chapter 5

Analysis Techniques

Remember that time is money.

—Benjamin Franklin (1706–1790)

Time and space are important words in computer science because we want fast algorithms and we want algorithms that don't use a lot of memory. The purpose of this chapter is to study some fundamental techniques and tools that can be used to analyze algorithms for the time and space that they require. Although the study of algorithm analysis is beyond our scope, we'll give a brief introduction to help show the need for the mathematical topics of the chapter.

The topics include techniques for finding closed forms or approximations for sums, counting permutations and combinations, discrete probability, solving recurrences, and comparing functions by rates of growth. Along the way we'll give examples to show how the topics apply to analyzing algorithms.

chapter guide

5.1 (Analyzing Algorithms) introduces techniques for finding the worst-case performance of an algorithm and for finding an optimal algorithm.

5.2 (Summations and Closed Forms) introduces techniques for finding closed forms for sums and for approximating sums that may not have a closed form.

5.3 (Permutations and Combinations) introduces basic counting techniques for permutations and combinations.

5.4 (Discrete Probability) introduces the ideas of discrete probability. We'll apply the results to the average-case performance of algorithms. We'll also introduce Markov chains and apply the results to software development.

5.5 (Solving Recurrences) introduces techniques for solving recurrences that crop up when analyzing an algorithm. We'll see solutions to recurrences that arise from some divide-and-conquer algorithms. We'll introduce generating functions and see how they can be used to solve recurrences.

5.6 (Comparing Rates of Growth) introduces approximation techniques for comparing the rates of growth of functions. We'll apply the results to functions that describe approximate running times of algorithms. We'll see some approximations for solutions to a variety of divide-and-conquer recurrences.

5.1 Analyzing Algorithms

An important question of computer science is: Can you convince another person that your algorithm is efficient? This takes some discussion. Let's start by stating the following problem.

The Optimal Algorithm Problem

Suppose algorithm A solves problem P. Is A the best solution to P?

What does "best" mean? Two typical meanings are *least time* and *least space*. In either case, we still need to clarify what it means for an algorithm to solve a problem in the least time or the least space. For example, an algorithm running on two different machines may take different amounts of time. Do we have to compare A to every possible solution of P on every type of machine? This is impossible. So we need to make a few assumptions in order to discuss the optimal algorithm problem. We'll concentrate on "least time" as the meaning of "best" because time is the most important factor in most computations.

5.1.1 Worst-Case Running Time

Instead of executing an algorithm on a real machine to find its running time, we'll analyze the algorithm by counting the number of certain operations that it will perform when executed on a real machine. In this way we can compare two algorithms by simply comparing the number of operations of the same type that each performs. If we make a good choice of the type of operations to count, we should get a good measure of an algorithm's performance. For example, we might count addition operations and multiplication operations for a numerical problem. On the other hand, we might choose to count comparison operations for a sorting problem.

The number of operations performed by an algorithm usually depends on the size or structure of the input. The size of the input again depends on the problem. For example, for a sorting problem, "size" usually means the number of items to be sorted. Sometimes inputs of the same size can have different structures that affect the number of operations performed. For example, some sorting algorithms perform very well on an input data set that is all mixed up but perform badly on an input set that is already sorted!

Because of these observations, we need to define the idea of a worst-case input for an algorithm A. An input of size n is a *worst-case input* if, when

compared to all other inputs of size n, it causes A to execute the largest number of operations. Now let's get down to business. For any input I we'll denote its size by size(I), and we'll let time(I) denote the number of operations executed by A on I. Then the *worst-case function* for A is defined as follows:

$$W_A(n) = \max\{\text{time}(I) \mid I \text{ is an input and size}(I) = n\}.$$

Now let's discuss comparing different algorithms that solve a problem P. We'll always assume that the algorithms we compare use certain specified operations that we intend to count. If A and B are algorithms that solve P and if $W_A(n) \leq W_B(n)$ for all $n > 0$, then we know algorithm A has worst-case performance that is better than or equal to that of algorithm B. This gives us the proper tool to describe the idea of an optimal algorithm.

Definition of Optimal in the Worst Case

An algorithm A is *optimal in the worst case* for problem P, if for any algorithm B that exists, or ever will exist, the following relationship holds:

$$W_A(n) \leq W_B(n) \text{ for all } n > 0.$$

How in the world can we ever find an algorithm that is optimal in the worst case for a problem P? The answer involves the following three steps:

1. (Find an algorithm) Find or design an algorithm A to solve P. Then do an analysis of A to find the worst-case function W_A.
2. (Find a lower bound) Find a function F such that $F(n) \leq W_B(n)$ for all $n > 0$ and for all algorithms B that solve P.
3. Compare F and W_A. If $F = W_A$, then A is optimal in the worst case.

Suppose we know that $F \neq W_A$ in Step 3. This means that $F(n) < W_A(n)$ for some n. In this case there are two possible courses of action to consider:

1. Put on your construction hat and try to build a new algorithm C such that $W_C(n) \leq W_A(n)$ for all $n > 0$.
2. Put on your analysis hat and try to find a new function G such that $F(n) \leq G(n) \leq W_B(n)$ for all $n > 0$ and for all algorithms B that solve P.

We should note that zero is always a lower bound, but it's not very interesting because most algorithms take more than zero time. A few problems have optimal algorithms. For the vast majority of problems that have solutions, optimal algorithms have not yet been found. The examples contain both kinds of problems.

example 5.1 Matrix Multiplication

We can "multiply" two n by n matrices A and B to obtain the product AB, which is the n by n matrix defined by letting the element in the ith row and jth column of AB be the value of the expression

$$A_{i1}B_{1j} + A_{i2}B_{2j} + \cdots + A_{in}B_{nj}.$$

For example, let A and B be the following 2 by 2 matrices:

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}, \quad B = \begin{pmatrix} e & f \\ g & h \end{pmatrix}.$$

The product AB is the following 2 by 2 matrix:

$$AB = \begin{pmatrix} ae + bg & af + bh \\ ce + dg & cf + dh \end{pmatrix}.$$

Notice that the computation of AB takes eight multiplications and four additions. The definition of matrix multiplication of two n by n matrices uses n^3 multiplication operations and $n^2(n-1)$ addition operations.

A known lower bound for the number of multiplication operations needed to multiply two n by n matrices is n^2. Strassen [1969] showed how to multiply two matrices with about $n^{2.81}$ multiplication operations. The number 2.81 is an approximation to the value of $\log_2(7)$. It stems from the fact that a pair of 2 by 2 matrices can be multiplied by using seven multiplication operations.

Multiplication of larger-size matrices is broken down into multiplying many 2 by 2 matrices. Therefore, the number of multiplication operations becomes less than n^3. This revelation got research going in two camps. One camp is trying to find a better algorithm. The other camp is trying to raise the lower bound above n^2. Pan [1978] gave an algorithm to multiply two 70×70 matrices using 143,640 multiplications, which is less than $70^{2.81}$ multiplication operations. Coppersmith and Winograd [1987] gave an algorithm that, for large values of n, uses $n^{2.376}$ multiplication operations. So it goes.

end example

example 5.2 Finding the Minimum

Let's examine an optimal algorithm to find the minimum number in an unsorted list of n numbers. We'll count the number of comparison operations that an algorithm makes between elements of the list. To find the minimum number in a list of n numbers, the minimum number must be compared with the other $n-1$ numbers. Therefore, $n-1$ is a lower bound on the number of comparisons needed to find the minimum number in a list of n numbers.

If we represent the list as an array a indexed from 1 to n, then the following algorithm is optimal because the operation $\leq$ is executed exactly $n-1$ times.

```
m := a[1];
for i := 2 to n do
    m := if m ≤ a[i] then m else a[i]
od
```

end example

5.1.2 Decision Trees

We can often use a tree to represent the decision processes that take place in an algorithm. A *decision tree* for an algorithm is a tree whose nodes represent decision points in the algorithm and whose leaves represent possible outcomes. Decision trees can be useful in trying to construct an algorithm or trying to find properties of an algorithm. For example, lower bounds may equate to the depth of a decision tree.

If an algorithm makes decisions based on the comparison of two objects, then it can be represented by a *binary decision tree*. Each nonleaf node in the tree represents a pair of objects to be compared by the algorithm, and each branch from that node represents a path taken by the algorithm based on the comparison. Each leaf can represent an outcome of the algorithm. A *ternary decision tree* is similar except that each nonleaf node represents a comparison that has three possible outcomes.

Lower Bounds for Decision Tree Algorithms

Let's see whether we can compute lower bounds for decision tree algorithms. If a decision tree has depth d, then some path from the root to a leaf contains $d + 1$ nodes. Since the leaf is a possible outcome, it follows that there are d decisions made on the path. Since no other path from the root to a leaf can have more than $d + 1$ nodes, it follows that d is the worst-case number of decisions made by the algorithm.

Now, suppose that a problem has n possible outcomes and it can be solved by a binary decision tree algorithm. What is the best binary decision tree algorithm? We may not know the answer, but we can find a lower bound for the depth of any binary decision tree to solve the problem. Since the problem has n possible outcomes, it follows that any binary decision tree algorithm to solve the problem must have at least n leaves, one for each of the n possible outcomes. Recall that the number of leaves in a binary tree of depth d is at most 2^d.

So if d is the depth of a binary decision tree to solve a problem with n possible outcomes, then we must have $n \leq 2^d$. We can solve this inequality for d by taking $\log_2$ of both sides to obtain $\log_2 n \leq d$. Since d is a natural number, it follows that

$$\lceil \log_2 n \rceil \leq d.$$

In other words, any binary decision tree algorithm to solve a problem with n possible outcomes must have a depth of at least $\lceil \log_2 n \rceil$.

We can do the same analysis for ternary decision trees. The number of leaves in a ternary tree of depth d is at most 3^d. If d is the depth of a ternary decision tree to solve a problem with n possible outcomes, then we must have $n \leq 3^d$. Solve the inequality for d to obtain

$$\lceil \log_3 n \rceil \leq d.$$

In other words, any ternary decision tree algorithm to solve a problem with n possible outcomes must have a depth of at least $\lceil \log_3 n \rceil$.

Many sorting and searching algorithms can be analyzed with decision trees because they perform comparisons. Let's look at some examples to illustrate the idea.

example 5.3 Binary Search

Suppose we search a sorted list in a binary fashion. That is, we check the middle element of the list to see whether it's the key we are looking for. If not, then we perform the same operation on either the left half or the right half of the list, depending on the value of the key. This algorithm has a nice representation as a decision tree. For example, suppose we have the following sorted list of 15 numbers:

$$x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}.$$

Suppose we're given a number key K, and we must find whether it is in the list. The decision tree for a binary search of the list has the number x_8 at its root. This represents the comparison of K with x_8. If $K = x_8$, then we are successful in one comparison. If $K < x_8$, then we go to the left child of x_8; otherwise we go to the right child of x_8. The result is a ternary decision tree in which the leaves are labeled with either S, for successful search, or U, for unsuccessful search. The decision tree is pictured in Figure 5.1.

Since the depth of the tree is 4, it follows that there will be four comparisons in the worst case to find whether K is in the list. Is this an optimal algorithm? To see that the answer is yes, we can observe that there are 31 possible outcomes for the given problem: 15 leaves labeled with S to represent successful searches; and 16 leaves labeled with U to represent the gaps where $K < x_1$, $x_i < K < x_{i+1}$ for $1 \leq i < 15$, and $x_{15} < K$. Therefore, a worst-case lower bound for the number of comparisons is $\lceil \log_3 31 \rceil = 4$. Therefore, the algorithm is optimal.

end example

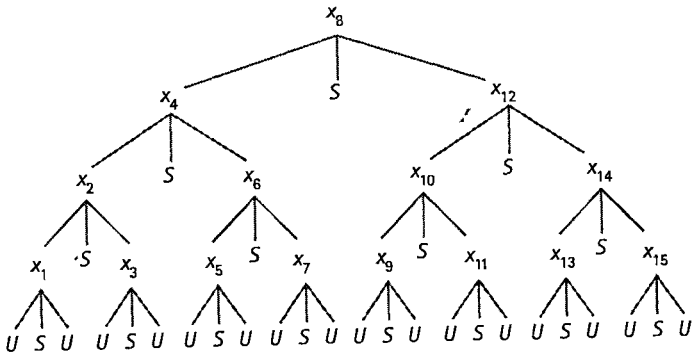


Figure 5.1 Decision tree for binary search.

example 5.4 Finding a Bad Coin

Suppose we are asked to use a pan balance to find the heavy coin among eight coins with the assumption that they all look alike and the other seven all have the same weight. One way to proceed is to always place coins in the two pans so that the bad coin is included and thus one pan will always go down.

This gives a binary decision tree, where each internal node of the tree represents the pan balance. If the left side goes down, then the heavy coin is on the left side of the balance. Otherwise, the heavy coin is on the right side of the balance. Each leaf represents one coin that is the heavy coin. Suppose we label the coins with the numbers 1, 2, . . . , 8.

The decision tree for one algorithm is shown in Figure 5.2, where the numbers on either side of a nonleaf node represent the coins on either side of the pan balance. This algorithm finds the heavy coin in three weighings. Can we do any better? Look at the next example.

end example

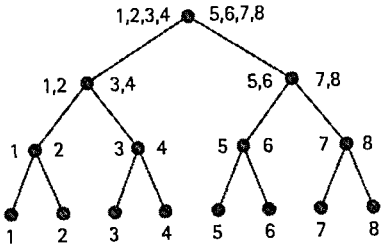


Figure 5.2 A binary decision tree.

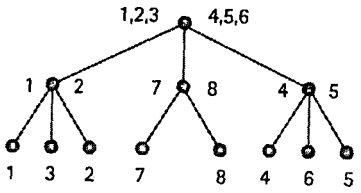


Figure 5.3 An optimal decision tree.

example 5.5 An Optimal Solution

The problem is the same as in Example 5.4. We are asked to use a pan balance to find the heavy coin among eight coins with the assumption that they all look alike and the other seven all have the same weight. In this case we'll weigh coins with the possibility that the two pans are balanced. So a decision tree can have nodes with three children.

We don't have to use all eight coins on the first weighing. For example, Figure 5.3 shows the decision tree for one algorithm. Notice that there is no middle branch on the middle subtree because, at this point, one of the coins, 7 or 8, must be the heavy one. This algorithm finds the heavy coin in two weighings.

This algorithm is an optimal pan-balance algorithm for the problem, where we are counting the number of weighings to find the heavy coin. To see this, notice that any one of the eight coins could be the heavy one. Therefore, there must be at least eight leaves on any algorithm's decision tree. Notice that a binary tree of depth d can have 2^d possible leaves. So to get eight leaves, we must have $2^d \geq 8$. This implies that $d \geq 3$. But a ternary tree of depth d can have 3^d possible leaves. So to get eight leaves, we must have $3^d \geq 8$, or $d \geq 2$. Therefore, 2 is a lower bound for the number of weighings. Since the algorithm solves the problem in two weighings, it is optimal.

end example

example 5.6 A Lower Bound Computation

Suppose we have a set of 13 coins in which at most one coin is bad and a bad coin may be heavier or lighter than the other coins. The problem is to use a pan balance to find the bad coin if it exists and say whether it is heavy or light. We'll find a lower bound on the heights of decision trees for pan-balance algorithms to solve the problem.

Any solution must tell whether a bad coin is heavy or light. Thus there are 27 possible outcomes: no bad coin and the 13 pairs of outcomes (ith coin light, ith coin heavy). Therefore, any decision tree for the problem must have at least 27 leaves. So a ternary decision tree of depth d must satisfy $3^d \geq 27$, or $d \geq 3$. This gives us a lower bound of 3.

Now the big question: Is there an algorithm to solve the problem, where the decision tree of the algorithm has depth 3? The answer is no. Just look at the cases of different initial weighings, and note in each case that the remaining possible outcomes cannot be distinguished with just two more weighings. Thus any decision tree for this problem must have depth 4 or more.

end example

Exercises

1. Draw a picture of the decision tree for an optimal algorithm to find the maximum number in the list x_1, x_2, x_3, x_4.
2. Suppose there are 95 possible answers to some problem. For each of the following types of decision tree, find a reasonable lower bound for the number of decisions necessary to solve the problem.
   a. Binary tree. b. Ternary tree. c. Four-way tree.
3. Find a nonzero lower bound on the number of weighings necessary for any ternary pan-balance algorithm to solve the following problem: A set of 30 coins contains at most one bad coin, which may be heavy or light. Is there a bad coin? If so, state whether it's heavy or light.
4. Find an optimal pan-balance algorithm to find a bad coin, if it exists, from 12 coins, where at most one coin is bad (i.e., heavier or lighter than the others). *Hint:* Once you've decided on the coins to weigh for the root of the tree, then the coins that you choose at the second level should be the same coins for all three branches of the tree.

