programs, we still want to describe situations in which the statement is true and situations in

Table 3.1: The truth table for Joe's statement

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

which the statement 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 use a variable to represent the truth value each symbolic statement. The truth table has one column for each of the original variables, and for each of the pieces we use to build up the compound statement. The truth table has one row for each possible way of assigning truth values to the original variables. Thus if we have two variables, we have, as above, four rows. If we have just one variable, then we have, as above, just two rows. If we have three variables then we will have $2^3 = 8$ rows, and so on.

In Table 3.1 we give the truth table for the symbolic statement that we derived from Line 1 of Joe's algorithm. The columns to the left of the double line contain the possible truth values of the variables; the columns to the right correspond to various sub-expressions whose truth values we need to compute. We give the truth table 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.

In Table 3.2 we give the truth table for the statement that we derived from Line 1 of Mary's algorithm.

Table 3.2: The truth table for Mary's 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

You will notice that the pattern of T's and F's that we used to the left of the double line in both Joe's and Mary's truth tables are the same—namely, reverse alphabetical order.[1] Thus

[1]Alphabetical order is sometimes called *lexicographic order*. Lexicography is the study of the principles and

row i of Table 3.1 represents exactly the same assignment of truth values to u , v, and w as row i of Table 3.2. The final columns of the two truth tables are identical, which means that Joe's symbolic statement and Mary's symbolic statement are true in exactly the same cases. Therefore, the two statements must say the same thing, and 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. Alternatively, two statements 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).

Tables 3.1 and 3.2 actually prove a *distributive law*:

Lemma 3.1 *The statements*

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

and

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

are equivalent.

DeMorgan's Laws

Exercise 3.1-2 *DeMorgan's Laws* say that $\neg(p \vee q)$ is equivalent to $\neg p \wedge \neg q$, and that $\neg(p \wedge q)$ is equivalent to $\neg p \vee \neg q$,. Use truth tables to demonstrate that DeMorgan's laws are correct.

Exercise 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 one of DeMorgan's laws to $\neg(\neg(p \vee q)) \wedge \neg(p \wedge q)$ to find another symbolic statement equivalent to the exclusive or.

To verify the first DeMorgan's Law, we create a pair of truth tables that we have condensed into one "double truth table" in Table 3.3. The second double vertical line separates the computation of the truth values of $\neg(p \vee q)$ and $\neg p \wedge \neg q$ We see that the fourth and the last columns are identical,

Table 3.3: Proving the first DeMorgan Law.

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

and therefore the first DeMorgan's Law is correct. We can verify the second of DeMorgan's Laws by a similar process.

To show that $p \oplus q$ is equivalent to $(p \vee q) \wedge \neg(p \wedge q)$, we use the "double truth table" in Table 3.4.

practices used in making dictionaries. Thus you will also see the order we used for the T's and F's called reverse lexicographic order, or reverse lex order for short.

Table 3.4: An equivalent statement to $p \oplus q$.

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 applying DeMorgan's law to $\neg(\neg(p \vee q)) \wedge \neg(p \wedge q)$, we see that $p \oplus q$ is also equivalent to $\neg(\neg(p \vee q) \vee (p \wedge q))$. It was easier to use DeMorgan's law to show this equivalence than to use another double truth table.

Implication

Another kind of compound statement occurs frequently in mathematics and computer science. Recall 2.21, Fermat's Little Theorem:

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

Fermat's Little Theorem combines two constituent statements,

> p is a prime

and

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

We can also restate Fermat's Little Theorem (a bit clumsily) as

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

or

> p is a prime implies $a^{p-1} \bmod p = 1$ for each non-zero $a \in Z_p$,

or

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

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 the four statements of Fermat's Little Theorem 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."

We summarize this discussion in the following definition:

Definition 3.1 *The following four English phrases are intended to mean the same thing. In other words, they are defined by the same truth table:*

- *s implies t,*
- *if s then t,*
- *t if s, and*
- *s only if t.*

Observe that the use of "only if" may seem a little different than the normal usage in English. Also observe that there are still other ways of making an "if ... then" statement in English. In a number of our lemmas, theorems, and corollaries (for example, Corollary 2.6 and Lemma 2.5) we have had two sentences. In the first we say "Suppose ...." In the second we say "Then ...." The two sentences "Suppose s." and "Then t." are equivalent to the single sentence $s \Rightarrow t$. When we have a statement equivalent to $s \Rightarrow t$, we call the statement s the *hypothesis* of the implication and we call the statement t the *conclusion* of the implication.

If and only if

The word "if" and the phrase "only if" frequently appear together in mathematical statements. For example, in Theorem 2.9 we proved

> A number a has a multiplicative inverse in Z_n if and only if there are integers x and y such that $ax + ny = 1$.

