Ministry of Education and Science of Ukraine
Sumy State University

Dvornichenko A. V.,
Lysenko O. V.

DISCRETE MATHEMATICS AND THEORY OF ALGORITHMS

Lecture notes

In four parts
Part II

Sumy
Sumy State University
2022

Ministry of Education and Science of Ukraine
Sumy State University

DISCRETE MATHEMATICS AND THEORY OF ALGORITHMS

Lecture notes

for students of speciality *113"Applied Mathematics"*
of full-time course of studies

In four parts
Part II

Approved at the meeting of the department of Applied Mathematics and Complex Systems Modeling as a lecture notes on the discipline "Discrete Mathematics and Theory of Algorithms".
Minutes №5 of 29.06.2022.

Sumy
Sumy State University
2022

Discrete Mathematics and Theory of Algorithms : lecture notes : in four parts / compilers: A. V. Dvornichenko, O. V. Lysenko. — Sumy : Sumy State University, 2022. — Part II. — 148 p.

Department of Applied Mathematics and Modeling of Complex Systems

Introduction

A discrete mathematics and theory of algorithms course has more than one purpose. Students should learn a particular set of mathematical facts and how to apply them; more importantly, such a course should teach students how to think logically and mathematically. To achieve these goals, this text stresses mathematical reasoning and the different ways problems are solved. Two important themes are interwoven in this text: inductions and recursions and counting. A successful discrete mathematics and theory of algorithms course should carefully blend and balance two these themes.

In the lecture materials, we used a number of books, among which we will single out the wonderful book by [1]. In the second part of the lecture notes, we consider the following sections: “Inductions and Recursions” and “Counting”.

Contents

Chapter 1

Induction and Recursion

Many mathematical statements assert that a property is true for all positive integers. Examples of such statements are that for every positive integer n : $n! \leq n^n$, $n^3 - n$ is divisible by 3; a set with n elements has 2^n subsets; and the sum of the first n positive integers is $n(n+1)/2$. A major goal of this chapter, and the book, is to give the student a thorough understanding of mathematical induction, which is used to prove results of this kind.

Proofs using mathematical induction have two parts. First, they show that the statement holds for the positive integer 1. Second, they show that if the statement holds for a positive integer then it must also hold for the next larger integer. Mathematical induction is based on the rule of inference that tells us that if $P(1)$ and $\forall k(P(k) \to P(k+1))$ are true for the domain of positive integers, then $\forall n P(n)$ is true. Mathematical induction can be used to prove a tremendous variety of results. Understanding how to read and construct proofs by mathematical induction is a key goal of learning discrete mathematics.

In Chapter 2 we explicitly defined sets and functions. That is, we described sets by listing their elements or by giving some property that characterizes these elements.We gave formulae for the values of functions. There is another important way to define such objects, based on mathematical induction. To define functions, some initial terms are specified, and a rule is given for finding subsequent values from values already known. (We briefly touched on this sort of definition in Chapter

2 when we showed how sequences can be defined using recurrence relations.) Sets can be defined by listing some of their elements and giving rules for constructing elements from those already known to be in the set. Such definitions, called *recursive definitions*, are used throughout discrete mathematics and computer science. Once we have defined a set recursively, we can use a proof method called structural induction to prove results about this set.

When a procedure is specified for solving a problem, this procedure must *always* solve the problem correctly. Just testing to see that the correct result is obtained for a set of input values does not show that the procedure always works correctly. The correctness of a procedure can be guaranteed only by proving that it always yields the correct result. The final section of this chapter contains an introduction to the techniques of program verification. This is a formal technique to verify that procedures are correct. Program verification serves as the basis for attempts under way to prove in a mechanical fashion that programs are correct.

1.1 Mathematical Induction

1.1.1 Introduction

Suppose that we have an infinite ladder, as shown in Figure 1.1, and we want to know whether we can reach every step on this ladder. We know two things:

1. We can reach the first rung of the ladder.
2. If we can reach a particular rung of the ladder, then we can reach the next rung.

Can we conclude that we can reach every runng? By (1), we know that we can reach the first rung of the ladder. Moreover, because we can reach the first rung, by (2), we can also reach the second rung; it is the next rung after the first rung. Applying (2) again, because we can reach the second rung, we can also reach the third rung. Continuing in this way, we can show that can reach the fourth rung, the fifth rung, and so on. For example, after 100 uses of (2), we know that we can reach the 101st rung. But can we conclude that we are able to reach every rung of this infinite ladder? The answer is yes, something we can verify using an important proof technique called **mathematical induction**. That is, we can show that $P(n)$ is true for every positive integer n, where $P(n)$ is the statement that we can reach the nth rung of the ladder. Mathematical induction is an extremely important proof technique that can be used to prove assertions of this type. As we will see in this section and in subsequent sections of this chapter and later chapters, mathematical induction is used extensively to prove results about a large variety of discrete objects. For example, it is used to prove results about the complexity of algorithms, the correctness of certain types of computer programs, theorems about graphs and trees, as well as a wide range of identities and inequalities.

In this section, we will describe how mathematical induction can be used and why it is a valid proof technique. It is extremely important to note that mathematical induction can be used only to prove results obtained in some other way. It is not a tool for discovering formulae or theorems.

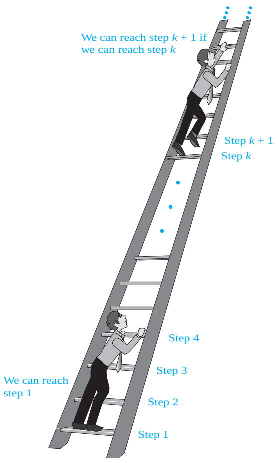


Figure 1.1: Climbing an Infinite Ladder.

1.1.2 Mathematical Induction

In general, mathematical induction [1] can be used to prove statements that assert that $P(n)$ is true for all positive integers n, where $P(n)$ is a propositional function. A proof by mathematical induction has two parts, a basis step, where we show that $P(1)$ is true, and an inductive step, where we show that for all positive integers k, if $P(k)$ is true, then $P(k+1)$ is true.

> *PRINCIPLE OF MATHEMATICAL INDUCTION* To prove that $P(n)$ is true for all positive integers n, where $P(n)$ is a propositional function, we complete two steps:
>
> *BASIS STEP:* We verify that $P(1)$ is true.

[1] Unfortunately, using the terminology "mathematical induction" clashes with the terminology used to describe different types of reasoning. In logic, **deductive reasoning** uses rules of inference to draw conclusions from premises, whereas **inductive reasoning** makes conclusions only supported, but not ensured, by evidence. Mathematical proofs, including arguments that use mathematical induction, are deductive, not inductive.

INDUCTIVE STEP: We show that the conditional statement $P(k) \to P(k+1)$ is true for all positive integers k.

To complete the inductive step of a proof using the principle of mathematical induction, we assume that $P(k)$ is true for an arbitrary positive integer k and show that under this assumption, $P(k+1)$ must also be true. The assumption that $P(k)$ is true is called the inductive hypothesis. Once we complete both steps in a proof by mathematical induction, we have shown that $P(n)$ is true for all positive integers, that is, we have shown that $\forall n P(n)$ is true where the quantification is over the set of positive integers. In the inductive step, we show that $\forall k(P(k) \to P(k+1))$ is true, where again, the domain is the set of positive integers.

Expressed as a rule of inference, this proof technique can be stated as

$$(P(1) \wedge \forall k(P(k) \to P(k+1))) \to \forall n P(n)$$

when the domain is the set of positive integers. Because mathematical induction is such an important technique, it isworthwhile to explain in detail the steps of a proof using this technique. The first thing we do to prove that $P(n)$ is true for all positive integers n is to show that $P(1)$ is true. This amounts to showing that the particular statement obtained when n is replaced by 1 in $P(n)$ is true. Then we must show that $P(k) \to P(k+1)$ is true for every positive integer k. To prove that this conditional statement is true for every positive integer k, we need to show that $P(k+1)$ cannot be false when $P(k)$ is true. This can be accomplished by assuming that $P(k)$ is true and showing that under this hypothesis $P(k+1)$ must also be true.

Remark! In a proof by mathematical induction it is not assumed that $P(k)$ is true for all positive integers! It is only shown that if it is assumed that $P(k)$ is true, then $P(k+1)$ is also true. Thus, a proof by mathematical induction is not a case of begging the question, or circular reasoning.

After completing the basis and inductive steps of a proof that $P(n)$ is true for all positive integers n, we know that $P(1)$ is true. That is

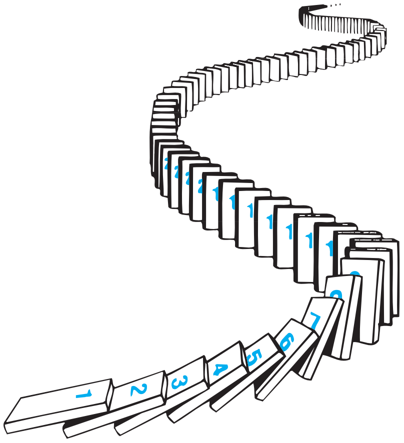


Figure 1.2: Illustrating How Mathematical InductionWorks Using Dominoes.

what is shown in the basis step. We can conclude that $P(2)$ is true, because we know that $P(1)$ is true and from the inductive step we know that $P(1) \rightarrow P(2)$. Furthermore, we know that $P(3)$ is true because $P(2)$ is true and we know that $P(2) \rightarrow P(3)$ from the inductive step. Continuing along these lines using a finite number of implications, we can show that $P(n)$ is true for any particular positive integer n.

WAYS TO REMEMBER HOW MATHEMATICAL INDUCTION WORKS

Thinking of the infinite ladder and the rules for reaching steps can help you remember how mathematical induction works. Note that statements (1) and (2) for the infinite ladder are exactly the basis step and inductive step, respectively, of the proof that $P(n)$ is true for all positive integers n, where $P(n)$ is the statement that we can reach the nth rung of the ladder. Consequently, we can invoke mathematical induction to conclude that we can reach every rung.

Another way to illustrate the principle of mathematical induction is to consider an infinite row of dominoes, labeled $1, 2, 3, \ldots, n, \ldots$, where each domino is standing up. Let P(n) be the proposition that domino n is knocked over. If the first domino is knocked over—i.e., if $P(1)$ is true—and if, whenever the kth domino is knocked over, it also knocks the $(k+1)$st domino over—i.e., if $P(k) \to P(k+1)$ is true for all positive integers k—then all the dominoes are knocked over. This is illustrated in Figure 1.2.

1.1.3 Why Mathematical Induction is Valid

Why is mathematical induction a valid proof technique? The reason comes from the wellordering property, as an axiom for the set of positive integers, which states that every nonempty subset of the set of positive integers has a least element. So, suppose we know that $P(1)$ is true and that the proposition $P(k) \to P(k+1)$ is true for all positive integers k. To show that $P(n)$ must be true for all positive integers n, assume that there is at least one positive integer for which $P(n)$ is false. Then the set S of positive integers for which $P(n)$ is false is nonempty. Thus, by the well-ordering property, S has a least element, which will be denoted by m. We know that m cannot be 1, because $P(1)$ is true. Because m is positive and greater than 1, $m-1$ is a positive integer. Furthermore, because $m-1$ is less than m, it is not in S, so $P(m-1)$ must be true. Because the conditional statement $P(m-1) \to P(m)$ is also true, it must be the case that $P(m)$ is true. This contradicts the choice of m. Hence, $P(n)$ must be true for every positive integer n.

1.1.4 Choosing the Correct Basis Step

Mathematical induction can be used to prove theorems other than those of the form “$P(n)$ is true for all positive integers n.” Often, we will need to show that $P(n)$ is true for $n = b,\ b+1,\ b+2, \ldots,$ where b is an integer other than 1. We can use mathematical induction to accomplish this, as long as we change the basis step by replacing $P(1)$ with $P(b)$. In other words, to use mathematical induction to show that $P(n)$ is true for $n = b,\ b+1,\ b+2, \ldots$, where b is an integer other than 1, we show that $P(b)$ is true in the basis step. In the inductive step, we show that the conditional statement $P(k) \to P(k+1)$ is true for $k = b,\ b+1,\ b+2, \ldots$. Note that b can be negative, zero, or positive. Following the domino analogy we used earlier, imagine that we begin by knocking down the bth domino (the basis step), and as each domino falls, it knocks down the next domino (the inductive step).

We will illustrate this notion in Example 3, which states that a summation formula is valid for all nonnegative integers. In this example,we need to prove that $P(n)$ is true for $n = 0,\ 1,\ 2, \ldots$. So, the basis step in Example 3 will show that $P(0)$ is true.

1.1.5 Guidelines for Proofs by Mathematical Induction

Examples 1–14 will illustrate how to use mathematical induction to prove a diverse collection of theorems. Each of these examples includes all the elements needed in a proof by mathematical induction. We will also present an example of an invalid proof by mathematical induction. Before we give these proofs, we will provide some useful guidelines for constructing correct proofs by mathematical induction.

Template for Proofs by Mathematical Induction

1. Express the statement that is to be proved in the form "for all $n \geq b$, $P(n)$" for a fixed integer b. For statements of the form "$P(n)$ for all positive integers n," let $b = 1$, and for statements of the form "$P(n)$ for all nonnegative integers n," let $b = 0$. For some statements of the form $P(n)$, such as inequalities, you may need to determine the appropriate value of b by checking the truth values of $P(n)$ for small values of n, as is done in Example 6.
2. Write out the words "Basis Step." Then show that $P(b)$ is true, taking care that the correct value of b is used. This completes the first part of the proof.
3. Write out the words "Inductive Step" and state, and clearly identify, the inductive hypothesis, in the form "Assume that $P(k)$ is true for an arbitrary fixed integer $k \geq b$."
4. State what needs to be proved under the assumption that the inductive hypothesis is true. That is, write out what $P(k+1)$ says.

5. Prove the statement $P(k+1)$ making use of the assumption $P(k)$. (Generally, this is the most difficult part of a mathematical induction proof. Decide on the most promising proof strategy and look ahead to see how to use the induction hypothesis to build your proof of the inductive step. Also, be sure that your proof is valid for all integers k with $k \geq b$, taking care that the proof works for small values of k, including $k = b$.)
6. Clearly identify the conclusion of the inductive step, such as by saying "This completes the inductive step."
7. After completing the basis step and the inductive step, state the conclusion, namely, "By mathematical induction, $P(n)$ is true for all integers n with $n \geq b$".

1.1.6 The Good and the Bad of Mathematical Induction

An important point needs to be made about mathematical induction before we commence a study of its use. The good thing about mathematical induction is that it can be used to prove a conjecture once it is has been made (and is true). The bad thing about it is that it cannot be used to find new theorems. Mathematicians sometimes find proofs by mathematical induction unsatisfying because they do not provide insights as to why theorems are true. Many theorems can be proved in many ways, including by mathematical induction. Proofs of these theorems by methods other than mathematical induction are often preferred because of the insights they bring. (See Example 8 and the subsequent remark for an example of this.)

1.1.7 Examples of Proofs by Mathematical Induction

Many theorems assert that $P(n)$ is true for all positive integers n, where $P(n)$ is a propositional function. Mathematical induction is a technique for proving theorems of this kind. In other words, mathematical induction can be used to prove statements of the form $\forall n P(n)$, where the domain is the set of positive integers. Mathematical induction can be used to prove an extremely wide variety of theorems, each of which is a statement of this form. (Remember, many mathematical assertions include an implicit universal quantifier. The statement “if n is a positive integer, then $n^3 - n$ is divisible by 3” is an example of this. Making the implicit universal quantifier explicit yields the statement “for every positive integer n, $n^3 - n$ is divisible by 3.”)

We will use a variety of examples to illustrate how theorems are proved using mathematical induction. The theorems we will prove include summation formulae, inequalities, identities for combinations of sets, divisibility results, theorems about algorithms, and some other creative results.

There are many opportunities for errors in induction proofs. We will describe some incorrect proofs by mathematical induction at the end of this section and in the exercises. To avoid making errors in proofs by mathematical induction, try to follow the guidelines for such proofs.

SEEING WHERE THE INDUCTIVE HYPOTHESIS IS USED

To help the reader understand each of the mathematical induction proofs in this section, we will note where the inductive hypothesis is used. We indicate this use in three different ways: by explicit mention in the text, by inserting the acronym IH (for inductive hypothesis) over an equals sign or a sign for an inequality, or by specifying the inductive hypothesis as the reason for a step in a multi-line display.

PROVING SUMMATION FORMULAE

We begin by using mathematical induction to prove several summation formulae. As we will see, mathematical induction is particularly well suited for proving that such formulae are valid. However, summation formulae can be proven in other ways. This is not surprising because there are often different ways to prove a theorem. The major disadvantage of using mathematical induction to prove a summation formula is that you cannot use it to derive this formula. That is, you must already have the formula before you attempt to prove it by mathematical induction.

Examples 1–4 illustrate how to use mathematical induction to prove summation formulae. The first summation formula we will prove by mathematical induction, in Example 1, is a closed formula for the sum of the smallest n positive integers.

EXAMPLE. 1

Show that if n is a positive integer, then

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

Solution: Let $P(n)$ be the proposition that the sum of the first n positive integers, $1 + 2 + \ldots n = \frac{n(n+1)}{2}$, is $n(n+1)/2$.bWe must do two things to prove that $P(n)$ is true for $n = 1,\ 2,\ 3, \ldots$. Namely, we must show that $P(1)$ is true and that the conditional statement $P(k)$ implies $P(k+1)$ is true for $k = 1,\ 2,\ 3, \ldots$.

BASIS STEP: $P(1)$ is true, because $1 = \frac{1(1+1)}{2}$. (The left-hand side of this equation is 1 because 1 is the sum of the first positive integer. The right-hand side is found by substituting 1 for n in $n(n+1)/2$.)

INDUCTIVE STEP: For the inductive hypothesis we assume that $P(k)$ holds for an arbitrary positive integer k. That is, we assume that

$$1 + 2 + \ldots + k = \frac{k(k+1)}{2}.$$

Under this assumption, it must be shown that $P(k+1)$ is true, namely, that

$$1 + 2 + \ldots + k + (k+1) = \frac{(k+1)[(k+1)+1]}{2} = \frac{(k+1)(k+2)}{2}$$

is also true.

We now look ahead to see how we might be able to prove that $P(k+1)$ holds under the assumption that $P(k)$ is true. We observe that the summation in the left-hand side of $P(k+1)$ is $k+1$ more than the summation in the left-hand side of $P(k)$. Our strategy will be to add $k+1$ to both sides of the equation in $P(k)$ and simplify the result algebraically to complete the inductive step.

We now return to the proof of the inductive step. When we add $k+1$ to both sides of the equation in $P(k)$, we obtain

$$\begin{aligned}1+2+\ldots+k+(k+1) &= \frac{k(k+1)}{2}+(k+1)\\ &= \frac{k(k+1)+2(k+1)}{2}\\ &= \frac{(k+1)(k+2)}{2}.\end{aligned}$$

This last equation shows that $P(k+1)$ is true under the assumption that $P(k)$ is true. This completes the inductive step.

We have completed the basis step and the inductive step, so by mathematical induction we know that $P(n)$ is true for all positive integers n. That is, we have proven that $1+2+\ldots+n = n(n+1)/2$ for all positive integers n. ∎

As we noted, mathematical induction is not a tool for finding theorems about all positive integers. Rather, it is a proof method for proving such results once they are conjectured. In Example 2, using mathematical induction to prove a summation formula, we will both formulate and then prove a conjecture.

EXAMPLE. 2

Conjecture a formula for the sum of the first n positive odd integers. Then prove your conjecture using mathematical induction.

Solution: The sums of the first n positive odd integers for $n = 1,\ 2,\ 3,\ 4,\ 5$ are

$$\begin{array}{lll} 1=1 & 1+3=4 & 1+3+5=9\\ 1+3+5+7=16 & 1+3+5+7+9=25 & \end{array}$$

From these values it is reasonable to conjecture that the sum of the first n positive odd integers is n^2, that is, $1+3+5+\ldots+(2n-1)=n^2$. We need a method to prove that this conjecture is correct, if in fact it is.

Let $P(n)$ denote the proposition that the sum of the first n odd positive integers is n^2. Our conjecture is that $P(n)$ is true for all positive integers n. To use mathematical induction to prove this conjecture, we must first complete the basis step; that is, we must show that $P(1)$ is true. Then we must carry out the inductive step; that is, we must show that $P(k+1)$ is true when $P(k)$ is assumed to be true. We now attempt to complete these two steps.

BASIS STEP: $P(1)$ states that the sum of the first one odd positive integer is 12. This is true because the sum of the first odd positive integer is 1. The basis step is complete.

INDUCTIVE STEP: To complete the inductive step we must show that the proposition $P(k) \to P(k+1)$ is true for every positive integer k. To do this, we first assume the inductive hypothesis. The inductive hypothesis is the statement that $P(k)$ is true for an arbitrary positive integer k, that is,

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

(Note that the kth odd positive integer is $(2k-1)$, because this integer is obtained by adding 2 a total of $k-1$ times to 1.)

To show that $\forall k(P(k) \to P(k+1))$ is true, we must show that if $P(k)$ is true (the inductive hypothesis), then $P(k+1)$ is true. Note that $P(k+1)$ is the statement that

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

Before we complete the inductive step, we will take a time out to figure out a strategy. At this stage of a mathematical induction proof it is time to look for a way to use the inductive hypothesis to show that $P(k+1)$ is true. Here we note that $1+3+5+\ldots+(2k-1)+(2k+1)$ is the sum of its first k terms $1+3+5+\ldots+(2k-1)$ and its last term $2k-1$. So, we can use our inductive hypothesis to replace $1+3+5+\ldots+(2k-1)$ by k^2.

We now return to our proof. We find that

$$\begin{aligned} 1 + 3 + 5 + \ldots + (2k-1) + (2k+1) &= [1 + 3 + \ldots + (2k-1)] + (2k+1) \\ &= k^2 + (2k+1) \\ &= k^2 + 2k + 1 \\ &= (k+1)^2. \end{aligned}$$

This shows that $P(k+1)$ follows from $P(k)$. Note that we used the inductive hypothesis $P(k)$ in the second equality to replace the sum of the first k odd positive integers by k^2.

We have now completed both the basis step and the inductive step. That is, we have shown that $P(1)$ is true and the conditional statement $P(k) \to P(k+1)$ is true for all positive integers k. Consequently, by the principle of mathematical induction we can conclude that $P(n)$ is true for all positive integers n. That is, we know that $1+3+5+\ldots+(2n-1) = n^2$ for all positive integers n. ∎

EXAMPLE. 3

Use mathematical induction to show that

$$1 + 2 + 2^2 + \ldots + 2^n = 2^{n+1} - 1$$

for all nonnegative integers n.

Solution: Let $P(n)$ be the proposition that $1+2+2^2+\ldots+2^n = 2^{n+1}-1$ for the integer n.

BASIS STEP: $P(0)$ is true because $2^0 = 1 = 2^1 - 1$. This completes the basis step.

INDUCTIVE STEP: For the inductive hypothesis, we assume that $P(k)$ is true for an arbitrary nonnegative integer k. That is, we assume that

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

To carry out the inductive step using this assumption, we must show that when we assume that $P(k)$ is true, then $P(k+1)$ is also true. That is, we must show that

$$1 + 2 + 2^2 + \ldots + 2^k + 2^{k+1} = 2^{(k+1)+1} - 1 = 2^{k+2} - 1$$

assuming the inductive hypothesis $P(k)$. Under the assumption of $P(k)$, we see that

$$\begin{aligned} 1 + 2 + 2^2 + \ldots + 2^k + 2^{k+1} &= (1 + 2 + 2^2 + \ldots + 2^k) + 2^{k+1} \\ &= (2^{k+1} - 1) + 2^{k+1} \\ &= 2 \cdot 2^{k+1} - 1 \\ &= 2^{k+2} - 1. \end{aligned}$$

Note that we used the inductive hypothesis in the second equation in this string of equalities to replace $1 + 2 + 2^2 + \ldots + 2^k$ by $2^{k+1} - 1$. We have completed the inductive step.

Becausewe have completed the basis step and the inductive step, by mathematical induction we know that $P(n)$ is true for all nonnegative integers n. That is, $1 + 2 + \ldots + 2^n = 2^{n+1} - 1$ for all nonnegative integers n. ∎

The formula given in Example 3 is a special case of a general result for the sum of terms of a geometric progression. We will use mathematical induction to provide an alternative proof of this formula.

EXAMPLE. 4

Sums of Geometric Progressions

Use mathematical induction to prove this formula for the sum of a finite number of terms of a geometric progression with initial term a and common ratio r:

$$\sum_{j=0}^{n} ar^j = a + ar + ar^2 + \ldots + ar^n = \frac{ar^{n+1} - a}{r-1} \qquad \text{when } r \neq 1,$$

where n is a nonnegative integer.

$\boxed{\textit{Solution:}}$ To prove this formula using mathematical induction, let $P(n)$ be the statement that the sum of the first $n+1$ terms of a geometric progression in this formula is correct.

BASIS STEP: $P(0)$ is true, because

$$\frac{ar^{0+1} - a}{r-1} = \frac{ar - a}{r-1} = \frac{a(r-1)}{r-1} = a.$$

INDUCTIVE STEP: The inductive hypothesis is the statement that $P(k)$ is true, where k is an arbitrary nonnegative integer. That is, $P(k)$ is the statement that

$$a + ar + ar^2 + \ldots + ar^k = \frac{ar^{k+1} - a}{r-1}.$$

To complete the inductive step we must show that if $P(k)$ is true, then $P(k+1)$ is also true. To show that this is the case, we first add ar^{k+1} to both sides of the equality asserted by $P(k)$. We find that

$$a + ar + ar^2 + \ldots + ar^k + ar^{k+1} = \frac{ar^{k+1} - a}{r-1} + ar^{k+1}.$$

Rewriting the right-hand side of this equation shows that

$$\begin{aligned} \tfrac{ar^{k+1}-a}{r-1} + ar^{k+1} &= \tfrac{ar^{k+1}-a}{r-1} + \tfrac{ar^{k+2}-ar^{k+1}}{r-1} \\ &= \tfrac{ar^{k+2}-a}{r-1}. \end{aligned}$$

Combining these last two equations gives

$$a + ar + ar^2 + \ldots + ar^k + ar^{k+1} = \frac{ar^{k+2} - a}{r-1}.$$

This shows that if the inductive hypothesis $P(k)$ is true, then $P(k+1)$ must also be true. This completes the inductive argument. We have completed the basis step and the inductive step, so by mathematical induction $P(n)$ is true for all nonnegative integers n. This shows that the formula for the sum of the terms of a geometric series is correct. ∎

As previously mentioned, the formula in Example 3 is the case of the formula in Example 4 with $a = 1$ and $r = 2$. The reader should verify that putting these values for a and r into the general formula gives the same formula as in Example 3.

PROVING INEQUALITIES

Mathematical induction can be used to prove a variety of inequalities that hold for all positive integers greater than a particular positive integer, as Examples 5–7 illustrate.

EXAMPLE. 5

Use mathematical induction to prove the inequality

$$n < 2^n$$

for all positive integers n.

Solution: Let $P(n)$ be the proposition that $n < 2^n$.

BASIS STEP: $P(1)$ is true, because $1 < 2^1 = 2$. This completes the basis step.