5.2 Summations and Closed Forms

In trying to count things we often come up with expressions or relationships that need to be simplified to a form that can be easily computed with familiar operations.

Definition of Closed Form

A *closed form* is an expression that can be computed by applying a fixed number of familiar operations to the arguments. A closed form can't have an ellipsis because the number of operations to evaluate the form would not be fixed. For example, the expression $n(n+1)/2$ is a closed form, but the expression $1 + 2 + \ldots + n$ is not a closed form. In this section we'll see some ways to find closed forms for sums.

5.2.1 Basic Summations and Closed Forms

When we count things, we most often add things together. When evaluating a sum of two or more expressions, it sometimes helps to write the sum in a different form. Here are six examples to demonstrate some simple relationships.

$$c + c + c = 3c$$
$$ca_1 + ca_2 + ca_3 = c(a_1 + a_2 + a_3)$$
$$(a_1 - a_0) + (a_2 - a_1) + (a_3 - a_2) = (a_3 - a_0)$$
$$(a_0 - a_1) + (a_1 - a_2) + (a_2 - a_3) = (a_0 - a_3)$$
$$(a_1 + a_2 + a_3) + (b_1 + b_2 + b_3) = (a_1 + b_1) + (a_2 + b_2) + (a_3 + b_3)$$
$$(a_1 x + a_2 x^2 + a_3 x^3) = x(a_1 + a_2 x + a_3 x^2)$$

To represent the sum of n terms $a_1, a_2, \ldots, a_n$ we use the following notation:

$$\sum_{k=1}^{n} a_k = a_1 + a_2 + \ldots + a_n.$$

The following list uses this notation to give some basic properties of sums, six of which generalize the preceding examples, and all of which are easily verified.

Summation Facts (5.1)

a. $\sum_{k=m}^{n} c = (n - m + 1)c.$ (sum of a constant)

b. $\sum_{k=m}^{n} c\, a_k = c \sum_{k=m}^{n} a_k.$

c. $\sum_{k=1}^{n} (a_k - a_{k-1}) = a_n - a_0$ and $\sum_{k=1}^{n} (a_{k-1} - a_k) = a_0 - a_n.$ (collapsing sums)

d. $\sum_{k=m}^{n} (a_k + b_k) = \sum_{k=m}^{n} a_k + \sum_{k=m}^{n} b_k.$

e. $\sum_{k=m}^{n} a_k = \sum_{k=m}^{i} a_k + \sum_{k=i+1}^{n} a_k.$ $(m \le i < n)$

f. $\sum_{k=m}^{n} a_k x^{k+i} = x^i \sum_{k=m}^{n} a_k x^k.$

g. $\sum_{k=m}^{n} a_{k+i} = \sum_{k=m+i}^{n+1} a_k.$ (change index limits)

These facts are very useful in manipulating sums into simpler forms from which we might be able to find closed forms. In the following examples, we'll derive some closed forms that have many applications.

Sums of Powers

A closed form for the sum of powers $\sum_{k=1}^{n} k^m$ can be derived from closed forms for the sums of powers less than m. We'll use a technique that begins by considering the simple expression

$$k^{m+1} - (k-1)^{m+1}. \tag{5.2}$$

Expand this expression and collect terms to obtain a polynomial of the following form, where $c_m \neq 0$.

$$k^{m+1} - (k-1)^{m+1} = c_0 + c_1 k + c_2 k^2 + \cdots + c_m k^m. \tag{5.3}$$

Now comes the interesting part. We take the sum of both sides of (5.3) from 1 to n and observe two things about the resulting equation. The sum of the left side collapses to obtain

$$\sum_{k=1}^{n} \left(k^{m+1} - (k-1)^{m+1}\right) = n^{m+1}.$$

The sum on the right side becomes

$$\sum_{k=1}^{n} (c_0 + c_1 k + c_2 k^2 + \cdots + c_m k^m) =$$

$$c_0 \sum_{k=1}^{n} 1 + c_1 \sum_{k=1}^{n} k + c_2 \sum_{k=1}^{n} k^2 + \cdots + c_m \sum_{k=1}^{n} k^m.$$

So by summing both sides of (5.3) we obtain the equation

$$n^{m+1} = c_0 \sum_{k=1}^{n} 1 + c_1 \sum_{k=1}^{n} k + c_2 \sum_{k=1}^{n} k^2 + \cdots + c_m \sum_{k=1}^{n} k^m.$$

If we know the closed forms for the sums of powers less than m, we can substitute them for the sums in the equation, which results in an equation with just the sum $\sum_{k=1}^{n} k^m$ as the unknown quantity. Put it on one side of the equation by itself and its closed form will be sitting on the other side. We'll do some samples to demonstrate the method.

example 5.7 Closed Forms for Sums of Low Powers

A closed form for the sum of the arithmetic progression 1, 2, ..., n is given in (4.25). We can write the sum and its closed form as

$$\sum_{k=1}^{n} k = \frac{n(n+1)}{2}. \tag{5.4}$$

To demonstrate this method we'll derive the closed form (5.4). So we begin with the expression

$$k^2 - (k-1)^2.$$

Expand the expression to obtain the equation

$$k^2 - (k-1)^2 = k^2 - k^2 + 2k - 1 = 2k - 1 = -1 + 2k.$$

Next we sum both sides from 1 to n and observe the resulting equation:

$$\sum_{k=1}^{n} (k^2 - (k-1)^2) = \sum_{k=1}^{n} (-1 + 2k). \tag{5.5}$$

The sum on the left side of (5.5) collapses to become

$$\sum_{k=1}^{n} (k^2 - (k-1)^2) = n^2 - 0^2 = n^2.$$

The sum on the right side of (5.5) becomes the following, where we use the known closed form (5.1a) for the sum of a constant.

$$\sum_{k=1}^{n} (-1 + 2k) = \sum_{k=1}^{n} (-1) + \sum_{k=1}^{n} 2k = -n + 2\sum_{k=1}^{n} k.$$

So Equation (5.5) becomes

$$n^2 = -n + 2\sum_{k=1}^{n} k.$$

Now solve for $\sum_{k=1}^{n} k$ to obtain the closed form in (5.4).

To make sure we have this method down, we'll find a closed form for $\sum_{k=1}^{n} k^2$. So we'll begin with the expression

$$k^3 - (k-1)^3.$$

Expand the expression to obtain the equation

$$k^3 - (k-1)^3 = k^3 - k^3 + 3k^2 - 3k + 1 = 1 - 3k + 3k^2.$$

Next we sum both sides from 1 to n and observe the resulting equation:

$$\sum_{k=1}^{n} (k^3 - (k-1)^3) = \sum_{k=1}^{n} (1 - 3k + 3k^2). \tag{5.6}$$

The sum on the left side of (5.6) collapses to become

$$\sum_{k=1}^{n} (k^3 - (k-1)^3) = n^3 - 0^3 = n^3.$$

The sum on the right side of (5.6) becomes the following, where we use the known closed forms (5.1a) and (5.4).

$$\sum_{k=1}^{n}(1-3k+3k^2) = \sum_{k=1}^{n}1 - 3\sum_{k=1}^{n}k + 3\sum_{k=1}^{n}k^2$$
$$= n - 3\frac{n(n+1)}{2} + 3\sum_{k=1}^{n}k^2.$$

So Equation (5.6) becomes

$$n^3 = n - 3\frac{n(n+1)}{2} + 3\sum_{k=1}^{n}k^2.$$

We can solve this equation for $\sum_{k=1}^{n}k^2$ to obtain the closed form

$$\sum_{k=1}^{n}k^2 = \frac{n(n+1)(2n+1)}{6}. \tag{5.7}$$

end example

example **5.8 Finding the Sum of a Geometric Progression**

A closed form for the sum of the geometric progression $1, a, a^2, \ldots, a^n$, where $a \neq 1$, is given in (4.27). We can write the sum and its closed form as

$$\sum_{k=0}^{n}a^k = \frac{a^{n+1}-1}{a-1}. \tag{5.8}$$

We'll derive (5.8) to demonstrate a technique similar to that used for sums of powers. In this case we'll start with an expression that is the difference between two successive general terms of the sum. So we start with the expression

$$a^{k+1} - a^k.$$

Rewrite the expression to obtain

$$a^{k+1} - a^k = (a-1)a^k.$$

Next we sum both sides of this equation from 1 to n to obtain the equation

$$\sum_{k=0}^{n}(a^{k+1}-a^k) = \sum_{k=0}^{n}(a-1)a^k. \tag{5.9}$$

The sum on the left side of (5.9) collapses to

$$\sum_{k=0}^{n}(a^{k+1}-a^k)=a^{n+1}-1.$$

The sum on the right side of (5.9) becomes

$$\sum_{k=0}^{n}(a-1)a^k=(a-1)\sum_{k=0}^{n}a^k.$$

So Equation (5.9) becomes

$$a^{n+1}-1=(a-1)\sum_{k=0}^{n}a^k.$$

Since $a \neq 1$, we can solve this equation for $\sum_{k=0}^{n} a^k$ to obtain the closed form in (5.8).

end example

example **5.9 Closed Form for a Sum of Products**

We'll derive the closed form for the sum

$$\sum_{k=1}^{n}ka^k=a+2a^2+3a^3+\cdots+na^n.$$

We'll use a technique that is similar to those in the preceding examples. That is, we'll start with an expression whose sums will collapse and that expands to an expression whose sums will contain the sum we want. The expression is the difference between two successive general terms of the sum. So we start with the expression

$$(k+1)a^{k+1}-ka^k.$$

Rewrite the expression to

$$(k+1)a^{k+1}-ka^k=ka^{k+1}+a^{k+1}-ka^k=(a-1)ka^k+a^{k+1}.$$

Now take the sum of both sides from 1 to n to obtain

$$\sum_{k=1}^{n}\left((k+1)a^{k+1}-ka^k\right)=\sum_{k=1}^{n}\left((a-1)ka^k+a^{k+1}\right). \tag{5.10}$$

The sum on the left side of (5.10) collapses to

$$\sum_{k=1}^{n}\left((k+1)a^{k+1}-ka^k\right)=(n+1)a^{n+1}-a.$$

The sum on the right side of (5.10) becomes the following, where we use the known closed form (5.8).

$$\begin{aligned}\sum_{k=1}^{n}\left((a-1)ka^k + a^{k+1}\right) &= (a-1)\sum_{k=1}^{n} ka^k + a\sum_{k=1}^{n} a^k \\ &= (a-1)\sum_{k=1}^{n} ka^k + a\left(\sum_{k=0}^{n} a^k - 1\right) \\ &= (a-1)\sum_{k=1}^{n} ka^k + a\left(\frac{a^{n+1}-1}{a-1}\right) - a.\end{aligned}$$

So Equation (5.10) becomes

$$(n+1)a^{n+1} - a = (a-1)\sum_{k=1}^{n} ka^k + a\left(\frac{a^{n+1}-1}{a-1}\right) - a.$$

Since $a \neq 1$, we can solve this equation for the sum to obtain the following closed form:

$$\sum_{k=1}^{n} ka^k = \frac{a-(n+1)a^{n+1}+na^{n+2}}{(a-1)^2}.$$

end example

The closed forms derived in the preceding examples are quite useful because they pop up in many situations when trying to count the number of operations performed by an algorithm. We'll list them here.

Closed Forms of Elementary Finite Sums (5.11)

a. $\displaystyle\sum_{k=1}^{n} k = \frac{n(n+1)}{2}.$

b. $\displaystyle\sum_{k=1}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}.$

c. $\displaystyle\sum_{k=0}^{n} a^k = \frac{a^{n+1}-1}{a-1}$ $(a \neq 1).$

d. $\displaystyle\sum_{k=1}^{n} ka^k = \frac{a-(n+1)a^{n+1}+na^{n+2}}{(a-1)^2}$ $(a \neq 1).$

A Sum of Products Transformation

The summation facts listed in (5.1) are basic and useful, but not exhaustive. In fact, we will hardly scratch the surface of known summation facts. Let's look at an easily verifiable transformation for general sums of products that can be useful in breaking down a problem into simpler problems, somewhat akin to the idea of divide and conquer.

Abel's Summation Transformation

$$\sum_{k=0}^{n} a_k b_k = A_n b_n + \sum_{k=0}^{n-1} A_k (b_k - b_{k+1}), \text{ where } A_k = \sum_{i=0}^{k} a_k. \tag{5.12}$$

Note that the formula also holds if the lower limit of 0 is replaced by any integer that is less than or equal to n.

This formula can be useful if either A_k or $b_k - b_{k+1}$ can be calculated. For example, we'll give an alternative derivation of the closed form in (5.11d). In this case we'll use the lower limit of 0 because (5.11d) has the same value whether we start at 1 or 0 and because the lower limit of 0 will simplify the closed form.

$$\begin{aligned}
\sum_{k=1}^{n} k a^k = \sum_{k=0}^{n} k a^k = \sum_{k=0}^{n} a^k k &= A_n n + \sum_{k=0}^{n-1} A_k (k - (k+1)) \\
&= A_n n - \sum_{k=0}^{n-1} A_k \\
&= \left(\frac{a^{n+1}-1}{a-1}\right) n - \sum_{k=0}^{n-1} \left(\frac{a^{k+1}-1}{a-1}\right) \\
&= \left(\frac{1}{a-1}\right) \left(n a^{n+1} - n - \sum_{k=0}^{n-1} (a^{k+1} - 1)\right) \\
&= \left(\frac{1}{a-1}\right) \left(n a^{n+1} - n - a \sum_{k=0}^{n-1} a^k + \sum_{k=0}^{n-1} 1\right) \\
&= \left(\frac{1}{a-1}\right) \left(n a^{n+1} - n - a \left(\frac{a^n - 1}{a-1}\right) + n\right) \\
&= \frac{a - (n+1)a^{n+1} + n a^{n+2}}{(a-1)^2}.
\end{aligned}$$

Some Applications

Let's look at a few examples of problems that can be solved by finding closed forms for summations.

example 5.10 The Polynomial Problem

Suppose we're interested in the number of arithmetic operations needed to evaluate the following polynomial at some number x.

$$c_0 + c_1x + c_2x^2 + \cdots + c_nx^n.$$

The number of operations performed will depend on how we evaluate it. For example, suppose that we compute each term in isolation and then add up all the terms. There are n addition operations and each term of the form c_ix^i takes i multiplication operations. So the total number of arithmetic operations is given by the following sum:

$$\begin{aligned} n + (0 + 1 + 2 + \cdots + n) &= n + \sum_{k=0}^{n} k \\ &= n + \frac{n(n+1)}{2} \\ &= \frac{n^2 + 3n}{2}. \end{aligned}$$

So for even small values of n there are many operations to perform. For example, if $n = 30$, then there are 495 arithmetic operations to perform. Can we do better? Sure, we can group terms so that we don't have to repeatedly compute the same powers of x. We'll continue the discussion after we've introduced recurrences in Section 5.5.

end example

example 5.11 A Simple Sort

In this example we'll construct a simple sorting algorithm and analyze it to find the number of comparison operations. We'll sort an array a of numbers indexed from 1 to n as follows: Find the smallest element in a and exchange it with the first element. Then find the smallest element in positions 2 through n and exchange it with the element in position 2. Continue in this manner to obtain a sorted array. To write the algorithm, we'll use a function "min" and a procedure "exchange," which are defined as follows:

- min(a, i, n) is the index of the minimum number among the elements $a[i]$, $a[i+1]$, ..., $a[n]$. We can easily modify the algorithm in Example 5.2 to accomplish this task with $n - i$ comparisons.
- exchange($a[i]$, $a[j]$) is the usual operation of swapping elements and does not use any comparisons.