Using s to stand for the statement "a number a has a multiplicative inverse in Z_n" and t to stand for the statement "there are integers x and y such that $ax+ny = 1$," we can write this statement symbolically as

> s if and only if t.

Referring to Definition 3.1, we parse this as

> s if t, and s only if t,

which again by the definition above is the same as

> $s \Rightarrow t$ and $t \Rightarrow s$.

We denote the statement "s if and only if t" by $s \Leftrightarrow t$. Statements of the form $s \Rightarrow t$ and $s \Leftrightarrow t$ are called *conditional statements*, and the connectives $\Rightarrow$ and $\Leftrightarrow$ are called *conditional connectives*.

Exercise 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. In particular if s is true and t is true, then we would say that both $s \Rightarrow t$ and $s \Leftrightarrow t$ are true. If s is true and t is false, we would say that both $s \Rightarrow t$ and $s \Leftrightarrow t$ are false. In the case that both s and t are false we would say that $s \Leftrightarrow t$ is true. What about $s \Rightarrow t$? Let us try an example. Suppose that s is the statement "it is supposed to rain" and t is the statement "I carry an umbrella." Then if, on a given day, it is not supposed to rain and I do not carry an umbrella, we would say that the statement "if it is supposed to rain then I carry an umbrella" is true on that day. This suggests that we also want to say $s \Rightarrow t$ is true if s is false and t is false.[2] Thus the truth tables are identical in rows one, two, and four. For "implies" and "if and only if" to mean different things, the truth tables must therefore be different in row three. Row three is the case where s is false and t is true. Clearly in this case we would want s if and only if t to be false, so our only choice is to say that $s \Rightarrow t$ is true in this case. This gives us the truth tables in Figure 3.2.

Figure 3.2: The truth tables for "implies" and for "if and only if."

IMPLIES

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

IF AND ONLY IF

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

Here is another place where (as with the usage for "or") 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"; 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.") In particular, the truth table marked IMPLIES is the truth table referred to in Definition 3.1. This truth table thus defines the mathematical meaning of s implies t, or any of the other three statements referred to in that definition.

Some people have difficulty using the truth table for $s \Rightarrow t$ because of this ambiguity in English. The following example can be helpful in resolving this ambiguity. Suppose that I hold

[2]Note that we are making this conclusion on the basis of one example. Why can we do so? We are not trying to prove something, but trying to figure out what the appropriate definition is for the $\Rightarrow$ connective. Since we have said that the truth or falsity of $s \Rightarrow t$ depends only on the truth or falsity of s and t, one example serves to lead us to an appropriate definition. If a different example led us to a different definition, then we would want to define two different kinds of implications, just as we have two different kinds of "ors," $\vee$ and $\oplus$. Fortunately, the only kinds of conditional statements we need for doing mathematics and computer science are "implies" and "if and only if."

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*

Principle 3.1 *A statement is true exactly when it is not false.*

This principle tells us that that my statement is true in the three cases where you wouldn't say I lied. We used this principle implicitly before when we introduced the principle of proof by contradiction, Principle 2.1. We were explaining the proof of Corollary 2.6, which states

> Suppose there is a b in Z_n such that the equation
>
> $$a \cdot_n x = b$$
>
> does not have a solution. Then a does not have a multiplicative inverse in Z_n.

We had assumed that the hypothesis of the corollary was true so that $a \cdot_n x = b$ does not have a solution. Then we assumed the conclusion that a does not have a multiplicative inverse was false. We saw that these two assumptions led to a contradiction, so that it was impossible for both of them to be true. Thus we concluded whenever the first assumption was true, the second had to be false. Why could we conclude this? Because the principle of the excluded middle says that the second assumption has to be either true or false. We didn't introduce the principle of the excluded middle at this point for two reasons. First, we expected that the reader would agree with our proof even if we didn't mention the principle, and second, we didn't want to confuse the reader's understanding of proof by contradiction by talking about two principles at once!

Important Concepts, Formulas, and Theorems

1. *Logical statements.* Logical statements may be built up from the following notation:
   - symbols (s, t, etc.) standing for statements (these will be called *variables*),
   - the symbol $\wedge$, standing for "and,"
   - the symbol $\vee$, standing for "or,"
   - the symbol $\oplus$ standing for "exclusive or,"
   - the symbol $\neg$, standing for "not,"

- the symbol $\Rightarrow$, standing for "implies," and
- the symbol $\Leftrightarrow$, standing for "if and only if."