INDUCTIVE STEP: We first assume the inductive hypothesis that $P(k)$ is true for an arbitrary positive integer k. That is, the inductive hypothesis $P(k)$ is the statement that $k < 2^k$. To complete the inductive step, we need to show that if $P(k)$ is true, then $P(k+1)$, which is the statement that $k+1 < 2k+1$ is true. That is, we need to show that if $k < 2^k$, then $k+1 < 2^{k+1}$. To show that this conditional statement is true for the positive integer k, we first add 1 to both sides of $k < 2^k$, and then note that $1 \leq 2^k$. This tells us that

$$k+1 < 2^k + 1 \leq 2^k + 2^k = 2 \cdot 2^k = 2^{k+1}.$$

This shows that $P(k+1)$ is true, namely, that $k+1 < 2^{k+1}$, based on the assumption that $P(k)$ is true. The induction step is complete.

Therefore, because we have completed both the basis step and the inductive step, by the principle of mathematical induction we have shown that $n < 2^n$ is true for all positive integers n. ■

EXAMPLE. 6

Use mathematical induction to prove that $2^n < n!$ for every integer n with $n \geq 4$. (Note that this inequality is false for $n = 1,\ 2$, and 3.)

Solution: Let $P(n)$ be the proposition that $2^n < n!$.

BASIS STEP: To prove the inequality for $n \geq 4$ requires that the basis step be $P(4)$. Note that $P(4)$ is true, because $2^4 = 16 < 24 = 4!$

INDUCTIVE STEP: For the inductive step, we assume that $P(k)$ is true for an arbitrary integer k with $k \geq 4$. That is, we assume that $2^k < k!$ for the positive integer k with $k \geq 4$. We must show that under this hypothesis, $P(k+1)$ is also true. That is, we must show that if $2^k < k!$ for an arbitrary positive integer k where $k \geq 4$, then $2k+1 < (k+1)!$. We have

$$\begin{array}{rcll} 2^{k+1} & = & 2 \cdot 2^k & \text{by definition of exponent} \\ & < & 2 \cdot k! & \text{by the inductive hypothesis} \\ & < & (k+1)k! & \text{because } 2 < k+1 \\ & = & (k+1)! & \text{by definition of factorial function.} \end{array}$$

This shows that $P(k+1)$ is true when $P(k)$ is true. This completes the inductive step of the proof.

We have completed the basis step and the inductive step. Hence, by mathematical induction $P(n)$ is true for all integers n with $n \geq 4$. That is, we have proved that $2^n < n!$ is true for all integers n with $n \geq 4$.

An important inequality for the sum of the reciprocals of a set of positive integers will be proved in Example 7.

EXAMPLE. 7

An Inequality for Harmonic Numbers. The harmonic numbers $H_j,\ j = 1,\ 2,\ 3, \ldots$, are defined by

$$H_j = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{j}.$$

For instance,

$$H_4 = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} = \frac{25}{12}.$$

Use mathematical induction to show that

$$H_{2^n} \geq 1 + \frac{n}{2},$$

whenever n is a nonnegative integer.

$\boxed{\textit{Solution:}}$To carry out the proof, let $P(n)$ be the proposition that $H_{2^n} \geq 1 + \frac{n}{2}$.

BASIS STEP: $P(0)$ is true, because $H_{2^0} = H_1 = 1 \geq 1 + \frac{0}{2}$.

INDUCTIVE STEP: The inductive hypothesis is the statement that $P(k)$ is true, that is, $H_{2^k} \geq 1 + \frac{k}{2}$, where k is an arbitrary nonnegative integer. We must show that if $P(k)$ is true, then $P(k+1)$, which states that $H_{2^{k+1}} \geq 1 + \frac{k+1}{2}$, is also true. So, assuming the inductive hypothesis, it follows that

$$\begin{array}{rcll}
H_{2^{k+1}} & = & 1 + \frac{1}{2} + \frac{1}{3} + \ldots + & \\
& + & \frac{1}{2^k} + \frac{1}{2^k+1} + \ldots + \frac{1}{2^{k+1}} & \text{by the definition of harmonic number} \\
& = & H_{2^k} + \frac{1}{2^k+1} + \ldots + \frac{1}{2^{k+1}} & \text{by the definition of } 2^k th \text{ harmonic number} \\
& \geq & (1 + \frac{k}{2}) + \frac{1}{2^k+1} + \ldots + \frac{1}{2^{k+1}} & \text{by the inductive hypothesis} \\
& \geq & (1 + \frac{k}{2}) + 2^k \cdot \frac{1}{2^{k+1}} & \text{because there are } 2^k \text{ terms each } \geq 1/2^{k+1} \\
& \geq & (1 + \frac{k}{2}) + \frac{1}{2} & \text{canceling a common factor of } 2^k \text{ in second term} \\
& = & 1 + \frac{k+1}{2} &
\end{array}$$

This establishes the inductive step of the proof.

We have completed the basis step and the inductive step. Thus, by mathematical induction $P(n)$ is true for all nonnegative integers n. That is, the inequality $H_{2^n} \geq 1 + \frac{n}{2}$ for the harmonic numbers holds for all nonnegative integers n. ∎

Remark! The inequality established here shows that the **harmonic series**

$$1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n} + \ldots$$

is a divergent infinite series. This is an important example in the study of infinite series.

PROVING DIVISIBILITY RESULTS

Mathematical induction can be used to prove divisibility results about integers. Although such results are often easier to prove using basic results in number theory, it is instructive to see how to prove such results using mathematical induction, as Examples 8 and 9 illustrate.

EXAMPLE. 8

Use mathematical induction to prove that $n^3 - n$ is divisible by 3 whenever n is a positive integer.

Solution: To construct the proof, let $P(n)$ denote the proposition: "$n^3 - n$ is divisible by 3."

BASIS STEP: The statement $P(1)$ is true because $1^3 - 1 = 0$ is divisible by 3. This completes the basis step.

INDUCTIVE STEP: For the inductive hypothesis we assume that $P(k)$ is true; that is, we assume that $k^3 - k$ is divisible by 3 for an arbitrary positive integer k. To complete the inductive step, we must show that when we assume the inductive hypothesis, it follows that $P(k+1)$, the statement that $(k+1)^3 - (k+1)$ is divisible by 3, is also true. That is, we must show that $(k+1)^3 - (k+1)$ is divisible by 3. Note that

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

Using the inductive hypothesis, we conclude that the first term $k^3 - k$ is divisible by 3. The second term is divisible by 3 because it is 3 times an integer. We know that $(k+1)^3 - (k+1)$ is also divisible by 3. This completes the inductive step.

Because we have completed both the basis step and the inductive step, by the principle of mathematical induction we know that $n^3 - n$ is divisible by 3 whenever n is a positive integer. ∎

Remark! We have included Example 8 as an illustration how a divisibility result can be proved by mathematical induction. However, there are simpler proofs. For example, we can prove that $n^3 - n$ is divisible by 3 for all positive integers n using the factorization $n^3 - n = n(n^2 - 1) = n(n-1)(n+1) = (n-1)n(n+1)$. Hence, $n^3 - 1$ is divisible by 3 because it is the product of three consecutive integers, one of which is divisible by 3.

Example 9 presents a more challenging proof by mathematical induction of a divisibility result.

EXAMPLE. 9

Use mathematical induction to prove that $7^{n+2} + 8^{2n+1}$ is divisible by 57 for every nonnegative integer n.

$\boxed{\textit{Solution:}}$To construct the proof, let $P(n)4$ denote the proposition: "$7^{n+2}+ 8^{2n+1}$ is divisible by 57"

BASIS STEP: To complete the basis step, we must show that $P(0)$ is true, because we want to prove that $P(n)$ is true for every nonnegative integer n. We see that $P(0)$ is true because $7^{0+2} + 8^{2\cdot 0+1} = 7^2 + 8^1 = 57$ is divisible by 57. This completes the basis step.

INDUCTIVE STEP: For the inductive hypothesis we assume that $P(k)$ is true for an arbitrary nonnegative integer k; that is, we assume that $7^{k+2} + 8^{2k+1}$ is divisible by 57. To complete the inductive step, we must show that when we assume that the inductive hypothesis $P(k)$ is true, then $P(k + 1)$, the statement that $7^{(k+1)+2} + 8^{2(k+1)+1}$ is divisible by 57, is also true.

The difficult part of the proof is to see how to use the inductive hypothesis. To take advantage of the inductive hypothesis, we use these steps:

$$\begin{aligned} 7^{(k+1)+2} + 8^{2(k+1)+1} &= 7^{k+3} + 8^{2k+3} \\ &= 7 \cdot 7^{k+2} + 8^2 \cdot 8^{2k+1} \\ &= 7 \cdot 7^{k+2} + 64 \cdot 8^{2k+1} \\ &= 7(7^{k+2} + 8^{2k+1}) + 57 \cdot 8^{2k+1}. \end{aligned}$$

We can now use the inductive hypothesis, which states that $7^{k+2} + 8^{2k+1}$ is divisible by 57. We conclude that the first termin this last sum, $7(7^{k+2} + 8^{2k+1})$, is divisible by 57. The second term in this sum, $57 \cdot 8^{2k+1}$, is divisible by 57. Hence, we conclude that $7(7^{k+2} + 8^{2k+1}) + 57 \cdot 8^{2k+1} = 7^{k+3} + 8^{2k+3}$ is divisible by 57. This completes the inductive step.

Because we have completed both the basis step and the inductive step, by the principle of mathematical induction we know that $7^{n+2}+8^{2n+1}$ is divisible by 57 for every nonnegative integer n. ∎

PROVING RESULTS ABOUT SETS

Mathematical induction can be used to prove many results about sets. In particular, in Example 10 we prove a formula for the number of subsets of a finite set and in Example 11 we establish a set identity.

EXAMPLE. 10

The Number of Subsets of a Finite Set Use mathematical induction to show that if S is a finite set with n elements, where n is a nonnegative integer, then S has 2^n subsets.

Solution: Let $P(n)$ be the proposition that a set with n elements has 2^n subsets.

BASIS STEP: $P(0)$ is true, because a set with zero elements, the empty set, has exactly $2^0 = 1$ subset, namely, itself.

INDUCTIVE STEP: For the inductive hypothesis we assume that $P(k)$ is true for an arbitrary nonnegative integer k, that is, we assume that every set with k elements has 2^k subsets. It must be shown that under this assumption, $P(k+1)$, which is the statement that every set with $k+1$ elements has $2k+1$ subsets, must also be true. To show this, let T be a set with $k+1$ elements. Then, it is possible to write $T = S \cup \{a\}$, where a is one of the elements of T and $S = T - \{a\}$ (and hence $|S| = k$). The subsets of T can be obtained in the following way. For each subset X of S there are exactly two subsets of T, namely, X and $X \cup \{a\}$. (This is illustrated in Figure 1.3.) These constitute all the subsets of T and are all distinct. We now use the inductive hypothesis to conclude that S has 2^k subsets, because it has k elements. We also know that there are two subsets of T for each subset of S. Therefore, there are $2 \cdot 2^k = 2^{k+1}$ subsets of T. This finishes the inductive argument.

Becausewe have completed the basis step and the inductive step, by mathematical induction it follows that $P(n)$ is true for all nonnegative integers n. That is, we have proved that a set with n elements has 2^n subsets whenever n is a nonnegative integer. ∎

EXAMPLE. 11

Use mathematical induction to prove the following generalization of one of De Morgan's laws:

$$\overline{\bigcap_{j=1}^{n} A_j} = \bigcup_{j=1}^{n} \bar{A}_j$$

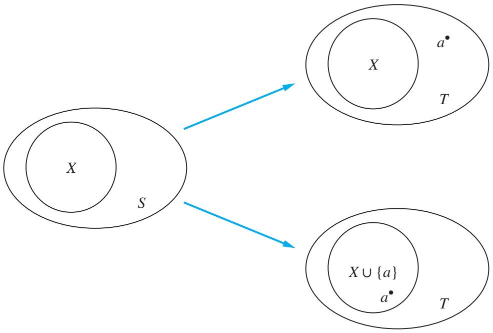


Figure 1.3: Generating subsets of a set with $k+1$ elements. Here $T = S \cup \{a\}$.

whenever $A_1, A_2, \ldots, A_n$ are subsets of a universal set U and $n \geq 2$.

$\boxed{\textit{Solution:}}$ Let $P(n)$ be the identity for n sets.

BASIS STEP: The statement $P(2)$ asserts that $\overline{A_1 \cap A_2} = \bar{A}_1 \cup \bar{A}_2$. This is one of De Morgan's laws.

INDUCTIVE STEP: The inductive hypothesis is the statement that $P(k)$ is true, where k is an arbitrary integer with $k \geq 2$; that is, it is the statement that

$$\overline{\bigcap_{j=1}^{k} A_j} = \bigcup_{j=1}^{k} \bar{A}_j$$

whenever $A_1, A_2, \ldots, A_k$ are subsets of the universal set U. To carry out the inductive step, we need to show that this assumption implies that $P(k+1)$ is true. That is, we need to show that if this equality holds for every collection of k subsets of U, then it must also hold for every collection of $k+1$ subsets of U. Suppose that $A_1, A_2, \ldots, A_k, A_{k+1}$ are subsets of U. When the inductive hypothesis is assumed to hold, it follows that

$$\begin{aligned}
\overline{\bigcap_{j=1}^{k+1} A_j} &= \overline{\left(\bigcap_{j=1}^{k} A_j\right) \cap A_{k+1}} && \text{by the definition of intersection} \\
&= \overline{\left(\bigcap_{j=1}^{k} A_j\right)} \cup \overline{A_{k+1}} && \text{by De Morgan's law (where the} \\
& && \text{two sets are } \cap_{j=1}^{k} A_j \text{ and } A_{k+1}) \\
&= \left(\bigcap_{j=1}^{k} \overline{A_j}\right) \cup \overline{A_{k+1}} && \text{by the inductive hypothesis} \\
&= \bigcup_{j=1}^{k+1} \overline{A_j} && \text{by the definition of union.}
\end{aligned}$$

This completes the inductive step.

Because we have completed both the basis step and the inductive step, by mathematical induction we know that $P(n)$ is true whenever n is a positive integer, $n \geq 2$. That is, we know that

$$\overline{\bigcap_{j=1}^{n} A_j} = \bigcup_{j=1}^{n} \overline{A_j}$$

whenever $A_1, A_2, \ldots, A_n$ are subsets of a universal set U and $n \geq 2$. ∎

PROVING RESULTS ABOUT ALGORITHMS Next, we provide an example (somewhat more difficult than previous examples) that illustrates one of many ways mathematical induction is used in the study of algorithms. We will show how mathematical induction can be used to prove that a greedy algorithm we introduced in Section 3.1 always yields an optimal solution.

EXAMPLE. 12

Recall the algorithm for scheduling talks discussed in Example 7 of Section 3.1. The input to this algorithm is a group of m proposed talks with preset starting and ending times. The goal is to schedule as many of these lectures as possible in the main lecture hall so that no two talks overlap. Suppose that talk t_j begins at time s_j and ends at time e_j.

Without loss of generality, we assume that the talks are listed in order of nondecreasing ending time, so that $e_1 \leq e_2 \leq \ldots \leq e_m$. The greedy algorithm proceeds by selecting at each stage a talk with the earliest ending time among all those talks that begin no sooner than when the last talk scheduled in the main lecture hall has ended. Note that a talk with the earliest end time is always selected first by the algorithm. We will show that this greedy algorithm is optimal in the sense that it always schedules the most talks possible in the main lecture hall. To prove the optimality of this algorithm we use mathematical induction on the variable n, the number of talks scheduled by the algorithm. We let $P(n)$ be the proposition that if the greedy algorithm schedules n talks in the main lecture hall, then it is not possible to schedule more than n talks in this hall.

BASIS STEP: Suppose that the greedy algorithm managed to schedule just one talk, t_1, in the main lecture hall. This means that no other talk can start at or after e_1, the end time of t_1. Otherwise, the first such talk we come to as we go through the talks in order of nondecreasing end times could be added. Hence, at time e_1 each of the remaining talks needs to use the main lecture hall because they all start before e_1 and end after e_1. It follows that no two talks can be scheduled because both need to use the main lecture hall at time e_1. This shows that $P(1)$ is true and completes the basis step.

INDUCTIVE STEP: The inductive hypothesis is that $P(k)$ is true, where k is an arbitrary positive integer, that is, that the greedy algorithm always schedules the most possible talks when it selects k talks, where k is a positive integer, given any set of talks, no matter how many. We must show that $P(k+1)$ follows from the assumption that $P(k)$ is true, that is, we must show that under the assumption of $P(k)$, the greedy algorithm always schedules the most possible talks when it selects $k+1$ talks.

Now suppose that the greedy algorithm has selected $k+1$ talks. Our first step in completing the inductive step is to show there is a schedule including the most talks possible that contains talk t_1, a talk with the earliest end time. This is easy to see because a schedule that begins with the talk t_i in the list, where $i > 1$, can be changed so that talk t_1 replaces talk t_i. To see this, note that because $e_1 \leq e_i$, all talks thatwere scheduled to followtalk t_i can still be scheduled. Once we included talk t_1, scheduling the talks so that as many as possible are scheduled is reduced to scheduling as many talks as possible that begin at or after time e_1. So, if we have scheduled as many talks as possible, the schedule of talks other than talk t_1 is an optimal schedule of the original talks that begin once talk t_1 has ended. Because the greedy algorithm schedules k talks when it creates this schedule, we can apply the inductive hypothesis to conclude that it has scheduled the most possible talks. It follows that the greedy algorithm has scheduled the most possible talks, $k+1$, when it produced a schedule with $k+1$ talks, so $P(k+1)$ is true. This completes the inductive step.

We have completed the basis step and the inductive step. So, by mathematical induction we know that $P(n)$ is true for all positive integers n. This completes the proof of optimality. That is, we have proved that when the greedy algorithm schedules n talks, when n is a positive integer, then it is not possible to schedule more than n talks. ∎

CREATIVE USES OF MATHEMATICAL INDUCTION

Mathematical induction can often be used in unexpected ways. We will illustrate two particularly clever uses of mathematical induction here, the first relating to survivors in a pie fight and the second relating to tilings with regular triominoes of checkerboards with one square missing.

EXAMPLE. 13

Odd Pie Fights An odd number of people stand in a yard at mutually distinct distances. At the same time each person throws a pie at their nearest neighbor, hitting this person. Use mathematical induction to show that there is at least one survivor, that is, at least one person who is not hit by a pie.

Solution: Let $P(n)$ be the statement that there is a survivor whenever $2n+1$ people stand in a yard at distinct mutual distances and each person throws a pie at their nearest neighbor. To prove this result, we will show that $P(n)$ is true for all positive integers n. This follows because as n runs through all positive integers, $2n+1$ runs through all odd integers greater than or equal to 3. Note that one person cannot engage in a pie fight because there is no one else to throw the pie at.

BASIS STEP: When $n = 1$, there are $2n+1 = 3$ people in the pie fight. Of the three people, suppose that the closest pair are A and B, and C is the third person. Because distances between pairs of people are different, the distance between A and C and the distance between B and C are both different from, and greater than, the distance between A and B. It follows that A and B throw pies at each other, while C throws a pie at either A or B, whichever is closer. Hence, C is not hit by a pie. This shows that at least one of the three people is not hit by a pie, completing the basis step.

INDUCTIVE STEP: For the inductive step, assume that $P(k)$ is true for an arbitrary odd integer k with $k \geq 3$. That is, assume that there is at least one survivor whenever $2k+1$ people stand in a yard at distinct mutual distances and each throws a pie at their nearest neighbor. We must show that if the inductive hypothesis $P(k)$ is true, then $P(k+1)$, the statement that there is at least one survivor whenever $2(k+1)+1 = 2k+3$ people stand in a yard at distinct mutual distances and each throws a pie at their nearest neighbor, is also true.

So suppose that we have $2(k+1)+1 = 2k+3$ people in a yard with distinct distances between pairs of people. Let A and B be the closest pair of people in this group of $2k+3$ people. When each person throws a pie at the nearest person, A and B throw pies at each other. We have two cases to consider, *(i)* when someone else throws a pie at either A or B and *(ii)* when no one else throws a pie at either A or B.

Case (i): Because A and B throw pies at each other and someone else throws a pie at either A and B, at least three pies are thrown at A and B, and at most $(2k+3)-3 = 2k$ pies are thrown at the remaining $2k+1$ people. This guarantees that at least one person is a survivor, for if each of these

$2k+1$ people was hit by at least one pie, a total of at least $2k+1$ pies would have to be thrown at them.

Case (ii): No one else throws a pie at either A and B. Besides A and B, there are $2k+1$ people. Because the distances between pairs of these people are all different, we can use the inductive hypothesis to conclude that there is at least one survivor S when these $2k+1$ people each throws a pie at their nearest neighbor. Furthermore, S is also not hit by either the pie thrown by A or the pie thrown by B because A and B throw their pies at each other, so S is a survivor because S is not hit by any of the pies thrown by these $2k+3$ people.

We have completed both the basis step and the inductive step, using a proof by cases. So by mathematical induction it follows that $P(n)$ is true for all positive integers n. We conclude that whenever an odd number of people located in a yard at distinct mutual distances each throws a pie at their nearest neighbor, there is at least one survivor. ∎

Example 14 illustrates how mathematical induction can be used to prove a result about covering checkerboards with right triominoes, pieces shaped like the letter "L."

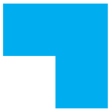


Figure 1.4: A right triomino.

EXAMPLE. 14

Let n be a positive integer. Show that every $2^n \times 2^n$ checkerboard with one square removed can be tiled using right triominoes, where these pieces cover three squares at a time, as shown in Figure 1.4.

$\boxed{\text{Solution:}}$ Let $P(n)$ be the proposition that every $2^n \times 2^n$ checkerboard with one square removed can be tiled using right triominoes. We can use mathematical induction to prove that $P(n)$ is true for all positive integers n.

BASIS STEP: $P(1)$ is true, because each of the four 2×2 checkerboards with one square removed can be tiled using one right triomino, as shown in Figure 1.5.

INDUCTIVE STEP: The inductive hypothesis is the assumption that $P(k)$ is true for the positive integer k; that is, it is the assumption that every $2^k \times 2^k$ checkerboard with one square removed can be tiled using right triominoes. It must be shown that under the assumption of the inductive hypothesis, $P(k+1)$ must also be true; that is, any $2^{k+1} \times 2^{k+1}$ checkerboard with one square removed can be tiled using right triominoes.

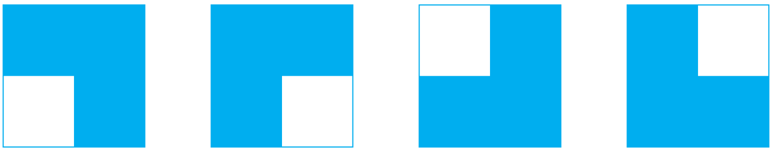


Figure 1.5: Tiling 2×2 checkerboards with one square removed.

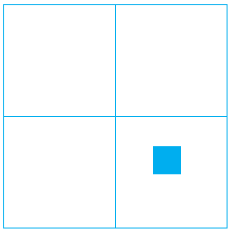


Figure 1.6: Dividing a $2^{k+1} \times 2^{k+1}$ checkerboard into four $2^k \times 2^k$ checkerboards

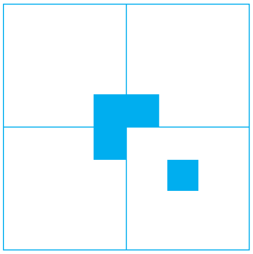


Figure 1.7: Tiling the $2^{k+1} \times 2^{k+1}$ checkerboard with one square removed.

To see this, consider a $2^{k+1} \times 2^{k+1}$ checkerboard with one square removed. Split this checkerboard into four checkerboards of size $2^k \times 2^k$, by dividing it in half in both directions. This is illustrated in Figure 1.6. No square has been removed from three of these four checkerboards. The fourth $2^k \times 2^k$ checkerboard has one square removed, so we now use the inductive hypothesis to conclude that it can be covered by right triominoes. Now temporarily remove the square from each of the other three $2^k \times 2^k$ checkerboards that has the center of the original, larger checkerboard as one of its corners, as shown in Figure 1.7. By the inductive hypothesis, each of these three $2^k \times 2^k$ checkerboards with a square removed can be tiled by right triominoes. Furthermore, the three squares that were temporarily removed can be covered by one right triomino. Hence, the entire $2^{k+1} \times 2^{k+1}$ checkerboard can be tiled with right triominoes.

We have completed the basis step and the inductive step. Therefore, by mathematical induction $P(n)$ is true for all positive integers n. This shows that we can tile every $2^n \times 2^n$ checkerboard, where n is a positive integer, with one square removed, using right triominoes. ∎

1.1.8 Mistaken Proofs By Mathematical Induction

As with every proof method, there are many opportunities for making errors when using mathematical induction. Many well-known mistaken, and often entertaining, proofs by mathematical induction of clearly false statements have been devised, as exemplified by Example 15. Often, it is not easy to find where the error in reasoning occurs in such mistaken proofs.

To uncover errors in proofs by mathematical induction, remember that in every such proof, both the basis step and the inductive step must be done correctly. Not completing the basis step in a supposed proof by mathematical induction can lead to mistaken proofs of clearly ridiculous statements such as "$n = n + 1$ whenever n is a positive integer." Locating the error in a faulty proof by mathematical induction, as Example 15 illustrates, can be quite tricky, especially when the error is hidden in the basis step.

EXAMPLE. 15

Find the error in this "proof" of the clearly false claim that every set of lines in the plane, no two of which are parallel, meet in a common point.

"*Proof:* :" Let $P(n)$ be the statement that every set of n lines in the plane, no two of which are parallel, meet in a common point. We will attempt to prove that $P(n)$ is true for all positive integers $n \geq 2$.

BASIS STEP: The statement $P(2)$ is true because any two lines in the plane that are not parallel meet in a common point (by the definition of parallel lines).

INDUCTIVE STEP: The inductive hypothesis is the statement that $P(k)$ is true for the positive integer k, that is, it is the assumption that every set of k lines in the plane, no two of which are parallel, meet in a common point. To complete the inductive step, we must show that if $P(k)$ is true, then $P(k+1)$ must also be true. That is, we must show that if every set of k lines in the plane, no two of which are parallel, meet in a common point, then every set of $k + 1$ lines in the plane, no two of which are parallel, meet in a common point. So, consider a set of $k + 1$ distinct lines in the plane. By the inductive hypothesis, the first k of these lines meet in a common point p1. Moreover, by the inductive hypothesis, the last k of these lines meet in a common point p_2. We will show that p_1 and p_2 must be the same point. If p_1 and p_2 were different points, all lines containing both of them must be the same line because two points determine a line. This contradicts our assumption that all these lines are distinct. Thus, p_1 and p_2 are the same point. We conclude that the point $p_1 = p_2$ lies on all $k+1$ lines. We have shown that $P(k+1)$ is true assuming that $P(k)$ is true. That is, we have shown that if we assume that every k, $k \geq 2$, distinct lines meet in a common point, then every $k+1$ distinct lines meet in a common point. This completes the inductive step.

We have completed the basis step and the inductive step, and supposedly we have a correct proof by mathematical induction.

Solution: Examining this supposed proof by mathematical induction it appears that everything is in order. However, there is an error, as there must be. The error is rather subtle. Carefully looking at the inductive step shows that this step requires that $k \geq 3$. We cannot show that $P(2)$ implies $P(3)$. When $k = 2$, our goal is to show that every three distinct lines meet in a common point. The first two lines must meet in a common point p_1 and the last two lines must meet in a common point p_2. But in this case, p_1 and p_2 do not have to be the same, because only the second line is common to both sets of lines. Here is where the inductive step fails. ∎

1.2 Recursive Definitions and Structural Induction

1.2.1 Introduction

Sometimes it is difficult to define an object explicitly. However, it may be easy to define this object in terms of itself. This process is called recursion. For instance, the picture shown in Figure 1 is produced recursively. First, an original picture is given. Then a process of successively superimposing centered smaller pictures on top of the previous pictures is carried out.