Now we can write the sorting algorithm as follows:

```
for i := 1 to n − 1 do
    j := min(a, i, n);
    exchange(a[i], a[j])
od
```

Now let's compute the number of comparison operations. The algorithm for $\min(a, i, n)$ makes $n - i$ comparisons. So as i moves from 1 to $n - 1$, the number of comparison operations moves from $n - 1$ to $n - (n - 1)$. Adding these comparisons gives the sum of an arithmetic progression,

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

The algorithm makes the same number of comparisons no matter what the form of the input array, even if it is sorted to begin with. So any arrangement of numbers is a worst-case input. For example, to sort 1,000 items it would take 499,500 comparisons, no matter how the items are arranged at the start.

There are many faster sorting algorithms. For example, an algorithm called "heapsort" takes no more than $2n\log_2 n$ comparisons for its worst-case performance. So for 1,000 items, heapsort would take a maximum of 20,000 comparisons—quite an improvement over our simple sort algorithm. In Section 5.3 we'll discover a good lower bound for the worst-case performance of comparison sorting algorithms.

end example

example **5.12 A Problem of Loops**

Let's look at a general problem of counting operations that occur within loops. We'll assume that we have an algorithm with a procedure P, where $P(j)$ executes $3j$ operations of a certain type. In the following algorithm we assume that n is a given positive integer.

```
i := 1;
while i < n do
    i := 2i;
    for j := 1 to i do P(j) od
od
```

Our task will be to count the total number of operations executed by P as a function of n, which we'll denote by $T(n)$. Note, for example, that if $n = 1$, then the while-loop can't be entered and we must have $T(1) = 0$.

To find a formula for $T(n)$ we might start by observing the for-loop. For each i the for-loop calls $P(1)$, $P(2)$, $P(3)$, ..., $P(i)$. Since each call to $P(j)$ executes $3j$ operations, it follows that for each i the number of operations executed by the

calls on P by the for-loop is $3(1+2+3+\ldots+i)$. Let $f(i)$ denote this expression. We can use (5.11a) to calculate $f(i) = 3i(i+1)/2$.

Now we must find the values of i that are used to enter the for-loop. Observe that the values of i to enter the while-loop are 1, 2, 4, 8, ..., 2^k, where $2^k < n \leq 2^{k+1}$. So the values of i to enter the for-loop are $2, 4, 8, \ldots, 2^{k+1}$. Now we can write an expression for $T(n)$.

$$\begin{aligned}
T(n) &= \sum_{m=1}^{k+1} f(2^m) = \sum_{m=1}^{k+1} (3/2)2^m(2^m+1) \\
&= 3\sum_{m=1}^{k+1} 2^{m-1}(2^m+1) \\
&= 3\sum_{m=0}^{k} 2^m(2^{m+1}+1) \\
&= 3\sum_{m=0}^{k} (2^{2m+1}+2^m) \\
&= 3\sum_{m=0}^{k} 2^{2m+1} + 3\sum_{m=0}^{k} 2^m \\
&= 6\sum_{m=0}^{k} 4^m + 3\sum_{m=0}^{k} 2^m \\
&= 2(4^{k+1}-1) + 3(2^{k+1}-1).
\end{aligned}$$

Now, to obtain a function of n, recall that $2^k < n \leq 2^{k+1}$. Apply $\log_2$ to the inequality to obtain $k < \log_2 n \leq k+1$. Therefore, $\lceil \log_2 n \rceil = k+1$ and we can substitute for $k+1$ to obtain the following expression for $T(n)$.

$$\begin{aligned}
T(n) &= 2(4^{\lceil \log_2 n \rceil} - 1) + 3(2^{\lceil \log_2 n \rceil} - 1) \\
&= 2 \cdot 4^{\lceil \log_2 n \rceil} + 3 \cdot 2^{\lceil \log_2 n \rceil} - 5.
\end{aligned}$$

end example

5.2.2 Approximating Sums

Sometimes a sum does not have a closed form or a closed form is hard to find. In such cases we can try to find an approximation for the sum.

Some Simple Approximation Techniques

If we have a sum that has a term of maximum value, then we can replace each term by that value to obtain an upper bound on the sum. Similarly, for a minimum-valued term. For example, suppose we have the following sum of logs:

$$\sum_{k=1}^{n} \log k = \log 1 + \log 2 + \cdots + \log n.$$

Since $\log n$ is the maximum value and $\log 1 = 0$ is the minimum value, we can say that

$$0 \leq \sum_{k=1}^{n} \log k \leq n \log n.$$

We can sometimes obtain closer bounds by splitting up the sum and bounding each part. For example, we'll split the sum up into two almost equal size sums as follows:

$$\begin{aligned}\sum_{k=1}^{n} \log k &= (\log 1 + \log 2 + \cdots + \log \lfloor n/2 \rfloor) + (\log (\lfloor n/2 \rfloor + 1) + \cdots + \log n) \\ &= \sum_{k=1}^{\lfloor n/2 \rfloor} \log k + \sum_{k=\lfloor n/2 \rfloor + 1}^{n} \log k.\end{aligned}$$

To get a better lower bound we can replace each term of the first sum by 0 and each term of the second sum by $\log \lfloor n/2 \rfloor$. To get a better upper bound we can replace each term of the first sum by $\log \lfloor n/2 \rfloor$ and each term of the second sum by $\log n$. This gives us

$$\lfloor n/2 \rfloor \log \lfloor n/2 \rfloor \leq \sum_{k=1}^{n} \log k \leq \lfloor n/2 \rfloor (\log \lfloor n/2 \rfloor + \log n).$$

5.2.3 Approximations with Definite Integrals

A powerful technique for establishing upper and lower bounds for a sum comes from elementary calculus. If you don't have a background in calculus, you can safely skip the next couple of paragraphs.

We are interested in approximating a sum of the following form, where f is a continuous function with nonnegative values.

$$\sum_{k=1}^{n} f(k) = f(1) + f(2) + \cdots + f(n). \tag{5.13}$$

Each number $f(k)$ can be thought of as the area of a rectangle of width 1 and height $f(k)$. In fact we'll be more specific and let the base of the rectangle be the

closed interval $[k, k+1]$. So the sum we want represents the area of n rectangles, whose bases consist of the following partition of the closed interval $[1, n+1]$.

$$[1, 2], [2, 3], \ldots, [n, n+1].$$

The area under the curve $f(x)$ above the x-axis for x in the closed interval $[1, n+1]$ is given by the definite integral

$$\int_1^{n+1} f(x)dx.$$

So this definite integral is an approximation for our sum:

$$\sum_{k=1}^{n} f(k) \approx \int_1^{n+1} f(x)dx. \tag{5.14}$$

Evaluating Definite Integrals

To evaluate a definite integral of f we need to find a function F that is an antiderivative of f (the derivative of $F(x)$ is $f(x)$) and then apply the fundamental theorem of calculus that relates the two functions as follows:

$$\int_a^b f(x)dx = F(x)\Big|_a^b = F(b) - F(a).$$

Here is a listing of some useful functions and antiderivatives.

$f(x)$	$F(x)$
$x^r (r \neq -1)$	$x^{r+1}/(r+1)$
$1/x$	$\ln x$
$\ln x$	$x \ln x - x$
e^x	e^x
c^x	$c^x / \ln x$

(5.15)

Bounds for Monotonic Functions

Assume f is a monotonic increasing function, which means that $x < y$ implies $f(x) \leq f(y)$. Then the area of each rectangle with base $[k, k+1]$ and height $f(k)$ is less than or equal to the area of region under the graph of f on the interval. So (5.14) gives us the following upper bound on the sum (5.13):

$$\sum_{k=1}^{n} f(k) \leq \int_1^{n+1} f(x)dx. \tag{5.16}$$

We can obtain a lower bound for the sum (5.13) by noticing that the area of each rectangle with base $[k-1, k]$ and height $f(k)$ is greater than or equal to the area of region under the graph of f on the interval. So in this case the sum (5.13) represents the area of n rectangles, whose bases consist of the following partition of the closed interval $[0, n]$.

$$[0, 1], [1, 2], \ldots, [n-1, n].$$

So we obtain the following lower bound on the sum (5.13):

$$\int_0^n f(x)dx \leq \sum_{k=1}^{n} f(k). \tag{5.17}$$

Putting (5.16) and (5.17) together we obtain the following bounds on the sum (5.13):

$$\int_0^n f(x)dx \leq \sum_{k=1}^{n} f(k) \leq \int_1^{n+1} f(x)dx. \tag{5.18}$$

If f is monotonic decreasing (i.e., $x < y$ implies $f(x) \geq f(y)$), then we use entirely similar reasoning to obtain the following bounds of the sum:

$$\int_1^{n+1} f(x)dx \leq \sum_{k=1}^{n} f(k) \leq \int_0^n f(x)dx. \tag{5.19}$$

A Note on Lower Limits

For ease of presentation we used a specific lower limit of 1 for the summation, which gave rise to lower limits of 0 and 1 in the bounding definite integrals. Any natural number m could replace 1 in the sum, with corresponding replacements of 0 by $m-1$ and 1 by m in the bounding definite integrals.

example 5.13 Some Sample Approximations

We'll find bounds for the following sum, where r is a real number and $r \neq -1$.

$$\sum_{k=1}^{n} k^r = 1^r + 2^r + \cdots + n^r.$$

If $r = 0$, then the sum becomes $1 + 1 + \cdots + 1 = n$. If $r > 0$, then x^r is increasing for $x \geq 0$. So we can obtain bounds by using (5.18) and the antiderivative from

(5.15) to obtain lower and upper bounds as follows:

$$\text{(lower bound)} \quad \sum_{k=1}^{n} k^r \geq \int_0^n x^r dx = \frac{x^{r+1}}{r+1}\Big|_0^n = \frac{n^{r+1}}{r+1}.$$

$$\text{(upper bound)} \quad \sum_{k=1}^{n} k^r \leq \int_1^{n+1} x^r dx = \frac{x^{r+1}}{r+1}\Big|_1^{n+1} = \frac{(n+1)^{r+1}}{r+1} - \frac{1}{r+1}.$$

If $r < 0$, then x^r is decreasing for $x > 0$, but is not defined at $x = 0$. So the upper bound from (5.19) does not exist. We can work around the problem by raising each lower limit of (5.19) by 1. This gives the inequality:

$$\int_2^{n+1} x^r dx \leq \sum_{k=2}^{n} k^r \leq \int_1^n x^r dx.$$

Notice that the middle sum is not what we want, but after we find the bounds, we can add the first term of the sum, which is 1 in this case, to the bounds. So we can evaluate the two definite integrals using the antiderivative from (5.15) to obtain bounds as follows:

$$\text{(upper bound)} \quad \sum_{k=2}^{n} k^r \leq \int_1^n x^r dx = \frac{x^{r+1}}{r+1}\Big|_1^n = \frac{n^{r+1}}{r+1} - \frac{1}{r+1}.$$

$$\text{(lower bound)} \quad \sum_{k=2}^{n} k^r \geq \int_2^{n+1} x^r dx = \frac{x^{r+1}}{r+1}\Big|_2^{n+1} = \frac{(n+1)^{r+1}}{r+1} - \frac{2^{r+1}}{r+1}.$$

Adding 1 to each bound gives us the bounds

$$1 + \frac{(n+1)^{r+1}}{r+1} - \frac{2^{r+1}}{r+1} \leq \sum_{k=1}^{n} k^r \leq 1 + \frac{n^{r+1}}{r+1} - \frac{1}{r+1}.$$

end example

5.2.4 Harmonic Numbers

For a positive integer n, the sum of the n numbers $1, 1/2, 1/3, \ldots, 1/n$ is called the nth *harmonic number* and it is usually denoted by H_n. In other words, the nth harmonic number is

$$H_n = \sum_{k=1}^{n} (1/k) = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}. \tag{5.20}$$

So $H_1 = 1$, $H_2 = 1 + 1/2$, and so on. The first few harmonic numbers are

$$1, \tfrac{3}{2}, \tfrac{11}{6}, \tfrac{25}{12}, \tfrac{137}{60}, \tfrac{49}{20}, \ldots.$$

Harmonic numbers are interesting for two reasons. The first is that they appear in a wide variety of counting problems and the second is that there is no closed form for the sum. This brings up the question of approximating H_n without having to actually add up all the terms.

Notice that $1/x$ is decreasing for $x > 0$, but it is not defined for $x = 0$. So the upper bound from (5.19) does not exist. We can work around the problem as we did in Example 5.13 by raising each lower limit of (5.19) by 1. This gives the inequality

$$\int_2^{n+1} (1/x)dx \leq \sum_{k=2}^{n} (1/k) \leq \int_1^n (1/x)dx.$$

Notice that the middle sum is now $H_n - 1$. The table in (5.15) tells us that an antiderivative of $1/x$ is $\ln x$. So we can evaluate the two definite integrals and obtain the following inequality:

$$\ln(n+1) - \ln 2 \leq H_n - 1 \leq \ln n.$$

Now add 1 to the terms of the inequality to obtain

$$\ln\left(\frac{n+1}{2}\right) + 1 \leq H_n \leq \ln(n) + 1. \tag{5.21}$$

The difference between these two bounds is less than $\ln 2 = 0.693\ldots$. There are better bounds than those of (5.21). We'll see another one in the exercises.

Sums with Terms That Contain Harmonic Numbers

We'll derive two well-known sums that involve harmonic numbers. The first sum simply adds up the first n harmonic numbers.

$$\sum_{k=1}^{n} H_k = (n+1)H_n - n. \tag{5.22}$$

We'll rearrange the terms of the sum to see whether we can discover a workable pattern.

$$\begin{aligned}\sum_{k=1}^{n} H_k &= H_1 + H_2 + H_3 + \cdots + H_n \\ &= (1) + (1 + 1/2) + (1 + 1/2 + 1/3) + \cdots + (1 + 1/2 + 1/3 + \cdots + 1/n). \\ &= n(1) + (n-1)(1/2) + (n-2)(1/3) + \cdots + (1)(1/n) \\ &= \sum_{k=1}^{n} (n - k + 1)(1/k) \\ &= \sum_{k=1}^{n} (n+1)(1/k) - \sum_{k=1}^{n} (k/k) \\ &= (n+1)\sum_{k=1}^{n} (1/k) - \sum_{k=1}^{n} 1 \\ &= (n+1)H_n - n.\end{aligned}$$

The second sum adds up the first n products of the form kH_k.

$$\sum_{k=1}^{n} kH_k = \frac{n(n+1)}{2} H_n - \frac{n(n-1)}{4}. \tag{5.23}$$

We can discover a workable pattern for this sum by rearranging terms as follows:

$$\begin{aligned}\sum_{k=1}^{n} kH_k &= 1H_1 + 2H_2 + 3H_3 + \cdots + nH_n \\ &= 1(1) + 2(1 + 1/2) + 3(1 + 1/2 + 1/3) + \cdots + n(1 + 1/2 + 1/3 + \cdots \\ &\quad + 1/n). \\ &= (1)(1 + 2 + 3 + \cdots + n) + (1/2)(2 + 3 + \cdots + n) + \\ &\quad (1/3)(3 + \cdots + n) + \cdots + (1/n)(n).\end{aligned}$$

We'll leave the remainder of the derivation as one of the exercises.

example 5.14 Sums within Sums

We'll consider the following algorithm with two loops:

```
k := 1;
while k < n do
    k := k + 1;
    for j := 1 to k do S(j) od
od
```

We will assume that each call to $S(j)$ executes about n/j operations of a type that we wish to count. For example, $S(j)$ might do some work on a collection of $\lfloor n/j \rfloor$ subsets of an n-element set, each of size j with the possibility of some subset of size less than j remaining. We should be using $\lfloor n/j \rfloor$ instead of n/j. But we'll work with n/j to keep the calculations simpler.

To start things off, we'll examine the for-loop for some value of k. This means that $S(j)$ is called k times with j taking values $1, 2, \ldots, k$. Since $S(j)$ executes n/j operations, the number of operations executed in each for-loop by S is

$$\sum_{j=1}^{k} (n/j) = n \sum_{j=1}^{k} (1/j) = nH_k.$$

Now we need to find the values of k at the for-loop. The values of k that enter the while-loop are 1, 2, ..., $n-1$. Since k gets incremented by 1 upon entry, the values of k at the for-loop are 2, 3, ..., n. So the number of operations by S is given by

$$\sum_{k=2}^{n} nH_k = n \sum_{k=2}^{n} H_k = n \left(\sum_{k=1}^{n} H_k - H_1 \right) = n \left(\sum_{k=1}^{n} H_k - 1 \right) = n(n+1)H_n - 2n.$$

Since we have an approximation for H_n, we can get a pretty good idea of the number of operations.

end example

We'll see some other ways to represent approximations in Section 5.6. To see other results about harmonic numbers, see Knuth [1968] or Graham, Knuth, and Patashnik [1989].