The operators $\wedge$, $\vee$, $\oplus$, $\Rightarrow$, $\Leftrightarrow$, and $\neg$ are called *logical connectives.* The operators $\Rightarrow$ and $\Leftrightarrow$ are called *conditional connectives.*

2. *Truth Tables.* The following are truth tables for the basic logical connectives:

AND

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

OR

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

XOR

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

NOT

s	$s \oplus t$
T	F
F	T

3. *Equivalence of logical statements.* We say that two symbolic compound statements are *equivalent* if they are true in exactly the same cases.

4. *Distributive Law.* The statements

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

and

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

are equivalent.

5. *DeMorgan's Laws.* DeMorgan's Laws say that $\neg(p \vee q)$ is equivalent to $\neg p \wedge \neg q$, and that $\neg(p \wedge q)$ is equivalent to $\neg p \vee \neg q$.

6. *Implication.* The following four English phrases are equivalent:

- s implies t,
- if s then t,
- t if s, and
- s only if t.

7. *Truth tables for implies and if and only if.*

IMPLIES

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

IF AND ONLY IF

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

8. *Principle of the Excluded Middle.* A statement is true exactly when it is not false.

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. Find at least two more examples of the use of some word or phrase equivalent to "implies" in lemmas, theorems, or corollaries in Chapters One or Two.
3. Find at least two more examples of the use of the phrase "if and only if" in lemmas, theorems, and corollaries in Chapters One or Two.
4. Show that the statements $s \Rightarrow t$ and $\neg s \vee t$ are equivalent.
5. Prove the DeMorgan law which states $\neg(p \wedge q) = \neg p \vee \neg q$.
6. Show that $p \oplus q$ is equivalent to $(p \wedge \neg q) \vee (\neg p \wedge q)$.
7. Give a simplified form of each of the following expressions (using T to stand for a statement that is always true and F to stand for a statement that is always false)[3]:
   - $s \vee s$,
   - $s \wedge s$,
   - $s \vee \neg s$,
   - $s \wedge \neg s$.
8. Use a truth table to show that $(s \vee t) \wedge (u \vee v)$ is equivalent to $(s \wedge u) \vee (s \wedge v) \vee (t \wedge u) \vee (t \wedge v)$. What algebraic rule is this similar to?
9. Use DeMorgan's Law, the distributive law, and Problems 7 and 8 to show that $\neg((s \vee t) \wedge (s \vee \neg t))$ is equivalent to $\neg s$.
10. 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).
11. 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).
12. 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.
13. Suppose that for each line of a 2-variable truth table, you are told whether the final column in that line should evaluate to true or to false. (For example, you might be told that the final column should contain T, F, T, and F in that order.) Explain how to create a logical statement using the symbols s, t, $\wedge$, $\vee$, and $\neg$ that has that pattern as its final column. Can you extend this procedure to an arbitrary number of variables?

[3] A statement that is always true is called a *tautology*; a statement that is always false is called a *contradiction*

14. In Problem 13, your solution may have used $\wedge$, $\vee$ and $\neg$. Is it possible to give a solution using only one of those symbols? Is it possible to give a solution using only two of these symbols?

15. 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

Variables and 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. For example, in some programming languages we may declare a variable to be a "boolean" or an "integer" or a "real."[4] In English and in mathematics, we also make statements about variables, but it is not always clear which words are being used as variables and what values these variables may take on. We use the phrase *varies over* to describe the set of values a variable 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. 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 $m = nq + r$." 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 possible values for a variable the *universe* of that variable.

In the statement "m is an even integer," it is clear that m is a variable, but the universe is not given. It might be the integers, just the even integers, or the rational numbers, or one of many other sets. The choice of the universe is crucial for determining the truth or falsity of a statement. If we choose the set of integers as the universe for m, then the statement is true for some integers and false for others. On the other hand, if we choose integer multiples of 10 as our universe, then the statement is always true. 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 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. The truth value of the statement is determined only after such a substitution.

Exercise 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, the universe must be clearly stated. Second, a statement about a variable can be true for some values of a variable and false for others.

[4]Note that to declare a variable x as an integer in, say, a C program does not mean that same thing as saying that x is an integer. In a C program, an integer may really be a 32-bit integer, and so it is limited to values between $2^{31} - 1$ and -2^{31}. Similarly a real has some fixed precision, and hence a real variable y may not be able to take on a value of, say, 10^{-985}.

Quantifiers

In contrast, the statement

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

is false; we do not need to qualify our answer by saying that it is true some of the time and false at other times. To determine whether Statement 3.1 is true or false, we could substitute various values for m into the simpler statement $m^2 > m$, and decide, for each of these values, whether the statement $m^2 > m$ is true or false. Doing so, we see that the statement $m^2 > m$ is true for values such as $m = -3$ 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 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 statement is also about the universe of integers, 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*, 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. Recall the following definition of the "big-O" notation you have probably used in earlier computer science courses:

Definition 3.2 *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 $x > n_0$.*

Exercise 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.