We can use recursion to define sequences, functions, and sets. The terms of a sequence are specified using an explicit formula. For instance, the sequence of powers of 2 is given by $a_n = 2^n$ for $n = 0,\ 1,\ 2, \ldots$. We can also define a sequence recursively by specifying how terms of the sequence are found from previous terms. The sequence of powers of 2 can also be defined by giving the first term of the sequence, namely, $a_0 = 1$, and a rule for finding a term of the sequence from the previous one, namely, $a_{n+1} = 2a_n$ for $n = 0,\ 1,\ 2, \ldots$. When we define a sequence recursively by specifying how terms of the sequence are found from previous terms, we can use induction to prove results about the sequence.

When we define a set recursively, we specify some initial elements in a basis step and provide a rule for constructing new elements from those we already have in the recursive step. To prove results about recursively defined sets we use a method called *structural induction.*

1.2.2 Recursively Defined Functions

We use two steps to define a function with the set of nonnegative integers as its domain:

BASIS STEP: Specify the value of the function at zero.

RECURSIVE STEP: Give a rule for finding its value at an integer from its values at smaller integers.

Such a definition is called a recursive or inductive definition. Note that a function $f(n)$ from the set of nonnegative integers to the set of a real numbers is the same as a sequence $a_0,\ a_1, \ldots,$ where a_i is a

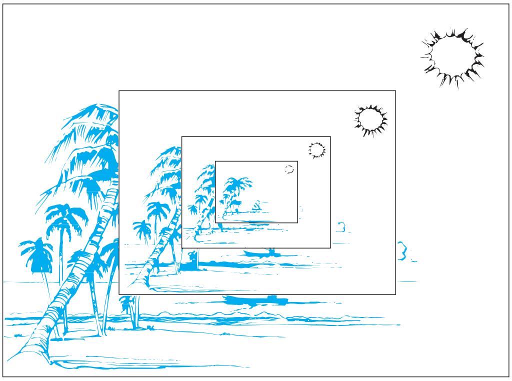


Figure 1.8: A recursively defined picture.

real number for every nonnegative integer i. So, defining a real-valued sequence $a_0,\ a_1, \ldots$ using a recurrence relation.

EXAMPLE. 1

Suppose that f is defined recursively by

$$f(0) = 3,$$
$$f(n+1) = 2f(n) + 3.$$

Find $f(1),\ f(2),\ f(3)$, and $f(4)$.

Solution: From the recursive definition it follows that

$$\begin{aligned} f(1) &= 2f(0) + 3 = 2 \cdot 3 + 3 = 9, \\ f(2) &= 2f(1) + 3 = 2 \cdot 9 + 3 = 21, \\ f(3) &= 2f(3) + 3 = 2 \cdot 45 + 3 = 93. \end{aligned} \qquad \blacksquare$$

Recursively defined functions are **well defined**. That is, for every positive integer, the value of the function at this integer is determined in an unambiguous way. This means that given any positive integer, we can use the two parts of the definition to find the value of the function at that integer, and that we obtain the same value no matter how we apply the two parts of the definition. This is a consequence of the principle of mathematical induction. Additional examples of recursive definitions are given in Examples 2 and 3.

EXAMPLE. 2

Give a recursive definition of a^n, where a is a nonzero real number and n is a nonnegative integer.

Solution: The recursive definition contains two parts. First a^0 is specified, namely, $a^0 = 1$. Then the rule for finding a^{n+1} from a^n, namely, $a^{n+1} = a \cdot a^n$, for $n = 0,\ 1,\ 2,\ 3, \ldots$, is given. These two equations uniquely define an for all nonnegative integers n. ∎

EXAMPLE. 3

Give a recursive definition of

$$\sum_{k=0}^{n} a_k$$

Solution: The first part of the recursive definition is

$$\sum_{k=0}^{0} a_k = a_0$$

The second part is

$$\sum_{k=0}^{n+1} a_k = \left(\sum_{k=0}^{n} a_k\right) + a_{n+1}. \qquad \blacksquare$$

In some recursive definitions of functions, the values of the function at the first k positive integers are specified, and a rule is given for determining the value of the function at larger integers from its values at some or all of the preceding k integers. That recursive definitions defined in this way produce well-defined functions follows from strong induction.

Recall from previous Section that the Fibonacci numbers, $f_0,\ f_1,\ f_2, \ldots$, are defined by the equations $f_0 = 0,\ f_1 = 1$, and

$$f_n = f_{n-1} + f_{n-2}$$

for $n = 2, 3, 4, \ldots$. [We can think of the Fibonacci number fn either as the nth term of the sequence of Fibonacci numbers $f_0,\ f_1, \ldots$ or as the value at the integer n of a function $f(n)$.]

We can use the recursive definition of the Fibonacci numbers to prove many properties of these numbers. We give one such property in Example 4.

EXAMPLE. 4

Show that whenever $n \geq 3,\ f_n > \alpha^{n-2}$, where $\alpha = (1 + \sqrt{5})/2$.

$\boxed{\textit{Solution:}}$ We can use strong induction to prove this inequality. Let $P(n)$ be the statement $f_n > \alpha^{n-2}$. We want to show that $P(n)$ is true whenever n is an integer greater than or equal to 3.

BASIS STEP: First, note that

$$\alpha < 2 = f_3,\ \alpha^2 = (3 + \sqrt{5})/2 < 3 = f_4$$

so $P(3)$ and $P(4)$ are true.

INDUCTIVE STEP: Assume that $P(j)$ is true, namely, that $f_j > \alpha^{j-2}$, for all integers j with $3 \leq j \leq k$, where $k \geq 4$. We must show that $P(k+1)$ is true, that is, that $f_{k+1} > \alpha^{k-1}$. Because α is a solution of $x^2 - x - 1 = 0$ (as the quadratic formula shows), it follows that $\alpha^2 = \alpha + 1$ Therefore,

$$\alpha^{k-1} = \alpha^2 \cdot \alpha^{k-3} = (\alpha + 1)\alpha^{k-3} = \alpha \cdot \alpha^{k-3} + 1 \cdot \alpha^{k-3} = \alpha^{k-2} + \alpha^{k-3}.$$

By the inductive hypothesis, because $k \geq 4$, we have

$$f_{k-1} > \alpha^{k-3},\ f_k > \alpha^{k-2}.$$

Therefore, it follows that

$$f_{k+1} = f_k + f_{k-1} > \alpha^{k-2} + \alpha^{k-3} = \alpha^{k-1}.$$

Hence, $P(k+1)$ is true. This completes the proof. ∎

Remark! The inductive step of the proof by strong induction in Example 4 shows that whenever $k \geq 4$, $P(k+1)$ follows from the assumption that $P(j)$ is true for $3 \leq j \leq k$. Hence, the inductive step does not show that $P(3) \to P(4)$. Therefore, we had to show that $P(4)$ is true separately.

We can now show that the Euclidean algorithm uses $O(\log b)$ divisions to find the greatest common divisor of the positive integers a and b, where $a \geq b$.

THEOREM 1.2.1: LAME'S THEOREM

Let a and b be positive integers with $a \geq b$. Then the number of divisions used by the Euclidean algorithm to find $gcd(a,b)$ is less than or equal to five times the number of decimal digits in b.

Proof: Recall that when the Euclidean algorithm is applied to find $gcd(a,b)$ with $a \geq b$, this sequence of equations (where $a = r_0$ and $b = r_1$) is obtained.

$$\begin{array}{lcll}
r_0 & = & r_1 q_1 + r_2 & 0 \leq r_2 < r_1, \\
r_1 & = & r_2 q_2 + r_2 & 0 \leq r_3 < r_2, \\
 & \cdot & & \\
 & \cdot & & \\
 & \cdot & & \\
r_{n-2} & = & r_{n-1} q_{n-1} + r_n & 0 \leq r_n < r_{n-1}, \\
r_{n-1} & = & r_n q_n & .
\end{array}$$

Here n divisions have been used to find $r_n = gcd(a,b)$. Note that the quotients $q_1, q_2, \ldots, q_{n-1}$ are all at least 1. Moreover, $q_n \geq 2$, because $r_n < r_{n-1}$. This implies that

$$\begin{array}{lcl} r_n & \geq & 1 = f_2, \\ r_{n-1} & \geq & 2r_n \geq 2f_2 = f_3, \\ r_{n-2} & \geq & 2r_{n-1} + r_n \geq f_3 + f_2 = f_4, \\ & \cdot & \\ & \cdot & \\ & \cdot & \\ r_2 & \geq & r_3 + r_4 \geq f_{n-1} + f_{n-2} = f_n, \\ b & = & r_1 \geq r_2 + r_3 \geq f_n + f_{n-1} = f_{n+1}. \end{array}$$

It follows that if n divisions are used by the Euclidean algorithm to find $gcd(a, b)$ with $a \geq b$, then $b \geq f_{n+1}$. By Example 4 we know that $f_{n+1} > \alpha^{n-1}$ for $n > 2$, where $\alpha = (1 + \sqrt{5})/2$. Therefore, it follows that $b > \alpha^{n-1}$. Furthermore, because $\log_{10} \alpha \approx 0.208 > 1/5$, we see that

$$\log_{10} b > (n - 1) \log_{10} \alpha > (n - 1)/5.$$

Hence, $n - 1 < 5 \cdot \log_{10} b$. Now suppose that b has k decimal digits. Then $b < 10^k$ and $\log_{10} b < k$. It follows that $n - 1 < 5k$, and because k is an integer, it follows that $n \leq 5k$. This finishes the proof. ∎

Because the number of decimal digits in b, which equals $\lfloor \log_{10} b \rfloor + 1$, is less than or equal to $\log_{10} b + 1$, Theorem 1.2.1 tells us that the number of divisions required to find $gcd(a, b)$ with $a > b$ is less than or equal to $5(\log_{10} b + 1)$. Because $5(\log_{10} b + 1)$ is $O(\log b)$, we see that $O(\log b)$ divisions are used by the Euclidean algorithm to find $gcd(a, b)$ whenever $a > b$.

1.2.3 Recursively Defined Sets and Structures

We have explored how functions can be defined recursively. We now turn our attention to how sets can be defined recursively. Just as in the recursive definition of functions, recursive definitions of sets have two parts, a **basis step** and a **recursive step**. In the basis step, an initial collection of elements is specified. In the recursive step, rules for forming new elements in the set from those already known to be in the set are provided. Recursive definitions may also include an **exclusion rule**, which specifies that a recursively defined set contains nothing other than those elements specified in the basis step or generated by applications of the recursive step. In our discussions, we will always tacitly assume that the exclusion rule holds and no element belongs to a recursively defined set unless it is in the initial collection specified in the basis step or can be generated using the recursive step one or more times. Later we will see how we can use a technique known as structural induction to prove results about recursively defined sets.

Examples 5, 6, 8, and 9 illustrate the recursive definition of sets. In each example, we show those elements generated by the first few applications of the recursive step.

EXAMPLE. 5

Consider the subset S of the set of integers recursively defined by

BASIS STEP: $3 \in S$.

RECURSIVE STEP: If $x \in S$ and $y \in S$, then $x + y \in S$.

The new elements found to be in S are 3 by the basis step, $3 + 3 = 6$ at the first application of the recursive step, $3 + 6 = 6 + 3 = 9$ and $6 + 6 = 12$ at the second application of the recursive step, and so on. We will show in Example 10 that S is the set of all positive multiples of 3. ∎

Recursive definitions play an important role in the study of strings. Recall from Section 2 that a string over an alphabet $\sum$ is a finite sequence of symbols from $\sum$. We can define $\sum^*$, the set of strings over $\sum$, recursively, as Definition 1 shows.

DEFINITION 1.2.1 *The set $\sum^*$ of **strings** over the alphabet $\sum$ is defined recursively by*

BASIS STEP: $\lambda \in \sum^*$ (where λ is the empty string containing no symbols).

RECURSIVE STEP: If $w \in \sum^*$ and $x \in \sum$, then $wx \in \sum^*$.

The basis step of the recursive definition of strings says that the empty string belongs to $\sum^*$. The recursive step states that new strings are produced by adding a symbol from $\sum$ to the end of strings in $\sum^*$. At each application of the recursive step, strings containing one additional symbol are generated.

EXAMPLE. 6

If $\sum = \{0,1\}$ the strings found to be in $\sum^*$ the set of all bit strings, are λ, specified to be in $\sum^*$ in the basis step, 0 and 1 formed during the first application of the recursive step, 00, 01, 10, and 11 formed during the second application of the recursive step, and so on. ∎

Recursive definitions can be used to define operations or functions on the elements of recursively defined sets. This is illustrated in Definition 2 of the concatenation of two strings and Example 7 concerning the length of a string.

DEFINITION 1.2.2 *Two strings can be combined via the operation of **concatenation**. Let $\sum$ be a set of symbols and $\sum^*$ the set of strings formed from symbols in $\sum$. We can define the concatenation of two strings, denoted by $\cdot$, recursively as follows.*

BASIS STEP: If $w \in \sum^*$, then $w \cdot \lambda = w$, where λ is the empty string.

RECURSIVE STEP: If $w_1 \in \sum^*$ and $w_2 \in \sum^*$ and $x \in \sum$, then $w_1 \cdot (w_2 x) = (w_1 \cdot w_2)x$

The concatenation of the strings w_1 and w_2 is often written as $w_1 w_2$ rather than $w_1 \cdot w_2$. By repeated application of the recursive definition, it follows that the concatenation of two strings w_1 and w_2 consists of the symbols in w_1 followed by the symbols in w_2. For instance, the concatenation of $w_1 = abra$ and $w_2 = cadabra$ is $w_1 w_2 = abracadabra$.

EXAMPLE. 7

Length of a String Give a recursive definition of $l(w)$, the length of the string w.

Solution: The length of a string can be recursively defined by

$$\begin{aligned} l(\lambda) &= 0; \\ l(wx) &= l(w) + 1 \text{if } w \in \textstyle\sum^* \text{ and } x \in \textstyle\sum. \end{aligned}$$

∎

Another important use of recursive definitions is to define **well-formed formulae** of various types. This is illustrated in Examples 8 and 9.

EXAMPLE. 8

Well-Formed Formulae in Propositional Logic We can define the set of well-formed formulae in propositional logic involving **T, F**, propositional variables, and operators from the set $\{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$.

BASIS STEP: **T, F**, and s, where s is a propositional variable, are well-formed formulae.

RECURSIVE STEP: If **E** and **F** arewell-formed formulae, then $(\neg E)$, $(E \wedge F)$, $(E \vee F)$, $(E \rightarrow F)$, and $(E \leftrightarrow F)$ are well-formed formulae.

For example, by the basis step we know that **T, F**, p, and q are well-formed formulae, where p and q are propositional variables. From an initial application of the recursive step, we know that $(p \vee q)$, $(p \rightarrow \mathbf{F})$, $(\mathbf{F} \rightarrow q)$, and $(q \wedge \mathbf{F})$ are well-formed formulae. A second application of the recursive step shows that $((p \vee q) \rightarrow (q \wedge \mathbf{F}))$, $(q \vee (p \vee q))$, and $((p \rightarrow \mathbf{F}) \rightarrow \mathbf{T})$ are well-formed formulae. We leave it to the reader to show that $p\neg\wedge q$, $pq\wedge$, and $\neg\wedge pq$ are not well-formed formulae, by showing that none can be obtained using the basis step and one or more applications of the recursive step. ∎

EXAMPLE. 9

Well-Formed Formulae of Operators and Operands We can define the set of well-formed formulae consisting of variables, numerals, and operators from the set $\{+, ?, *, /, \uparrow\}$ (where $*$ denotes multiplication and $\uparrow$ denotes exponentiation) recursively.

BASIS STEP: x is a well-formed formula if x is a numeral or a variable.

RECURSIVE STEP: If F and G are well-formed formulae, then $(F + G)$, $(F - G)$, $(F * G), (F/G)$, and $(F \uparrow G)$ are well-formed formulae.

For example, by the basis step we see that $x, y, 0$, and 3 are well-formed formulae (as is any variable or numeral). Well-formed formulae generated by applying the recursive step once include $(x + 3)$, $(3 + y)$, $(x - y)$, $(3 - 0)$, $(x * 3)$, $(3 * y)$, $(3/0)$, (x/y), $(3 \uparrow x)$, and $(0 \uparrow 3)$. Applying the recursive step twice shows that formulae such as $((x + 3) + 3)$ and $(x - (3 * y))$ are well-formed formulae. We leave it to the reader to show that each of the formulae

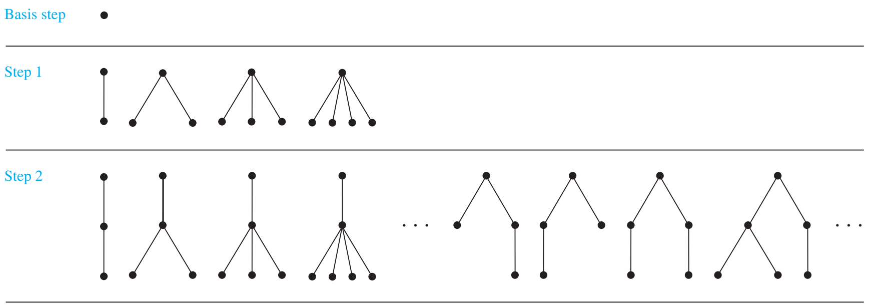


Figure 1.9: Building up rooted trees.

$x3+$, $y * +x$, and $*x/y$ is not a well-formed formula by showing that none of them can be obtained from the basis step and one or more applications of the recursive step. ∎

DEFINITION 1.2.3 *The set of* ***rooted trees****, where a rooted tree consists of a set of vertices containing a distinguished vertex called the* ***root****, and edges connecting these vertices, can be defined recursively by these steps:*

BASIS STEP: A single vertex r is a rooted tree.

RECURSIVE STEP: Suppose that $T_1, T_2, \ldots, T_n$ are disjoint rooted trees with roots $r_1, r_2, \ldots, r_n$, respectively. Then the graph formed by starting with a root r, which is not in any of the rooted trees $T_1, T_2, \ldots, T_n$, and adding an edge from r to each of the vertices $r_1, r_2, \ldots, r_n$, is also a rooted tree.

In Figure 1.9 we illustrate some of the rooted trees formed starting with the basis step and applying the recursive step one time and two times. Note that infinitely many rooted trees are formed at each application of the recursive definition.

Binary trees are a special type of rooted trees. We will provide recursive definitions of two types of binary trees—full binary trees and extended binary trees. In the recursive step of the definition of each type of binary tree, two binary trees are combined to form a new tree with one of these trees designated the left subtree and the other the right

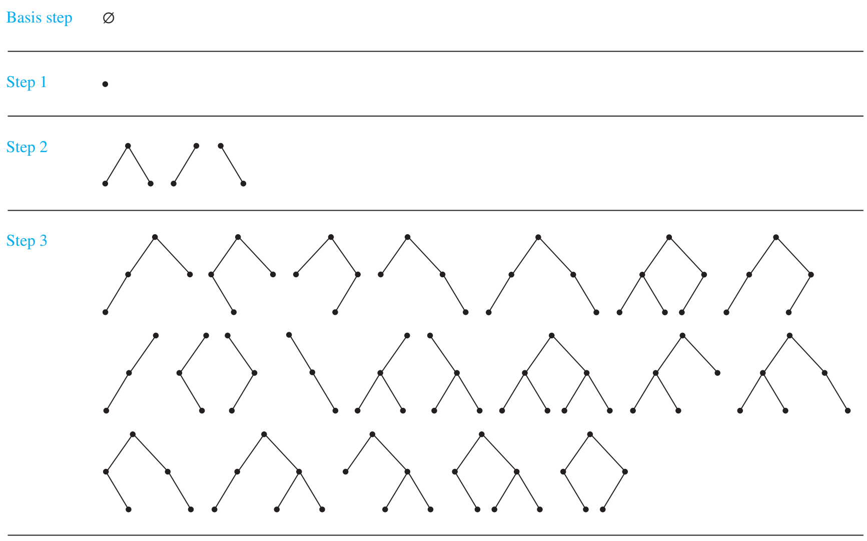


Figure 1.10: Building up extended binary trees.

subtree. In extended binary trees, the left subtree or the right subtree can be empty, but in full binary trees this is not possible. Binary trees are one of the most important types of structures in computer science.

DEFINITION 1.2.4 *The set of* ***extended binary trees*** *can be defined recursively by these steps:*

BASIS STEP: The empty set is an extended binary tree.

RECURSIVE STEP: If T_1 and T_2 are disjoint extended binary trees, there is an extended binary tree, denoted by $T_1 \cdot T_2$, consisting of a root r together with edges connecting the root to each of the roots of the left subtree T_1 and the right subtree T_2 when these trees are nonempty.

Figure 1.10 shows how extended binary trees are built up by applying the recursive step from one to three times. We now show how to define the set of full binary trees. Note that the difference between this recursive definition and that of extended binary trees lies entirely in the basis step.

DEFINITION 1.2.5 *The set of full binary trees can be defined recursively by these steps:*

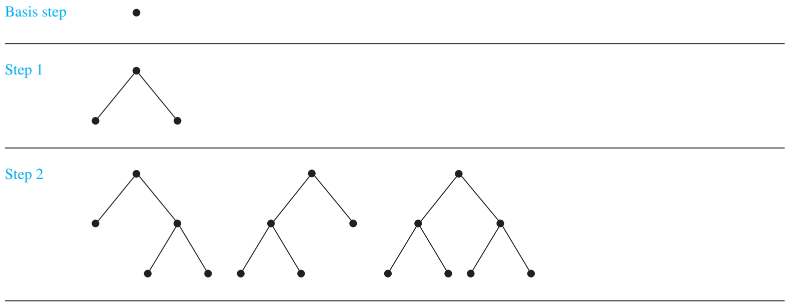


Figure 1.11: Building up extended binary trees.

BASIS STEP: There is a full binary tree consisting only of a single vertex r.

RECURSIVE STEP: If T_1 and T_2 are disjoint full binary trees, there is a full binary tree, denoted by $T_1 \cdot T_2$, consisting of a root r together with edges connecting the root to each of the roots of the left subtree T_1 and the right subtree T_2.

Figure 1.11 shows how full binary trees are built up by applying the recursive step one and two times.

1.2.4 Structural Induction

To prove results about recursively defined sets, we generally use some form of mathematical induction. Example 10 illustrates the connection between recursively defined sets and mathematical induction.

EXAMPLE. 10

Show that the set S defined in Example 5 by specifying that $3 \in S$ and that if $x \in S$ and $y \in S$, then $x + y \in S$, is the set of all positive integers that are multiples of 3.

Solution: Let A be the set of all positive integers divisible by 3. To prove that $A = S$, we must show that A is a subset of S and that S is a subset of A. To prove that A is a subset of S, we must show that every positive integer divisible by 3 is in S. We will use mathematical induction to prove this.

Let $P(n)$ be the statement that $3n$ belongs to S. The basis step holds because by the first part of the recursive definition of S, $3 \cdot 1 = 3$ is in S. To establish the inductive step, assume that $P(k)$ is true, namely, that $3k$ is in S. Because $3k$ is in S and because 3 is in S, it follows from the second part of the recursive definition of S that $3k + 3 = 3(k + 1)$ is also in S.

To prove that S is a subset of A, we use the recursive definition of S. First, the basis step of the definition specifies that 3 is in S. Because $3 = 3 \cdot 1$, all elements specified to be in S in this step are divisible by 3 and are therefore in A. To finish the proof, we must show that all integers in S generated using the second part of the recursive definition are in A. This consists of showing that $x + y$ is in A whenever x and y are elements of S also assumed to be in A. Now if x and y are both in A, it follows that $3|x$ and $3|y$ and it follows that $3|(x + y)$ completing the proof. ∎

In Example 10 we used mathematical induction over the set of positive integers and a recursive definition to prove a result about a recursively defined set. However, instead of using mathematical induction directly to prove results about recursively defined sets, we can use a more convenient form of induction known as **structural induction**. A proof by structural induction consists of two parts. These parts are

BASIS STEP: Show that the result holds for all elements specified in the basis step of the recursive definition to be in the set.

RECURSIVE STEP: Show that if the statement is true for each of the elements used to construct new elements in the recursive step of the definition, the result holds for these new elements.

The validity of structural induction follows from the principle of mathematical induction for the nonnegative integers. To see this, let $P(n)$ state that the claim is true for all elements of the set that are generated by n or fewer applications of the rules in the recursive step of a recursive definition. We will have established that the principle of mathematical induction implies the principle of structural induction if we can show that $P(n)$ is true whenever n is a positive integer. In the basis step of a proof by structural induction we show that $P(0)$ is true. That is, we show that the result is true of all elements specified to be in the set in the basis step of the definition. A consequence of the recursive step is that if we assume $P(k)$ is true, it follows that $P(k+1)$ is true. When we have completed a proof using structural induction, we have shown that $P(0)$ is true and that $P(k)$ implies $P(k+1)$. By mathematical induction it follows that $P(n)$ is true for all nonnegative integers n. This also shows that the result is true for all elements generated by the recursive definition, and shows that structural induction is a valid proof technique.

EXAMPLES OF PROOFS USING STRUCTURAL INDUCTION Structural induction can be used to prove that all members of a set constructed recursively have a particular property. We will illustrate this idea by using structural induction to prove results about well-formed formulae, strings, and binary trees. For each proof, we have to carry out the appropriate basis step and the appropriate recursive step. For example, to use structural induction to prove a result about the set of well-formed formulae defined in Example 8, where we specify that **T, F**, and every propositional variable s are well-formed formulae and where we specify that if E and F are well-formed formulae, then $(\neg E)$, $(E \wedge F)$, $(E \vee F)$, $(E \rightarrow F)$, and $(E \leftrightarrow F)$ are well-formed formulae, we need to complete this basis step and this recursive step.

BASIS STEP: Show that the result is true for **T, F**, and s whenever s is a propositional variable.

RECURSIVE STEP: Show that if the result is true for the compound propositions p and q, it is also true for $(\neg p)$, $(p \vee q)$, $(p \wedge q)$, $(p \rightarrow q)$, and $(p \leftrightarrow q)$.

Example 11 illustrates how we can prove results about well-formed formulae using structural induction.

EXAMPLE. 11

Show that every well-formed formula for compound propositions, as defined in Example 8, contains an equal number of left and right parentheses.

Solution:

BASIS STEP: Each of the formula **T, F**, and s contains no parentheses, so clearly they contain an equal number of left and right parentheses.

RECURSIVE STEP: Assume p and q are well-formed formulae, each containing an equal number of left and right parentheses. That is, if l_p and l_q are the number of left parentheses in p and q, respectively, and r_p and r_q are the number of right parentheses in p and q, respectively, then $l_p = r_p$ and $l_q = r_q$.

To complete the inductive step, we need to show that each of $(\neg p)$, $(p \vee q)$, $(p \wedge q)$, $(p \rightarrow q)$, and $(p \leftrightarrow q)$ also contains an equal number of left and right parentheses. The number of left parentheses in the first of these compound propositions equals $l_p + 1$ and in each of the other compound propositions equals $l_p + l_q + 1$. Similarly, the number of right parentheses in the first of these compound propositions equals $r_p + 1$ and in each of the other compound propositions equals $r_p + r_q + 1$. Because $l_p = r_p$ and $l_q = r_q$, it follows that each of these compound expressions contains the same number of left and right parentheses. This completes the proof by structural induction. ∎

Suppose that $P(w)$ is a propositional function over the set of strings $w \in \sum^*$. To use structural induction to prove that $P(w)$ holds for all strings $w \in \sum^*$, we need to complete both a basis step and a recursive step. These steps are:

BASIS STEP: Show that $P(\lambda)$ is true.