5.2.5 Polynomials and Partial Fractions

Before we proceed further with summations, let's recall a few facts about polynomials and partial fractions that we can use to simplify quotients of polynomials (also known as rational functions). So we will consider expressions of the following form, where $p(x)$ and $q(x)$ are two polynomials:

$$\frac{p(x)}{q(x)}.$$

To work with partial fractions the degree of $p(x)$ must be less than the degree of $q(x)$. If the degree of $p(x)$ is greater than or equal to the degree of $q(x)$, then we can transform the expression into the following form, where $s(x)$, $p_1(x)$, and $q_1(x)$ are polynomials and the degree of $p_1(x)$ is less than the degree of $q_1(x)$:

$$\frac{p(x)}{q(x)} = s(x) + \frac{p_1(x)}{q_1(x)}.$$

The transformation can be carried out by using long division for polynomials.

Dividing Polynomials

For example, suppose we have the following quotient of polynomials.

$$\frac{2x^3+1}{x^2+3x+2}.$$

The degree of the numerator, which is 3, is greater than or equal to the degree of the denominator, which is 2. So we can divide the numerator by the denominator (the divisor) by using long division as follows, where division takes place between the terms of highest degree:

$$\begin{array}{r} 2x \\ x^2+3x+2 \;\overline{)\; 2x^3 \qquad\qquad +1} \\ \underline{2x^3+6x^2+4x} \\ -6x^2-4x+1 \end{array}.$$

Since the degree of the remainder, which is 2, is still greater than or equal to the degree of the divisor, we need to carry out the division one more step to obtain

$$\begin{array}{r} 2x-6 \\ x^2+3x+2 \;\overline{)\; 2x^3 \qquad\qquad +1} \\ \underline{2x^3+6x^2+4x} \\ -6x^2-4x+1 \\ \underline{-6x^2-18x-12} \\ 14x+13 \end{array}.$$

The degree of the remainder, which is 1, is now less than the degree of the divisor. So we can proceed just like the division algorithm for integers to write the dividend as the divisor times the quotient plus the remainder as follows:

$$2x^3+1=(x^2+3x+2)(2x-6)+(14x+13).$$

Now divide the equation by the divisor x^2+3x+2 to obtain the following desired form:

$$\frac{2x^3+1}{x^2+3x+2}=(2x-6)+\frac{14x+13}{x^2+3x+2}.$$

Partial Fractions

Now assume that we have the following quotient of polynomials $p(x)$ and $q(x)$, where the degree of $p(x)$ is less than the degree of $q(x)$.

$$\frac{p(x)}{q(x)}.$$

Then the *partial fraction* representation (or expansion or decomposition) of the expression is a sum of terms that satisfy the following rules, where $q(x)$ has been factored into a product of linear and/or quadratic factors.

Partial Fractions

1. If the linear polynomial $ax + b$ is repeated k times as a factor of $q(x)$, then add the following terms to the partial fraction representation, where A_1, ..., A_k are constants to be determined:

$$\frac{A_1}{ax+b} + \frac{A_2}{(ax+b)^2} + \cdots + \frac{A_k}{(ax+b)^k}.$$

2. If the quadratic polynomial $cx^2 + dx + e$ is repeated k times as a factor of $q(x)$, then add the following terms to the partial fraction representation, where A_i and B_i are constants to be determined:

$$\frac{A_1x+B_1}{cx^2+dx+e} + \frac{A_2x+B_2}{(cx^2+dx+e)^2} + \cdots + \frac{A_kx+B_k}{(cx^2+dx+e)^k}.$$

Some Sample Partial Fractions

Here are a few samples of partial fractions that can be obtained from the two rules.

$$\frac{x-1}{x(x-2)(x+1)} = \frac{A}{x} + \frac{B}{x-2} + \frac{C}{x+1}$$

$$\frac{x^3-1}{x^2(x-2)^3} = \frac{A}{x} + \frac{B}{x^2} + \frac{C}{x-2} + \frac{D}{(x-2)^2} + \frac{E}{(x-2)^3}$$

$$\frac{x^2}{(x-1)(x^2+x+1)} = \frac{A}{x-1} + \frac{Bx+C}{x^2+x+1}$$

$$\frac{x}{(x-1)(x^2+1)^2} = \frac{A}{x-1} + \frac{Bx+C}{x^2+1} + \frac{Dx+C}{(x^2+1)^2}.$$

Determining the Constants

To determine the constants in a partial fraction representation, we can solve simultaneous equations. If there are n constants to be found, then we need to create n equations. One way to accomplish this is to multiply the equation by the greatest common denominator, collect terms, and equate coefficients. Another was is to pick n values for x, with the restriction that no value of x makes any denominator zero.

For example, suppose we want to represent the following expression as the sum of partial fractions.

$$\frac{x+1}{(2x-1)(3x-1)}.$$

Since the degree of the numerator is less than the degree of the denominator, the rules tell us to write

$$\frac{x+1}{(2x-1)(3x-1)} = \frac{A}{2x-1} + \frac{B}{3x-1}.$$

Now we need to create two equations in A and B. One way to proceed is to multiply the equation by the greatest common denominator and then equate coefficients in the resulting equation. So we multiply both sides by $(2x-1)(3x-1)$ to obtain

$$x+1 = A(3x-1) + B(2x-1).$$

Collect terms on the right side to obtain

$$x+1 = (3A+2B)x + (-A-B).$$

Now equate coefficients to obtain the two equations

$$1 = -A - B$$
$$1 = 3A + 2B.$$

Solving for A and B, we get $A = 3$ and $B = -4$.

An alternative way to obtain two equations in A and B is to pick two values to substitute for x in the equation to obtain two equations. For example, let $x = 0$ and $x = 1$ to obtain the two equations

$$1 = -A - B$$
$$1 = A + (1/2)B.$$

Solving for A and B, we get $A = 3$ and $B = -4$, as shown. So the partial fraction representation of the expression is

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

example 5.15 Partial Fractions and Collapsing Sums

Consider the following summation.

$$\sum_{k=1}^{n} \frac{1}{k^2+k}.$$

It's always fun to write out a few terms of the sum to see whether a pattern of some kind emerges. You might try it. But with new tools available, we might notice that the summand has a partial fraction representation as

$$\frac{1}{k^2+k} = \frac{1}{k(k+1)} = \frac{A}{k} + \frac{B}{k+1} = \frac{1}{k} - \frac{1}{k+1}.$$

So we can write the sum as follows, where the sum of differences collapses to the answer.

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

end example

example **5.16 A Sum with a Harmonic Answer**

We'll evaluate the following sum:

$$\sum_{k=1}^{n} \frac{14k+13}{k^2+3k+2}.$$

We can use partial fractions to put the expression into the following workable form:

$$\frac{14k+13}{k^2+3k+2} = \frac{14k+13}{(k+1)(k+2)} = \frac{-1}{k+1} + \frac{15}{k+2}.$$

So we can calculate the sum as follows:

$$\begin{aligned}
\sum_{k=1}^{n} \frac{14k+13}{k^2+3k+2} &= \sum_{k=1}^{n} \left(\frac{15}{k+2} - \frac{1}{k+1}\right) \\
&= 15\sum_{k=1}^{n} \frac{1}{k+2} - \sum_{k=1}^{n} \frac{1}{k+1} \\
&= 15\sum_{k=3}^{n+2} \frac{1}{k} - \sum_{k=2}^{n+1} \frac{1}{k} \\
&= 15(H_{n+2} - H_2) - (H_{n+1} - H_1).
\end{aligned}$$

end example

Exercises

Closed Forms for Sums

1. Expand each expression into a sum of terms. Don't evaluate.

   a. $\sum_{k=1}^{5}(2k+3)$. b. $\sum_{k=1}^{5}k3^k$. c. $\sum_{k=0}^{4}(5-k)3^k$.

2. (*Changing Limits of Summation*). Given the following summation expression:

$$\sum_{k=1}^{n} g(k-1)a_k x^{k+1}.$$

   For each of the following lower limits of summation, find an equivalent summation expression that starts with that lower limit:

   a. $k=0$. b. $k=2$. c. $k=-1$. d. $k=3$. e. $k=-2$.

3. Find a closed form for each of the following sums:

   a. $3+6+9+12+\cdots+3n$.
   b. $3+9+15+21+\cdots+(6n+3)$.
   c. $3+6+12+24+\cdots+3(2^n)$.
   d. $3+(2)3^2+(3)3^3+(4)3^4+\cdots+n3^n$.

4. Use summation facts and known closed forms to transform each of the following summations into a closed form:

   a. $\sum_{k=1}^{n}(2k+2)$. b. $\sum_{k=1}^{n}(2k-1)$.
   c. $\sum_{k=1}^{n}(2k+3)$. d. $\sum_{k=1}^{n}(4k-1)$.
   e. $\sum_{k=1}^{n}(4k-2)$. f. $\sum_{k=1}^{n}k2^k$.
   g. $\sum_{k=1}^{n}k(k+1)$. h. $\sum_{k=2}^{n}(k^2-k)$.

5. Verify the two collapsing sum formulas in (5.1c) for the case $n=3$.

6. Verify Abel's summation transformation (5.12) for the case $n=2$.

7. Verify the following *Summation by Parts* formula for the case $n=2$.

$$\sum_{k=0}^{n} a_k(b_{k+1}-b_k) = a_{n+1}b_{n+1} - a_0b_0 + \sum_{k=0}^{n} b_{k+1}(a_{k+1}-a_k).$$

8. Use properties of sums and logs to calculate the given value for each of the following sums:

   a. $\sum_{i=0}^{k-1} 2^i (1/5^i)^{\log_5 2} = k.$

   b. $\sum_{i=0}^{k-1} 3^{2i} (1/5^i)^{\log_5 3} = (3^k - 1)/2.$

   c. $\sum_{i=0}^{k-1} (1/5^i)^{\log_5 2} = 2 - (1/2)^{k-1}.$

9. Use properties of sums and logs to show that the following equation holds, where H_k is the kth harmonic number and $n = 2^k$.

$$\sum_{i=0}^{k-1} \frac{1}{\log_2(n/2^i)} = H_k.$$

10. In each case find a closed form for the sum in terms of harmonic numbers.

   a. $\sum_{k=0}^{n} \frac{1}{2k+1}.$ b. $\sum_{k=1}^{n} \frac{1}{2k-1}.$

11. In each case find a closed form for the sum in terms of harmonic numbers by using division and partial fractions to simplify the summand.

   a. $\sum_{k=1}^{n} \frac{k}{k+1}.$ b. $\sum_{k=1}^{n} \frac{k^2}{k+1}.$

12. Finish the derivation of the sum (5.23) starting from where the sum was rearranged.

Analyzing Algorithms

13. Given the following algorithm, find a formula in terms of n for each case:

```
for i := 1 to n do
    for j := 1 to i do x := x + f(x) od;
    x := x + g(x)
od
```

   a. Find the number of times that the assignment statement (:=) is executed during the running of the program.

   b. Find the number of times that the addition operation (+) is executed during the running of the program.

14. Given the following algorithm, find a formula in terms of n for each case:

```
i := 1;
while i < n + 1 do
    i := i + 1;
    for j := 1 to i do S od
od
```

a. Find the number of times that the statement S is executed during the running of the program.

b. Find the number of times that the assignment statement (:=) is executed during the running of the program.

15. Given the following algorithm, find a formula in terms of n for each case:

```
i := 1;
while i ¡ n + 1 do
    i := i + 2;
    for j := 1 to i do S od
od
```

a. Find the number of times that the statement S is executed during the running of the program.

b. Find the number of times that the assignment statement (:=) is executed during the running of the program.

Challenges

16. Use the collapsing sums technique from Example 5.7 to find a closed form for $\sum_{k=1}^{n} k^3$.

17. Use the technique introduced in Example 5.9 to derive a closed form for the sum

$$\sum_{k=1}^{n} k^2 a^k \text{ by using the known closed forms for } \sum_{k=1}^{n} k a^k \text{ and } \sum_{k=1}^{n} a^k.$$

18. In each case use (5.18) to find upper and lower bounds for the sum and compare the results with the actual closed form for the sum.

a. $\sum_{k=1}^{n} k.$ b. $\sum_{k=1}^{n} k^2.$

19. Verify that the two bounds in (5.21) differ by less than ln 2.

20. Notice that the average value $(1/2)(1/k + 1/(k+ 1))$ of the areas of the two rectangles with base $[k, k + 1]$ and heights $1/k$ and $1/(k + 1)$, respectively (i.e., the area of the trapezoid with sides of length $1/k$ and $1/(k + 1)$) is larger than the definite integral of $1/x$ on the interval $[k, k + 1]$.

a. Use this fact to obtain the following lower bound for H_n.

$$\ln n + \frac{1}{2n} + \frac{1}{2}.$$

b. Show that the lower bound in Part (a) is better (i.e., greater) than the lower bound given in (5.21).

5.3 Permutations and Combinations

Whenever we need to count the number of ways that some things can be arranged or the number of subsets of things, there are some nice techniques that can help out. That's what our discussion of permutations and combinations will be about.

Two Counting Rules

Before we start, we should recall two useful rules of counting.

The *rule of sum* states that if there are m choices for some event to occur and n choices for another event to occur and the events are disjoint, then there are $m + n$ choices for either event to occur. This is just another way to say that the cardinality of the union of disjoint sets is the sum of the cardinalities of the two sets (1.11).

The *rule of product* states that if there are m choices for some event and n choices for another event, then there are mn choices for both events. This is just another way to say that the cardinality of the Cartesian product of two finite sets is the product of the cardinalities of the two sets (1.17).

For example, if a vending machine has 6 types of drink and 18 types of snack food, then there are $6 + 18 = 24$ possible choices in the machine and $6 \cdot 18 = 108$ possible choices of a drink and a snack.

Both rules are sometimes used to answer a question. For example, suppose we have a bookshelf with 5 technical books, 12 biographies, and 37 novels. We intend to take two books of different type to read while on vacation. There are $5 \cdot 12 = 60$ ways to choose a technical book and a biography, $5 \cdot 37 = 185$ ways to choose a technical book and a novel, and $12 \cdot 37 = 444$ ways to choose a biography and novel. So there are $50 + 185 + 444 = 689$ ways to choose two books of different type.

5.3.1 Permutations (Order Is Important)

An arrangement (i.e., an ordering) of distinct objects is called a *permutation* of the objects. For example, one permutation of the three letters in the set

$\{a, b, c\}$ is *bca*. We can count the number of permutations by noticing that there are 3 choices for the first letter. For each choice of a first letter there are 2 choices remaining for the second letter, and upon picking a second letter there is one letter remaining. Therefore there are $3 \cdot 2 \cdot 1 = 6$ permutations of a 3-element set. The six permutations of $\{a, b, c\}$ can be listed as

$$abc, acb, bac, bca, cab, cba.$$

The idea generalizes to permutations of an n-element set. There are n choices for the first element. For each of these choices there are $n - 1$ choices for the second element. Continuing in this way, we obtain $n \cdot (n-1) \cdots 2 \cdot 1$ different permutations of n elements. The notation for this product is the symbol $n!$, which we read as "n factorial." So we have

$$n! = n \cdot (n-1) \cdots 2 \cdot 1.$$

By convention, we set $0! = 1$. So here's the first rule of permutations.

Permutations (5.24)

The number of permutations of n distinct objects is n factorial.

Now suppose we want to count the number of permutations of r elements chosen from an n-element set, where $1 \leq r \leq n$. There are n choices for the first element. For each of these choices there are $n-1$ choices for the second element. We continue this process r times to obtain the answer,

$$n(n-1) \cdots (n-r+1) = \frac{n!}{(n-r)!}. \tag{5.25}$$

This number is denoted by $P(n, r)$. Here's the definition and the rule.