Exercise 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.

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

As Exercise 3.2-2 points out, English has many different ways to express quantifiers. For example, the sentences, "All hammers are tools", "Each sandwich is delicious", "No one in their right mind would do that", "Somebody loves me", and "Yes Virginia, there is a Santa Claus" all contain quantifiers. For Exercise 3.2-3, we can say "It is not the case that there is a child who wants a toothache." Our quantifier is the phrase "there is."

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 $m^2 > m$) 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 give more details about how to show a statement about a variable is true or false for every member of our universe later in this section.

Mathematical statements of theorems, lemmas, and corollaries often have quantifiers. For example in Lemma 2.5 the phrase "for any" is a quantifier, and in Corollary 2.6 the phrase "there is" is a quantifier.

Standard notation for quantification

Each of the many variants of language that describe quantification 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.

All quantified statements have one of these two forms. We use the standard shorthand of $\forall$ for the phrase "for all" and the standard shorthand of $\exists$ for the phrase "there exists." We also adopt the convention that we parenthesize the expression that is subject to the quantification. For example, using Z to stand for the universe of all integers, we 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 $\forall$.

As another example, we rewrite the definition of the "Big Oh" notation with these symbols. 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 statement is identical to the definition of "big Oh" that we gave earlier in Definition 3.2, except for more precision as to what c and n_0 actually are.

Exercise 3.2-5 How would you rewrite Euclid's division theorem, Theorem 2.12 using the shorthand notation we have introduced for quantifiers? Use Z^+ to to stand for the positive integers and N to stand for the nonnegative integers.

We can rewrite Euclid's division theorem as

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

Statements about variables

To talk about statements about variables, we need a notation to use for such statements. For example, we can use $p(n)$ to stand for the statement $n^2 > n$. Now, we can say that $p(4)$ and $p(-3)$ are true, while $p(1)$ and $p(.5)$ are false. 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 $\exists c(\exists n_0(\forall x(p(c, n_0, x))))$, where $p(c, n_0, x)$ 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.)

Exercise 3.2-6 Rewrite Euclid's division theorem, using the notation above for statements about variables. Leave out the references to universes so that you can see clearly the order in which the quantifiers occur.

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

Rewriting statements to encompass larger universes

It is sometimes useful to rewrite a quantified statement so that the universe is larger, and the statement itself serves to limit the scope of the universe.

Exercise 3.2-7 Let R to stand for the real numbers and R^+ to stand for the positive real numbers. Consider the following two statements:

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

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

Rewrite these statements so that the universe is all the real numbers, but the statements say the same thing in everyday English that they did before.

For Exercise 3.2-7, 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 can state this rule as a general theorem:

Theorem 3.2 *Let U_1 be a universe, and let U_2 be another universe with $U_1 \subseteq U_2$. Suppose that $q(x)$ is a statement such that*

$$U_1 = \{x |\ q(x) \text{ is true}\}. \tag{3.2}$$

Then if $p(x)$ is a statement about U_2, it may also be interpreted as a statement about U_1, and

(a) $\forall x \in U_1(p(x))$ *is equivalent to* $\forall x \in U_2(q(x) \Rightarrow p(x))$.

(b) $\exists x \in U_1(p(x))$ *is equivalent to* $\exists x \in U_2(q(x) \wedge p(x))$.

Proof: By Equation 3.2 the statement $q(x)$ must be true for all $x \in U_1$ and false for all x in U_2 but not U_1. To prove part (a) we must show that $\forall x \in U_1(p(x))$ is true in exactly the same cases as the statement $\forall x \in U_2(q(x) \Rightarrow p(x))$. For this purpose, suppose first that $\forall x \in U_1(p(x))$ is true. Then $p(x)$ is true for all x in U_1. Therefore, by the truth table for "implies" and our remark about Equation 3.2, the statement $\forall x \in U_2(q(x) \Rightarrow p(x))$ is true. Now suppose $\forall x \in U_1(p(x))$ is false. Then there exists an x in U_1 such that $p(x)$ is false. Then by the truth table for "implies," the statement $\forall x \in U_2(q(x) \Rightarrow p(x))$ is false. Thus the statement $\forall x \in U_1(p(x))$ is true if and only if the statement $\forall x \in U_2(q(x) \Rightarrow p(x))$ is true. Therefore the two statements are true in exactly the same cases. Part (a) of the theorem follows.