RECURSIVE STEP: Assume that $P(w)$ is true, where $w \in \sum^*$. Show that if $x \in \sum$, then $P(wx)$ must also be true.

Example 12 illustrates how structural induction can be used in proofs about strings.

EXAMPLE. 12

Use structural induction to prove that $l(xy) = l(x) + l(y)$, where x and y belong to $\sum^*$, the set of strings over the alphabet $\sum$.

Solution: We will base our proof on the recursive definition of the set $\sum^*$ given in Definition 1 and the definition of the length of a string in Example 7, which specifies that $l(\lambda) = 0$ and $l(wx) = l(w) + 1$ when $w \in \sum^*$ and $x \in \sum$. Let $P(y)$ be the statement that $l(xy) = l(x) + l(y)$ whenever x belongs to $\sum^*$.

BASIS STEP: To complete the basis step, we must show that $P(\lambda)$ is true. That is, we must show that $l(x\lambda) = l(x) + l(\lambda)$ for all $x \in \sum^*$. Because $l(x\lambda) = l(x) = l(x) + 0 = l(x) + l(\lambda)$ for every string x, it follows that $P(\lambda)$ is true.

RECURSIVE STEP: To complete the inductive step, we assume that $P(y)$ is true and show that this implies that $P(ya)$ is true whenever $a \in \sum$. What we need to show is that $l(xya) = l(x) + l(ya)$ for every $a \in \sum$. To show this, note that by the recursive definition of $l(w)$ (given in Example 7), we have $l(xya) = l(xy) + 1$ and $l(ya) = l(y) + 1$. And, by the inductive hypothesis,

$l(xy) = l(x)+l(y)$. We conclude that $l(xya) = l(x)+l(y)+1 = l(x)+l(ya)$. ∎

We can prove results about trees or special classes of trees using structural induction.

BASIS STEP: Show that the result is true for the tree consisting of a single vertex.

RECURSIVE STEP: Show that if the result is true for the full binary trees T_1 and T_2, then it is true for tree $T_1 \cdot T_2$ consisting of a root r, which has T_1 as its left subtree and T_2 as its right subtree.

Before we provide an example showing how structural induction can be used to prove a result about full binary trees, we need some definitions. We will recursively define the height $h(T)$ and the number of vertices $n(T)$ of a full binary tree T.bWe begin by defining the height of a full binary tree.

DEFINITION 1.2.6 *We define the height $h(T)$ of a full binary tree T recursively.*

BASIS STEP: The height of the full binary tree T consisting of only a root r is $h(T) = 0$.

RECURSIVE STEP: If T_1 and T_2 are full binary trees, then the full binary tree $T = T_1 \cdot T_2$ has height $h(T) = 1 + max(h(T_1), h(T_2))$.

If we let $n(T)$ denote the number of vertices in a full binary tree, we observe that $n(T)$ satisfies the following recursive formula:

BASIS STEP: The number of vertices $n(T)$ of the full binary tree T consisting of only a root r is $n(T) = 1$.

RECURSIVE STEP: If T_1 and T_2 are full binary trees, then the number of vertices of the full binary tree $T = T_1 \cdot T_2$ is $n(T) = 1 + n(T_1) + n(T_2)$.

We now show how structural induction can be used to prove a result about full binary trees.

THEOREM 1.2.2

If T is a full binary tree T, then $n(T) \leq 2^{h(T)+1} - 1$.

Proof: We prove this inequality using structural induction.

BASIS STEP: For the full binary tree consisting of just the root r the result is true because $n(T) = 1$ and $h(T) = 0$, so that $n(T) = 1 \leq 2^{0+1} - 1 = 1$.

RECURSIVE STEP: For the inductive hypothesis we assume that $n(T_1) \leq 2^{h(T_1)+1} - 1$ and $n(T_2) \leq 2^{h(T_2)+1} - 1$ whenever T_1 and T_2 are full binary trees. By the recursive formulae for $n(T)$ and $h(T)$ we have $n(T) = 1 + n(T_1) + n(T_2)$ and $h(T) = 1 + max(h(T_1),\ h(T_2))$. We find that

$$\begin{array}{rcll}
n(T) & = & 1 + n(T_1) + n(T_2) & \text{by the recursive formula} \\
 & & & \text{for } n(T) \\
 & \leq & 1 + (2^{h(T_1)+1} - 1) + (2^{h(T_2)+1} - 1) & \text{by the inductive hypothesis} \\
 & \leq & 2 \cdot max(2^{h(T_1)+1},\ 2^{h(T_2)+1}) - 1 & \text{because the sum of two terms} \\
 & & & \text{is at most 2 times the larger} \\
 & = & 2 \cdot 2^{max(h(T_1),h(T_2))+1} - 1 & \text{because } max(2^x, 2^y) \\
 & & & = 2^{max(x,y)} \\
 & = & 2 \cdot 2^{h(T)} - 1 & \text{by the recursive definition} \\
 & & & \text{of } h(T) \\
 & = & 2^{h(T)+1} - 1. &
\end{array}$$

This completes the recursive step. ∎

1.2.5 Generalized Induction

We can extend mathematical induction to prove results about other sets that have the wellordering property besides the set of integers. We provide an example here to illustrate the usefulness of such an approach.

As an example, note that we can define an ordering on $\mathbf{N} \times \mathbf{N}$, the ordered pairs of nonnegative integers, by specifying that (x_1, y_1) is less than or equal to (x_2, y_2) if either $x_1 < x_2$, or $x_1 = x_2$ and $y_1 < y_2$; this is called the **lexicographic ordering**. The set $N \times N$ with this ordering has the property that every subset of $N \times N$ has a least element. This implies that we can recursively define the terms $a_{m,n}$, with $m \in N$ and $n \in N$, and prove results about them using a variant of mathematical induction, as illustrated in Example 13.

EXAMPLE. 13

Suppose that $a_{m,n}$ is defined recursively for $(m, n) \in N \times N$ by $a_{0,0} = 0$ and

$$a_{m,n} = \begin{cases} a_{m-1,n} + 1 & \text{if } n = 0 \text{and } m > 0 \\ a_{m,n-1} + n & \text{if } n > 0. \end{cases}$$

Show that $am, n = m + n(n + 1)/2$ for all $(m, n) \in \mathbf{N} \times \mathbf{N}$, that is, for all pairs of nonnegative integers.

$\boxed{\textit{Solution:}}$We can prove that $a_{m,n} = m + n(n + 1)/2$ using a generalized version of mathematical induction. The basis step requires that we show that this formula is valid when $(m, n) = (0, 0)$. The induction step requires that we show that if the formula holds for all pairs smaller than (m, n) in the lexicographic ordering of $\mathbf{N} \times \mathbf{N}$, then it also holds for (m, n).

BASIS STEP: Let $(m, n) = (0, 0)$. Then by the basis case of the recursive definition of $a_{m,n}$ we have $a_{0,0} = 0$. Furthermore, when $m = n = 0$, $m + n(n + 1)/2 = 0 + (0 \cdot 1)/2 = 0$. This completes the basis step.

INDUCTIVE STEP: Suppose that $a_{m',n'} = m' + n'(n' + 1)/2$ whenever (m', n') is less than (m, n) in the lexicographic ordering of $\mathbf{N} \times \mathbf{N}$. By the recursive definition, if $n = 0$, then $a_{m,n} = a_{m-1,n+1}$. Because $(m - 1, n)$ is smaller than (m, n), the inductive hypothesis tells us that $a_{m-1,n} = m - 1 + n(n + 1)/2$, so that $a_{m,n} = m - 1 + n(n + 1)/2 + 1 = m + n(n + 1)/2$, giving us the desired equality. Now suppose that $n > 0$, so $a_{m,n} = a_{m,n?1} + n$. Because $(m, n - 1)$ is smaller than (m, n), the inductive hypothesis tells us that $a_{m,n?1} = m + (n-1)n/2$, so $a_{m,n} = m + (n-1)n/2 + n = m + (n^2 - n + 2n)/2 = m + n(n + 1)/2$. This finishes the inductive step. ∎

1.3 Recursive Algorithms

1.3.1 Introduction

Sometimes we can reduce the solution to a problem with a particular set of input values to the solution of the same problem with smaller input values. For instance, the problem of finding the greatest common divisor of two positive integers a and b, where $b > a$, can be reduced to finding the greatest common divisor of a pair of smaller integers, namely, $b\mathbf{mod}a$ and a, because $gcd(b\mathbf{mod}a,\ a) = gcd(a, b)$.

When such a reduction can be done, the solution to the original problem can be found with a sequence of reductions, until the problem has been reduced to some initial case for which the solution is known. For instance, for finding the greatest common divisor, the reduction continues until the smaller of the two numbers is zero, because $gcd(a, 0) = a$ when $a > 0$.

We will see that algorithms that successively reduce a problem to the same problem with smaller input are used to solve a wide variety of problems.

DEFINITION 1.3.1 *An algorithm is called* ***recursive*** *if it solves a problem by reducing it to an instance of the same problem with smaller input.*

We will describe a variety of different recursive algorithms in this section.

EXAMPLE. 1

Give a recursive algorithm for computing $n!$, where n is a nonnegative integer.

Solution: We can build a recursive algorithm that finds $n!$, where n is a nonnegative integer, based on the recursive definition of $n!$, which specifies that $n! = n\cdot(n-1)!$ when n is a positive integer, and that $0! = 1$. To find $n!$ for a particular integer n, we use the recursive step n times, each time replacing a value of the factorial function with the value of the factorial function at the next smaller integer. At this last step, we insert the value of $0!$. The recursive algorithm we obtain is displayed as Algorithm 1.

To help understand how this algorithm works, we trace the steps used by the algorithm to compute $4!$. First, we use the recursive step to write $4! = 4\cdot 3!$.

We then use the recursive step repeatedly to write $3! = 3 \cdot 2!$, $2! = 2 \cdot 1!$, and $1! = 1 \cdot 0!$. Inserting the value of $0! = 1$, and working back through the steps, we see that $1! = 1 \cdot 1 = 1$, $2! = 2 \cdot 1! = 2$, $3! = 3 \cdot 2! = 3 \cdot 2 = 6$, and $4! = 4 \cdot 3! = 4 \cdot 6 = 24$. ∎

ALGORITHM 1

A Recursive Algorithm for Computing $n!$.

```
procedure factorial(n: nonnegative integer)
if $n = 0$ then return 1
else return $n \cdot factorial(n - 1)$
{output is $n!$}
```

Example 2 shows how a recursive algorithm can be constructed to evaluate a function from its recursive definition.

EXAMPLE. 2

Give a recursive algorithm for computing a^n, where a is a nonzero real number and n is a nonnegative integer.

Solution: We can base a recursive algorithm on the recursive definition of a^n. This definition states that $a^{n+1} = a \cdot a^n$ for $n > 0$ and the initial condition $a^0 = 1$. To find a^n, successively use the recursive step to reduce the exponent until it becomes zero. We give this procedure in Algorithm 2. ∎

ALGORITHM 2

A Recursive Algorithm for Computing a^n.

```
procedure power($a$: nonzero real number, $n$: nonnegative integer)
if $n = 0$ then return 1
else return $a \cdot power(a, n - 1)$
{output is $a^n$}
```

Next we give a recursive algorithm for finding greatest common divisors.

EXAMPLE. 3

Give a recursive algorithm for computing the greatest common divisor of two nonnegative integers a and b with $a < b$.

Solution: We can base a recursive algorithm on the reduction $gcd(a, b) = gcd(b\mathbf{mod}a, a)$ and the condition $gcd(0, b) = b$ when $b > 0$. This produces the procedure in Algorithm 3, which is a recursive version of the Euclidean algorithm.

We illustrate the workings of Algorithm 3 with a trace when the input is $a = 5,\ b = 8$. With this input, the algorithm uses the "else" clause to find that $gcd(5, 8) = gcd(8\mathbf{mod}5, 5) = gcd(3, 5)$. It uses this clause again to find that $gcd(3, 5) = gcd(5\mathbf{mod}3, 3) = gcd(2, 3)$, then to get $gcd(2, 3) = gcd(3\mathbf{mod}2, 2) = gcd(1, 2)$, then to get $gcd(1, 2) = gcd(2\mathbf{mod}1, 1) = gcd(0, 1)$. Finally, to find $gcd(0, 1)$ it uses the first step with $a = 0$ to find that $gcd(0, 1) = 1$. Consequently, the algorithm finds that $gcd(5, 8) = 1$. ∎

ALGORITHM 3

A Recursive Algorithm for Computing $gcd(a, b)$.

```
procedure gcd($a, b$: nonnegative integers with $a < b$)
if $a = 0$ then return $b$
else return gcd($b$ mod $a, a$)
{output is gcd($a, b$)}
```

EXAMPLE. 4

Devise a recursive algorithm for computing $b^n\mathbf{mod}m$, where $b,\ n$, and m are integers with $m \geq 2,\ n \geq 0$, and $1 \leq b < m$.

Solution: We can base a recursive algorithm on the fact that

$$b^n \ \mathbf{mod}\ m = (b \cdot (b^{n-1} \ \mathbf{mod}\ m)) \ \mathbf{mod}\ m,$$

and the initial condition $b^0\mathbf{mod}m = 1$. However, we can devise a much more efficient recursive algorithm based on the observation that

$$b^n textbfmod\ m = (b^{n/2}\ \mathbf{mod}\ m)^2\ \mathbf{mod}\ m$$

when n is even and

$$b^n\ \mathbf{mod}\ m = ((b^{\lfloor n/2 \rfloor}\mathbf{mod}\ m)^2\mathbf{mod}\ m \cdot b\ \mathbf{mod}\ m)\ \mathbf{mod}\ m$$

when n is odd, which we describe in pseudocode as Algorithm 4.

We trace the execution of Algorithm 4 with input $b = 2$, $n = 5$, and $m = 3$ to illustrate how it works. First, because $n = 5$ is odd we use the "else" clause to see that $mpower(2,\ 5,\ 3) = (mpower(2,\ 2,\ 3)^2\ \mathbf{mod}\ 3 \cdot 2\ \mathbf{mod}\ 3)\ \mathbf{mod}\ 3$. We next use the "else if" clause to see that $mpower(2,\ 2,\ 3) = mpower(2,\ 1,\ 3)^2\ \mathbf{mod}\ 3$. Using the "else" clause again, we see that $mpower(2,\ 1,\ 3) = (mpower(2,\ 0,\ 3)^2\ \mathbf{mod}\ 3 \cdot 2\ \mathbf{mod}\ 3)\ \mathbf{mod}\ 3$. Finally, using the "if" clause, we see that $mpower(2,\ 0,\ 3) = 1$. Working backwards, it follows that $mpower(2,\ 1,\ 3) = (1^2\ \mathbf{mod}\ 3 \cdot 2\ \mathbf{mod}\ 3)\ \mathbf{mod}\ 3 = 2$, so $mpower(2,\ 2,\ 3) = 2^2\ \mathbf{mod}\ 3 = 1$, and finally $mpower(2,\ 5,\ 3) = (1^2\ \mathbf{mod}\ 3 \cdot 2\ \mathbf{mod}\ 3)\ \ \mathbf{mod}\ 3 = 2$. ∎

ALGORITHM 4
Recursive Modular Exponentiation.

```
procedure $mpower(b, n, m$: integers with $b > 0$ and $m \geq 2,\ n \geq 0)$
if $n = 0$ then
    return 1
else if $n$ is even then
    return $mpower(b,\ n/2,\ m)^2$ mod $m$
else
    return $(mpower(b,\ \lfloor n/2 \rfloor, m)^2$ mod $m \cdot b$ mod $m)$ mod $m$
{output is $b^n$ mod $m$}
```

EXAMPLE. 5

Express the linear search algorithm as a recursive procedure.

Solution: To search for the first occurrence of x in the sequence $a_1,\ a_2, \ldots, a_n$, at the ith step of the algorithm, x and a_i are compared. If x equals a_i, then the algorithm returns i, the location of x in the sequence. Otherwise, the search for the first occurrence of x is reduced to a search in a sequence with one fewer element, namely, the sequence $a_{i+1}, \dots, a_n$. The algorithm returns 0 when x is never found in the sequence after all terms have been examined. We can now give a recursive procedure, which is displayed as pseudocode in Algorithm 5.

Let search (i, j, x) be the procedure that searches for the first occurrence of x in the sequence $a_i,\ a_{i+1}, \dots, a_j$. The input to the procedure consists of the triple $(1,\ n,\ x)$. The algorithm terminates at a step if the first termof the remaining sequence is x or if there is only one termof the sequence and this is not x. If x is not the first term and there are additional terms, the same procedure is carried out but with a search sequence of one fewer term, obtained by deleting the first term of the search sequence. If the algorithm terminates without x having been found, the algorithm returns the value 0. ∎

ALGORITHM 5

A Recursive Linear Search Algorithm.

```
procedure search($i, j, x$ : integers, $1 \leq i \leq j \leq n$)
if $a_i = x$ then
    return $i$
else if $i = j$ then
    return 0
else
    return search($i + 1,\ j,\ x$)
{output is the location of $x$ in $a_1,\ a_2, \dots, a_n$ if it appears; otherwise it is 0}
```

EXAMPLE. 6

Construct a recursive version of a binary search algorithm.

Solution: Suppose we want to locate x in the sequence $a_1,\ a_2, \dots, a_n$ of integers in increasing order. To perform a binary search, we begin by comparing x with the middle term, $a_{\lfloor (n+1)/2 \rfloor}$. Our algorithm will terminate if x

equals this term and return the location of this term in the sequence. Otherwise, we reduce the search to a smaller search sequence, namely, the first half of the sequence if x is smaller than the middle term of the original sequence, and the second half otherwise. We have reduced the solution of the search problem to the solution of the same problem with a sequence at most half as long. If we have never encountered the search term x, our algorithm returns the value 0.We express this recursive version of a binary search algorithm as Algorithm 6. ∎

ALGORITHM 6

A Recursive Binary Search Algorithm.

```
**procedure** *binary search*($i, j, x$: integers, $1 \leq i \leq j \leq n$)
$m := \lfloor (i+j)/2 \rfloor$
**if** $x = a_m$ **then**
    **return** $m$
**else if** ($x < a_m$ and $i < m$) **then**
    **return** *binary search*($i, m-1, x$)
**else if** ($x > a_m$ and $j > m$) **then**
    **return** *binary search*($m+1, j, x$)
**else return 0**
{output is location of $x$ in $a_1,\ a_2, \ldots, a_n$ if it appears; otherwise
it is 0}
```

1.3.2 Proving Recursive Algorithms Correct

Mathematical induction, and its variant strong induction, can be used to prove that a recursive algorithm is correct, that is, that it produces the desired output for all possible input values. Examples 7 and 8 illustrate how mathematical induction or strong induction can be used to prove that recursive algorithms are correct. First, we will show that Algorithm 2 is correct.

EXAMPLE. 7

Prove that Algorithm 2, which computes powers of real numbers, is correct.

Solution: We use mathematical induction on the exponent n.

BASIS STEP: If $n = 0$, the first step of the algorithm tells us that *power* $(a, 0) = 1$. This is correct because $a^0 = 1$ for every nonzero real number a. This completes the basis step.

INDUCTIVE STEP: The inductive hypothesis is the statement that power $(a, k) = a^k$ for all $a \neq 0$ for an arbitrary nonnegative integer k. That is, the inductive hypothesis is the statement that the algorithm correctly computes a^k. To complete the inductive step, we show that if the inductive hypothesis is true, then the algorithm correctly computes a^{k+1}. Because $k + 1$ is a positive integer, when the algorithm computes a^{k+1}, the algorithm sets *power* $(a, k + 1) = a \cdot power(a, k)$. By the inductive hypothesis, we have $power(a, k) = a^k$, so $power(a, k + 1) = a \cdot power(a, k) = a \cdot a^k = a^{k+1}$. This completes the inductive step.

We have completed the basis step and the inductive step, so we can conclude that Algorithm 2 always computes an correctly when $a \neq 0$ and n is a nonnegative integer. ∎

Generally, we need to use strong induction to prove that recursive algorithms are correct, rather than just mathematical induction. Example 8 illustrates this; it shows how strong induction can be used to prove that Algorithm 4 is correct.

EXAMPLE. 8

Prove that Algorithm 4, which computes modular powers, is correct.

Solution: We use strong induction on the exponent n.

BASIS STEP: Let b be an integer and m an integer with $m \geq 2$. When $n = 0$, the algorithm sets $mpower(b,\ n,\ m)$ equal to 1. This is correct because $b^0 \ \mathbf{mod}\ m = 1$. The basis step is complete.

INDUCTIVE STEP: For the inductive hypothesis we assume that

$mpower(b, j, m) = b^j \ \mathbf{mod}\ m$ for all integers $0 \leq j < k$ whenever b is a positive integer and m is an integer with $m \geq 2$. To complete the inductive step, we show that if the inductive hypothesis is correct, then $mpower(b, k, m) = b^k \ \mathbf{mod}\ m$. Because the recursive algorithm handles odd and even values of k differently, we split the inductive step into two cases.

When k is even, we have

$$\begin{aligned}(b,k,m) &= (mpower(b,k/2,m))^2 \textbf{ mod } m\\ &= (b^{k/2} \textbf{ mod } m)^2 \textbf{ mod } m\\ &= b^k \textbf{ mod } m,\end{aligned}$$

where we have used the inductive hypothesis to replace $mpower(b,k/2,m)$ by $b^{k/2}$ **mod** m.

When k is odd, we have

$$\begin{aligned}mpower(b,k,m) &= ((mpower(b,\lfloor k/2\rfloor,m))^2 \textbf{ mod } m \cdot b \textbf{ mod } m)\textbf{mod } m\\ &= ((b^{\lfloor k/2\rfloor} \textbf{ mod } m)^2 \textbf{ mod } m \cdot b \textbf{ mod } m) \textbf{ mod } m\\ &= b^{2\lfloor k/2\rfloor+1} \textbf{ mod } m = b^k \textbf{ mod } m,\end{aligned}$$

because $2\lfloor k/2\rfloor + 1 = 2(k-1)/2 + 1 = k$ when k is odd. Here we have used the inductive hypothesis to replace $mpower(b,\lfloor k/2\rfloor,m)$ by $b^{\lfloor k/2\rfloor}$ **mod** m. This completes the inductive step.

We have completed the basis step and the inductive step, so by strong induction we know that Algorithm 4 is correct. ∎

1.3.3 Recursion and Iteration

A recursive definition expresses the value of a function at a positive integer in terms of the values of the function at smaller integers. This means that we can devise a recursive algorithm to evaluate a recursively defined function at a positive integer. Instead of successively reducing the computation to the evaluation of the function at smaller integers, we can start with the value of the function at one or more integers, the base cases, and successively apply the recursive definition to find the values of the function at successive larger integers. Such a procedure is called **iterative**. Often an iterative approach for the evaluation of a recursively defined sequence requires much less computation than a procedure using recursion (unless special-purpose recursive machines are used). This is illustrated by the iterative and recursive procedures for finding the nth Fibonacci number. The recursive procedure is given first.

ALGORITHM 7

A Recursive Algorithm for Fibonacci Numbers.

```
procedure fibonacci($n$: nonnegative integer)
if $n = 0$ then return 0
else if $n = 1$ then return 1
else return fibonacci($n - 1$)+ fibonacci($n - 2$)
{output is fibonacci($n$)}
```

When we use a recursive procedure to find f_n, we first express f_n as $f_{n-1} + f_{n-2}$. Then we replace both of these Fibonacci numbers by the sum of two previous Fibonacci numbers, and so on. When f_1 or f_0 arises, it is replaced by its value.

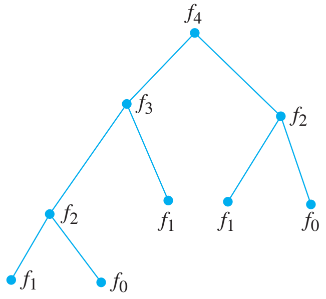


Figure 1.12: Evaluating f_4 recursively.

Note that at each stage of the recursion, until f_1 or f_0 is obtained, the number of Fibonacci numbers to be evaluated has doubled. For instance, when we find f_4 using this recursive algorithm, we must carry out all the computations illustrated in the tree diagram in Figure 1.12.

This tree consists of a root labeled with f_4, and branches from the root to vertices labeled with the two Fibonacci numbers f_3 and f_2 that occur in the reduction of the computation of f_4. Each subsequent reduction produces two branches in the tree. This branching ends when f_0 and f_1 are reached. The reader can verify that this algorithm requires $f_{n+1} - 1$ additions to find f_n. Now consider the amount of computation required to find f_n using the iterative approach in Algorithm 8.

ALGORITHM 8

An Iterative Algorithm for Computing Fibonacci Numbers.

```
procedure iterative fibonacci($n$ : nonnegative integer)
if $n = 0$ then return 0
else
  $x := 0$
  $y := 1$
   for $i := 1$ to $n - 1$
     $z := x + y$
     $x := y$
     $y := z$
   return $y$
{output is the $n$th Fibonacci number}
```

This procedure initializes x as $f_0 = 0$ and y as $f_1 = 1$. When the loop is traversed, the sum of x and y is assigned to the auxiliary variable z. Then x is assigned the value of y and y is assigned the value of the auxiliary variable z. Therefore, after going through the loop the first time, it follows that x equals f_1 and y equals $f_0 + f_1 = f_2$. Furthermore, after going through the loop $n - 1$ times, x equals f_{n-1} and y equals f_n (the reader should verify this statement). Only $n - 1$ additions have been used to find f_n with this iterative approach when $n > 1$. Consequently, this algorithm requires far less computation than does the recursive algorithm.

We have shown that a recursive algorithm may require far more computation than an iterative one when a recursively defined function is evaluated. It is sometimes preferable to use a recursive procedure even if it is less efficient than the iterative procedure. In particular, this is true when the recursive approach is easily implemented and the iterative approach is not.

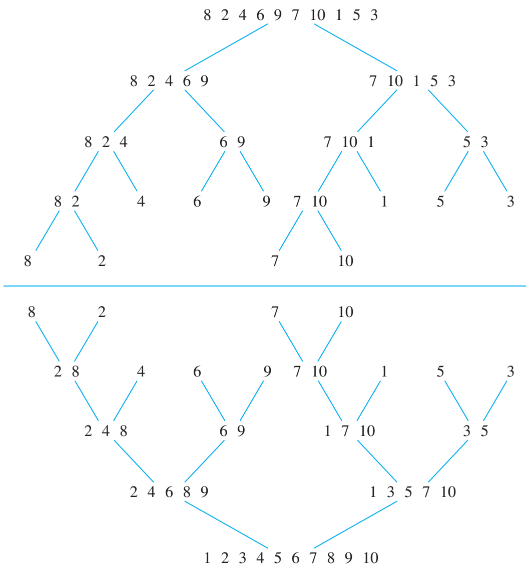


Figure 1.13: The merge sort of 8, 2, 4, 6, 9, 7, 10, 1, 5, 3.

1.3.4 The Merge Sort

We nowdescribe a recursive sorting algorithm called the **merge sort** algorithm. We will demonstrate how the merge sort algorithm works with an example before describing it in generality.

EXAMPLE. 9

Use the merge sort to put the terms of the list 8, 2, 4, 6, 9, 7, 10, 1, 5, 3 in increasing order.