Permutations

An *r-permutation* of n distinct objects is a permutation of r of the objects. The number of r-permutations of n distinct objects is

$$P(n,\ r) = \frac{n!}{(n-r)!}. \tag{5.26}$$

Notice that $P(n,\ 1) = n$ and $P(n,\ n) = n!$. If $S = \{a,\ b,\ c,\ d\}$, then there are 12 permutations of two elements from S, given by the formula $P(4,\ 2) = 4!/2! = 12$. The permutations are listed as follows:

$$ab,\ ba,\ ac,\ ca,\ ad,\ da,\ bc,\ cb,\ bd,\ db,\ cd,\ dc.$$

Permutations with Repeated Elements

Permutations can be thought of as arrangements of objects selected from a set *without replacement*. In other words, we can't pick an element from the set more than once. If we can pick an element more than once, then the objects are said to be selected *with replacement*. In this case the number of arrangements of r objects from an n-element set is just n^r. We can state this idea in terms of bags as follows: The number of distinct permutations of r objects taken from a bag containing n distinct objects, each occurring r times, is n^r. For example, consider the bag $B = [a, a, b, b, c, c]$. Then the number of distinct permutations of two objects chosen from B is 3^2, and they can be listed as follows:

$$aa,\ ab,\ ac,\ ba,\ bb,\ bc,\ ca,\ cb,\ cc.$$

Let's look now at permutations of all the elements in a bag. For example, suppose we have the bag $B = [a, a, b, b, b]$. We can write down the distinct permutations of B as follows:

$$aabbb,\ ababb,\ abbab,\ abbba,\ baabb,\ babab,\ babba,\ bbaab,\ bbaba,\ bbbaa.$$

There are 10 strings. Let's see how to compute the number 10 from the information we have about the bag B. One way to proceed is to place subscripts on the elements in the bag, obtaining the five distinct elements a_1, a_2, b_1, b_2, b_3. Then we get $5! = 120$ permutations of the five distinct elements. Now we remove all the subscripts on the elements, and we find that there are many repeated strings among the original 120 strings.

For example, suppose we remove the subscripts from the two strings,

$$a_1b_1b_2a_2b_3 \quad \text{and} \quad a_2b_1b_3a_1b_2.$$

Then we obtain two occurrences of the string *abbab*. If we wrote all occurrences down, we would find 12 strings, all of which reduce to the string *abbab* when subscripts are removed. This is because there are 2! permutations of the letters a_1 and a_2, and there are 3! permutations of the letters b_1, b_2, and b_3. So there are $2!3! = 12$ distinct ways to write the string *abbab* when we use subscripts. Of course, the number is the same for any string of two a's and three b's. Therefore, the number of distinct strings of two a's and three b's is found by dividing the total number of subscripted strings by 2!3! to obtain $5!/2!3! = 10$. This argument generalizes to obtain the following result about permutations that can contain repeated elements.

Permutations of a Bag **(5.27)**

Let B be an n-element bag with k distinct elements, where each of the numbers $m_1, \ldots, m_k$ denotes the number of occurrences of each element. Then the number of permutations of the n elements of B is

$$\frac{n!}{m_1! \cdots m_k!}.$$

Now let's look at a few examples to see how permutations (5.24)–(5.27) can be used to solve a variety of problems. We'll start with an important result about sorting.

example 5.17 Worst-Case Lower Bound for Comparison Sorting

Let's find a lower bound for the number of comparisons performed by any algorithm that sorts by comparing elements in the list to be sorted. Assume that we have a set of n distinct numbers. Since there are $n!$ possible arrangements of these numbers, it follows that any algorithm to sort a list of n numbers has $n!$ possible input arrangements. Therefore, any decision tree for a comparison sorting algorithm must contain at least $n!$ leaves, one leaf for each possible outcome of sorting one arrangement.

We know that a binary tree of depth d has at most 2^d leaves. So the depth d of the decision tree for any comparison sort of n items must satisfy the inequality

$$n! \leq 2^d.$$

We can solve this inequality for the natural number d as follows:

$$\log_2 n! \leq d$$
$$\lceil \log_2 n! \rceil \leq d.$$

In other words, $\lceil \log_2 n! \rceil$ is a worst-case lower bound for the number of comparisons to sort n items. The number $\lceil \log_2 n! \rceil$ is hard to calculate for large values of n. We'll see in Section 5.6 that it is approximately $n \log_2 n$.

end example

example 5.18 People in a Circle

In how many ways can 20 people be arranged in a circle if we don't count a rotation of the circle as a different arrangement? There are 20! arrangements of 20 people in a line. We can form a circle by joining the two ends of a line. Since there are 20 distinct rotations of the same circle of people, it follows that there are

$$\frac{20!}{20} = 19!$$

distinct arrangements of 20 people in a circle. Another way to proceed is to put one person in a certain fixed position of the circle. Then fill in the remaining 19 people in all possible ways to get 19! arrangements.

end example

example 5.19 Rearranging a String

How many distinct strings can be made by rearranging the letters of the word *banana*? One letter is repeated twice, one letter is repeated three times, and one letter stands by itself. So we can answer the question by finding the number of permutations of the bag of letters $[b, a, n, a, n, a]$. Therefore, (5.27) gives us the result

$$\frac{6!}{1!2!3!} = 60.$$

end example

example 5.20 Strings with Restrictions

How many distinct strings of length 10 can be constructed from the two digits 0 and 1 with the restriction that five characters must be 0 and five must be 1? The answer is

$$\frac{10!}{5!5!} = 252$$

because we are looking for the number of permutations from a 10-element bag with five 1's and five 0's.

example 5.21 Constructing a Code

Suppose we want to build a code to represent each of 29 distinct objects with a binary string having the same minimal length n, where each string has the same number of 0's and 1's. Somehow we need to solve an inequality like

$$\frac{n!}{k!k!} \geq 29,$$

where $k = n/2$. We find by trial and error that $n = 8$. Try it.

end example

5.3.2 Combinations (Order Is Not Important)

When we choose, or select, some objects from a set of objects, without regard to order, the choice is called a *combination*. Since order does not matter we can represent combinations as sets. For example, one combination of two elements from the set $\{a, b, c, d\}$ is the subset $\{a, b\}$. There are six 2-element combinations from $\{a, b, c, d\}$, which we can list as

$$\{a, b\}, \{a, c\}, \{a, d\}, \{b, c\}, \{b, d\}, \{c, d\}.$$

We want to count combinations without listing them. We can get the idea from our little example. We'll start by observing that the number of 2-permutations of the set is $4 \cdot 3 = 12$. But we're not counting permutations. We've counted each 2-element combination twice. For example, $\{a, b\}$ has been counted twice, once for each of its permutations ab and ba. So we must divide 12 by 2 to obtain the correct number of combinations.

We can generalize this idea to find a formula for the number of r-element combinations of an n-element set. First count the number of r-permutations of n elements, which is given by the formula

$$P(n, r) = \frac{n!}{(n-r)!}.$$

Now observe that each r-element combination has been counted $r!$ times, once for each of its permutations. So we must divide $P(n, r)$ by $r!$ to obtain the number of r-element combinations of an n-element set

$$\frac{P(n, r)}{r!} = \frac{n!}{r!(n-r)!}.$$

This formula is denoted by the expression $C(n, r)$. Here's the definition and the rule.

end examp

Combinations **(5.28)**

An *r-combination* of n distinct objects is a combination of r of the objects. The number of *r-combinations* chosen from n distinct objects is

$$C(n, r) = \frac{n!}{r!(n-r)!}.$$

$C(n, r)$ is often read "n choose r."

example **5.22 Subsets of the Same Size**

Let $S = \{a, b, c, d, e\}$. We'll list all the three-element subsets of S:

$$\{a, b, c\}, \{a, b, d\}, \{a, b, e\}, \{a, c, d\}, \{a, c, e\},$$
$$\{a, d, e\}, \{b, c, d\}, \{b, c, e\}, \{b, d, e\}, \{c, d, e\}.$$

There are 10 such subsets, which we can verify by the formula

$$C(5, 3) = \frac{5!}{3!2!} = 10.$$

end example

Binomial Coefficients

Notice how $C(n, r)$ crops up in the following binomial expansion of the expression $(x + y)^4$:

$$\begin{aligned}(x+y)^4 &= x^4 + 4x^3y + 6x^2y^2 + 4xy^3 + y^4 \\ &= C(4,0)\,x^4 + C(4,1)\,x^3y + C(4,2)\,x^2y^2 + C(4,3)\,xy^3 + C(4,4)\,y^4.\end{aligned}$$

A useful way to represent $C(n, r)$ is with the *binomial coefficient symbol:*

$$\binom{n}{r} = C(n, r).$$

Using this symbol, we can write the expansion for $(x + y)^4$ as follows:

$$\begin{aligned}(x+y)^4 &= x^4 + 4x^3y + 6x^2y^2 + xy^3 + y^4 \\ &= \binom{4}{0}x^4 + \binom{4}{1}x^3y + \binom{4}{2}x^2y^2 + \binom{4}{3}xy^3 + \binom{4}{4}y^4.\end{aligned}$$

This is an instance of a well-known formula called the *binomial theorem*, which can be written as follows, where n is a natural number:

Binomial Theorem (5.29)

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

Pascal's Triangle

The binomial coefficients for the expansion of $(x + y)^n$ can be read from the nth row of the table in Figure 5.4. The table is called *Pascal's triangle*—after the philosopher and mathematician Blaise Pascal (1623–1662). However, prior to the time of Pascal, the triangle was known in China, India, the Middle East, and Europe. Notice that any interior element is the sum of the two elements above and to its left.

But how do we really know that the following statement is correct?

Elements in Pascal's Triangle (5.30)

The nth row kth column entry of Pascal's triangle is $\binom{n}{k}$.

	0	1	2	3	4	5	6	7	8	9	10
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				
7	1	7	21	35	35	21	7	1			
8	1	8	28	56	70	56	28	8	1		
9	1	9	36	84	126	126	84	36	9	1	
10	1	10	45	120	210	252	210	120	45	10	1

Figure 5.4 Pascal's triangle.

Proof: For convenience we will designate a position in the triangle by an ordered pair of the form (row, column). Notice that the edge elements of the triangle are all 1, and they occur at positions $(n, 0)$ or (n, n). Notice also that

$$\binom{n}{0} = 1 = \binom{n}{n}.$$

So (5.30) is true when $k = 0$ or $k = n$. Next, we need to consider the interior elements of the triangle. So let $n > 1$ and $0 < k < n$. We want to show that the element in position (n, k) is $\binom{n}{k}$. To do this, we need the following useful result about binomial coefficients:

$$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}. \tag{5.31}$$

To prove (5.31), just expand each of the three terms and simplify. Continuing with the proof of (5.30), we'll use well-founded induction. To do this, we need to define a well-founded order on something. For our purposes we will let the something be the set of positions in the triangle. We agree that any position in row $n - 1$ precedes any position in row n. In other words, if $n' < n$, then (n', k') precedes (n, k) for any values of k' and k. Now we can use well-founded induction. We pick position (n, k) and assume that (5.30) is true for all pairs in row $n - 1$. In particular, we can assume that the elements in positions $(n - 1, k)$ and $(n - 1, k - 1)$ have values

$$\binom{n-1}{k} \quad \text{and} \quad \binom{n-1}{k-1}.$$

Now we use this assumption along with (5.31) to tell us that the value of the element in position (n, k) is $\binom{n}{k}$. QED.

Can you find some other interesting patterns in Pascal's triangle? There are lots of them. For example, look down the column labeled 2 and notice that, for each $n \geq 2$, the element in position $(n, 2)$ is the value of the arithmetic sum $1 + 2 + \cdots + (n-1)$. In other words, we have the formula

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

Combinations with Repeated Elements

Let's continue our discussion about combinations by counting bags of things rather than sets of things. Suppose we have the set $A = \{a, b, c\}$. How many 3-element bags can we construct from the elements of A? We can list them as follows:

$$[a,a,a], [a,a,b], [a,a,c], [a,b,c], [a,b,b],$$
$$[a,c,c], [b,b,b], [b,b,c], [b,c,c], [c,c,c].$$

So there are ten 3-element bags constructed from the elements of $\{a, b, c\}$.

Let's see if we can find a general formula for the number of k-element bags that can be constructed from an n-element set. For convenience, we'll assume that the n-element set is $A = \{1, 2, \ldots, n\}$. Suppose that $b = [x_1, x_2, x_3, \ldots, x_k]$ is some k-element bag with elements chosen from A, where the elements of b are written so that $x_1 \leq x_2 \leq \cdots \leq x_k$. This allows us to construct the following k-element set:

$$B = \{x_1, x_2 + 1, x_3 + 2, \ldots, x_k + (k-1)\}.$$

The numbers $x_i + (i-1)$ are used to ensure that the elements of B are distinct elements in the set $C = \{1, 2, \ldots, n + (k-1)\}$. So we've associated each k-element bag b over A with a k-element subset B of C. Conversely, suppose that $\{y_1, y_2, y_3, \ldots, y_k\}$ is some k-element subset of C, where the elements are written so that $y_1 \leq y_2 \leq \cdots \leq y_k$. This allows us to construct the k-element bag

$$[y_1, y_2 - 1, y_3 - 2, \ldots, y_k - (k-1)],$$

whose elements come from the set A. So we've associated each k-element subset of C with a k-element bag over A.

Therefore, the number of k-element bags over an n-element set is exactly the same as the number of k-element subsets of a set with $n + (k-1)$ elements. This gives us the following result.

Bag Combinations **(5.32)**

The number of k-element bags whose distinct elements are chosen from an n-element set, where k and n are positive, is given by

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

example 5.23 Selecting Coins

In how many ways can four coins be selected from a collection of pennies, nickels, and dimes? Let $S = \{\text{penny, nickel, dime}\}$. Then we need the number of 4-element bags chosen from S. The answer is

$$\binom{3+4-1}{4} = \binom{6}{4} = 15.$$

end example

example 5.24 Selecting a Committee

In how many ways can five people be selected from a collection of Democrats, Republicans, and Independents? Here we are choosing five-element bags from a set of three characteristics {Democrat, Republican, Independent}. The answer is

$$\binom{3+5-1}{5} = \binom{7}{5} = 21.$$

end example

Exercises

Permutations and Combinations

1. Evaluate each of the following expressions.

a. $P(6, 6)$. b. $P(6, 0)$. **c.** $P(6, 2)$.
d. $P(10, 4)$. **e.** $C(5, 2)$. f. $C(10, 4)$.

2. Let $S = \{a, b, c\}$. Write down the objects satisfying each of the following descriptions.

a. All permutations of the three letters in S.
b. All permutations consisting of two letters from S.
c. All combinations of the three letters in S.
d. All combinations consisting of two letters from S.
e. All bag combinations consisting of two letters from S.

3. For each part of Exercise 2, write down the formula, in terms of P or C, for the number of objects requested.

4. Given the bag $B = [a, a, b, b]$, write down all the bag permutations of B, and verify with a formula that you wrote down the correct number.

5. Find the number of ways to arrange the letters in each of the following words. Assume all letters are lowercase.

a. Computer. b. Radar. **c.** States.
d. Mississippi. **e.** Tennessee.

6. A *derangement* of a string is a permutation of the letters such that each letter changes its position. For example, a derangement of the string ABC is BCA. But ACB is not a derangement of ABC, since A does not change position. Write down all derangements for each of the following strings.

a. A. b. AB. **c.** ABC. d. $ABCD$.

7. Suppose we want to build a code to represent 29 objects in which each object is represented as a binary string of length n, which consists of k 0's and m 1's, and $n = k + m$. Find n, k, and m, where n has the smallest possible value.

8. We wish to form a committee of seven people chosen from five Democrats, four Republicans, and six Independents. The committee will contain two Democrats, two Republicans, and three Independents. In how many ways can we choose the committee?

9. Each of the following problems refers to a property of Pascal's triangle (Figure 5.4).

a. Each row has a largest number. Find a formula to describe which column contains the largest number in row n.
b. Show that the sum of the numbers in row n is 2^n.