Similarly, for Part (b), we observe that if $\exists x \in U_1(p(x))$ is true, then for some $x' \in U_1$, $p(x')$ is true. For that x', $q(x')$ is also true, and hence $p(x') \wedge q(x')$ is true, so that $\exists x \in U_2(q(x) \wedge p(x))$ is true as well. On the other hand, if $\exists x \in U_1(p(x))$ is false, then no $x \in U_1$ has $p(x)$ true. Therefore by the truth table for "and" $q(x) \wedge p(x)$ won't be true either. Thus the two statements in Part (b) are true in exactly the same cases and so are equivalent. ■

Proving quantified statements true or false

Exercise 3.2-8 Let R stand for the real numbers and R^+ 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) $\exists x \in R(x \geq 0 \wedge \forall y \in R^+(y > x))$

In Exercise 3.2-8, 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 real number x there is a real number y bigger than x, which we know is true. Statement (d) says that every y in R is larger than every x in R, and so it is false. 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.

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

Principle 3.2 (The meaning of quantified statements)

- *The statement* $\exists x \in U(p(x))$ *is true if there is at least one value of* x *in* U *for which the statement* $p(x)$ *is true.*
- *The statement* $\exists x \in U(p(x))$ *is false if there is no* $x \in U$ *for which* $p(x)$ *is true.*
- *The statement* $\forall x \in U(p(x))$ *is true if* $p(x)$ *is true for each value of* x *in* U*.*
- *The statement* $\forall x \in U(p(x))$ *is false if* $p(x)$ *is false for at least one value of* x *in* U*.*

Negation of quantified statements

An interesting connection between $\forall$ and $\exists$ arises from the negation of statements.

Exercise 3.2-9 What does the statement "It is not the case that for all integers n, $n^2 > 0$" mean?

From our knowledge of English we see that since the statement $\neg\forall n \in Z(n^2 > 0)$ asserts that it is not the case that, for all integers n, we have $n^2 > 0$, there must be some integer n such that $n^2 \not> 0$. In other words, it says there is some integer n such that $n^2 \leq 0$. 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. By considering the case when $p(x)$ is true for all $x \in U$, (we call this case "always true") and the case when $p(x)$ is false for at least one $x \in U$ (we call this case "not always true") we can analyze the equivalence. The theorem that follows, which formalizes the example above in which $p(x)$ was the statement $x^2 > 0$, is proved by dividing these cases into two possibilities.

Theorem 3.3 *The statements* $\neg\forall x \in U(p(x))$ *and* $\exists x \in U(\neg p(x))$ *are equivalent.*

Proof: Consider the following table which we have set up much like a truth table, except that the relevant cases are not determined by whether $p(x)$ is true or false, but by whether $p(x)$ is true for all x in the universe U or not.

$p(x)$	$\neg p(x)$	$\forall x \in U(p(x))$	$\neg\forall x \in U(p(x))$	$\exists x \in U(\neg p(x))$
always true	always false	true	false	false
not always true	not always false	false	true	true

Since the last two columns are identical, the theorem holds. ■

Corollary 3.4 *The statements* $\neg\exists x \in U(q(x))$ *and* $\forall x \in U(\neg q(x))$ *are equivalent.*

Proof: Since the two statements in Theorem 3.3 are equivalent, their negations are also equivalent. We then substitute $\neg q(x)$ for $p(x)$ to prove the corollary. ■

Put another way, to negate a quantified statement, you switch the quantifier and "push" the negation inside.

To deal with the negation of more complicated statements, we simply take them one quantifier at a time. Recall Definition 3.2, the definition of big Oh notation,

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

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

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

After one application of Corollary 3.4 we get

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

After another application of Corollary 3.4 we obtain

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

Now we apply Theorem 3.3 and obtain

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

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

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

Thus $f(x)$ is *not* $O(g(x))$ if for every c in $R+$ and every n_0 in R^+, there is an x such that $x > n_0$ and $f(x) \not\leq cg(x)$.

In our next exercise, we use the "Big Theta" notation defined as follows:

Definition 3.3 *$f(x) = \Theta(g(x))$ means that $f(x) = O(g(x))$* **and** *$g(x) = O(f(x))$.*

Exercise 3.2-10 Express $\neg(f(x) = \Theta(g(x)))$ in terms similar to those we used to describe $f(x) \neq O(g(x))$.

Exercise 3.2-11 Suppose the universe for a statement $p(x)$ 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.

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 3.2-11 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

Exercise 3.2-12 Are there any quantifiers in the statement "The sum of even integers is even?"