Solution: A merge sort begins by splitting the list into individual elements by successively splitting lists in two. The progression of sublists for this example is represented with the balanced binary tree of height 4 shown in the upper half of Figure 1.13.

Sorting is done by successively merging pairs of lists. At the first stage, pairs of individual elements are merged into lists of length two in increasing order. Then successive merges of pairs of lists are performed until the entire list is put into increasing order. The succession of merged lists in increasing order is represented by the balanced binary tree of height 4 shown in the lower half of Figure 1.13 (note that this tree is displayed “upside down”). ∎

In general, a merge sort proceeds by iteratively splitting lists into two sublists of equal length (or where one sublist has one more element than the other) until each sublist contains one element. This succession of sublists can be represented by a balanced binary tree. The procedure continues by successively merging pairs of lists, where both lists are in increasing order, into a larger list with elements in increasing order, until the original list is put into increasing order. The succession of merged lists can be represented by a balanced binary tree.

We can also describe the merge sort recursively. To do a merge sort, we split a list into two sublists of equal, or approximately equal, size, sorting each sublist using the merge sort algorithm, and then merging the two lists. The recursive version of the merge sort is given in Algorithm 9. This algorithm uses the subroutine merge, which is described in Algorithm 10.

ALGORITHM 9

A Recursive Merge Sort.

```
procedure $mergesort(L = a_1, \dots, a_n)$
if $n > 1$ then
  $m := \lfloor n/2 \rfloor$
  $L_1 := a_1, a_2, \dots, a_m$
  $L_2 := a_{m+1}, a_{m+2}, \dots, a_n$
$quadL := merge(mergesort(L_1), mergesort(L_2))$
$\{L$ is now sorted into elements in nondecreasing order$\}$
```

An efficient algo rithm for merging two ordered lists into a larger ordered list is needed to implement the merge sort. We will now describe such a procedure.

EXAMPLE. 10

Merge the two lists 2, 3, 5, 6 and 1, 4.

Table 1.1: Merging the Two Sorted Lists 2, 3, 5, 6 and 1, 4.

First List	Second List	Merged List	Comparison
2 3 5 6	1 4		$1 < 2$
2 3 5 6	4	1	$2 < 4$
3 5 6	4	1 2	$3 < 4$
5 6	4	1 2 3	$4 < 5$
5 6		1 2 3 4	
		1 2 3 4 5 6	

Solution: Table 1.1 illustrates the steps we use. First, compare the smallest elements in the two lists, 2 and 1, respectively. Because 1 is the smaller, put it at the beginning of the merged list and remove it from the second list. At this stage, the first list is 2, 3, 5, 6, the second is 4, and the combined list is 1.

Next, compare 2 and 4, the smallest elements of the two lists. Because 2 is the smaller, add it to the combined list and remove it from the first list. At this stage the first list is 3, 5, 6, the second is 4, and the combined list is 1, 2.

Continue by comparing 3 and 4, the smallest elements of their respective lists. Because 3 is the smaller of these two elements, add it to the combined list and remove it from the first list. At this stage the first list is 5, 6, and the second is 4. The combined list is 1, 2, 3.

Then compare 5 and 4, the smallest elements in the two lists. Because 4 is the smaller of these two elements, add it to the combined list and remove it from the second list. At this stage the first list is 5, 6, the second list is empty, and the combined list is 1, 2, 3, 4.

Finally, because the second list is empty, all elements of the first list can be appended to the end of the combined list in the order they occur in the first list. This produces the ordered list 1, 2, 3, 4, 5, 6. ∎

We will now consider the general problem of merging two ordered lists L_1 and L_2 into an ordered list L. We will describe an algorithm for solving this problem. Start with an empty list L. Compare the smallest elements of the two lists. Put the smaller of these two elements at the right end of L, and remove it from the list it was in. Next, if one of L_1 and L_2 is empty, append the other (nonempty) list to L, which completes the merging. If neither L_1 nor L_2 is empty, repeat this process. Algorithm 10 gives a pseudocode description of this procedure.

We will need estimates for the number of comparisons used to merge two ordered lists in the analysis of the merge sort. We can easily obtain such an estimate for Algorithm 10. Each time a comparison of a_n element from L_1 and an element from L_2 is made, an additional element is added to the merged list L. However, when either L_1 or L_2 is empty, no more comparisons are needed. Hence, Algorithm 10 is least efficient when $m+n-2$ comparisons are carried out, where m and n are the number of elements in L_1 and L_2, respectively, leaving one element in each of L_1 and L_2. The next comparison will be the last one needed, because it will make one of these lists empty. Hence, Algorithm 10 uses no more than $m+n-1$ comparisons. Lemma 1 summarizes this estimate.

ALGORITHM 10

Merging Two Lists.

```
procedure merge($L_1, L_2$ : sorted lists)
$L$ := empty list
while $L_1$ and $L_2$ are both nonempty
    remove smaller of first elements of $L_1$ and $L_2$ from its list;
    put it at the right end of $L$
    if this removal makes one list empty then remove all elements
    from the other list and append them to $L$
return $L$\{$L$ is the merged list with elements in increasing order\}
```

LEMMA 1.3.1

Two sorted lists with m elements and n elements can be merged into a sorted list using no more than $m+n-1$ comparisons.

Sometimes two sorted lists of length m and n can be merged using far fewer than $m+n-1$ comparisons. For instance, when $m=1$, a binary search procedure can be applied to putv the one element in the first list into the second list. This requires only $\lceil \log n \rceil$ comparisons, which is much smaller than $m+n-1=n$, for $m=1$. On the other hand, for some values of m and n, Lemma 1 gives the best possible bound. That is, there are lists with m and n elements that cannot be merged using fewer than $m+n-1$ comparisons.

We can now analyze the complexity of the merge sort. Instead of studying the general problem, we will assume that n, the number of elements in the list, is a power of 2, say 2^m. This will make the analysis less complicated, but when this is not the case, various modifications can be applied that will yield the same estimate.

At the first stage of the splitting procedure, the list is split into two sublists, of 2^{m-1} elements each, at level 1 of the tree generated by the splitting. This process continues, splitting the two sublists with 2^{m-1} elements into four sublists of 2^{m-2} elements each at level 2, and so on. In general, there are 2^{k-1} lists at level $k-1$, each with 2^{m-k+1} elements. These lists at level $k-1$ are split into 2^k lists at level k, each with 2^{m-k} elements. At the end of this process, we have 2^m lists each with one element at level m.

We start merging by combining pairs of the 2^m lists of one element into 2^{m-1} lists, at level $m-1$, each with two elements. To do this, 2^{m-1} pairs of lists with one element each are merged. The merger of each pair requires exactly one comparison. The procedure continues, so that at level $k(k = m, m-1, m-2, \ldots, 3, 2, 1)$, 2^k lists each with 2^{m-k} elements are merged into 2^{k-1} lists, each with 2^{m-k+1} elements, at level $k-1$. To do this a total of 2^{k-1} mergers of two lists, each with 2^{m-k} elements, are needed. But, by Lemma 1, each of these mergers can be carried out using at most $2^{m-k} + 2^{m-k} - 1 = 2^{m-k+1} - 1$ comparisons. Hence, going from level k to $k-1$ can be accomplished using at most $2^{k-1}(2^{m-k+1} - 1)$ comparisons.

Summing all these estimates shows that the number of comparisons required for the merge sort is at most

$$\sum_{k=1}^{m} 2^{k-1}(2^{m-k+1} - 1) = \sum_{k=1}^{m} 2^m - \sum_{k=1}^{m} 2^{k-1} = m2^m - (2^m - 1) = n \log n - n + 1,$$

because $m = \log n$ and $n = 2^m$.

Theorem 1.3.1 summarizes what we have discovered about the worst-case complexity of the merge sort algorithm.

THEOREM 1.3.1

The number of comparisons needed to merge sort a list with n elements is $O(n \log n)$.

Theorem 1 tells us that the merge sort achieves this best possible big-O estimate for the complexity of a sorting algorithm.

1.4 Program Correctness

1.4.1 Introduction

Suppose that we have designed an algorithm to solve a problem and have written a program to implement it. How can we be sure that the program always produces the correct answer? After all the bugs have been removed so that the syntax is correct, we can test the program with sample input. It is not correct if an incorrect result is produced for any sample input. But even if the program gives the correct answer for all sample input, it may not always produce the correct answer (unless all possible input has been tested). We need a proof to show that the program *always* gives the correct output.

Program verification, the proof of correctness of programs, uses the rules of inference and proof techniques described in this chapter, including mathematical induction. Because an incorrect program can lead to disastrous results, a large amount of methodology has been constructed for verifying programs. Efforts have been devoted to automating program verification so that it can be carried out using a computer. However, only limited progress has been made toward this goal. Indeed, some mathematicians and theoretical computer scientists argue that it will never be realistic to mechanize the proof of correctness of complex programs.

Some of the concepts and methods used to prove that programs are correct will be introduced in this section. Many different methods have been devised for proving that programs are correct.

1.4.2 Program Verification

A program is said to be **correct** if it produces the correct output for every possible input. A proof that a program is correct consists of two parts. The first part shows that the correct answer is obtained if the program terminates. This part of the proof establishes the **partial correctness** of the program. The second part of the proof shows that the program always terminates.

To specify what it means for a program to produce the correct output, two propositions are used. The first is the **initial assertion**, which gives the properties that the input values must have. The second is the **final assertion**, which gives the properties that the output of the program should have, if the program did what was intended. The appropriate initial and final assertions must be provided when a program is checked.

DEFINITION 1.4.1 *A program, or program segment, S is said to be* ***partially correct with respect*** *to the initial assertion p and the final assertion q if whenever p is true for the input values of S and S terminates, then q is true for the output values of S. The notation $p\{S\}q$ indicates that the program, or program segment, S is partially correct with respect to the initial assertion p and the final assertion q.*

Note: The notation $p\{S\}q$ is known as a Hoare triple. Tony Hoare introduced the concept of partial correctness.

Note that the notion of partial correctness has nothing to do with whether a program terminates; it focuses only on whether the program does what it is expected to do if it terminates.

A simple example illustrates the concepts of initial and final assertions.

EXAMPLE. 1

Show that the program segment

$$\begin{aligned} y &:= 2 \\ z &:= x + y \end{aligned}$$

is correct with respect to the initial assertion $p : x = 1$ and the final assertion $q : z = 3$.

Solution: Suppose that p is true, so that $x = 1$ as the program begins. Then y is assigned the value 2, and z is assigned the sum of the values of x and y, which is 3. Hence, S is correct with respect to the initial assertion p and the final assertion q. Thus, $p\{S\}q$ is true. ∎

1.4.3 Rules of Inference

A useful rule of inference proves that a program is correct by splitting the program into a sequence of subprograms and then showing that each subprogram is correct.

Suppose that the program S is split into subprograms S_1 and S_2. Write $S = S_1; S_2$ to indicate that S is made up of S_1 followed by S_2. Suppose that the correctness of S_1 with respect to the initial assertion p and final assertion q, and the correctness of S_2 with respect to the initial assertion q and the final assertion r, have been established. It follows that if p is true and S_1 is executed and terminates, then q is true; and if q is true, and S_2 executes and terminates, then r is true. Thus, if p is true and $S = S_1; S_2$ is executed and terminates, then r is true. This rule of inference, called the **composition rule**, can be stated as

$$\begin{array}{ll} & p\{S_1\}q \\ & q\{S_2\}r \\ \hline \therefore & p\{S_1; S_2\}r \end{array}$$

This rule of inference will be used later in this section.

Next, some rules of inference for program segments involving conditional statements and loops will be given. Because programs can be split into segments for proofs of correctness, this will let us verify many different programs.

1.4.4 Conditional Statements

First, rules of inference for conditional statements will be given. Suppose that a program segment has the form

```
if condition then
    S
```

where S is a block of statements. Then S is executed if *condition* is true, and it is not executed when *condition* is false. To verify that this segment is correct with respect to the initial assertion p and final assertion q, two things must be done. First, it must be shown that when

p is true and *condition* is also true, then q is true after S terminates. Second, it must be shown that when p is true and condition is false, then q is true (because in this case S does not execute).

This leads to the following rule of inference:

$$\begin{array}{l} (p \wedge \mathit{condition}\ \{S\}q) \\ (p \wedge \neg \mathit{condition}\ \rightarrow q) \\ \hline \therefore\ \ p\{\textbf{if}\ \mathit{condition}\ \textbf{then}\ S\}q \end{array}$$

Example 2 illustrates how this rule of inference is used.

EXAMPLE. 2

Verify that the program segment

```
if $x > y$ then
    $y := x$
```

is correct with respect to the initial assertion **T** and the final assertion $y \geq x$.

Solution: When the initial assertion is true and $x > y$, the assignment $y := x$ is carried out. Hence, the final assertion, which asserts that $y \geq x$, is true in this case. Moreover, when the initial assertion is true and $x > y$ is false, so that $x \leq y$, the final assertion is again true. Hence, using the rule of inference for program segments of this type, this program is correct with respect to the given initial and final assertions. ∎

Similarly, suppose that a program has a statement of the form

```
if condition then
    $S_1$
else
    $S_2$
```

If *condition* is true, then S_1 executes; if *condition* is false, then S_2 executes. To verify that this program segment is correct with respect to the initial assertion p and the final assertion q, two things must be done. First, it must be shown that when p is true and *condition* is true, then q is true after S_1 terminates. Second, it must be shown that when p is true and *condition* is false, then q is true after S_2 terminates. This leads to the following rule of inference:

$$\begin{array}{l} \quad (p \wedge \mathit{condition}\ \{S_1\}q) \\ \quad (p \wedge \neg\mathit{condition}\ \{S_2\}q) \\ \hline \therefore\ \ p\{\textbf{if}\ \mathit{condition}\ \textbf{then}\ \ S_1\ \textbf{else}\ S_2\}q \end{array}$$

Example 3 illustrates how this rule of inference is used.

EXAMPLE. 3

Verify that the program segment

```
if $x < 0$ then
    $abs := -x$
else
    $abs := x$
```

is correct with respect to the initial assertion **T** and the final assertion $abs = |x|$.

Solution: Two things must be demonstrated. First, it must be shown that if the initial assertion is true and $x < 0$, then $abs = |x|$. This is correct, because when $x < 0$ the assignment statement $abs := -x$ sets $abs = -x$, which is $|x|$ by definition when $x < 0$. Second, it must be shown that if the initial assertion is true and $x < 0$ is false, so that $x \geq 0$, then $abs = |x|$. This is also correct, because in this case the program uses the assignment statement $abs := x$, and x is $|x|$ by definition when $x \geq 0$, so $abs := x$. Hence, using the rule of inference for program segments of this type, this segment is correct with respect to the given initial and final assertions. ∎

1.4.5 Loop Invariants

Next, proofs of correctness of **while** loops will be described. To develop a rule of inference for program segments of the type

```
while condition
    S
```

note that S is repeatedly executed until condition becomes false. An assertion that remains true each time S is executed must be chosen. Such an assertion is called a loop invariant. In other words, p is a loop invariant if $(p \wedge$ *condition*$)\{S\}p$ is true.

Suppose that p is a loop invariant. It follows that if p is true before the program segment is executed, p and $\neg$ *condition* are true after termination, if it occurs. This rule of inference is

$$\frac{(p \wedge condition\ \{S\}p)}{\therefore\ \ p\{\textbf{while}\ condition S\}(\neg condition\ \ \wedge p)}$$

The use of a loop invariant is illustrated in Example 4.

EXAMPLE. 4

A loop invariant is needed to verify that the program segment

```
$i := 1$
$factorial := 1$
while $i < n$
    $i := i + 1$
    $factorial := factorial \cdot i$
```

terminates with $factorial = n!$ when n is a positive integer.

Let p be the assertion "$factorial = i!$ and $i \leq n$." We first prove that p is a loop invariant.

Suppose that, at the beginning of one execution of the **while** loop, p is true and the condition of the **while** loop holds; in other words, assume that $factorial = i!$ and that $i < n$. The new values i_{new} and $factorial_{new}$ of i and $factorial$ are $i_{new} = i + 1$ and $factorial_{new} = factorial \cdot (i + 1) = (i + 1)! = i_{new}!$. Because $i < n$, we also have $i_{new} = i + 1 \leq n$. Thus, p is true at the end of the execution of the loop. This shows that p is a loop invariant.

Now we consider the program segment. Just before entering the loop, $i = 1 \leq n$ and $factorial = 1 = 1! = i!$ both hold, so p is true. Because p is a loop invariant, the rule of inference just introduced implied that if the **while** loop terminates, it terminates with p true and with $i < n$ false. In this case, at the end, $factorial = i!$ and $i \leq n$ are true, but $i < n$ is false; in other words, $i = n$ and $factorial = i! = n!$, as desired.

Finally, we need to check that the **while** loop actually terminates. At the beginning of the program i is assigned the value 1, so after $n - 1$ traversals of the loop, the new value of i will be n, and the loop terminates at that point. ∎

A final example will be given to show how the various rules of inference can be used to verify the correctness of a longer program.

EXAMPLE. 5

We will outline how to verify the correctness of the program S for computing the product of two integers.

```
    procedure multiply(m, n: integers)

              S_1  { if n < 0 then a := -n
                   { elsea := n
              S_2  { k := 0
                   { x := 0
                   { whilek < a
              S_3  {     x := x + m
                   {     k := k + 1
              S_4  { if n < 0 then product := -x
                   { else product := x

return product
    {product equals mn}
```

The goal is to prove that after S is executed, *product* has the value mn. The proof of correctness can be carried out by splitting S into four segments, with $S = S_1;\ S_2;\ S_3;\ S_4$, as shown in the listing of S. The rule of composition can be used to build the correctness proof. Here is how the argument proceeds. The details will be left as an exercise for the reader.

Let p be the initial assertion "mand n are integers." Then, it can be shwn that $p\{S_1\}q$ is true, when q is the proposition $p \wedge (a = |n|)$. Next, let r be the proposition $q \wedge (k = 0) \wedge (x = 0)$. It is easily verified that $q\{S_2\}r$ is true.

It can be shown that "$x = mk$ and $k \leq a$" is an invariant for the loop in S_3. Furthermore, it is easy to see that the loop terminates after a iterations, with $k = a$, so $x = ma$ at this point. Because r implies that $x = m \cdot 0$ and $0 \leq a$, the loop invariant is true before the loop is entered. Because the loop terminates with $k = a$, it follows that $r\{S_3\}s$ is true, where s is the proposition "$x = ma$ and $a = |n|$." Finally, it can be shown that S_4 is correct with respect to the initial assertion s and final assertion t, where t is the proposition "$product = mn$."

Putting all this together, because $p\{S_1\}q$, $q\{S_2\}r$, $r\{S_3\}s$, and $s\{S_4\}t$ are all true, it follows from the rule of composition that $p\{S\}t$ is true. Furthermore, because all four segments terminate, S does terminate. This verifies the correctness of the program. ∎

Chapter 2

Counting

Combinatorics, the study of arrangements of objects, is an important part of discrete mathematics. This subject was studied as long ago as the seventeenth century, when combinatorial questions arose in the study of gambling games. Enumeration, the counting of objects with certain properties, is an important part of combinatorics. We must count objects to solve many different types of problems. For instance, counting is used to determine the complexity of algorithms. Counting is also required to determine whether there are enough telephone numbers or Internet protocol addresses to meet demand. Recently, it has played a key role in mathematical biology, especially in sequencing DNA. Furthermore, counting techniques are used extensively when probabilities of events are computed.

We can phrase many counting problems in terms of ordered or unordered arrangements of the objects of a set with or without repetitions. These arrangements, called permutations and combinations, are used in many counting problems. For instance, suppose the 100 top finishers on a competitive exam taken by 2000 students are invited to a banquet. We can count the possible sets of 100 students that will be invited, as well as the ways in which the top 10 prizes can be awarded.

Another problem in combinatorics involves generating all the arrangements of a specified kind. This is often important in computer simulations. We will devise algorithms to generate arrangements of various types.

2.1 The Basics of Counting

2.1.1 Introduction

Suppose that a password on a computer system consists of six, seven, or eight characters. Each of these characters must be a digit or a letter of the alphabet. Each password must contain at least one digit. How many such passwords are there? The techniques needed to answer this question and a wide variety of other counting problems will be introduced in this section.

Counting problems arise throughout mathematics and computer science. For example, we must count the successful outcomes of experiments and all the possible outcomes of these experiments to determine probabilities of discrete events.We need to count the number of operations used by an algorithm to study its time complexity.

We will introduce the basic techniques of counting in this section. These methods serve as the foundation for almost all counting techniques.

2.1.2 Basic Counting Principles

We first present two basic counting principles, the **product rule** and the **sum rule**. Then we will show how they can be used to solve many different counting problems.

The product rule applies when a procedure is made up of separate tasks.

THE PRODUCT RULE
Suppose that a procedure can be broken down into a sequence of two tasks. If there are n_1 ways to do the first task and for each of these ways of doing the first task, there are n_2 ways to do the second task, then there are $n_1 n_2$ ways to do the procedure.

Examples 1–10 show how the product rule is used.

EXAMPLE. 1

A new company with just two employees, Sanchez and Patel, rents a floor of a building with 12 offices. How many ways are there to assign different offices to these two employees?

Solution: The procedure of assigning offices to these two employees consists of assigning an office to Sanchez, which can be done in 12 ways, then assigning an office to Patel different from the office assigned to Sanchez, which can be done in 11 ways. By the product rule, there are $12 \cdot 11 = 132$ ways to assign offices to these two employees. ∎

EXAMPLE. 2

The chairs of an auditorium are to be labeled with an uppercase English letter followed by a positive integer not exceeding 100. What is the largest number of chairs that can be labeled differently?

Solution: The procedure of labeling a chair consists of two tasks, namely, assigning to the seat one of the 26 uppercase English letters, and then assigning to it one of the 100 possible integers. The product rule shows that there are $26 \cdot 100 = 2600$ different ways that a chair can be labeled. Therefore, the largest number of chairs that can be labeled differently is 2600. ∎

EXAMPLE. 3

There are 32 computers in a data center in the cloud. Each of these computers has 24 ports. How many different computer ports are there in this data center?

Solution: The procedure of choosing a port consists of two tasks, first picking a computer and then picking a port on this computer. Because there are 32 ways to choose the computer and 24 ways to choose the port no matter which computer has been selected, the product rule shows that there are $32 \cdot 24 = 768$ ports. ∎

An extended version of the product rule is often useful. Suppose that a procedure is carried out by performing the tasks $T_1, T_2, \ldots, T_m$ in sequence. If each task $T_i,\ i = 1, 2, \ldots, n$, can be done in ni ways, regardless of how the previous tasks were done, then there are $n_1 \cdot n_2 \cdot \ldots \cdot n_m$ ways to carry out the procedure. This version of the product rule can be proved by mathematical induction from the product rule for two tasks.

EXAMPLE. 4

How many different bit strings of length seven are there?

$\boxed{\textit{Solution:}}$ Each of the seven bits can be chosen in two ways, because each bit is either 0 or 1. Therefore, the product rule shows there are a total of $2^7 = 128$ different bit strings of length seven. ∎

EXAMPLE. 5

How many different license plates can be made if each plate contains a sequence of three uppercase English letters followed by three digits (and no sequences of letters are prohibited, even if they are obscene)?

$\boxed{\textit{Solution:}}$ There are 26 choices for each of the three uppercase English letters and 10 choices for each of the three digits. Hence, by the product rule there are a total of $26 \cdot 26 \cdot 26 \cdot 10 \cdot 10 \cdot 10 = 17,576,000$ possible license plates. ∎

EXAMPLE. 6

Counting Functions How many functions are there from a set with m elements to a set with n elements?

$\boxed{\textit{Solution:}}$ A function corresponds to a choice of one of the n elements in the codomain for each of the m elements in the domain. Hence, by the product rule there are $n \cdot n \cdot \ldots \cdot n = n^m$ functions from a set with m elements to one with n elements. For example, there are $5^3 = 125$ different functions from a set with three elements to a set with five elements. ∎

EXAMPLE. 7

Counting One-to-One Functions How many one-to-one functions are there from a set with m elements to one with n elements?

$\boxed{\textit{Solution:}}$ First note that when $m > n$ there are no one-to-one functions from a set with m elements to a set with n elements.

Now let $m \leq n$. Suppose the elements in the domain are $a_1, a_2, \ldots, a_m$. There are n ways to choose the value of the function at a_1. Because the function is one-to-one, the value of the function at a_2 can be picked in $n-1$ ways (because the value used for a_1 cannot be used again). In general, the value of the function at a_k can be chosen in $n-k+1$ ways. By the product rule, there are $n(n-1)(n-2)\ldots(n-m+1)$ one-to-one functions from a set with m elements to one with n elements.

For example, there are $5 \cdot 4 \cdot 3 = 60$ one-to-one functions from a set with three elements to a set with five elements. ∎

EXAMPLE. 8

The Telephone Numbering Plan The *North American numbering plan (NANP)* specifies the format of telephone numbers in the U.S., Canada, and many other parts of North America. A telephone number in this plan consists of 10 digits, which are split into a three-digit area code, a three-digit office code, and a four-digit station code. Because of signaling considerations, there are certain restrictions on some of these digits. To specify the allowable format, let X denote a digit that can take any of the values 0 through 9, let N denote a digit that can take any of the values 2 through 9, and let Y denote a digit that must be a 0 or a 1. Two numbering plans, which will be called the old plan, and the new plan, will be discussed. (The old plan, in use in the 1960s, has been replaced by the new plan, but the recent rapid growth in demand for new numbers for mobile phones and devices will eventually make even this new plan obsolete. In this example, the letters used to represent digits follow the conventions of the *North American Numbering Plan.*) As will be shown, the new plan allows the use of more numbers.

In the old plan, the formats of the area code, office code, and station code are NYX, NNX, and $XXXX$, respectively, so that telephone numbers had the form $NYX - NNX - XXXX$. In the new plan, the formats of these codes are NXX, NXX, and $XXXX$, respectively, so that telephone numbers have the form $NXX - NXX - XXXX$. How many different North American telephone numbers are possible under the old plan and under the new plan?

Solution: By the product rule, there are $8 \cdot 2 \cdot 10 = 160$ area codes with format NYX and $8 \cdot 10 \cdot 10 = 800$ area codes with format NXX. Similarly, by the product rule, there are $8 \cdot 8 \cdot 10 = 640$ office codes with format NNX. The product rule also shows that there are $10 \cdot 10 \cdot 10 \cdot 10 = 10,000$ station codes with format $XXXX$.

Consequently, applying the product rule again, it follows that under the old plan there are

$$160 \cdot 640 \cdot 10,000 = 1,024,000,000$$

different numbers available in North America. Under the new plan, there are

$$800 \cdot 800 \cdot 10,000 = 6,400,000,000$$

different numbers available. ∎

EXAMPLE. 9

What is the value of k after the following code, where $n_1, n_2, \ldots, n_m$ are positive integers, has been executed?

```
$k := 0$
for $i_1 := 1$ to $n_1$
    for $i_2 := 1$ to $n_2$
          .
          .
          .
          .
          for $i_m := 1$ to $n_m$
                  $k := k + 1$
```

Solution: The initial value of k is zero. Each time the nested loop is traversed, 1 is added to k. Let T_i be the task of traversing the ith loop. Then the number of times the loop is traversed is the number of ways to do the tasks $T_1, T_2, \cdots, T_m$. The number of ways to carry out the task $T_j, j = 1, 2, \cdots, m$, is n_j, because the jth loop is traversed once for each integer i_j with $1 \leq i_j \leq n_j$. By the product rule, it follows that the nested loop is traversed $n_1 n_2 \cdots n_m$ times. Hence, the final value of k is $n_1 n_2 \ldots n_m$.
∎

EXAMPLE. 11

Counting Subsets of a Finite Set Use the product rule to show that the number of different subsets of a finite set S is $2^{|S|}$.