10. A deck of 52 playing cards has four suits (Clubs, Diamonds, Hearts, Spades) with each suit having thirteen cards with ranks from high to low: Ace, King, Queen, Jack, 10, 9, 8, 7, 6, 5, 4, 3, and 2. In each case, find the number of possible 5-card hands with the given property.

a. The total number of 5-card hands.
b. Straight flush: A 5-card sequence of the same suit, such as 3, 4, 5, 6, 7 of Clubs. (Note that the Ace can also be used as the low card in a straight.)
c. Four of a kind: Four cards of the same rank and one of another rank.
d. Full house: Three of one rank and two of another rank such as 10, 10, 10, Jack, Jack.
e. Flush: All five cards the same suit but not a straight flush.
f. Straight: A 5-card sequence but not a straight flush.
g. Three of a kind: Three of the same rank and two others not of the same rank, such as Queen, Queen, Queen, 4, 5.
h. Two pair: Each pair has a different rank with a 5^{th} card of neither rank, such as 4, 4, Queen, Queen, 5.

i. One pair: Two cards of the same rank and three other different rank cards, such as Ace, Ace, 4, 9, Jack.

j. High card: Five different ranks but not a flush and not a straight.

Challenges

11. Test the birthday problem on a group of people.

12. Suppose an operating system must schedule the execution of n processes, where each process consists of k separate actions that must be done in order. Assume that any action of one process may run before or after any action of another process. How many execution schedules are possible?

13. Count the number of strings consisting of n 0's and n 1's such that each string is subject to the following restriction: As we scan a string from left to right, the number of 0's is never greater than the number of 1's. For example, the string 110010 is OK, but the string 100110 is not. *Hint:* Count the total number of strings of length $2n$ with n 0's and n 1's. Then try to count the number that are not OK, and subtract this number from the total number.

14. Given a nonempty finite set S with n elements, prove that there are $n!$ bijections from S to S.

5.4 Discrete Probability

The founders of probability theory were Blaise Pascal (1623–1662) and Pierre Fermat (1601–1665). They developed the principles of the subject in 1654 during a correspondence about games of chance. It started when Pascal was asked about a gambling problem. The problem asked how the stakes of a "points" game should be divided up between two players if they quit before either had enough points to win.

Probability comes up whenever we ask about the chance of something happening. To answer such a question requires one to make some kind of assumption. For example, we might ask about the average behavior of an algorithm. That is, instead of the worst-case performance, we might be interested in the average-case performance. This can be a bit tricky because it usually forces us to make one or two assumptions. Some people hate to make assumptions. But it's not so bad. Let's do an example.

Suppose we have a sorted list of the first 15 prime numbers, and we want to know the average number of comparisons needed to find a number in the list, using a binary search. The decision tree for a binary search of the list is pictured in Figure 5.5.

After some thought, you might think it reasonable to add up all the path lengths from the root to a leaf marked with an S (for successful search) and

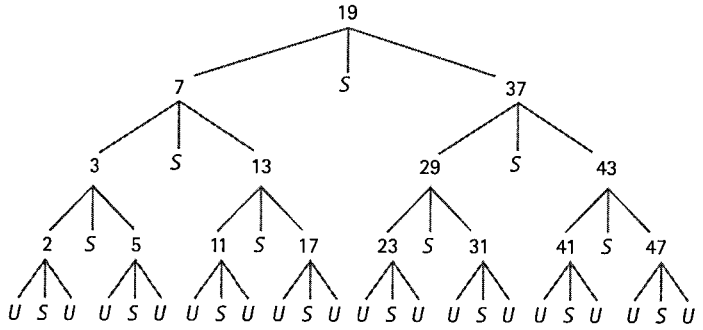


Figure 5.5 Binary search decision tree.

divide by the number of S leaves, which is 15. In this case there are eight paths of length 4, four paths of length 3, two paths of length 2, and one path of length 1. So we get

$$\text{Average path length} = \frac{32+12+4+1}{15} = \frac{49}{15} \approx 3.27.$$

This gives us the average number of comparisons needed to find a number in the list. Or does it? Have we made any assumptions here? Yes, we assumed that each path in the tree has the same chance of being traversed as any other path. Of course, this might not be the case. For example, suppose that we always wanted to look up the number 37. Then the average number of comparisons would be two. So our calculation was made under the assumption that each of the 15 numbers had the same chance of being picked.

5.4.1 Probability Terminology

Let's pause here and introduce some notions and notations for *discrete probability*, which gives us methods to calculate the likelihood of events that have a finite number of outcomes. If some operation or experiment has n possible outcomes and each outcome has the same chance of occurring, then we say that each outcome has *probability* $1/n$. In the preceding example we assumed that each number had probability $1/15$ of being picked. As another example, let's consider the coin-flipping problem. If we flip a fair coin, then there are two possible outcomes, assuming that the coin does not land on its edge. Thus the probability of a head is $1/2$, and the probability of a tail is $1/2$. If we flip the coin 1,000 times, we should expect about 500 heads and 500 tails. So probability has something to do with expectation.

Now for some terminology. The set of all possible outcomes of an experiment is called a *sample space* or *probability space*. The elements in a sample space are called *sample points* or simply *points*. Further, any subset of a sample space is called an *event*. For example, suppose we flip two coins and are interested in the set of possible outcomes. Let H and T stand for head and tail, respectively, and

let the string HT mean that the first coin lands H and the second coin lands T. Then the sample space for this experiment is the set

$$\{HH, HT, TH, TT\}.$$

For example, the event that one coin lands as a head and the other coin lands as a tail can be represented by the subset $\{HT, TH\}$.

To discuss probability we need to make assumptions or observations about the probabilities of sample points. Here is the terminology.

Probability Distribution
A *probability distribution* on a sample space S is an assignment of probabilities to the points of S such that the sum of all the probabilities is 1.

Let's describe a probability distribution from a more formal point of view. Let $S = \{x_1, x_2, \ldots, x_n\}$ be a sample space. A probability distribution P on S is a function

$$P : S \to [0, 1]$$

such that

$$P(x_1) + P(x_2) + \cdots + P(x_n) = 1.$$

For example, in the two-coin-flip experiment it makes sense to define the following probability distribution on the sample space $S = \{HH, HT, TH, TT\}$:

$$P(HH) = P(HT) = P(TH) = P(TT) = \frac{1}{4}.$$

Probability of an Event

Once we have a probability distribution P defined on the points of a sample space S, we can use P to define the probability of any event E in S.

Probability of an Event
The *probability* of an event E is denoted by $P(E)$ and is defined by

$$P(E) = \sum_{x \in E} P(x).$$

In particular, we have $P(S) = 1$ and $P(\varnothing) = 0$. If A and B are two events, then the following formula follows directly from the definition and the inclusion-exclusion principle.

$$P(A \cup B) = P(A) + P(B) - P(A \cap B).$$

This formula has a very useful consequence. If E' is the complement of E in S, then $S = E \cup E'$ and $E \cap E' = \varnothing$. So it follows from the formula that

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

example **5.25 Complement of an Event**

In our two-coin-flip example, let E be the event that at least one coin is a tail. Then $E = \{HT, TH, TT\}$. We can calculate $P(E)$ as follows:

$$P(E) = P(\{HT, TH, TT\}) = P(HT) + P(TH) + P(TT) = \tfrac{1}{4} + \tfrac{1}{4} + \tfrac{1}{4} = \tfrac{3}{4}.$$

But we also could observe that the complement E' is the event that both coins are heads. So we could calculate

$$P(E) = 1 - P(E') = 1 - P(HH) = 1 - \tfrac{1}{4} = \tfrac{3}{4}.$$

end example

Classic Example: The Birthday Problem

Suppose we ask 25 people, chosen at random, their birthday (month and day). Would you bet that they all have different birthdays? It seems a likely bet that no two have the same birthday since there are 365 birthdays in the year. But, in fact, the probability that two out of 25 people have the same birthday is greater than 1/2. Again, we're assuming some things here, which we'll get to shortly. Let's see why this is the case. The question we want to ask is:

> Given n people in a room, what is the probability that at least two of the people have the same birthday (month and day)?

We'll neglect leap year and assume that there are 365 days in the year. So there are 365^n possible n-tuples of birthdays for n people. This set of n-tuples is our sample space S. We'll also assume that birthdays are equally distributed throughout the year. So for any n-tuple $(b_1, \ldots, b_n)$ of birthdays, we have $P(b_1, \ldots, b_n) = 1/365^n$. The event E that we are concerned with is the subset of S consisting of all n-tuples that contain two or more equal entries. So our question can be written as follows:

What is $P(E)$?

To answer the question, let's use the complement technique. That is, we'll compute the probability of the event $E' = S - E$, consisting of all n-tuples that have distinct entries. In other words, no two of the n people have the same birthday. Then the probability that we want is $P(E) = 1 - P(E')$. So let's concentrate on E'.

n	$P(E)$
10	0.117
20	0.411
23	0.507
30	0.706
40	0.891

Figure 5.6 Birthday table.

An n-tuple is in E' exactly when all its components are distinct. The cardinality of E' can be found in several ways. For example, there are 365 possible values for the first element of an n-tuple in E'. For each of these 365 values there are 364 values for the second element of an n-tuple in E'. Thus we obtain

$$365 \cdot 364 \cdot 363 \cdots (365 - n + 1)$$

n-tuples in E'. Since each n-tuple of E' is equally likely with probability $1/365^n$, it follows that

$$P(E') = \frac{365 \cdot 364 \cdot 363 \cdots (365 - n + 1)}{365^n}.$$

Thus the probability that we desire is

$$P(E) = 1 - P(E') = 1 - \frac{365 \cdot 364 \cdot 363 \cdots (365 - n + 1)}{365^n}.$$

The table in Figure 5.6 gives a few calculations for different values of n. Notice the case when $n = 23$. The probability is better than 0.5 that two people have the same birthday. Try this out the next time you're in a room full of people. It always seems like magic when two people have the same birthday.

example 5.26 Switching Pays

Suppose there is a set of three numbers. One of the three numbers will be chosen as the winner of a three-number lottery. We pick one of the three numbers. Later, one of the two remaining numbers is identified and we are told that it is not the winner, and we are given the chance to keep the number that we picked or to switch and choose the remaining number. What should we do? We should switch.

To see this, notice that once we pick a number, the probability that we did not pick the winner is 2/3. In other words, it is more likely that one of the other two numbers is a winner. So when we are told that one of the other numbers is not the winner, it follows that the remaining other number has probability 2/3

of being the winner. So go ahead and switch. Try this experiment a few times with a friend to see that in the long run it's better to switch.

Another way to see that switching is the best policy is to modify the problem to a set of 50 numbers and a 50-number lottery. If we pick a number, then the probability that we did not pick a winner is 49/50. Later we are told that 48 of the remaining numbers are not winners, but we are given the chance to keep the number we picked or switch and choose the remaining number. What should we do? We should switch because the chance that the remaining number is the winner is 49/50.

end example

5.4.2 Conditional Probability

If we ask a question about the chance of something happening given that something else has happened, we are using conditional probability.

Conditional Probability
If A and B are events and $P(B) \neq 0$, then *the conditional probability of A given B* is denoted by $P(A|B)$ and defined by

$$P(A|B) = \frac{P(A \cap B)}{P(B)}.$$

We can think of $P(A|B)$ as the probability of the event $A \cap B$ when the sample space is restricted to B once we make an appropriate adjustment to the probability distribution.

example **5.27 Majoring in Two Subjects**

In a university it is known that 1% of students major in mathematics and 2% major in computer science. Further, it is known that 0.1% of students major in both mathematics and computer science. If we learn that a student is a computer science major, what is the probability that the student is a mathematics major? If we learn that a student is a mathematics major, what is the probability that the student is a computer science major?

To answer the questions, let A and B be the sets of mathematics majors and computer science majors, respectively. Then we have the three probabilities

$$P(A) = 0.01,\ P(B) = 0.02, \text{ and } P(A \cap B) = 0.001.$$

The two questions are answered by $P(A \mid B)$ and $P(B \mid A)$, respectively. Here are the calculations.

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{0.001}{0.02} = 0.05 \text{ and } P(B|A) = \frac{P(B \cap A)}{P(A)} = \frac{0.001}{0.01} = 0.10.$$

end example

A Useful Consequence of the Definition

From the definition of conditional probability we can obtain a simple relationship that has many applications. Given the definition

$$P(A|B) = \frac{P(A \cap B)}{P(B)}.$$

Multiply both sides by $P(B)$ to obtain the equation

$$P(A \cap B) = P(B)P(A|B).$$

Since intersection of sets is commutative, we also obtain the equation

$$P(A \cap B) = P(A)P(B|A).$$

example **5.28 Online Advertising**

A company advertises its product on a website. The company estimates that the ad will be read by 20% of the people who visit the site. It further estimates that if the ad is read, then the probability that the reader will buy the product is 0.005. What is the probability that a visitor to the website will read the ad and buy the product?

Let A be the event "read the ad" and let B be the event "buy the product." So we are given $P(A) = 0.20$ and $P(B|A) = 0.005$. We are asked to find the probability $P(A \cap B)$, which is

$$P(A \cap B) = P(A)P(B|A) = (0.20)(0.005) = 0.001.$$

end example

After-the-Fact (*a posteriori*) Probabilities

We usually use probability to give information regarding whether an event will happen in the future based on past experience. This is often called *a priori* probability. But probability can sometimes be used to answer questions about whether an event has happened based on new information. This is often called *a posteriori* probability (or after-the-fact probability).

example **5.29 Sports and Weather**

Suppose a sports team wins 75% of the games it plays in good weather, but the team wins only 50% of the games it plays in bad weather. The historic weather pattern for September has good weather two-thirds of the time and bad weather the rest of the time. If we read in the paper that the team has won a game on September 12, what is the probability that the weather was bad on that day?

Let W and L be the events win and lose, respectively. Let G and B be the events good weather and bad weather, respectively. We are given the conditional probabilities $P(W \mid G) = 3/4$ and $P(W \mid B) = 1/2$. The probabilities for the weather in September are $P(G) = 2/3$ and $P(B) = 1/3$. The question asks us to find $P(B \mid W)$. The definition gives us the formula

$$P(B|W) = \frac{P(B \cap W)}{P(W)}.$$

How do we find $P(W)$? Since G and B are mutually exclusive, we have

$$W = (G \cap W) \cup (B \cap W).$$

Therefore, we have

$$P(W) = P(G \cap W) + P(B \cap W).$$

So our calculation becomes

$$P(B|W) = \frac{P(B \cap W)}{P(W)} = \frac{P(B \cap W)}{P(G \cap W) + P(B \cap W)}.$$

Although we are not given $P(B \cap W)$ and $P(G \cap W)$, we can calculate the values using the given information together with the equations

$$P(B \cap W) = P(B)P(W|B) \text{ and } P(G \cap W) = P(G)P(W|G).$$

So our calculation becomes

$$\begin{aligned} P(B|W) = \frac{P(B \cap W)}{P(W)} &= \frac{P(B \cap W)}{P(G \cap W) + P(B \cap W)} \\ &= \frac{P(W|B)P(B)}{P(W|G)P(G) + P(W|B)P(B)} \\ &= \frac{(1/2)(1/3)}{(3/4)(2/3) + (1/2)(1/3)} = 1/4. \end{aligned}$$

end example

Bayes' Theorem

The formula we derived in the previous example is an instance of a well-known theorem called *Bayes' theorem.* Suppose a sample space is partitioned into disjoint (i.e., mutually exclusive) events $H_1, \ldots, H_n$ and E is another event such that $P(E) \neq 0$. Then we have the following formula for each $P(H_i \mid E)$.

$$P(H_i|E) = \frac{P(H_i \cap E)}{P(H_1 \cap E) + \cdots + P(H_n \cap E)}$$
$$= \frac{P(H_i)P(E|H_i)}{P(H_1)P(E|H_1) + \cdots + P(H_n)P(E|H_n)}.$$

Each $P(H_i)$ is the *a priori* probability of H_i and each $P(H_i|E)$ is the *a posteriori* probability of H_i given E.

Bayes' theorem is used in many contexts. For example, H_1, ..., H_n might be a set of mutually exclusive hypotheses that are possible causes of some outcome that can be tested by experiment and E could be an experiment that results in the outcome.

example 5.30 Software Errors

Suppose that the input data set for a program is partitioned into two types, one makes up 60% of the data and the other makes up 40%. Suppose further that inputs from the two types cause warning messages 15% of the time and 20% of the time, respectively. If a random warning message is received, what are the chances that it was caused by an input of each type?