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 in Exercise 3.2-12, we said we were stating a more general fact about the 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. In fact, we frequently rely on context in writing mathematical statements with implicit quantifiers because, in context, it makes the statements easier to read. For example, in Lemma 2.8 we said

The equation

$$a \cdot_n x = 1$$

has a solution in Z_n if and only if there exist integers x and y such that

$$ax + ny = 1.$$

In context it was clear that the a we were talking about was an arbitrary member of Z_n. It would simply have made the statement read more clumsily if we had said

For every $a \in Z_n$, the equation

$$a \cdot_n x = 1$$

has a solution in Z_n if and only if there exist integers x and y such that

$$ax + ny = 1.$$

On the other hand, we were making a transition from talking about Z_n to talking about the integers, so it was important for us to include the quantified statement "there exist integers x and y such that $ax+ny = 1$." More recently in Theorem 3.3, we also did not feel it was necessary to say "For all universes U and for all statements p about U," at the beginning of the theorem. We felt the theorem would be easier to read if we kept those quantifiers implicit and let the reader (not necessarily consciously) infer them from context.

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 how we would prove this statement? 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 by definition $m + n$ 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 that 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 allowed us to 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 of 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.

Important Concepts, Formulas, and Theorems

1. *Varies over.* We use the phrase *varies over* to describe the set of values a variable may take on.

2. *Universe.* We call the set of possible values for a variable the *universe* of that variable.

3. *Free variables.* Variables that are not constrained in any way whatever are called *free variables.*

4. *Quantifier.* A phrase that converts a symbolic statement about potentially any member of our universe into a statement about the universe instead is called a *quantifier.* There are two types of quantifiers:

   - *Universal 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.*
   - *Existential quantifier.* A quantifier that asserts a statement about a variable is true for at least one value of the variable in its universe is called an *existential quantifier.*

5. *Larger universes.* Let U_1 be a universe, and let U_2 be another universe with $U_1 \subseteq U_2$. Suppose that $q(x)$ is a statement such that

   $$U_1 = \{x|\ q(x) \text{ is true}\}.$$

   Then if $p(x)$ is a statement about U_2, it may also be interpreted as a statement about U_1, and

   (a) $\forall x \in U_1(p(x))$ is equivalent to $\forall x \in U_2(q(x) \Rightarrow p(x))$.
   (b) $\exists x \in U_1(p(x))$ is equivalent to $\exists x \in U_2(q(x) \wedge p(x))$.

6. *Proving quantified statements true or false.*

   - The statement $\exists x \in U(p(x))$ is true if there is at least one value of x in U for which the statement $p(x)$ is true.

- The statement $\exists x \in U(p(x))$ is false if there is no $x \in U$ for which $p(x)$ is true.
- The statement $\forall x \in U(p(x))$ is true if $p(x)$ is true for each value of x in U.
- The statement $\forall x \in U(p(x))$ is false if $p(x)$ is false for at least one value of x in U.

7. *Negation of quantified statements.* To negate a quantified statement, you switch the quantifier and push the negation inside.
   - The statements $\neg\forall x \in \mathrm{U}(p(x))$ and $\exists x \in U(\neg p(x))$ are equivalent.
   - The statements $\neg\exists x \in \mathrm{U}(p(x))$ and $\forall x \in U(\neg p(x))$ are equivalent.
8. *Big-Oh* 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 $x > n_0$.
9. *Big-Theta.* $f(x) = \Theta(g(x))$ means that $f = O(g(x))$ **and** $g = O(f(x))$.
10. *Some notation for sets of numbers.* We use R to stand for the real numbers, R^+ to stand for the positive real numbers, Z to stand for the integers (positive, negative, and zero), Z^+ to stand for the positive integers, and N to stand for the nonnegative integers.

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 it 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 a variable or some variables for its factors.)
5. Write down the definition of a greatest common divisor of m and n in such a way that 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 $x < y$? (You need not include references to the universes for the variables.)
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 (implicit) quantifiers in the statement "The product of odd integers is odd?" If so, what are they?

9. Rewrite the statement "The product of odd integers is odd," with all quantifiers (including any in the definition of odd integers) 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. Consider the following slight modification of Theorem 3.2. For each part below, either prove that it is true or give a counterexample.

    Let U_1 be a universe, and let U_2 be another universe with $U_1 \subseteq U_2$. Suppose that $q(x)$ is a statement such that $U_1 = \{x |\ q(x) \text{ is true}\}$.

    (a) $\forall x \in U_1(p(x))$ is equivalent to $\forall x \in U_2(q(x) \wedge p(x))$.

    (b) $\exists x \in U_1(p(x))$ is equivalent to $\exists x \in U_2(q(x) \Rightarrow p(x))$.

12. 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.)

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

14. 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?

15. 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 "everyday 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 two even numbers is even. That proof contained several crucial ingredients. First, we introduced symbols for members of the universe of integers. 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? There are many possible answers to this question, but 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 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 were even. We then used the definition of even numbers, and, as our previous parenthetic comment suggests, it was natural 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).