Solution: Let S be a finite set. List the elements of S in arbitrary order. Recall from Section 2.2 that there is a one-to-one correspondence between subsets of S and bit strings of length $|S|$. Namely, a subset of S is associated with the bit string with a 1 in the ith position if the ith element in the list is in the subset, and a 0 in this position otherwise. By the product rule, there are $2|S|$ bit strings of length $|S|$. Hence, $|P(S)| = 2^{|S|}$. ∎

The product rule is often phrased in terms of sets in this way: If $A_1, A_2, \cdots, A_m$ are finite sets, then the number of elements in the Cartesian product of these sets is the product of the number of elements in each set. To relate this to the product rule, note that the task of choosing an element in the Cartesian product $A_1 \times A_2 \times \cdots \times A_m$ is done by choosing an element in A_1, an element in $A_2, \cdots$, and an element in A_m. By the product rule it follows that

$$|A_1 \times A_2 \times \cdots \times A_m| = |A_1| \cdot |A_2| \cdot \cdots \cdot A_m|$$

EXAMPLE. 11

DNA and Genomes The hereditary information of a living organism is encoded using deoxyribonucleic acid (DNA), or in certain viruses, ribonucleic acid (RNA). DNA and RNA are extremely complex molecules, with different molecules interacting in a vast variety of ways to enable living process. For our purposes, we give only the briefest description of how DNA and RNA encode genetic information.

DNA molecules consist of two strands consisting of blocks known as nucleotides. Each nucleotide contains subcomponents called **bases**, each of which is adenine (A), cytosine (C), guanine (G), or thymine (T). The two strands of DNA are held together by hydrogen bonds connecting different bases, with A bonding only with T, and C bonding only with G. Unlike DNA, RNA is single stranded, with uracil (U) replacing thymine as a base. So, in DNA the possible base pairs are A-T and C-G, while in RNA they are A-U, and C-G. The DNA of a living creature consists of multiple pieces of DNA forming separate chromosomes. A **gene** is a segment of a DNA molecule that encodes a particular protein. The entirety of genetic information of an organism is called its **genome**.

Sequences of bases in DNA and RNA encode long chains of proteins called amino acids. There are 22 essential amino acids for human beings. We can quickly see that a sequence of at least three bases are needed to encode these 22 different amino acid. First note, that because there are four possibilities for each base in DNA, A, C, G, and T, by the product rule there are $4^2 = 16 < 22$ different sequences of two bases. However, there are $4^3 = 64$ different sequences of three bases, which provide enough different sequences to encode the 22 different amino acids (even after taking into account that several different sequences of three bases encode the same amino acid).

The DNA of simple living creatures such as algae and bacteria have between 10^5 and 10^7 links, where each link is one of the four possible bases. More complex organisms, such as insects, birds, and mammals, have between 10^8 and 10^{10} links in their DNA. So, by the product rule, there are at least 4^{10^5} different sequences of bases in the DNA of simple organisms and at least 4^{10^8} different sequences of bases in the DNA of more complex organisms. These are both Soon it won't be that costly to have your own genetic code found. incredibly huge numbers, which helps explain why there is such tremendous variability among living organisms. In the past several decades techniques have been developed for determining the genome of different organisms. The first step is to locate each gene in the DNA of an organism. The next task, called **gene sequencing**, is the determination of the sequence of links on each gene. (The specific sequence of links on these genes depends on the particular individual representative of a species whose DNA is analyzed.) For example, the human genome includes approximately 23,000 genes, each with 1000 or more links. Gene sequencing techniques take advantage of many recently developed algorithms and are based on numerous new ideas in combinatorics. Many mathematicians and computer scientistswork on problems involving genomes, taking part in the fast moving fields of bioinformatics and computational biology. ▮

We now introduce the sum rule.

THE SUM RULE If a task can be done either in one of n_1 ways or in one of n_2 ways, where none of the set of n_1 ways is the same as any of the set of n_2 ways, then there are $n_1 + n_2$ ways to do the task.

Example 12 illustrates how the sum rule is used.

EXAMPLE. 12

Suppose that either a member of the mathematics faculty or a student who is a mathematics major is chosen as a representative to a university committee. How many different choices are there for this representative if there are 37 members of the mathematics faculty and 83 mathematics majors and no one is both a faculty member and a student?

Solution: There are 37 ways to choose a member of the mathematics faculty and there are 83 ways to choose a student who is a mathematics major. Choosing a member of the mathematics faculty is never the same as choosing a student who is a mathematics major because no one is both a faculty member and a student. By the sum rule it follows that there are $37 + 83 = 120$ possible ways to pick this representative. ∎

We can extend the sum rule to more than two tasks. Suppose that a task can be done in one of n_1 ways, in one of n_2 ways, $\cdots$, or in one of nm ways, where none of the set of n_i ways of doing the task is the same as any of the set of n_j ways, for all pairs i and j with $1 \leq i < j \leq m$. Then the number of ways to do the task is $n_1 + n_2 + \cdots + n_m$. This extended version of the sum rule is often useful in counting problems.

EXAMPLE. 13

A student can choose a computer project from one of three lists. The three lists contain 23, 15, and 19 possible projects, respectively. No project is on more than one list. How many possible projects are there to choose from?
Solution: The student can choose a project by selecting a project from the first list, the second list, or the third list. Because no project is on more than one list, by the sum rule there are $23+15+19 = 57$ ways to choose a project. ∎

EXAMPLE. 14

What is the value of k after the following code, where $n_1, n_2, \cdots, n_m$ are positive integers, has been executed?

```
$k := 0$
for $i_1 := 1$ to $n_1$
    $k := k + 1$
for $i_2 := 1$ to $n_2$
    $k := k + 1$
        ⋮
for $i_m := 1$ to $n_m$
    $k := k + 1$
```

Solution: The initial value of k is zero. This block of code is made up of m different loops. Each time a loop is traversed, 1 is added to k. To determine the value of k after this code has been executed, we need to determine how many times we traverse a loop. Note that there are n_i ways to traverse the ith loop. Because we only traverse one loop at a time, the sum rule shows that the final value of k, which is the number of ways to traverse one of the m loops is $n_1 + n_2 + \cdots + n_m$. ■

The sum rule can be phrased in terms of sets as: If $A_1, A_2, \cdots, A_m$ are pairwise disjoint finite sets, then the number of elements in the union of these sets is the sum of the numbers of elements in the sets. To relate this to our statement of the sum rule, note there are $|A_i|$ ways to choose an element from A_i for $i = 1, 2, \cdots, m$. Because the sets are pairwise disjoint, when we select an element from one of the sets A_i, we do not also select an element from a different set A_j. Consequently, by the sum rule, because we cannot select an element from two of these sets at the same time, the number of ways to choose an element from one of the sets, which is the number of elements in the union, is

$$|A_1 \cup A_2 \cup \cdots \cup A_m| = |A_1| + |A_2| + \cdots + A_m|, \text{when} \quad A_i \cap A_j = \text{for all} \quad i, j$$

This equality applies only when the sets in question are pairwise disjoint. The situation is much more complicated when these sets have elements in common. That situation will be briefly discussed later in this section.

2.1.3 More Complex Counting Problems

Many counting problems cannot be solved using just the sum rule or just the product rule. However, many complicated counting problems can be solved using both of these rules in combination.

EXAMPLE. 15

In a version of the computer language BASIC, the name of a variable is a string of one or two alphanumeric characters, where uppercase and lowercase letters are not distinguished. Moreover, a variable name must begin with a letter and must be different from the five strings of two characters that are reserved for programming use. How many different variable names are there in this version of BASIC?

Solution: Let V equal the number of different variable names in this version of BASIC. Let V_1 be the number of these that are one character long and V_2 be the number of these that are two characters long. Then by the sum rule, $V = V_1 + V_2$. Note that $V_1 = 26$, because a one-character variable name must be a letter. Furthermore, by the product rule there are $26 \cdot 36$ strings of length two that begin with a letter and end with an alphanumeric character. However, five of these are excluded, so $V_2 = 26 \cdot 36?5 = 931$. Hence, there are $V = V_1 + V_2 = 26 + 931 = 957$ different names for variables in this version of BASIC. ∎

EXAMPLE. 16

Each user on a computer system has a password, which is six to eight characters long, where each character is an uppercase letter or a digit. Each password must contain at least one digit. How many possible passwords are there?

Solution:: Let P be the total number of possible passwords, and let P_6, P_7, and P_8 denote the number of possible passwords of length $6, 7$, and 8, respectively. By the sum rule, $P = P_6 + P_7 + P_8$. We will now find P_6, P_7, and P_8. Finding P_6 directly is difficult. To find P_6 it is easier to find the number of strings of uppercase letters and digits that are six characters long, including those with no digits, and subtract from this the number of strings with no digits. By the product rule, the number of strings of six characters is

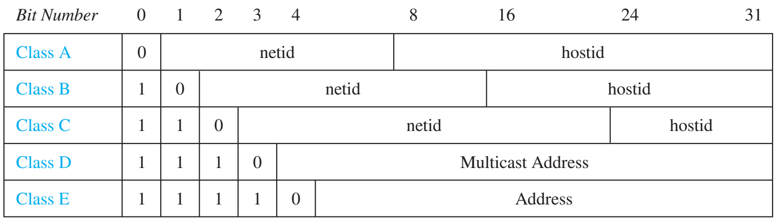


Figure 2.1: Internet addresses (IPv4).

36^6, and the number of strings with no digits is 26^6. Hence,

$$P_6 = 36^6 - 26^6 = 2,176,782,336 - 308,915,776 = 1,867,866,560.$$

Similarly, we have

$$P_7 = 36^7 - 26^7 = 78,364,164,096 - 8,031,810,176 = 70,332,353,920$$

and

$$P_8 = 36^8 - 26^8 = 2,821,109,907,456 - 208,827,064,576 = 2,612,282,842,880.$$

Consequently,

$$P = P_6 + P_7 + P_8 = 2,684,483,063,360.$$

∎

EXAMPLE. 17

Counting Internet Addresses. In the Internet, which is made up of interconnected physical networks of computers, each computer (or more precisely, each network connection of a computer) is assigned an *Internet address*. In Version 4 of the Internet Protocol (IPv4), still in use today, an address is a string of 32 bits. It begins with a *network number (netid)*. The netid is followed by a *host number (hostid)*, which identifies a computer as a member of a particular network.

Three forms of addresses are used, with different numbers of bits used for netids and hostids. **Class A addresses**, used for the largest networks, consist of 0, followed by a 7-bit netid and a 24-bit hostid. **Class B addresses**, used for medium-sized networks, consist of 10, followed by a 14-bit netid and a 16-bit hostid. **Class C addresses**, used for the smallest networks, consist of 110, followed by a 21-bit netid and an 8-bit hostid. There are several restrictions on addresses because of special uses: 1111111 is not available as the netid of a Class A network, and the hostids consisting of all 0s and all 1s are not available for use in any network. A computer on the Internet has either a Class A, a Class B, or a Class C address. (Besides Class A, B, and C addresses, there are also Class D addresses, reserved for use in multicasting when multiple computers are addressed at a single time, consisting of 1110 followed by 28 bits, and Class E addresses, reserved for future use, consisting of 11110 followed by 27 bits. Neither Class D nor Class E addresses are assigned as the IPv4 address of a computer on the Internet.) The lack of available IPv4 address has become a crisis! Figure 1 illustrates IPv4 addressing. (Limitations on the number of Class A and Class B netids have made IPv4 addressing inadequate; IPv6, a new version of IP, uses 128-bit addresses to solve this problem.) How many different IPv4 addresses are available for computers on the Internet?

Solution: Let x be the number of available addresses for computers on the Internet, and let x_A, x_B, and x_C denote the number of Class A, Class B, and Class C addresses available, respectively. By the sum rule, $x = x_A + x_B + x_C$.

To find x_A, note that there are $2^7 - 1 = 127$ Class A netids, recalling that the netid 1111111 is unavailable. For each netid, there are $2^{24} - 2 = 16,777,214$ hostids, recalling that the hostids consisting of all 0s and all 1s are unavailable. Consequently, $x_A = 127 \cdot 16,777,214 = 2,130,706,178$.

To find x_B and x_C, note that there are $2^{14} = 16,384$ Class B netids and $2^{21} = 2,097,152$ Class C netids. For each Class B netid, there are $2^{16} - 2 = 65,534$ hostids, and for each Class C netid, there are $2^8 - 2 = 254$ hostids, recalling that in each network the hostids consisting of all 0s and all 1s are unavailable. Consequently, $x_B = 1,073,709,056$ and $x_C = 532,676,608$. We conclude that the total number of IPv4 addresses available is $x = x_A + x_B + x_C = 2,130,706,178 + 1,073,709,056 + 532,676,608 = 3,737,091,842$.

∎

2.1.4 The Subtraction Rule (Inclusion–Exclusion for Two Sets)

Suppose that a task can be done in one of two ways, but some of the ways to do it are common to both ways. In this situation, we cannot use the sum rule to count the number of ways to do the task. If we add the number of ways to do the tasks in these two ways, we get an overcount of the total number of ways to do it, because the ways to do the task that are common to the two ways are counted twice.

To correctly count the number of ways to do the two tasks, we must subtract the number of ways that are counted twice. This leads us to an important counting rule.

THE SUBTRACTION RULE If a task can be done in either n_1 ways or n_2 ways, then the number of ways to do the task is n_1+n_2 minus the number of ways to do the task that are common to the two different ways.

The subtraction rule is also known as the principle of inclusion–exclusion, especially when it is used to count the number of elements in the union of two sets. Suppose that A_1 and A_2 are sets. Then, there are $|A_1|$ ways to select an element from A_1 and $|A_2|$ ways to select an element from A_2. The number of ways to select an element from A_1 or from A_2, that is, the number of ways to select an element from their union, is the sum of the number of ways to select an element from A_1 and the number of ways to select an element from A_2, minus the number of ways to select an element that is in both A_1 and A_2. Because there are $|A_1 \cup A_2|$ ways to select an element in either A_1 or in A_2, and $|A_1 \cap A_2|$ ways to select an element common to both sets, we have

$$|A_1 \cup A_2| = |A_1| + |A_2| - |A_1 \cap A_2|$$

This is the formula given for the number of elements in the union of two sets. Example 18 illustrates how we can solve counting problems using the subtraction principle.

EXAMPLE. 18

How many bit strings of length eight either start with a 1 bit or end with the two bits 00?

Solution: Figure 2.2 illustrates the three counting problems we need to solve before we can apply the principle of inclusion–exclusion. We can construct a bit string of length eight that either starts with a 1 bit or ends with the two bits 00, by constructing a bit string of length eight beginning with a 1 bit or by constructing a bit string of length eight that ends with the two bits 00. We can construct a bit string of length eight that begins with a 1 in $2^7 = 128$ ways. This follows by the product rule, because the first bit can be chosen in only one way and each of the other seven bits can be chosen in two ways. Similarly, we can construct a bit string of length eight ending with the two bits 00, in $2^6 = 64$ ways. This follows by the product rule, because each of the first six bits can be chosen in two ways and the last two bits can be chosen in only one way. Some of the ways to construct a bit string of length eight starting with a 1 are the same as the ways to construct a bit string of length eight that ends with the two bits 00. There are $2^5 = 32$ ways to construct such a string. This follows by the product rule, because the first bit can be chosen in only one way, each of the second through the sixth bits can be chosen in two ways, and the last two bits can be chosen in one way. Consequently, the number of bit strings of length eight that begin with a 1 or end with a 00, which equals the number of ways to construct a bit string of length eight that begins with a 1 or that ends with 00, equals $128 + 64 - 32 = 160$. ∎

We present an example that illustrates how the formulation of the principle of inclusion– exclusion can be used to solve counting problems.

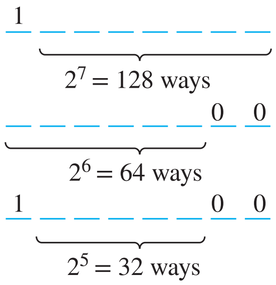


Figure 2.2: 8-Bit strings starting with 1 or ending with 00.

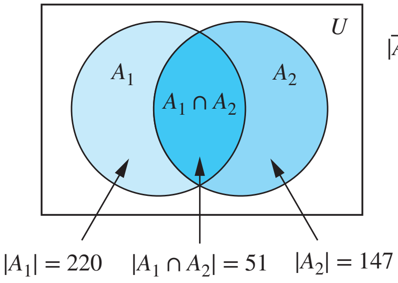
EXAMPLE. 19

A computer company receives 350 applications from college graduates for a job planning a line of new web servers. Suppose that 220 of these applicants majored in computer science, 147 majored in business, and 51 majored both in computer science and in business. How many of these applicants majored neither in computer science nor in business?

Solution: To find the number of these applicants who majored neither in computer science nor in business, we can subtract the number of students who majored either in computer science or in business (or both) from the total number of applicants. Let A_1 be the set of students who majored in computer science and A_2 the set of students who majored in business. Then $A_1 \cup A_2$ is the set of students who majored in computer science or business (or both), and $A_1 \cap A_2$ is the set of students who majored both in computer science and in business. By the subtraction rule the number of students who majored either in computer science or in business (or both) equals

$$|A_1 \cup A_2| = |A_1| + |A_2| - |A_1 \cap A_2| = 220 + 147 - 51 = 316.$$

We conclude that $350 - 316 = 34$ of the applicants majored neither in computer science nor in business. A Venn diagram for this example is shown in Figure 2.3. ∎



$$\begin{aligned}\overline{|A_1 \cup A_2|} &= |U| - |A_1 \cup A_2| \\ &= |U| - (|A_1| + |A_2| - |A_1 \cap A_2|) \\ &= 350 - (220 + 147 - 51) \\ &= 350 - 316 \\ &= 34\end{aligned}$$

Figure 2.3: Applicants who majored in neither computer science nor business.

The subtraction rule, or the principle of inclusion–exclusion, can be generalized to find the number of ways to do one of n different tasks or, equivalently, to find the number of elements in the union of n sets, whenever n is a positive integer. We will study the inclusion–exclusion principle and some of its many applications in next Chapter.

2.1.5 The Division Rule

We have introduced the product, sum, and subtraction rules for counting. You may wonder whether there is also a division rule for counting. In fact, there is such a rule, which can be useful when solving certain types of enumeration problems.

THE DIVISION RULE There are n/d ways to do a task if it can be done using a procedure that can be carried out in n ways, and for every way w, exactly d of the n ways correspond to way w.

We can restate the division rule in terms of sets: "If the finite set A is the union of n pairwise disjoint subsets each with d elements, then $n = |A|/d$."

We can also formulate the division rule in terms of functions: "If f is a function from A to B where A and B are finite sets, and that for every value $y \in B$ there are exactly d values $x \in A$ such that $f(x) = y$ (in which case, we say that f is d-to-one), then $|B| = |A|/d$."

Remark! The division rule comes in handy when it appears that a task can be done in n different ways, but it turns out that for each way of doing the task, there are d equivalent ways of doing it. Under these circumstances, we can conclude that there are n/d inequivalent ways of doing the task.

We illustrate the use of the division rule for counting with two examples.

EXAMPLE. 20

Suppose that an automated system has been developed that counts the legs of cows in a pasture. Suppose that this system has determined that in a farmer's pasture there are exactly 572 legs. How many cows are there is this pasture, assuming that each cow has four legs and that there are no other animals present?

Solution: Let n be the number of cow legs counted in a pasture. Because each cow has four legs, by the division rule we know that the pasture contains

$n/4$ cows. Consequently, the pasture with 572 cow legs has $572/4 = 143$ cows in it. ∎

EXAMPLE. 21

How many different ways are there to seat four people around a circular table, where two seatings are considered the same when each person has the same left neighbor and the same right neighbor?

Solution: We arbitrarily select a seat at the table and label it seat 1. We number the rest of the seats in numerical order, proceeding clockwise around the table.Note that are fourways to select the person for seat 1, three ways to select the person for seat 2, two ways to select the person for seat 3, and one way to select the person for seat 4. Thus, there are $4! = 24$ ways to order the given four people for these seats. However, each of the four choices for seat 1 leads to the same arrangement, as we distinguish two arrangements only when one of the people has a different immediate left or immediate right neighbor. Because there are four ways to choose the person for seat 1, by the division rule there are $24/4 = 6$ different seating arrangements of four people around the circular table. ∎

2.1.6 Tree Diagrams

Counting problems can be solved using tree diagrams. A tree consists of a root, a number of branches leaving the root, and possible additional branches leaving the endpoints of other branches. To use trees in counting, we use a branch to represent each possible choice. We represent the possible outcomes by the leaves, which are the endpoints of branches not having other branches starting at them. Note that when a tree diagram is used to solve a counting problem, the number of choices of which branch to follow to reach a leaf can vary as in Example 22.

EXAMPLE. 22

How many bit strings of length four do not have two consecutive 1s?

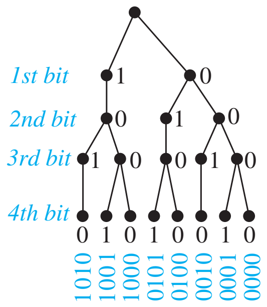


Figure 2.4: Bit strings of length four without consecutive 1s.

Solution: The tree diagram in Figure 2.4 displays all bit strings of length four without two consecutive 1s. We see that there are eight bit strings of length four without two consecutive 1s. ∎

EXAMPLE. 23

A playoff between two teams consists of at most five games. The first team that wins three games wins the playoff. In how many different ways can the playoff occur? Solution: The tree diagram in Figure 2.5 displays all the ways the playoff can proceed, with the winner of each game shown. We see that there are 20 different ways for the playoff to occur. ∎

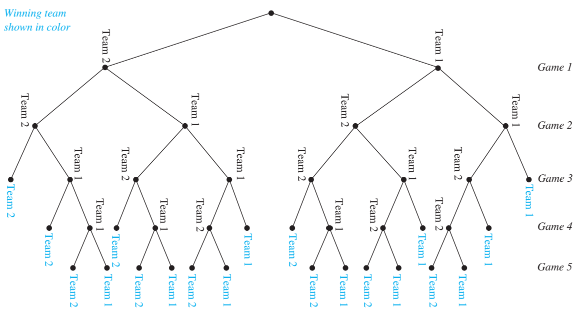


Figure 2.5: Best three games out of five playoffs.

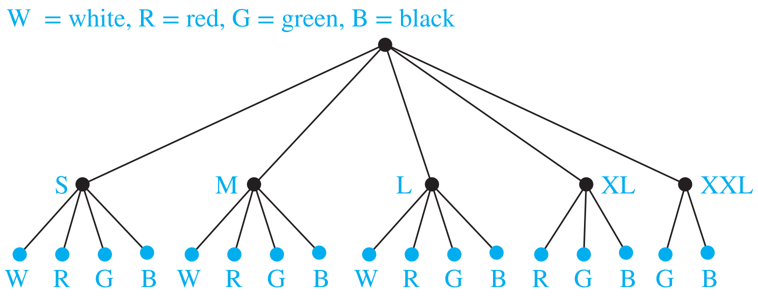

W = white, R = red, G = green, B = black
S
M
L
XL
XXL
W R G B W R G B W R G B R G B G B

Figure 2.6: Counting varieties of T-shirts.

EXAMPLE. 24

Suppose that "I Love New Jersey" T-shirts come in five different sizes: S, M, L, XL, and XXL. Further suppose that each size comes in four colors, white, red, green, and black, except for XL, which comes only in red, green, and black, and XXL, which comes only in green and black. How many different shirts does a souvenir shop have to stock to have at least one of each available size and color of the T-shirt?

Solution: The tree diagram in Figure 2.6 displays all possible size and color pairs. It follows that the souvenir shop owner needs to stock 17 different T-shirts.

2.2 The Pigeonhole Principle

2.2.1 Introduction

Suppose that a flock of 20 pigeons flies into a set of 19 pigeonholes to roost. Because there are 20 pigeons but only 19 pigeonholes, a least one of these 19 pigeonholes must have at least two pigeons in it. To see why this is true, note that if each pigeonhole had at most one pigeon in it, at most 19 pigeons, one per hole, could be accommodated. This illustrates a general principle called the **pigeonhole principle**, which states that if there are more pigeons than pigeonholes, then there must be at least one pigeonhole with at least two pigeons in it (see Figure 2.7). This principle is extremely useful; it applies to much more than pigeons

THEOREM 2.2.1: THE PIGEONHOLE PRINCIPLE

If k is a positive integer and $k+1$ or more objects are placed into k boxes, then there is at least one box containing two or more of the objects.

Proof: We prove the pigeonhole principle using a proof by contraposition. Suppose that none of the k boxes contains more than one object. Then the total number of objects would be at most k. This is a contradiction, because there are at least $k+1$ objects. ∎

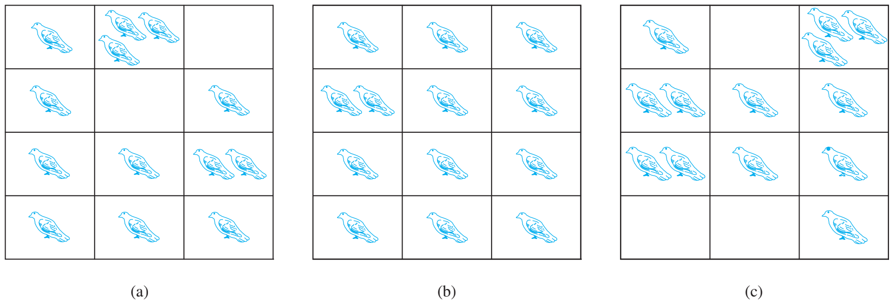


Figure 2.7: There are more pigeons than pigeonholes.

The pigeonhole principle is also called the **Dirichlet drawer principle**, after the nineteenthcentury German mathematician G. Lejeune Dirichlet, who often used this principle in his work. It is an important additional proof technique supplementing those we have developed in earlier chapters. We introduce it in this chapter because of its many important applications to combinatorics.

We will illustrate the usefulness of the pigeonhole principle. We first show that it can be used to prove a useful corollary about functions.

COROLLARY A function f from a set with $k+1$ or more elements to a set with k elements is not one-to-one.

Proof: Suppose that for each element y in the codomain of f we have a box that contains all elements x of the domain of f such that $f(x)=y$. Because the domain contains $k+1$ or more elements and the codomain contains only k elements, the pigeonhole principle tells us that one of these boxes contains two or more elements x of the domain. This means that f cannot be one-to-one. ∎

Examples 1–3 show how the pigeonhole principle is used.

EXAMPLE. 1

Among any group of 367 people, there must be at least two with the same birthday, because there are only 366 possible birthdays. ∎

EXAMPLE. 2

In any group of 27 English words, there must be at least two that begin with the same letter, because there are 26 letters in the English alphabet. ∎

EXAMPLE. 3

How many students must be in a class to guarantee that at least two students receive the same score on the final exam, if the exam is graded on a scale from 0 to 100 points?

Solution: There are 101 possible scores on the final. The pigeonhole principle shows that among any 102 students there must be at least 2 students with the same score. ∎

The pigeonhole principle is a useful tool in many proofs, including proofs of surprising results, such as that given in Example 4.

EXAMPLE. 4

Show that for every integer n there is a multiple of n that has only 0s and 1s in its decimal expansion.

