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

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

The sum below the horizontal line has $n-1$ terms each equal to n, and 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] < A[i])
(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 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 $A[i] < A[i-1]$? 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?