To solve the problem we can use Bayes' theorem. Let E_1 and E_2 be the two sets of data and let B be the set of data that causes warning messages. Then we want to find $P(E_1|B)$ and $P(E_2|B)$. Now we are given the following probabilities:

$$P(E_1) = 0.6,\ P(E_2) = 0.4,\ P(B|E_1) = 0.15,\ P(B|E_2) = 0.2$$

So we can calculate $P(E_1 \mid B)$ as follows:

$$P(E_1|B) = \frac{P(E_1)\,P(B|E_1)}{P(E_1)\,P(B|E_1) + P(E_2)\,P(B|E_2)}$$
$$= \frac{(0.6)\,(0.15)}{(0.6)\,(0.15) + (0.4)\,(0.2)} = \frac{0.09}{0.17} \approx 0.53.$$

A similar calculation gives $P(E_2|B) \approx 0.47$.

end example

5.4.3 Independent Events

Informally, two events A and B are independent if they don't influence each other. If A and B don't influence each other and their probabilities are nonzero, we would like to say that $P(A|B) = P(A)$ and $P(B|A) = P(B)$. This condition will follow from the definition of independence. Two events A and B are *independent* if the following equation holds:

$$P(A \cap B) = P(A)\,P(B).$$

It's interesting to note that if A and B are independent events, then so are the three pairs of events A and B', A' and B, and A' and B'. We'll discuss this in the exercises.

The nice thing about independent events is that they simplify the task of assigning probabilities and computing probabilities.

example 5.31 Independence of Events

In the two-coin-flip example, let A be the event that the first coin is heads and let B be the event that the two coins come up different. Then $A = \{HT, HH\}$, $B = \{HT, TH\}$, and $A \cap B = \{HT\}$. If each coin is fair, then A and B are independent because $P(A) = P(B) = 1/2$ and $P(A \cap B) = 1/4$.

Of course many events are not independent. For example, if C is the event that at least one coin is tails, then $C = \{HT, TH, TT\}$. It follows that $A \cap C = \{HT\}$ and $B \cap C = \{HT, TH\}$. If the coins are fair, then it follows that A and C are dependent events and also that B and C are dependent events.

end example

Repeated Independent Trials

Independence is often used to assign probabilities for repeated trials of the same experiment. We'll be content here to discuss repeated trials of an experiment with two outcomes, where the trials are independent. For example, if we flip a coin n times, it's reasonable to assume that each flip is independent of the other flips. To make things a bit more general, we'll assume that a coin comes up either heads with probability p or tails with probability $1 - p$. Here is the question that we want to answer.

What is the probability that the coin comes up heads exactly k times?

To answer this question we need to consider the independence of the flips. For example, if we let A_i be the event that the ith flip comes up heads, then $P(A_i) = p$ and $P(A_i') = 1 - p$. Suppose now that we ask the probability that the first k flips come up heads and the last $n - k$ flips come up tails. Then we are asking about the probability of the event

$$A_1 \cap \cdots \cap A_k \cap A_{k+1}' \cap \cdots \cap A_n'.$$

Since each event in the intersection is independent of the other events, the probability of the intersection is the product of probabilities

$$p^k(1 -_, p)^{n-k}.$$

We get the same answer for each arrangement of k heads and $n - k$ tails. So we'll have an answer to the question if we can find the number of different arrangements of k heads and $n - k$ tails. By (5.27) there are

$$\frac{n!}{k!\,(n-k)!}$$

such arrangements. This is also $C(n, k)$, which we can represent by the binomial coefficient symbol. So if a coin flip is repeated n times, then the probability of k successes is given by the expression

$$\binom{n}{k} p^k(1-p)^{n-k}.$$

This set of probabilities is called the *binomial distribution.* The name fits because by the binomial theorem, the sum of the probabilities as k goes from 0 to n is 1. We should note that although we used coin flipping to introduce the ideas, the binomial distribution applies to any experiment with two outcomes that has repeated trials.

example **5.32 A Good Golfer**

A golfer wins 60% of the tournaments that s/he enters. What is the probability that the golfer will win exactly five of the next seven tournaments? There are two outcomes, win and lose, with probabilities 0.6 and 0.4, respectively. So the probability of exactly five wins in seven tournaments is given by

$$\binom{7}{5} (0.6)^5(0.4)^2 \approx 0.26.$$

end example

5.4.4 Expectation and Average Behavior

Let's get back to talking about averages and expectations. We all know that the average of a bunch of numbers is the sum of the numbers divided by the number of numbers. So what's the big deal? The deal is that we often assign numbers to each outcome in a sample space. For example, in our beginning discussion we assigned a path length to each of the first 15 prime numbers. We added up the 15 path lengths and divided by 15 to get the average. Makes sense, doesn't it? But remember, we assumed that each number was equally likely to occur. This

is not always the case. So we also have to consider the probabilities assigned to the points in the sample space.

Let's look at another example to set the stage for a definition of expectation. Suppose we agree to flip a coin. If the coin comes up heads, we agree to pay 4 dollars; if it comes up tails, we agree to accept 5 dollars. Notice here that we have assigned a number to each of the two possible outcomes of this experiment. What is our expected take from this experiment? It depends on the coin. Suppose the coin is fair. After one flip we are either 4 dollars poorer or 5 dollars richer. Suppose we play the game 10 times. What then? Well, since the coin is fair, it seems likely that we can expect to win five times and lose five times. So we can expect to pay 20 dollars and receive 25 dollars. Thus our expectation from 10 flips is 5 dollars.

Suppose we knew that the coin was biased with $P(\text{head}) = 2/5$ and $P(\text{tail}) = 3/5$. What would our expectation be? Again, we can't say much for just one flip. But for 10 flips we can expect about four heads and six tails. Thus we can expect to pay out 16 dollars and receive 30 dollars, for a net profit of 14 dollars. An equation to represent our reasoning follows:

$$10P(\text{head})(-4) + 10P(\text{tail})(5) = 10(2/5)(-4) + 10(3/5)(5) = 70/5 = 14.$$

Can we learn anything from this equation? Yes, we can. The 14 dollars represents our take over 10 flips. What's the average profit? Just divide by 10 to get \$1.40. This can be expressed by the following equation:

$$P(\text{head})(-4) + P(\text{tail})(5) = (2/5)(-4) + (3/5)(5) = 7/5 = 1.4.$$

So we can compute the average profit per flip without using the number of coin flips. The average profit per flip is \$1.40 no matter how many flips there are. That's what probability gives us. It's called *expectation*, and we'll generalize from this example to define expectation for any sample space having an assignment of numbers to the sample points.

Definition of Expectation

Let S be a sample space, P a probability distribution on S, and $V : S \to \mathbb{R}$ an assignment of numbers to the points of S. Suppose $S = \{x_1, x_2, \ldots, x_n\}$. Then the *expected value* (or *expectation*) of V is defined by the following formula.

$$E(V) = V(x_1)P(x_1) + V(x_2)P(x_2) + \cdots + V(x_n)P(x_n).$$

So when we want the average behavior, we're really asking for the expectation. For example, in our little coin-flip example we have $S = \{\text{head, tail}\}$, $P(\text{head}) = 2/5$, $P(\text{tail}) = 3/5$, $V(\text{head}) = -4$, and $V(\text{tail}) = 5$. So the expectation of V is calculated by $E(V) = (-4)(2/5) + 5(3/5) = 1.4$.

We should note here that in probability theory the function V is called a *random variable.*

Average Performance of an Algorithm

To compute the average performance of an algorithm A, we must do several things: First, we must decide on a sample space to represent the possible inputs of size n. Suppose our sample space is $S = \{I_1, I_2, \ldots, I_k\}$. Second, we must define a probability distribution P on S that represents our idea of how likely it is that the inputs will occur. Third, we must count the number of operations required by A to process each sample point. We'll denote this count by the function $V : S \to \mathbb{N}$. Lastly, we'll let $\text{Avg}_A(n)$ denote the average number of operations to execute A as a function of input size n. Then $\text{Avg}_A(n)$ is just the expectation of V:

$$\text{Avg}_A(n) = E(V) = V(I_1)P(I_1) + V(I_2)P(I_2) + \cdots + V(I_k)P(I_k).$$

To show that an algorithm A is *optimal in the average case* for some problem, we need to specify a particular sample space and probability distribution. Then we need to show that $\text{Avg}_A(n) \leq \text{Avg}_B(n)$ for all $n > 0$ and for all algorithms B that solve the problem. The problem of finding lower bounds for the average case is just as difficult as finding lower bounds for the worst case. So we're often content to just compare known algorithms to find the best of the bunch.

Analysis of Sequential Search

Suppose we have the following algorithm to search for an element X in an array L, indexed from 1 to n. If X is in L, the algorithm returns the index of the rightmost occurrence of X. The index 0 is returned if X is not in L:

```
i := n;
while i ≥ 1 and X ≠ L[i] do
    i := i - 1
od
```

We'll count the average number of comparisons $X \neq L[i]$ performed by the algorithm. First we need a sample space. Suppose we let I_i denote the input case where the rightmost occurrence of X is at the ith position of L. Let I_{n+1} denote the case in which X is not in L. So the sample space is the set

$$\{I_1, I_2, \ldots, I_{n+1}\}.$$

Let $V(I)$ denote the number of comparisons made by the algorithm when the input has the form I. Looking at the algorithm, we obtain

$$V(I_i) = n - i + 1 \quad \text{for } 1 \leq i \leq n,$$
$$V(I_{n+1}) = n.$$

Suppose we let q be the probability that X is in L. Thus $1 - q$ is the probability that X is not in L. Let's also assume that whenever X is in L, its position is

random. This gives us the following probability distribution P over the sample space:

$$P(I_i) = \frac{q}{n} \quad \text{for } 1 \le i \le n,$$
$$P(I_{n+1}) = 1 - q.$$

Therefore, the expected number of comparisons made by the algorithm for this probability distribution is given by the expected value of V:

$$\begin{aligned}\text{Avg}_A(n) = E(V) &= V(I_1)P(I_1) + \cdots + V(I_{n+1})P(I_{n+1}) \\ &= \frac{q}{n}(n + (n-1) + \cdots + 1) + (1-q)n \\ &= q\left(\frac{n+1}{2}\right) + (1-q)n.\end{aligned}$$

Let's observe a few things about the expected number of comparisons. If we know that X is in L, then $q = 1$. So the expectation is $(n + 1)/2$ comparisons. If we know that X is not in L, then $q = 0$, and the expectation is n comparisons. If X is in L and it occurs at the first position, then the algorithm takes n comparisons. So the worst case occurs for the two input cases I_{n+1} and I_1, and we have $W_A(n) = n$.

5.4.5 Finite Markov Chains

We mentioned before that probability comes up whenever we ask about the chance of something happening. To answer such a question requires one to make some kind of assumption. Now we'll look at processes that change state over time where each change of state depends only on the previous state and a given probability distribution about the chances of changing from any state to any other state. Such a process is called a *Markov chain*. The main property is that the next state depends only on the current state and the given probability for changing states.

We'll introduce the ideas of *finite Markov chains*, where the number of states is finite, with a two-state example to keep things manageable. For example, in weather forecasting, the two states might be sunny and cloudy. We must find a probability distribution about the chances of changing states. For example, the probability of a sunny day tomorrow given that today is sunny might be 0.6 and the probability of a sunny day tomorrow given that today is cloudy might be 0.3. From these conditional probabilities we can then forecast the weather. Of course, there can also be several states. For example, the weather example might have the three states rain, sunny, and partly cloudy.

So the process is very general, with almost unlimited applications. It applies to any process where a change of state depends only on the given state of affairs and a given (i.e., assumed) probability of moving from that state to another state. Now let's get down to business and introduce the ideas. The account we

give is based on that given in the elegant little booklet on probability by Bates [1965].

Introduction with a Two-State Example

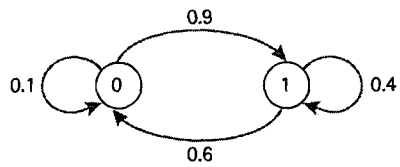
For ease of notation, we'll assume that there are two states, 0 and 1. We'll assume that the probability of moving from 0 to 1 is 0.6, so the probability of moving from 0 to 0 is 0.4. We'll assume that the probability of moving from 1 to 1 is 0.3; so the probability of moving from 1 to 0 is 0.7.

We can represent these four probabilities by the following matrix where the first row gives the probabilities of entering states 0 and 1 when starting from state 0 and the second row gives the probabilities of entering states 0 and 1 when starting from state 1.

$$P = \begin{pmatrix} 0.1 & 0.9 \\ 0.6 & 0.4 \end{pmatrix}.$$

The matrix P is called the *transition matrix* of the chain.

We can also picture the situation with a directed graph where the nodes are the states and each edge is labeled with the probability of traversing the edge to the next state.



For example, suppose the process starts in state 0 and we are interested in the chance of entering state 1 after two stages (or transitions). There are two possible paths of length two from state 0 that end in state 1: the path 0, 1, 1 and the path 0, 0, 1. In other words, we can move to state 1 and then again to state 1 with probability (0.9)(0.4) or we can move to state 0 and then to state 1 with probability (0.1)(0.9). We'll represent these two possible paths by the event {011, 001}. So the desired probability is,

$$P(\{011, 001\}) = (0.9)(0.4) + (0.1)(0.9) = 0.45.$$

For example, if we start in state 0 the chance of entering state 0 after two stages is represented by the event {000, 010}. So the desired probability is,

$$P(\{000, 010\}) = (0.1)(0.1) + (0.9)(0.6) = 0.55.$$

Similarly, to calculate the two-stage probabilities for starting in state 1 we have the two events {100, 110} and {101, 111}. So the desired probabilities are,

$$P(\{100, 110\}) = (0.6)(0.1) + (0.4)(0.6) = 0.30$$
$$P(\{101, 111\}) = (0.6)(0.9) + (0.4)(0.4) = 0.70.$$

We can represent these four probabilities by the following matrix, where the first row gives the probabilities for starting in state 0 and ending in states 0 and 1, and the second row gives the probabilities for starting in state 1 and ending in states 0 and 1:

$$\begin{pmatrix} 0.55 & 0.45 \\ 0.30 & 0.70 \end{pmatrix}.$$

Now, we come to an interesting relationship. The preceding matrix of two-stage probabilities is just the product of the matrix P with itself, which you should verify for yourself:

$$P^2 = PP = \begin{pmatrix} 0.1 & 0.9 \\ 0.6 & 0.4 \end{pmatrix} \begin{pmatrix} 0.1 & 0.9 \\ 0.6 & 0.4 \end{pmatrix} = \begin{pmatrix} 0.55 & 0.45 \\ 0.30 & 0.70 \end{pmatrix}.$$

The nice part is that this kind of relationship works for any number of stages. In other words, if we are interested in the probability of entering some state after n stages, we can find the answer in the matrix P^n.

Let's demonstrate this relationship for three stages of the process. For example, if we start in state 0 the chance of entering state 1 after three stages is represented by the event {0001, 0011, 0101, 0111}. So the desired probability is,

$$P(\{0001, 0011, 0101, 0111\})$$
$$= (0.1)(0.1)(0.9) + (0.1)(0.9)(0.4) + (0.9)(0.6)(0.9) + (0.9)(0.4)(0.4) = 0.675.$$

Now we could calculate the other probabilities in the same way. But instead, we'll calculate the matrix P^3.

$$P^3 = PP^2 = \begin{pmatrix} 0.1 & 0.9 \\ 0.6 & 0.4 \end{pmatrix} \begin{pmatrix} 0.55 & 0.45 \\ 0.30 & 0.70 \end{pmatrix} = \begin{pmatrix} 0.325 & 0.675 \\ 0.450 & 0.550 \end{pmatrix}.$$

Notice, for example, that the entry in the second row first column is 0.675. As expected, it is the probability of entering state 1 after three stages starting from state 0. As an exercise, you should compute the other three probabilities and confirm that they are the other three entries of P^3.

It can be shown that this process extends to any n. In other words, we have the following general property for any finite Markov chain.

Markov Chain Property

Given the transition matrix P, the probabilities after n stages of the process are given by P^n, the nth power P.