Solution: Let n be a positive integer. Consider the $n+1$ integers 1, 11, 111,$\cdots$, 11 $cdots$1 (where the last integer in this list is the integer with $n+1$ 1s in its decimal expansion). Note that there are n possible remainders when an integer is divided by n. Because there are $n+1$ integers in this list, by the pigeonhole principle there must be two with the same remainder when divided by n. The larger of these integers less the smaller one is a multiple of n, which has a decimal expansion consisting entirely of 0s and 1s. ∎

2.2.2 The Generalized Pigeonhole Principle

The pigeonhole principle states that there must be at least two objects in the same box when there are more objects than boxes. However, even more can be said when the number of objects exceeds a multiple of the number of boxes. For instance, among any set of 21 decimal digits there must be 3 that are the same. This follows because when 21 objects are distributed into 10 boxes, one box must have more than 2 objects.

THEOREM 2.2.2

If N objects are placed into k boxes, then there is at least one box containing at least $\lceil N/k \rceil$ objects.

Proof: We will use a proof by contraposition. Suppose that none of the boxes contains more than $\lceil N/k \rceil - 1$ objects. Then, the total number of objects is at most

$$k\left(\left\lceil \frac{N}{k} \right\rceil - 1\right) < k\left(\left(\frac{N}{k} + 1\right) - 1\right) = N$$

where the inequality $\lceil N/k \rceil < (N/k) + 1$ has been used. Thus, the total number of objects is less than N. This completes the proof by contraposition. ∎

A common type of problem asks for the minimum number of objects such that at least r of these objects must be in one of k boxes when these objects are distributed among the boxes. When we have N objects, the generalized pigeonhole principle tells us there must be at least r objects in one of the boxes as long as $\lceil N/k \rceil \geq r$. The smallest integer N with $N/k > r-1$, namely, $N = k(r-1)+1$, is the smallest integer satisfying the inequality $\lceil N/k \rceil \geq r$. Could a smaller value of N suffice? The answer is no, because if we had $k(r-1)$ objects, we could put $r-1$ of them in each of the k boxes and no box would have at least r objects.

When thinking about problems of this type, it is useful to consider howyou can avoid having at least r objects in one of the boxes as you add successive objects. To avoid adding a rth object to any box, you eventually end up with $r-1$ objects in each box. There is no way to add the next object without putting an rth object in that box.

Examples 5–8 illustrate how the generalized pigeonhole principle is applied.

EXAMPLE. 5

Among 100 people there are at least $\lceil 100/12 \rceil = 9$ who were born in the same month. ∎

EXAMPLE. 6

What is the minimum number of students required in a discrete mathematics class to be sure that at least six will receive the same grade, if there are five possible grades, A, B, C, D, and F?

Solution: The minimum number of students needed to ensure that at least six students receive the same grade is the smallest integer N such that $\lceil N/5 \rceil = 6$. The smallest such integer is $N = 5 \cdot 5 + 1 = 26$. If you have only 25 students, it is possible for there to be five who have received each grade so that no six students have received the same grade. Thus, 26 is the minimum number of students needed to ensure that at least six students will receive the same grade. ∎

EXAMPLE. 7

a) How many cards must be selected from a standard deck of 52 cards to guarantee that at least three cards of the same suit are selected?

b) How many must be selected from a standard deck of 52 cards to guarantee that at least three hearts are selected?

Solution: **a)** Suppose there are four boxes, one for each suit, and as cards are selected they are placed in the box reserved for cards of that suit. Using the generalized pigeonhole principle, we see that if N cards are selected, there is at least one box containing at least $\lceil N/4 \rceil$ cards. Consequently, we know that at least three cards of one suit are selected if $\lceil N/4 \rceil \geq 3$ The smallest integer N such that $\lceil N/4 \rceil \geq 3$ is $N = 2 \cdot 4 + 1 = 9$, so nine cards suffice. Note that if eight cards are selected, it is possible to have two cards of each suit, so more than eight cards are needed. Consequently, nine cards must be selected to guarantee that at least three cards of one suit are chosen. One good way to think about this is to note that after the eighth card is chosen, there is no way to avoid having a third card of some suit.

b) We do not use the generalized pigeonhole principle to answer this question, because we want to make sure that there are three hearts, not just three cards of one suit. Note that in the worst case, we can select all the clubs, diamonds, and spades, 39 cards in all, before we select a single heart. The next three cards will be all hearts, so we may need to select 42 cards to get three hearts. ∎

EXAMPLE. 8

What is the least number of area codes needed to guarantee that the 25 million phones in a state can be assigned distinct 10-digit telephone numbers?

(Assume that telephone numbers are of the form $NXX - NXX - XXXX$, where the first three digits form the area code, N represents a digit from 2 to 9 inclusive, and X represents any digit.)

$\boxed{\textit{Solution:}}$There are eight million different phone numbers of the form $NXX - XXXX$. Hence, by the generalized pigeonhole principle, among 25 million telephones, at least $\lceil 25,000,000/8,000,000 \rceil = 4$ of them must have identical phone numbers. Hence, at least four area codes are required to ensure that all 10-digit numbers are different. ∎

Example 9, although not an application of the generalized pigeonhole principle, makes use of similar principles.

EXAMPLE. 9

Suppose that a computer science laboratory has 15 workstations and 10 servers. A cable can be used to directly connect a workstation to a server. For each server, only one direct connection to that server can be active at any time.We want to guarantee that at any time any set of 10 or fewer workstations can simultaneously access different servers via direct connections. Although we could do this by connecting every workstation directly to every server (using 150 connections), what is the minimum number of direct connections needed to achieve this goal?

$\boxed{\textit{Solution:}}$Suppose thatwe label theworkstations $W_1, W_2, \cdots, W_{15}$ and the servers $S_1, S_2, \cdots, S_{10}$. First, we would like to find a way for there to be far fewer than 150 direct connections between workstations and servers to achieve our goal. One promising approach is to directly connect W_k to S_k for $k = 1, 2, \cdots, 10$ and then to connect each of $W_{11}, W_{12}, W_{13}, W_{14}$, and W_{15} to all 10 servers. This gives us a total of $10 + 5 \cdot 10 = 60$ direct connections. We need to determine whether with this configuration any set of 10 or fewer workstations can simultaneously access different servers. We note that if workstation W_j is included with $1 \leq j \leq 10$, it can access server S_j, and for each workstation W_k with $k \geq 11$ included, there must be a corresponding workstation W_j with $1 \leq j \leq 10$ not included, so W_k can access server S_j. (This follows because there are at least as many available servers S_j as there are workstations W_j with $1 \leq j \leq 10$ not included.) So, any set of 10 or fewer workstations are able to simultaneously access different servers.

But can we use fewer than 60 direct connections? Suppose there are fewer than 60 direct connections between workstations and servers. Then some server would be connected to at most $\lfloor 59/10 \rfloor = 5$ workstations. (If all servers were connected to at least six workstations, there would be at least $6 \cdot 10 = 60$ direct connections.) This means that the remaining nine servers are not enough for the other 10 or more workstations to simultaneously access different servers. Consequently, at least 60 direct connections are needed. It follows that 60 is the answer. ■

2.2.3 Some Elegant Applications of the Pigeonhole Principle

In many interesting applications of the pigeonhole principle, the objects to be placed in boxes must be chosen in a clever way. A few such applications will be described here.

EXAMPLE. 10

During a month with 30 days, a baseball team plays at least one game a day, but no more than 45 games. Show that there must be a period of some number of consecutive days during which the team must play exactly 14 games.

Solution: Let a_j be the number of games played on or before the jth day of the month. Then $a_1, a_2, \cdots, a_{30}$ is an increasing sequence of distinct positive integers, with $1 \leq a_j \leq 45$. Moreover, $a_1 + 14, a_2 + 14, \cdots, a_{30} + 14$ is also an increasing sequence of distinct positive integers, with $15 \leq a_j + 14 \leq 59$.

The 60 positive integers $a_1, a_2, \cdots, a_{30}, a_1 + 14, a_2 + 14, \cdots, a_{30} + 14$ are all less than or equal to 59. Hence, by the pigeonhole principle two of these integers are equal. Because the integers $a_j, j = 1, 2, \cdots, 30$ are all distinct and the integers $a_j + 14, j = 1, 2, \cdots, 30$ are all distinct, there must be indices i and j with $a_i = a_j + 14$. This means that exactly 14 games were played from day $j + 1$ to day i. ■

EXAMPLE. 11

Show that among any $n + 1$ positive integers not exceeding 2n there must be an integer that divides one of the other integers.

Solution: Write each of the $n + 1$ integers $a_1, a_2, \cdots, a_{n+1}$ as a power of 2 times an odd integer. In other words, let $a_j = 2^{k_j} q_j$ for $j = 1, 2, \cdots, n + 1$, where k_j is a nonnegative integer and q_j is odd. The integers $q_1, q_2, \cdots, q_{n+1}$ are all odd positive integers less than $2n$. Because there are only n odd positive integers less than $2n$, it follows from the pigeonhole principle that two of the integers $q_1, q_2, \cdots, q_{n+1}$ must be equal. Therefore, there are distinct integers i and j such that $q_i = q_j$. Let q be the common value of q_i and q_j. Then, $a_i = 2^{k_i} q$ and $a_j = 2^{k_j} q$. It follows that if $k_i < k_j$, then a_i divides a_j; while if $k_i > k_j$, then a_j divides a_i. ∎

A clever application of the pigeonhole principle shows the existence of an increasing or a decreasing subsequence of a certain length in a sequence of distinct integers. We review some definitions before this application is presented. Suppose that $a_1, a_2, \cdots, a_N$ is a sequence of real numbers. A **subsequence** of this sequence is a sequence of the form $a_{i_1}, a_{i_2}, \cdots, a_{i_m}$, where $1 \le i_1 < i_2 < \cdots < i_m \le N$. Hence, a subsequence is a sequence obtained from the original sequence by including some of the terms of the original sequence in their original order, and perhaps not including other terms. A sequence is called **strictly increasing** if each term is larger than the one that precedes it, and it is called strictly decreasing if each term is smaller than the one that precedes it.

THEOREM 2.2.3

Every sequence of $n^2 + 1$ distinct real numbers contains a subsequence of length $n + 1$ that is either strictly increasing or strictly decreasing.

We give an example before presenting the proof of Theorem 2.2.3.

EXAMPLE. 12

The sequence 8, 11, 9, 1, 4, 6, 12, 10, 5, 7 contains 10 terms. Note that $10 = 3^2 + 1$. There are four strictly increasing subsequences of length four, namely, 1, 4, 6, 12; 1, 4, 6, 7; 1, 4, 6, 10; and 1, 4, 5, 7. There is also a strictly decreasing subsequence of length four, namely, 11, 9, 6, 5. ∎

The proof of the theorem will now be given.

Proof: Let $a_1, a_2, \ldots, a_{n^2+1}$ be a sequence of $n^2 + 1$ distinct real numbers. Associate an ordered pair with each term of the sequence, namely, associate (i_k, d_k) to the term a_k, where i_k is the length of the longest increasing subsequence starting at a_k, and d_k is the length of the longest decreasing subsequence starting at a_k.

Suppose that there are no increasing or decreasing subsequences of length $n + 1$. Then i_k and d_k are both positive integers less than or equal to n, for $k = 1, 2, \ cdots, n^2 + 1$. Hence, by the product rule there are n^2 possible ordered pairs for (i_k, d_k). By the pigeonhole principle, two of these $n^2 + 1$ ordered pairs are equal. In other words, there exist terms as and at, with $s < t$ such that $i_s = i_t$ and $d_s = d_t$. We will show that this is impossible. Because the terms of the sequence are distinct, either $a_s < a_t$ or $a_s > a_t$. If $a_s < a_t$, then, because $i_s = i_t$, an increasing subsequence of length $i_t + 1$ can be built starting at a_s, by taking a_s followed by an increasing subsequence of length it beginning at a_t. This is a contradiction. Similarly, if $a_s > a_t$, the same reasoning shows that d_s must be greater than d_t, which is a contradiction. ∎

The final example shows how the generalized pigeonhole principle can be applied to an important part of combinatorics called **Ramsey theory**, after the English mathematician F. P. Ramsey. In general, Ramsey theory deals with the distribution of subsets of elements of sets.

EXAMPLE. 13

Assume that in a group of six people, each pair of individuals consists of two friends or two enemies. Show that there are either three mutual friends or three mutual enemies in the group.

Solution: Let A be one of the six people. Of the five other people in the group, there are either three or more who are friends of A, or three or more who are enemies of A. This follows from the generalized pigeonhole principle, because when five objects are divided into two sets, one of the sets has at least $\lceil 5/2 \rceil = 3$ elements. In the former case, suppose that B, C, and D are friends of A. If any two of these three individuals are friends, then these two and A form a group of three mutual friends. Otherwise, B, C, and D form a set of three mutual enemies. The proof in the latter case, when there are three or more enemies of A, proceeds in a similar manner. ∎

The **Ramsey number** $R(m, n)$, where m and n are positive integers greater than or equal to 2, denotes the minimum number of people at a party such that there are either m mutual friends or n mutual enemies, assuming that every pair of people at the party are friends or enemies. Example 13 shows that $R(3, 3) \leq 6$. We conclude that $R(3, 3) = 6$ because in a group of five people where every two people are friends or enemies, there may not be three mutual friends or three mutual enemies.

It is possible to prove some useful properties about Ramsey numbers, but for the most part it is difficult to find their exact values. Note that by symmetry it can be shown that $R(m, n) = R(n, m)$. We also have $R(2, n) = n$ for every positive integer $n \geq 2$. The exact values of only nine Ramsey numbers $R(m, n)$ with $3 \leq m \leq n$ are known, including $R(4, 4) = 18$. Only bounds are known for many other Ramsey numbers, including $R(5, 5)$, which is known to satisfy $43 \leq R(5, 5) \leq 49$. The reader interested in learning more about Ramsey numbers should consult [2] or [3].

2.3 Permutations and Combinations

2.3.1 Introduction

number of distinct elements of a set of a particular size, where the order of these elements matters. Many other counting problems can be solved by finding the number of ways to select a particular number of elements from a set of a particular size, where the order of the elements selected does not matter. For example, in how many ways can we select three students from a group of five students to stand in line for a picture? How many different committees of three students can be formed from a group of four students? In this section we will develop methods to answer questions such as these.

2.3.2 Permutations

We begin by solving the first question posed in the introduction to this section, as well as related questions.

EXAMPLE. 1

In how many ways can we select three students from a group of five students to stand in line for a picture? In how many ways can we arrange all five of these students in a line for a picture?

Solution: First, note that the order in which we select the students matters. There are five ways to select the first student to stand at the start of the line. Once this student has been selected, there are four ways to select the second student in the line. After the first and second students have been selected, there are three ways to select the third student in the line. By the product rule, there are $5 \cdot 4 \cdot 3 = 60$ ways to select three students from a group of five students to stand in line for a picture.

To arrange all five students in a line for a picture, we select the first student in five ways, the second in four ways, the third in three ways, the fourth in two ways, and the fifth in one way. Consequently, there are $5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 120$ ways to arrange all five students in a line for a picture. ∎

Example 1 illustrates how ordered arrangements of distinct objects can be counted. This leads to some terminology.

A **permutation** of a set of distinct objects is an ordered arrangement of these objects. We Links also are interested in ordered arrangements of some of the elements of a set. An ordered arrangement of r elements of a set is called an ***r*-permutation**.

EXAMPLE. 2

Let $S = \{1, 2, 3\}$. The ordered arrangement 3, 1, 2 is a permutation of S. The ordered arrangement 3, 2 is a 2-permutation of S. ∎

The number of r-permutations of a set with n elements is denoted by $P(n, r)$. We can find $P(n, r)$ using the product rule.

EXAMPLE. 3

Let $S = \{a, b, c\}$. The 2-permutations of S are the ordered arrangements a, b;a, c; b, a; b, c; c, a; and c, b. Consequently, there are six 2-permutations of this set with three elements. There are always six 2-permutations of a set with three elements. There are three ways to choose the first element of the arrangement. There are two ways to choose the second element of the arrangement, because it must be different from the first element. Hence, by the product rule, we see that $P(3, 2) = 3 \cdot 2 = 6$. the first element. By the product rule, it follows that $P(3, 2) = 3 \cdot 2 = 6$. ∎

We now use the product rule to find a formula for $P(n, r)$ whenever n and r are positive integers with $1 \leq r \leq n$.

THEOREM 2.3.1

If n is a positive integer and r is an integer with $1 \leq r \leq n$, then there are $P(n, r) = n(n-1)(n-2)\cdots(n-r+1)$ r-permutations of a set with n distinct elements.

Proof: : We will use the product rule to prove that this formula is correct. The first element of the permutation can be chosen in n ways because there are n elements in the set. There are $n - 1$ ways to choose the second element of the permutation, because there are $n-1$ elements left in the set after using the element picked for the first position. Similarly, there are $n-2$ ways to choose the third element, and so on, until there are exactly $n-(r-1) = n-r+1$ ways to choose the rth element. Consequently, by the product rule, there are $n(n-1)(n-2)\cdots(n-r+1)$ r-permutations of the set. $\blacksquare$

Note that $P(n,0) = 1$ whenever n is a nonnegative integer because there is exactly one way to order zero elements. That is, there is exactly one list with no elements in it, namely the empty list. We now state a useful corollary of Theorem 2.3.1.

COROLLARY If n and r are integers with $0 \le r \le n$, then

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

Proof: When n and r are integers with $1 \le r \le n$, by Theorem 2.3.1 we have

$$P(n,r) = n(n-1)(n-2)\cdots(n-r+1) = \frac{n!}{(n-r)!}$$

Because $\frac{n!}{(n-0)!}$ whenever n is a nonnegative integer, we see that the formula $P(n,r) = \frac{n!}{(n-r)!}$ also holds when $r = 0$. $\blacksquare$

By Theorem 2.3.1 we know that if n is a positive integer, then $P(n,n) = n$. We will illustrate this result with some examples.

EXAMPLE. 4

How many ways are there to select a first-prize winner, a second-prize winner, and a third-prize winner from 100 different people who have entered a contest?

Solution: Because it matters which person wins which prize, the number of ways to pick the three prize winners is the number of ordered selections of three elements from a set of 100 elements, that is, the number of 3-permutations of a set of 100 elements. Consequently, the answer is

$$P(100,3) = 100 \cdot 99 \cdot 98 = 970,200.$$

EXAMPLE. 5

Suppose that there are eight runners in a race. The winner receives a gold medal, the secondplace finisher receives a silver medal, and the third-place finisher receives a bronze medal. How many different ways are there to award these medals, if all possible outcomes of the race can occur and there are no ties?

Solution: The number of different ways to award the medals is the number of 3-permutations of a set with eight elements. Hence, there are $P(8,3) = 8 \cdot 7 \cdot 6 = 336$ possible ways to award the medals. ∎

EXAMPLE. 6

Suppose that a saleswoman has to visit eight different cities. She must begin her trip in a specified city, but she can visit the other seven cities in any order she wishes. How many possible orders can the saleswoman use when visiting these cities?

Solution: The number of possible paths between the cities is the number of permutations of seven elements, because the first city is determined, but the remaining seven can be ordered arbitrarily. Consequently, there are $7! = 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 5040$ ways for the saleswoman to choose her tour. If, for instance, the saleswoman wishes to find the path between the cities with minimum distance, and she computes the total distance for each possible path, she must consider a total of 5040 paths! ∎

EXAMPLE. 7

How many permutations of the letters $ABCDEFGH$ contain the string ABC ?

Solution: Because the letters ABC must occur as a block, we can find the answer by finding the number of permutations of six objects, namely, the block ABC and the individual letters D, E, F, G, and H. Because these six objects can occur in any order, there are $6! = 720$ permutations of the letters $ABCDEFGH$ in which ABC occurs as a block. ∎

2.3.3 Combinations

We now turn our attention to counting unordered selections of objects. We begin by solving a question posed in the introduction to this section of the chapter.

EXAMPLE. 8

How many different committees of three students can be formed from a group of four students?

Solution: To answer this question, we need only find the number of subsets with three elements from the set containing the four students. We see that there are four such subsets, one for each of the four students, because choosing three students is the same as choosing one of the four students to leave out of the group. This means that there are four ways to choose the three students for the committee, where the order in which these students are chosen does not matter. ∎

Example 8 illustrates that many counting problems can be solved by finding the number of subsets of a particular size of a set with n elements, where n is a positive integer.

An r-**combination** of elements of a set is an unordered selection of r elements from the set. Thus, an r-combination is simply a subset of the set with r elements.

EXAMPLE. 9

Let S be the set $\{1, 2, 3, 4\}$. Then $\{1, 3, 4\}$ is a 3-combination from S. (Note that $\{4, 1, 3\}$ is the same 3-combination as $\{1, 3, 4\}$, because the order in which the elements of a set are listed does not matter.) ∎

The number of r-combinations of a set with n distinct elements is denoted by $C(n, r)$. Note that $C(n, r)$ is called a **binomial coefficient**. We will learn where this terminology comes from in next Section.

EXAMPLE. 10

We see that $C(4,2)=6$, because the 2-combinations of $\{a,b,c,d\}$ are the six subsets $\{a,b\}$, $\{a,c\}$, $\{a,d\}$, $\{b,c\}$, $\{b,d\}$, and $\{c,d\}$. ■

We can determine the number of r-combinations of a set with n elements using the formula for the number of r-permutations of a set. To do this, note that the r-permutations of a set can be obtained by first forming r-combinations and then ordering the elements in these combinations. The proof of Theorem 2.3.2, which gives the value of $C(n,r)$, is based on this observation.

THEOREM 2.3.2

The number of r-combinations of a set with n elements, where n is a nonnegative integer and r is an integer with $0 \le r \le n$, equals

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

Proof: The $P(n,r)$ r-permutations of the set can be obtained by forming the $C(n,r)$ r-combinations of the set, and then ordering the elements in each r-combination, which can be done in $P(r,r)$ ways. Consequently, by the product rule,

$$P(n,r) = C(n,r) \cdot P(r,r).$$

This implies that

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

We can also use the division rule for counting to construct a proof of this theorem. Because the order of elements in a combination does not matter and there are $P(r,r)$ ways to order r elements in an r-combination of n elements, each of the $C(n,r)$ r-combinations of a set with n elements corresponds to exactly $P(r,r)$ r-permutations. Hence, by the division rule, $C(n,r) = \frac{P(n,r)}{P(r,r)}$, which implies as before that $C(n,r) = \frac{n!}{r!(n-r)!}$. ■

The formula in Theorem 2.3.2, although explicit, is not helpful when $C(n,r)$ is computed for large values of n and r. The reasons are that it is practical to compute exact values of factorials exactly only for small integer values, and when floating point arithmetic is used, the formula in Theorem 2.3.2 may produce a value that is not an integer. When computing $C(n,r)$, first note that when we cancel out $(n-r)!$ from the numerator and denominator of the expression for $C(n,r)$ in Theorem 2.3.2, we obtain

$$C(n,r) = \frac{n!}{r!(n-r)!} = \frac{(n(n-1)\cdot(n-r+1)}{r!}.$$

Consequently, to compute $C(n,r)$ you can cancel out all the terms in the larger factorial in the denominator from the numerator and denominator, then multiply all the terms that do not cancel in the numerator and finally divide by the smaller factorial in the denominator. [When doing this calculation by hand, instead of by machine, it is also worthwhile to factor out common factors in the numerator $n(n-1)\cdots(n-r+1)$ and in the denominator $r!$.] Note that many computational programs can be used to find $C(n,r)$. [Such functions may be called *choose* (n,k) or *binom* (n,k).]

Example 11 illustrates how $C(n,k)$ is computed when k is relatively small compared to n and when k is close to n. It also illustrates a key identity enjoyed by the numbers $C(n,k)$.

EXAMPLE. 11

How many poker hands of five cards can be dealt from a standard deck of 52 cards? Also, how many ways are there to select 47 cards from a standard deck of 52 cards?

Solution: Because the order in which the five cards are dealt from a deck of 52 cards does not matter, there are

$$C(52,5) = \frac{52!}{5!47!}$$

different hands of five cards that can be dealt. To compute the value of $C(52,5)$, first divide the numerator and denominator by 47! to obtain

$$C(52,5) = \frac{52\cdot 51\cdot 50\cdot 49\cdot 48}{5\cdot 4\cdot 3\cdot 2\cdot 1}.$$

This expression can be simplified by first dividing the factor 5 in the denominator into the factor 50 in the numerator to obtain a factor 10 in the numerator, then dividing the factor 4 in the denominator into the factor 48 in the numerator to obtain a factor of 12 in the numerator then dividing the factor 3 in the denominator into the factor 51 in the numerator to obtain a factor of 17 in the numerator, and finally, dividing the factor 2 in the denominator into the factor 52 in the numerator to obtain a factor of 26 in the numerator. We find that

$$C(52,5) = 26 \cdot 17 \cdot 10 \cdot 49 \cdot 12 = 2,598,960.$$

Consequently, there are 2,598,960 different poker hands of five cards that can be dealt from a standard deck of 52 cards.

Note that there are

$$C(52,47) = \frac{52!}{47!5!}$$

different ways to select 47 cards from a standard deck of 52 cards. We do not need to compute this value because $C(52,47) = C(52,5)$. (Only the order of the factors 5! and 47! is different in the denominators in the formulae for these quantities.) It follows that there are also 2,598,960 different ways to select 47 cards from a standard deck of 52 cards. ∎

In Example 11 we observed that $C(52,5) = C(52,47)$. This is not surprising because selecting five cards out of 52 is the same as selecting the 47 that we leave out. The identity $C(52,5) = C(52,47)$ is a special case of the useful identity for the number of $r-$combinations of a set given in next Corollary.

COROLLARY Let n and r be nonnegative integers with $r \leq n$. Then $C(n,r) = C(n, n-r)$.

Proof: From Theorem 2.3.2 it follows that

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

and

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

Hence, $C(n,r) = C(n,n-r)$. ∎

We can also prove Corollary without relying on algebraic manipulation. Instead, we can use a combinatorial proof. We describe this important type of proof in Definition.

DEFINITION 2.3.1 *A* ***combinatorial proof*** *of an identity is a proof that uses counting arguments to prove that both sides of the identity count the same objects but in different ways or a proof that is based on showing that there is a bijection between the sets of objects counted by the two sides of the identity. These two types of proofs are called* ***double counting proofs*** *and* ***bijective proofs****, respectively.*

Proof: We will use a bijective proof to show that $C(n, r) = C(n, n-r)$ for all integers n and r with $0 \leq r \leq n$. Suppose that S is a set with n elements. The function that maps a subset A of S to A is a bijection between subsets of S with r elements and subsets with $n - r$ elements (as the reader should verify). The identity $C(n, r) = C(n, n-r)$ follows because when there is a bijection between two finite sets, the two sets must have the same number of elements.

Alternatively, we can reformulate this argument as a double counting proof. By definition, the number of subsets of S with r elements equals $C(n, r)$. But each subset A of S is also determined by specifying which elements are not in A, and so are in A. Because the complement of a subset of S with r elements has $n - r$ elements, there are also $C(n, n - r)$ subsets of S with r elements. It follows that $C(n, r) = C(n, n - r)$. ∎

EXAMPLE. 12

How many ways are there to select five players from a 10-member tennis team to make a trip to a match at another school?

Solution: The answer is given by the number of 5-combinations of a set with 10 elements. By Theorem 2, the number of such combinations is

$$C(10, 5) = \frac{10!}{5!5!} = 252.$$

EXAMPLE. 13

A group of 30 people have been trained as astronauts to go on the first mission to Mars. How many ways are there to select a crew of six people to go on this mission (assuming that all crew members have the same job)?