Principle 3.3 (Direct inference) *From p and $p \Rightarrow q$ we may conclude q.*

This common-sense principle is a cornerstone of logical arguments. But why is it true? In Table 3.5 we take another look at the truth table for implication.

Table 3.5: Another look at implication

p	q	$p \Rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

The only line which has a T in both the p column and the $p \Rightarrow q$ column is the first line. In this line q is true also, and we therefore conclude that if p and $p \Rightarrow q$ hold then q must hold also. While this may seem like a somewhat "inside out" application of the truth table, it is simply a different way of using a truth table.

There are quite a few rules (called rules of inference) like the principle of direct inference that people commonly use in proofs without explicitly stating them. Before beginning a formal study of rules of inference, we complete our analysis of which rules we used in the proof that the sum of two even integers is even. After concluding that $m = 2i$ and $n = 2j$, we next 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 to say that $m + n$ was even. We then used a rule of inference which says

Principle 3.4 (Conditional Proof) *If, by assuming p, we may prove q, then the statement $p \Rightarrow q$ is true.*

Using this principle, we reached the conclusion that if m and n are even integers, then $m + n$ is an even integer. In order to conclude that this statement is true for all integers m and n, we used another rule of inference, one of the more difficult to describe. We originally 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

Principle 3.5 (Universal Generalization) *If we can prove a statement about x by assuming x is a member of our universe, then we can conclude the statement 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 for direct proofs

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 *direct proof* consists of a sequence of statements, each of which is either a hypothesis[5], a generally accepted fact, or the result of one of the following rules of inference for compound statements.

Rules of Inference for Direct Proofs

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 $\forall x \in U(p(x))$.

[5]If we are proving an implication $s \Rightarrow t$, we call s a hypothesis. If we make assumptions by saying "Let ...," "Suppose ...," or something similar before we give the statement to be proved, then these assumptions are hypotheses as well.

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

The first rule is a statement of the principle of the excluded middle as it applies to statements about variables. The next four four rules are in effect a description of the truth tables for "and," "or," "implies" and "if and only if." 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 earlier discussion, is the principle of direct inference, and describes 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, the principle of conditional proof, which is also exemplified earlier, may be regarded as yet another description of one row of the truth table of $p \Rightarrow q$. Rule 9 is the principle of universal generalization, discussed and exemplified earlier. Rule 10 specifies what we mean by the truth of an existentially quantified statement, according to Principle 3.2.

Although some of our rules of inference are redundant, they are useful. For example, we could have written a portion of our proof that the sum of even numbers is even as follows without using Rule 8.

> "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.

Exercise 3.3-1 Prove that if m is even, then m^2 is even. Explain which steps of the proof use one of the rules of inference above.

For Exercise 3.3-1, we can mimic the proof that the sum of even integers is even.

> Let m be integer. Suppose that m is even. If m is even, then there is a k with $m = 2k$. Thus, there is a k such that $m^2 = 4k^2$. Therefore, there is an integer $h = 2k^2$ such that $m^2 = 2h$. Thus if m is even, m^2 is even. Therefore, for all integers m, if m is even, then m^2 is even.

In our first sentence we are setting things up to use Rule 9. In the second sentence we are simply stating an implicit hypothesis. In the next two sentences we use Rule 6, the principle of direct inference. When we said "Therefore, there is an integer $h = 2k^2$ such that $m^2 = 2h$," we were simply stating an algebraic fact. In our next sentence we used Rule 8. Finally, we used Rule 9. You might have written the proof in a different way and used different rules of inference.

Contrapositive rule of inference.

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

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

To do Exercise 3.3-2, we construct the double truth table in Table 3.6. Since the columns under $p \Rightarrow q$ and under $\neg q \Rightarrow \neg p$ are exactly the same, we know the two statements are equivalent. This exercise tells us that if we know that $\neg q \Rightarrow \neg p$, then we can conclude that $p \Rightarrow q$. This is

Table 3.6: A double truth table for $p \Rightarrow q$ and $\neg q \Rightarrow \neg p$.

p	q	$p \Rightarrow q$	$\neg p$	$\neg q$	$\neg q \Rightarrow \neg p$
T	T	T	F	F	T
T	F	F	F	T	F
F	T	T	T	F	T
F	F	T	T	T	T

called the *principle of proof by contraposition.*

Principle 3.6 (Proof by Contraposition) *The statement $p \Rightarrow q$ and the statement $\neg q \Rightarrow \neg p$ are equivalent, and so a proof of one is a proof of the other.*

The statement $\neg q \Rightarrow \neg p$ is called the *contrapositive* of the statement $p \Rightarrow q$. The following example demonstrates the utility of the principle of proof by contraposition.