The Starting State

At first glance it appears that the process will give a different result depending on whether we start in state 0 or state 1. For example, suppose we flip a coin to choose the starting state. Then the probability of entering state 0 after one stage is

$$(0.5)(0.1) + (0.5)(0.6) = 0.35,$$

and the probability of entering state 1 after one stage is

$$(0.5)(0.9) + (0.5)(0.4) = 0.65.$$

Notice that we can calculate these one-stage results by multiplying the vector (0.5, 0.5) times the matrix P to obtain

$$(0.5,\ 0.5)\,P = (0.5,\ 0.5)\begin{pmatrix} 0.1 & 0.9 \\ 0.6 & 0.4 \end{pmatrix} = (0.35,\ 0.65)\,.$$

In a similar way, we can obtain the two-stage probabilities by multiplying the initial vector (0.5, 0.5) times P^2 to obtain

$$\begin{aligned}(0.5,\ 0.5)\,P^2 &= (0.5,\ 0.5)\begin{pmatrix} 0.1 & 0.9 \\ 0.6 & 0.4 \end{pmatrix}\begin{pmatrix} 0.1 & 0.9 \\ 0.6 & 0.4 \end{pmatrix} \\ &= (0.35,\ 0.65)\begin{pmatrix} 0.1 & 0.9 \\ 0.6 & 0.4 \end{pmatrix} = (0.425,\ 0.575)\,.\end{aligned}$$

Let's observe what happens to these vectors if we continue this process for a few stages. Let X_n be the vector of probabilities after n stages of the process that starts with

$$X_0 = (0.5, 0.5).$$

X_0 is called the *initial probability vector* of the process. We've already calculated X_1 and X_2 as

$$\begin{aligned}X_1 &= X_0P = (0.5, 0.5)P = (0.35, 0.65), \\ X_2 &= X_0P^2 = (X_0P)P = X_1P = (0.35, 0.65)P = (0.425, 0.575).\end{aligned}$$

You can verify that the next three values are

$$X_3 = X_0P^3 = X_2P = (0.3875, 0.6125),$$
$$X_4 = X_0P^4 = X_3P = (0.40625, 0.59375),$$
$$X_5 = X_0P^5 = X_4P = (0.396875, 0.603125).$$

Some Questions

Notice that the values of the vectors appear to be approaching some kind of constant value that looks like it might be (0.4, 0.6). Do we have to go on calculating to see the value? Does the value, if it exists, depend on the initial vector X_0? In other words, in the long run does it make a difference which state we start with?

Some Answers

These questions can be answered if we consider the following question that may seem a bit strange. Is there a probability vector $X = (p, q)$, where $p + q = 1$, such that $XP = X$? To see that the answer is yes, we'll solve the equation for the unknown p. The equation $XP = X$ becomes

$$(p, q)\begin{pmatrix} 0.1 & 0.9 \\ 0.6 & 0.4 \end{pmatrix} = (p, q).$$

This gives us the two equations

$$(0.1)p + (0.6)q = p,$$
$$(0.9)p + (0.4)q = q.$$

Each equation gives the relation

$$p = (2/3)q.$$

Since $p + q = 1$, we can solve for p and q to obtain $p = 0.4$ and $q = 0.6$, which tells us that the probability vector $X = (0.4, 0.6)$ satisfies the equation $XP = X$. So the answer to our strange question is yes.

Now, we noticed that the sequence of probability vectors X_0, X_1, X_2, X_3, X_4, X_5 appears to be approaching the vector (0.4, 0.6), which just happens to be the solution to the equation $XP = X$. Is there some connection? The following theorem tells all.

Markov Chain Theorem

If P is the transition matrix of a finite Markov chain such that some power of P has no zero entries, then the following statements hold:

a. There is a unique probability vector X such that $XP = X$ and X has no zero entries.

Continued ➡

b. As n increases, matrix P^n approaches the matrix that has X in each row.

c. If X_0 is any initial probability vector, then as n increases, the vector $X_n = X_0P^n$ approaches the unique probability vector X.

Now we can answer the questions we asked about the process. Part (a) answers the strange question that asks whether there is a probability vector X such that $XP = X$. Part (c) answers the question that asks about whether the values of X_n appear to be approaching some kind of constant value. The constant value is the X from Part (a). Parts (b) and (c) tell us that X does not depend on the initial vector X_0. In other words, in the long run, it makes no difference how the process starts. So in our example, the probability of entering state 0 in the long run is 0.4 and the probability of entering state 1 is 0.6, no matter how we chose the starting state.

Many-State Examples

We introduced Markov chains with a two-state example. But the ideas extend to any finite number of states. For example, in weather forecasting, we might have three or more conditions such as sunny, partly cloudy, rainy, etc. We might have a golf example, where a golfer on any hole has par, less than par, or greater than par. We could have a baseball example with the three hitting conditions for a player of single, double, and other hit (triple or home run). In business, the applications are endless. Similarly, in computer science, applications to performance of software/hardware abound. In the following example, we'll apply Markov chains to software development.

example 5.33 Software Development

We'll assume that a set of instructions for a program is partitioned into three sequences, which we'll call A, B, and C, where the last instruction in each sequence is a jump/branch instruction. While testing the software, we find that once the execution enters one of these sequences, the jump/branch instructions at the end of the sequences have the following transfer pattern:

A transfers back to A 60% of the time, to B 20% of the time, and to C 20% of the time.

B transfers to A 20% of the time, back to B 50% of the time, and to C 30% of the time.

C transfers to A 40% of the time, B 40% of the time, and back to C 20% of the time.

The transition matrix for this set of probabilities is

$$P = \begin{pmatrix} 0.6 & 0.2 & 0.2 \\ 0.2 & 0.5 & 0.3 \\ 0.4 & 0.4 & 0.2 \end{pmatrix}.$$

Now we can answer some questions, like, which code sequence will be used the most during long executions? Since P has no zero entries, we can find the probability vector X such that $XP = X$. To solve for X, we'll let $X = (x, y, z)$, where $x + y + z = 1$. Then the matrix equation $XP = X$ becomes

$$(x,\ y,\ z)\begin{pmatrix} 0.6 & 0.2 & 0.2 \\ 0.2 & 0.5 & 0.3 \\ 0.4 & 0.4 & 0.2 \end{pmatrix} = (x,\ y,\ z).$$

This gives us the three equations,

$$(0.6)x + (0.2)y + (0.4)z = x,$$
$$(0.2)x + (0.5)y + (0.4)z = y,$$
$$(0.2)x + (0.3)y + (0.2)z = z.$$

Now we can solve these equations subject to the probability restriction,

$$x + y + z = 1.$$

After collecting terms in the first three equations, we obtain the following set of four equations:

$$\begin{aligned} x + \quad y + \quad z &= 1, \\ -0.4x + \ 0.2y + \ 0.4z &= 0, \\ 0.2x + -0.5y + \ 0.4z &= 0, \\ 0.2x + \ 0.3y + -0.8z &= 0. \end{aligned}$$

At this point, let's recall two elementary operations on sets of linear equations that can be performed without changing the solutions:

a. Replace an equation by multiplying it by a nonzero number.

b. Replace an equation by adding it to a nonzero multiple of another equation.

By using these two elementary operations we can transform the set of equations to a simpler form that is easy to solve. The process is called *Gaussian elimination.*

For example, if we like to work with whole numbers, we can use operation (a) and multiply each of the last three equations by 10. This gives us the following set of equations:

$$\begin{aligned} x + \quad y + \quad z &= 1, \\ -4x + \quad 2y + \quad 4z &= 0, \\ 2x + -5y + \quad 4z &= 0, \\ 2x + \quad 3y + -8z &= 0. \end{aligned}$$

We can use operation (b) to eliminate x from the second equation by adding 4 times the first equation to the second equation. Similarly, we can eliminate x from the third and fourth equations by adding -2 times the first equation to each equation. The result is the following set of equations:

$$\begin{aligned} x + \quad y + \quad z &= 1, \\ 6y + \quad 8z &= 4, \\ -7y + \quad 2z &= -2, \\ y + -10z &= -2. \end{aligned}$$

Now we can eliminate y from the second equation by adding -6 times the last equation to the second equation. Similarly, we can eliminate y from the third equation by adding 7 times the last equation to the third equation. The result is the following set of equations:

$$\begin{aligned} x + y + \quad z &= 1, \\ 68z &= 16, \\ -68z &= -16, \\ y + -10z &= -2. \end{aligned}$$

Now we can back substitute to find $z = 4/17$, $y = -2 + 10z = 6/17$, and $x = 1 - y - z = 7/17$. So the probability vector is

$$X = (x, y, z) = (7/17, 6/17, 4/17) \approx (0.41, 0.35, 0.24).$$

We can conclude that no matter which sequence is entered initially, in executions that require many jump/branch instructions, sequence A will be executed about 41% of the time and those in sequences B and C will be executed 35% and 24% of the time, respectively. This information can be used to decide which instructions to place in cache memory. In a dynamic setting, when a jump/branch instruction is executed, the probability vector can be used to predict the next instruction to execute so that it can be pre-fetched and ready to execute.

end example

5.4.6 Approximations (Monte Carlo Method)

Sometimes it is not so easy to find a formula to solve a problem. In some of these cases we can find reasonable approximations by repeating some experiment many times and then observing the results. For example, suppose we have an irregular shape drawn on a piece of paper and we would like to know the area of the shape. The *Monte Carlo method* would have us randomly choose a large number of points on the paper. Then the area of the shape would be pretty close to the percentage of points that lie within the shape multiplied by the area of the paper.

The Monte Carlo method is useful in probability not only to check a calculated answer for a problem, but to find reasonable answers to problems for which we have no other answer. For example, a computer simulating thousands of repetitions of an experiment can give a pretty good approximation to the average outcome of the experiment.

Exercises

Discrete Probability

1. Suppose three fair coins are flipped. Find the probability for each of the following events.
   a. Exactly one coin is a head.
   b. Exactly two coins are tails.
   c. At least one coin is a head.
   d. At most two coins are tails.

2. Suppose a pair of dice are flipped. Find the probability for each of the following events.
   a. The sum of the dots is 7.
   b. The sum of the dots is even.
   c. The sum of the dots is either 7 or 11.
   d. The sum of the dots is at least 5.

3. A team has probability 2/3 of winning whenever it plays. Find each of the following probabilities that the team will win.
   a. Exactly 4 out of 5 games.
   b. At most 4 out of 5 games.
   c. Exactly 4 out of 5 games given that it has already won the first 2 games of a 5-game series.

4. A baseball player's batting average is .250. Find each of the following probabilities that he will get hits.
   a. Exactly 2 hits in 4 times at bat.
   b. At least one hit in 4 times at bat.

5. A computer program uses one of three procedures for each piece of input. The procedures are used with probabilities 1/3, 1/2, and 1/6. Negative results are detected at rates of 10%, 20%, and 30% by the three procedures, respectively. Suppose a negative result is detected. Find the probabilities that each of the procedures was used.

6. A commuter crosses one of three bridges, A, B, or C, to go home from work, crossing A with probability 1/3, B with probability 1/6, and C with probability 1/2. The commuter arrives home by 6 p.m. 75%, 60%, and 80% of the time by crossing bridges A, B, and C, respectively. If the commuter arrives home by 6 p.m., find the probability that bridge A was used. Also find the probabilities for bridges B and C.

7. A student is chosen at random from a class of 80 students that has 20 honor students, 30 athletes, and 40 that are neither honor students nor athletes.
   a. What is the probability that the student selected is an athlete given that he or she is an honor student?
   b. What is the probability that the student selected is an honor student given that he or she is an athlete?
   c. Are the events "honor student" and "athlete" independent?

8. Suppose we have an algorithm that must perform 2,000 operations as follows: The first 1,000 operations are performed by a processor with a capacity of 100,000 operations per second. Then the second 1,000 operations are performed by a processor with a capacity of 200,000 operations per second. Find the average number of operations per second performed by the two processors to execute the 2,000 operations.

9. Consider each of the following lottery problems.
   a. Find the chances of winning a lottery that allows you to pick six numbers from the set $\{1, 2, \ldots, 49\}$.
   b. Suppose that a lottery consists of choosing a set of five numbers from the set $\{1, 2, \ldots, 49\}$. Suppose further that smaller prizes are given to people with four of the five winning numbers. What is the probability of winning a smaller prize?
   c. Suppose that a lottery consists of choosing a set of six numbers from the set $\{1, 2, \ldots, 49\}$. Suppose further that smaller prizes are given to people with four or five of the six winning numbers. What is the probability of winning a smaller prize?
   d. Find a formula for the probability of winning a smaller prize that goes with choosing k of the winning m numbers from the set $\{1, \ldots, n\}$, where $k < m < n$.

10. For each of the following problems, compute the expected value.

a. The expected number of dots that show when a die is tossed.

b. The expected score obtained by guessing all 100 questions of a true-false exam in which a correct answer is worth 1 point and an incorrect answer is worth $-1/2$ point.

11. Use the Monte Carlo method to answer each question about throwing darts.

a. A dart lands inside a square of side length x. What is the probability that the dart landed outside the circle that is inscribed in the square?

b. An irregular shape S is drawn within a square of side length x. A number of darts are thrown at the square with 70 landing inside the square but outside S and 45 landing inside S. What is the approximate area of S?

Markov Chains

12. This exercise will use the transition matrix P from the introductory example.

$$P = \begin{pmatrix} 0.1 & 0.9 \\ 0.6 & 0.4 \end{pmatrix}$$

a. Calculate P^4, P^8, and P^{16}. Notice that it does not require a large value of n for P^n to get close to the matrix

$$\begin{pmatrix} 0.4 & 0.6 \\ 0.4 & 0.6 \end{pmatrix}$$

described in Part (b) of the Markov chain theorem.

b. For the initial probability vector $X_0 = (0.5, 0.5)$ from the introductory example, calculate the probability vectors $X_4 = X_0P^4$, $X_8 = X_0P^8$, and $X_{16} = X_0P^{16}$.

c. Suppose the starting state of the Markov chain is chosen by tossing a fair die, where state 0 is chosen if the top of the die is six, and otherwise state 1 is chosen. In other words, the initial probability vector is $X_0 = (1/6, 5/6)$. Calculate the probability vectors X_4, X_8, and X_{16}, and compare your results with those of Part (b).

13. A company has gathered statistics on three of its products A, B, and C. (You can think of A, B, and C as three breakfast cereals, or as three models of automobile, or as any three products that compete with each other for

market share.) The statistics show that customers switch between products according to the following transition matrix:

$$P = \begin{pmatrix} 0 & 0.5 & 0.5 \\ 0.5 & 0.2 & 0.3 \\ 0.3 & 0 & 0.7 \end{pmatrix}.$$

a. Calculate P^2 and observe that it has no zero entries.

b. Since Part (a) shows that P^2 has no zero entries, the Markov theorem tells us that there is a unique probability vector X such that $XP = X$ and X has no zero entries. Find the probability vector X such that $XP = X$.

c. Calculate P^4 and P^8. Notice that the sequence P, P^2, P^4, P^8 gives good evidence of the fact that P^n approaches the matrix with X in each row.

d. Let $X_0 = (0.1, 0.8, 0.1)$ be the initial probability vector with respect to customers buying the products A, B, and C. Compute $X_1 = X_0P$, $X_2 = X_0P^2$, $X_4 = X_0P^4$, and $X_8 = X_0P^8$.

e. Let $X_0 = (0.3, 0.1, 0.6)$ be the initial probability vector with respect to customers buying the products A, B, and C. Compute $X_1 = X_0P$, $X_2 = X_0P^2$, $X_4 = X_0P^4$, and $X_8 = X_0P^8$. Compare the results with those of Part (d).

f. Suppose that the company manufactures the three products A, B, and C on the same assembly line that can produce 1200 items in a certain period of time. How many items of each type should be manufactured?

Challenges

14. Show that if S is a sample space and A is an event, then S and A are independent events. What about the independence of two events A and B that are disjoint?

15. Prove that if A and B are independent events, then so are the three pairs of events A and B', A' and B, and A' and B'.

16. (*Average-Case Analysis of Binary Search*).

a. Assume that we have a sorted list of 15 elements, x_1, x_2, ..., x_{15}. Calculate the average number of comparisons made by a binary search algorithm to look for a key that may or may not be in the list. Assume that the key has probability 1/2 of being in the list and that each of the events "key = x_i" is equally likely for $1 \leq i \leq 15$.