Lemma 3.5 *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 is not greater than 10, n^2 is not greater than 100. Then, by the principle of proof by contraposition, if $n^2 > 100$, n must be greater than 10.■

We adopt Principle 3.6 as a rule of inference, called the *contrapositive* rule of inference.

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

In our proof of the Chinese Remainder Theorem, Theorem 2.24, 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, the major topic of the remainder of this section, 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.

For Exercise 3.3-3, a quick look at the double truth table for $p \Rightarrow q$ and $q \Rightarrow p$ in Table 3.7 demonstrates that these two statements are *not* equivalent. The statement $q \Rightarrow p$ is called the *converse* of $p \Rightarrow q$. Notice that $p \Leftrightarrow q$ is true exactly when $p \Rightarrow q$ and its converse are true. It is surprising how often people, even professional mathematicians, absent-mindedly try to prove the converse of a statement when they mean to prove the statement itself. Try not to join this crowd!

Table 3.7: A double truth table for $p \Rightarrow q$ and $q \Rightarrow p$.

p	q	$p \Rightarrow q$	$q \Rightarrow p$
T	T	T	T
T	F	F	T
F	T	T	F
F	F	T	T

Proof by contradiction

Proof by contrapositive inference is an example of what we call *indirect proof*. We have actually seen another example indirect proof, the principle of proof by contradiction. In our proof of Corollary 2.6 we introduced the principle of proof by contradiction, Principle 2.1. We were trying to prove the statement

> Suppose there is a b in Z_n such that the equation
>
> $$a \cdot_n x = b$$
>
> does not have a solution. Then a does not have a multiplicative inverse in Z_n.

We assumed that the hypothesis that $a \cdot_n x = b$ does not have a solution was true. We also assumed that the conclusion that a does not have a multiplicative inverse was false. We showed that these two assumptions together led to a contradiction. Then, using the principle of the excluded middle, Principle 3.1 (without saying so), we concluded that if the hypothesis is in fact true, then the only possibility was that the conclusion is true as well.

We used the principle again later in our proof of Euclid's Division Theorem. Recall that in that proof 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, by the principle of the excluded middle, our assumption had to be incorrect. We adopt the principle of proof by contradiction (also called the principle of *reduction to absurdity*) as our last rule of inference.

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

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 $x \neq 0$. ■

Any proof that uses one of the indirect methods of inference is called an indirect proof. 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.

Exercise 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.

Exercise 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 $n^2 < 9$. Therefore, by the principle of proof by contradiction, $n < 3$.

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

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

or

$$m^2 = 5n^2.$$

Now m^2 must have an even number of prime factors (counting each prime factor as many times as it occurs) as must n^2. But $5n^2$ has an odd number of prime factors. Thus a product of an even number of prime factors is equal to a product of an odd number of prime factors, which is a contradiction since each positive integer may be expressed uniquely as a product of (positive) prime numbers. Thus by the principle of proof by contradiction, $\sqrt{5}$ is not rational.

Important Concepts, Formulas, and Theorems

1. *Principle of direct inference or modus ponens.* From p and $p \Rightarrow q$ we may conclude q.
2. *Principle of conditional proof.* If, by assuming p, we may prove q, then the statement $p \Rightarrow q$ is true.
3. *Principle of universal generalization.* If we can prove a statement about x by assuming x is a member of our universe, then we can conclude it is true for every member of our universe.
4. *Rules of Inference.* 12 rules of inference appear in this chapter. They are
   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 $\exists x \in U(p(x))$.
   11) From $\neg q(x) \Rightarrow \neg p(x)$ we may conclude $p(x) \Rightarrow q(x)$.
   12) If from assuming $p(x)$ and $\neg q(x)$, we can derive both $r(x)$ and $\neg r(x)$ for some statement r, then we may conclude $p(x) \Rightarrow q(x)$.

5. *Contrapositive of $p \Rightarrow q$.* The contrapositive of the statement $p \Rightarrow q$ is the statement $\neg q \Rightarrow \neg p$.

6. *Converse of $p \Rightarrow q$.* The converse of the statement $p \Rightarrow q$ is the statement $q \Rightarrow p$.

7. *Contrapositive rule of inference.* From $\neg q \Rightarrow \neg p$ we may conclude $p \Rightarrow q$.

8. *Principle of proof by contradiction.* If from assuming p and $\neg q$, we can derive both r and $\neg r$ for some statement r, then we may conclude $p \Rightarrow q$.

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 a 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, then $m + n$ 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 integer such that 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 $f(x)$, $g(x)$ and $h(x)$ are functions from R^+ to R^+ such that $f(x) = O(g(x))$ and $g(x) = O(h(x))$, then $f(x) = O(h(x))$.

Index