Solution: The number of ways to select a crew of six from the pool of 30 people is the number of 6-combinations of a set with 30 elements, because the order in which these people are chosen does not matter. By Theorem 2.3.2, the number of such combinations is

$$C(30,6) = \frac{30!}{6!24!} = \frac{30 \cdot 29 \cdot 28 \cdot 27 \cdot 26 \cdot 25}{6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1} = 593,775.$$

EXAMPLE. 14

How many bit strings of length n contain exactly r 1s?

Solution: The positions of r 1s in a bit string of length n form an r-combination of the set $\{1, 2, 3, \cdots, n\}$. Hence, there are $C(n, r)$ bit strings of length n that contain exactly r 1s. ∎

EXAMPLE. 15

Suppose that there are 9 faculty members in the mathematics department and 11 in the computer science department. How many ways are there to select a committee to develop a discrete mathematics course at a school if the committee is to consist of three faculty members from the mathematics department and four from the computer science department?

Solution: By the product rule, the answer is the product of the number of 3-combinations of a set with nine elements and the number of 4-combinations of a set with 11 elements. By Theorem 2.3.2, the number of ways to select the committee is

$$C(9,3) \cdot C(11,4) = \frac{9!}{3!6!} \cdot \frac{11!}{4!7!} = 84 \cdot 330 = 27,720.$$

2.4 Binomial Coefficients and Identities

As we remarked in previous Section, the number of r-combinations from a set with n elements is often denoted by $\binom{n}{r}$. This number is also called a **binomial coefficient** because these numbers occur as coefficients in the expansion of powers of binomial expressions such as $(a+b)^n$. We will discuss the binomial theorem, which gives a power of a **binomial expression** as a sum of terms involving binomial coefficients. We will prove this theorem using a combinatorial proof. We will also show how combinatorial proofs can be used to establish some of the many different identities that express relationships among binomial coefficients.

2.4.1 The Binomial Theorem

The binomial theorem gives the coefficients of the expansion of powers of binomial expressions. A **binomial** expression is simply the sum of two terms, such as $x + y$. (The terms can be products of constants and variables, but that does not concern us here.)

Example 1 illustrates how the coefficients in a typical expansion can be found and prepares us for the statement of the binomial theorem.

EXAMPLE. 1

The expansion of $(x+y)^3$ can be found using combinatorial reasoning instead of multiplying the three terms out. When $(x+y)^3 = (x+y)(x+y)(x+y)$ is expanded, all products of a term in the first sum, a term in the second sum, and a term in the third sum are added. Terms of the form x^3, x^2y, xy^2, and y^3 arise. To obtain a term of the form x^3, an x must be chosen in each of the sums, and this can be done in only one way. Thus, the x^3 term in the product has a coefficient of 1. To obtain a term of the form x^2y, an x must be chosen in two of the three sums (and consequently a y in the other sum). Hence, the number of such terms is the number of 2-combinations of three objects, namely, $\binom{3}{2}$. Similarly, the number of terms of the form xy^2 is the number of ways to pick one of the three sums to obtain an x (and consequently take a y from each of the other two sums). This can be done in $\binom{3}{1}$ ways. Finally, the only way to obtain a y^3 term is to choose the y for each of the three sums in the product, and this can be done in exactly one way. Consequently, it follows that

$$\begin{aligned}(x+y)^3 &= (x+y)(x+y)(x+y) = (xx+xy+yx+yy)(x+y)\\ &= xxx+xxy+xyx+xyy+yxx+yxy+yyx+yyy\\ &= x^3+3x^2y+3xy^2+y^3 \quad ∎\end{aligned}$$

We now state the binomial theorem.

THEOREM 2.4.1: THEBINOMIAL THEOREM

Let x and y be variables, and let n be a nonnegative integer. Then

$$\begin{aligned}(x+y)^n &= \sum_{j=0}^{n}\binom{n}{j}x^{n-j}y^j =\\ &= \binom{n}{0}x^n+\binom{n}{1}x^{n-1}y+\cdots+\binom{n}{n-1}xy^{n-1}+\binom{n}{n}y^n\end{aligned}$$

Proof: : We use a combinatorial proof. The terms in the product when it is expanded are of the form $x^{n-j}y^j$ for $j = 0, 1, 2, \cdots, n$. To count the number of terms of the form $x^{n-j}y^j$, note that to obtain such a term it is necessary to choose $n-j$ xs from the n binomial factors (so that the other j terms in the product are ys). Therefore, the coefficient of $x^{n-j}y^j$ is $\binom{n}{n-j}$, which is equal to $\binom{n}{j}$. This proves the theorem. ∎

Some computational uses of the binomial theorem are illustrated in Examples 2–4.

EXAMPLE. 2

What is the expansion of $(x+y)^4$?

Solution: From the binomial theorem it follows that

$$\begin{aligned}(x+y)^4 &= \sum_{j=0}^{4}\binom{4}{j}x^{4-j}y^j =\\ &= \binom{4}{0}x^4+\binom{4}{1}x^3y+\binom{4}{2}x^2y^2+\binom{4}{3}xy^3+\binom{4}{4}y^4 \quad ∎\\ &= x^4+4x^3y+6x^2y^2+4xy^3+y^4\end{aligned}$$

EXAMPLE. 3

What is the coefficient of $x^{12}y^{13}$ in the expansion of $(x+y)^{25}$?

Solution: From the binomial theorem it follows that this coefficient is

$$\binom{25}{13} = \frac{25!}{13!12!} = 5.200.300 \quad \blacksquare$$

EXAMPLE. 4

What is the coefficient of $x^{12}y^{13}$ in the expansion of $(2x-3y)^{25}$?

Solution: First, note that this expression equals $(2x+(-3y))^{25}$. By the binomial theorem, we have

$$(2x+(-3y))^{25} = \sum_{j=0}^{25} \binom{25}{j} (2x)^{25-j}(-3y)^j$$

Consequently, the coefficient of $x^{12}y^{13}$ in the expansion is obtained when $j = 13$, namely,

$$\binom{25}{13} 2^{12}(-3)^{13} = -\frac{25!}{13!12!} 2^{12}3^{13}.$$

Note that another way to find the solution is to first use the binomial theorem to see that

$$(u+v)^{25} = \sum_{j=0}^{25} \binom{25}{j} u^{25-j}v^j$$

Setting $u = 2x$ and $v = -3y$ in this equation yields the same result. $\blacksquare$

We can prove some useful identities using the binomial theorem, as Corollaries 1, 2, and 3 demonstrate.

COROLLARY 1. Let n be a nonnegative integer. Then

$$\sum_{k=0}^{n} \binom{n}{k} = 2^n$$

Proof: : Using the binomial theorem with $x = 1$ and $y = 1$, we see that

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

This is the desired result. ∎

Proof: : A set with n elements has a total of 2^n different subsets. Each subset has zero elements, one element, two elements,$\cdots$, or n elements in it. There are $\binom{n}{0}$ subsets with zero elements,$\binom{n}{1}$ subsets with one element, $\binom{n}{2}$ subsets with two elements,$\cdots$, and $\binom{n}{n}$ subsets with n elements. Therefore,

$$\sum_{k=0}^{n}\binom{n}{k}$$

counts the total number of subsets of a set with n elements. By equating the two formulas we have for the number of subsets of a set with n elements, we see that

$$\sum_{k=0}^{n}\binom{n}{k} = 2^n \qquad \blacksquare$$

COROLLARY 2. Let n be a positive integer. Then

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

Proof: : When we use the binomial theorem with $x = -1$ and $y = 1$, we see that

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

This proves the corollary. ∎

Remark! Corollary 2 implies that
$\binom{n}{0}+\binom{n}{2}+\binom{n}{4}+\cdots = \binom{n}{1}+\binom{n}{3}+\binom{n}{5}+\cdots$.

COROLLARY 3. Let n be a nonnegative integer. Then

$$\sum_{k=0}^{n}2^k\binom{n}{k} = 3^n$$

Proof: We recognize that the left-hand side of this formula is the expansion of $(1+2)^n$ provided by the binomial theorem. Therefore, by the binomial theorem, we see that

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

Hence

$$\sum_{k=0}^{n} \binom{n}{k} 2^k = 3^n \qquad \blacksquare$$

2.4.2 Pascal's Identity and Triangle

The binomial coefficients satisfy many different identities. We introduce one of the most important of these now.

THEOREM 2.4.2: PASCAL'S IDENTITY

Let n and k be positive integers with $n \geq k$. Then

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

Proof: We will use a combinatorial proof. Suppose that T is a set containing $n+1$ elements. Let a be an element in T, and let $S = T - \{a\}$. Note that there are $\binom{n+1}{k}$ subsets of T containing k elements. However, a subset of T with k elements either contains a together with $k-1$ elements of S, or contains k elements of S and does not contain a. Because there are $\binom{n}{k-1}$ subsets of $k-1$ elements of S, there are $\binom{n}{k-1}$ subsets of k elements of T that contain a. And there are $\binom{n}{k}$ subsets of k elements of T that do not contain a, because there are $\binom{n}{k}$ subsets of k elements of S. Consequently,

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

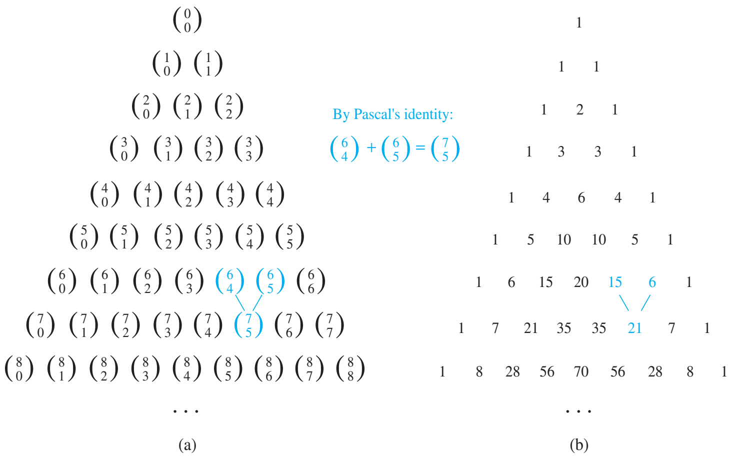


Figure 2.8: Pascal's triangle.

Remark! It is also possible to prove this identity by algebraic manipulation from the formula for $\binom{n}{r}$.

Remark! Pascal's identity, together with the initial conditions $\binom{n}{0} = \binom{n}{n} = 1$ for all integers n, can be used to recursively define binomial coefficients. This recursive definition is useful in the computation of binomial coefficients because only addition, and not multiplication, of integers is needed to use this recursive definition.

Pascal's identity is the basis for a geometric arrangement of the binomial coefficients in a triangle, as shown in Figure 2.8.

The nth row in the triangle consists of the binomial coefficients

$$\binom{n}{k}, k = 0, 1, \cdots, n.$$

This triangle is known as **Pascal's triangle**, named after the French mathematician Blaise Pascal. Pascal's identity shows that when two adjacent binomial coefficients in this triangle are added, the binomial coefficient in the next row between these two coefficients is produced.

Pascal's triangle has a long and ancient history, predating Pascal by many centuries. In the East, binomial coefficients and Pascal's identity were known in the second century B.C.E. by the Indian mathematician Pingala. Later, Indian mathematicians included commentaries relating to Pascal's triangle in their books written in the first half of the last millennium. The Persian mathematician Al-Karaji and the multi-talented Omar Khayy'am wrote about Pasca''s triangle in the eleventh and twelfth centuries, respectively; in Iran, Pascal's triangle is known as Khayy'am's triangle. The triangle was known by the Chinese mathematician Jia Xian in the eleventh century and was written about in the 13th century by Yang Hui; in Chinese Pascal's triangle is often known as Yang Hui's triangle.

In the West, Pascal's triangle appears on the frontispiece of a 1527 book on business calculation written by the German scholar Petrus Apianus. In Italy, Pascal's triangle is called Tartaglia's triangle, after the Italian mathematician Niccol'o Fontana Tartaglia who published the first few rows of the triangle in 1556. In his book Trait'e du triangle arithm?etique, published posthumously 1665, Pascal presented results about Pascal's triangle and used them to solve probability theory problems. Later French mathematicians named this triangle after Pascal; in 1730 Abraham de Moivre coined the name "Pascal's Arithmetic Triangle," which later became "Pascal's Triangle."

2.4.3 Other Identities Involving Binomial Coefficients

We conclude this section with combinatorial proofs of two of the many identities enjoyed by the binomial coefficients.

THEOREM 2.4.3: VANDERMONDE'S IDENTITY

Let m, n, and r be nonnegative integers with r not exceeding either m or n. Then

$$\binom{m+n}{r} = \sum_{k=0}^{r} \binom{m}{r-k}\binom{n}{k}$$

Proof: Suppose that there are m items in one set and n items in a second set. Then the total number of ways to pick r elements from the union of these sets is $\binom{m+n}{r}$.

Another way to pick r elements from the union is to pick k elements from the second set and then $r-k$ elements from the first set, where k is an integer with $0 \le k \le r$. Because there are $\binom{n}{k}$ ways to choose k elements from the second set and $\binom{m}{r-k}$ ways to choose $r-k$ elements from the first set, the product rule tells us that this can be done in $\binom{m}{r-k}\binom{n}{k}$ ways. Hence, the total number of ways to pick r elements from the union also equals $\sum_{k=0}^{r}\binom{m}{r-k}\binom{n}{k}$.

We have found two expressions for the number of ways to pick r elements from the union of a set with m items and a set with n items. Equating them gives us Vandermonde's identity. ∎

Corollary 4 follows from Vandermonde's identity.

COROLLARY 4. If n is a nonnegative integer, then

$$\binom{2n}{n} = \sum_{k=0}^{n}\binom{n}{k}^2$$

Proof: We use Vandermonde's identity with $m = r = n$ to obtain

$$\binom{2n}{n} = \sum_{k=0}^{n}\binom{n}{n-k}\binom{n}{k} = \sum_{k=0}^{n}\binom{n}{k}^2$$

The last equality was obtained using the identity $\binom{n}{k} = \binom{n}{n-k}$. ∎

We can prove combinatorial identities by counting bit strings with different properties, as the proof of Theorem 2.4.4 will demonstrate.

THEOREM 2.4.4

Let n and r be nonnegative integers with $r \le n$. Then

$$\binom{n+1}{r+1} = \sum_{j=r}^{n}\binom{j}{r}$$

Proof: We use a combinatorial proof. By Example 14 in previous Section, the left-hand side, $\binom{n+1}{r+1}$, counts the bit strings of length $n+1$ containing $r+1$ ones.

We show that the right-hand side counts the same objects by considering the cases corresponding to the possible locations of the final 1 in a string with $r+1$ ones. This final one must occur at position $r+1, r+2, \cdots$, or $n+1$. Furthermore, if the last one is the kth bit there must be r ones among the first $k-1$ positions. Consequently, there are $\binom{k-1}{r}$ such bit strings. Summing over k with $r+1 \leq k \leq n+1$, we find that there are

$$\sum_{k=r+1}^{n+1} \binom{k-1}{r} = \sum_{j=r}^{n} \binom{j}{r}$$

bit strings of length n containing exactly $r+1$ ones. (Note that the last step follows from the change of variables $j = k-1$.) Because the left-hand side and the right-hand side count the same objects, they are equal. This completes the proof. ∎

2.5 Generalized Permutations and Combinations

2.5.1 Introduction

In many counting problems, elements may be used repeatedly. For instance, a letter or digit may Links be used more than once on a license plate. When a dozen donuts are selected, each variety can be chosen repeatedly. This contrasts with the counting problems discussed earlier in the chapter where we considered only permutations and combinations in which each item could be used at most once. In this section we will show how to solve counting problems where elements may be used more than once.

Also, some counting problems involve indistinguishable elements. For instance, to count the number of ways the letters of the word $SUCCESS$ can be rearranged, the placement of identical letters must be considered. This contrasts with the counting problems discussed earlier where all elements were considered distinguishable. In this section we will describe how to solve counting problems in which some elements are indistinguishable.

Moreover, in this section we will explain how to solve another important class of counting problems, problems involving counting the ways distinguishable elements can be placed in boxes. An example of this type of problem is the number of different ways poker hands can be dealt to four players.

2.5.2 Permutations with Repetition

Counting permutations when repetition of elements is allowed can easily be done using the product rule, as Example 1 shows.

EXAMPLE. 1

How many strings of length r can be formed from the uppercase letters of the English alphabet?

Solution: By the product rule, because there are 26 uppercase English letters, and because each letter can be used repeatedly, we see that there are

4 apples	4 oranges	4 pears
3 apples, 1 orange	3 apples, 1 pear	3 oranges, 1 apple
3 oranges, 1 pear	3 pears, 1 apple	3 pears, 1 orange
2 apples, 2 oranges	2 apples, 2 pears	2 oranges, 2 pears
2 apples, 1 orange, 1 pear	2 oranges, 1 apple, 1 pear	2 pears, 1 apple, 1 orange

26^r strings of uppercase English letters of length r. ∎

The number of r-permutations of a set with n elements when repetition is allowed is given in Theorem 2.5.1.

THEOREM 2.5.1

The number of r-permutations of a set of n objects with repetition allowed is n^r.

Proof: There are n ways to select an element of the set for each of the r positions in the r-permutation when repetition is allowed, because for each choice all n objects are available. Hence, by the product rule there are n^r r-permutations when repetition is allowed. ∎

2.5.3 Combinations with Repetition

Consider these examples of combinations with repetition of elements allowed.

EXAMPLE. 2

How many ways are there to select four pieces of fruit from a bowl containing apples, oranges, and pears if the order in which the pieces are selected does not matter, only the type of fruit and not the individual piece matters, and there are at least four pieces of each type of fruit in the bowl?

Solution: To solve this problem we list all the ways possible to select the fruit. There are 15 ways:

The solution is the number of 4-combinations with repetition allowed from a three-element set, $\{apple, orange, pear\}$. ∎

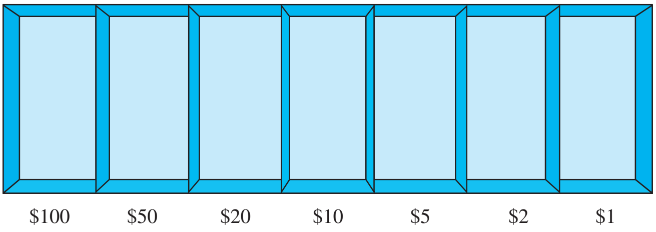


Figure 2.9: Cash box with seven types of bills.

To solve more complex counting problems of this type, we need a general method for counting the r-combinations of an n-element set. In Example 3 we will illustrate such a method.

EXAMPLE. 3

How many ways are there to select five bills from a cash box containing $1 bills, $2 bills, $5 bills, $10 bills, $20 bills, $50 bills, and $100 bills? Assume that the order in which the bills are chosen does not matter, that the bills of each denomination are indistinguishable, and that there are at least five bills of each type.

Solution: Because the order in which the bills are selected does not matter and seven different types of bills can be selected as many as five times, this problem involves counting 5- combinations with repetition allowed from a set with seven elements. Listing all possibilities would be tedious, because there are a large number of solutions. Instead, we will illustrate the use of a technique for counting combinations with repetition allowed.

Suppose that a cash box has seven compartments, one to hold each type of bill, as illustrated in Figure2.9. These compartments are separated by six dividers, as shown in the picture. The choice of five bills corresponds to placing five markers in the compartments holding different types of bills. Figure 2.10 illustrates this correspondence for three different ways to select five bills, where the six dividers are represented by bars and the five bills by stars.

The number of ways to select five bills corresponds to the number of ways to arrange six bars and five stars in a row with a total of 11 positions. Consequently, the number of ways to select the five bills is the number of ways to select the positions of the five stars from the 11 positions. This corresponds to the number of unordered selections of 5 objects from a set of 11 objects,

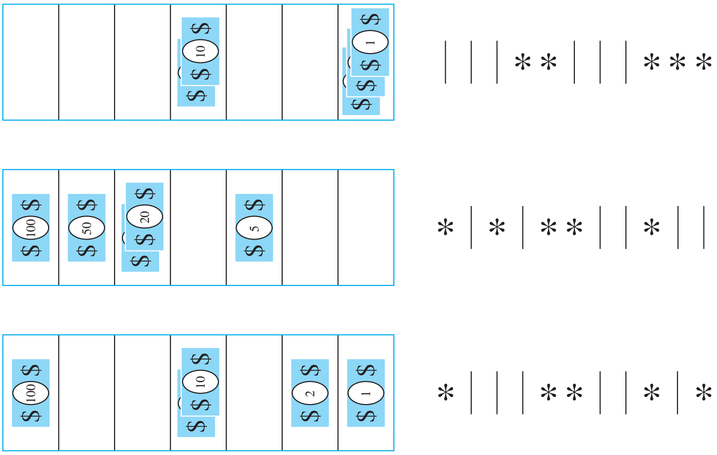


Figure 2.10: Examples of ways to select five bills.

which can be done in $C(11,5)$ ways. Consequently, there are

$$C(11,5) = \frac{11!}{5!6!} = 462$$

ways to choose five bills from the cash box with seven types of bills. ∎

Theorem 2.5.2 generalizes this discussion.

THEOREM 2.5.2

There are $C(n+r-1,r) = C(n+r-1,n-1)$ r-combinations from a set with n elements when repetition of elements is allowed.

Proof: Each r-combination of a set with n elements when repetition is allowed can be represented by a list of $n-1$ bars and r stars. The $n-1$ bars are used to mark off n different cells, with the ith cell containing a star for each time the ith element of the set occurs in the combination. For instance, a 6-combination of a set with four elements is represented with three bars and six stars. Here

$$\star\star|\star| \quad |\star\star\star$$

represents the combination containing exactly two of the first element, one of the second element, none of the third element, and three of the fourth element of the set. As we have seen, each different list containing $n-1$ bars and r stars corresponds to an rcombination of the set with n elements, when repetition is allowed. The number of such lists is $C(n-1+r, r)$, because each list corresponds to a choice of the r positions to place the r stars from the $n-1+r$ positions that contain r stars and $n-1$ bars. The number of such lists is also equal to $C(n-1+r, n-1)$, because each list corresponds to a choice of the $n-1$ positions to place the $n-1$ bars. ∎

Examples 4–6 show how Theorem 2.5.2 is applied.

EXAMPLE. 4

Suppose that a cookie shop has four different kinds of cookies. How many different ways can six cookies be chosen? Assume that only the type of cookie, and not the individual cookies or the order in which they are chosen, matters.

Solution: The number of ways to choose six cookies is the number of 6-combinations of a set with four elements. From Theorem 2.5.2 this equals $C(4+6-1, 6) = C(9, 6)$. Because

$$C(9,6) = C(9,3) = \frac{9 \cdot 8 \cdot 7}{1 \cdot 2 \cdot 3} = 84,$$

there are 84 different ways to choose the six cookies. ∎

Theorem 2.5.2 can also be used to find the number of solutions of certain linear equations where the variables are integers subject to constraints. This is illustrated by Example 5.

EXAMPLE. 5

How many solutions does the equation

$$x_1 + x_2 + x_3 = 11$$

have, where x_1, x_2, and x_3 are nonnegative integers?

Solution: To count the number of solutions, we note that a solution corresponds to a way of selecting 11 items from a set with three elements so that

x_1 items of type one, x_2 items of type two, and x_3 items of type three are chosen. Hence, the number of solutions is equal to the number of 11-combinations with repetition allowed from a set with three elements. From Theorem 2.5.2 it follows that there are

$$C(3+11-1,11)=C(13,11)=C(13,2)=\frac{13\cdot 12}{1\cdot 2}=78$$

solutions.

The number of solutions of this equation can also be found when the variables are subject to constraints. For instance, we can find the number of solutions where the variables are integers with $x_1 \geq 1, x_2 \geq 2$, and $x_3 \geq 3$. A solution to the equation subject to these constraints corresponds to a selection of 11 items with x_1 items of type one, x_2 items of type two, and x_3 items of type three, where, in addition, there is at least one item of type one, two items of type two, and three items of type three. So, a solution corresponds to a choice of one item of type one, two of type two, and three of type three, together with a choice of five additional items of any type. By Theorem 2.5.2 this can be done in

$$C(3+5-1,5)=C(7,5)=C(7,2)=\frac{7\cdot 6}{1\cdot 2}=21$$

ways. Thus, there are 21 solutions of the equation subject to the given constraints. ∎

Example 6 shows how counting the number of combinations with repetition allowed arises in determining the value of a variable that is incremented each time a certain type of nested loop is traversed.

EXAMPLE. 6

What is the value of k after the following pseudocode has been executed?

```
$k := 0$
for $i_1 := 1$ to $n$
    for $i_2 := 1$ to $i_1$
          .
          .
          .
          for $i_m := 1$ to $i_{m-1}$
                $k := k+1$
```

Table 2.1: Combinations and PermutationsWith andWithout Repetition.

Type	Repetition Allowed?	Formula
r-permutations	No	$\frac{n!}{(n-r)!}$
r-combinations	No	$\frac{n!}{r!(n-r)!}$
r-permutations	Yes	n^r
r-combinations	Yes	$\frac{(n+r-1)!}{r!(n-1)!}$

Solution: Note that the initial value of k is 0 and that 1 is added to k each time the nested loop is traversed with a sequence of integers $i_1, i_2, \cdot, i_m$ such that

$$1 \le i_m \le i_{m-1} \le \cdots \le i_1 \le n$$

The number of such sequences of integers is the number of ways to choose m integers from $\{1, 2, \cdots, n\}$, with repetition allowed. (To see this, note that once such a sequence has been selected, if we order the integers in the sequence in nondecreasing order, this uniquely defines an assignment of $i_m, i_{m-1}, \cdots, i_1$. Conversely, every such assignment corresponds to a unique unordered set.) Hence, from Theorem 2.5.2 , it follows that $k = C(n + m - 1, m)$ after this code has been executed. ∎

The formulae for the numbers of ordered and unordered selections of r elements, chosen with and without repetition allowed from a set with n elements, are shown in Table 2.1.

2.5.4 Permutations with Indistinguishable Objects

Some elements may be indistinguishable in counting problems. When this is the case, care must be taken to avoid counting things more than once. Consider Example 7.

EXAMPLE. 7

How many different strings can be made by reordering the letters of the word SUCCESS?

Solution: Because some of the letters of SUCCESS are the same, the answer is not given by the number of permutations of seven letters. This word contains three Ss, two Cs, one U, and one E. To determine the number of different strings that can be made by reordering the letters, first note that the three Ss can be placed among the seven positions in $C(7,3)$ different ways, leaving four positions free. Then the two Cs can be placed in $C(4,2)$ ways, leaving two free positions. The U can be placed in $C(2,1)$ ways, leaving just one position free. Hence E can be placed in $C(1,1)$ way. Consequently, from the product rule, the number of different strings that can be made is

$$C(7,3)C(4,2)C(2,1)C(1,1) = \frac{7!}{3!4!} \cdot \frac{4!}{2!2!} \cdot \frac{2!}{1!1!} \cdot \frac{1!}{1!0!} = 420. \quad \blacksquare$$

We can prove Theorem2.5.3 using the same sort of reasoning as in Example 7.

THEOREM 2.5.3

The number of different permutations of n objects, where there are n_1 indistinguishable objects of type 1, n_2 indistinguishable objects of type 2,$\cdots$, and n_k indistinguishable objects of type k, is

$$\frac{n!}{n1!n2!\cdots n_k!}$$

.

Proof: : To determine the number of permutations, first note that the n_1 objects of type one can be placed among the n positions in $C(n, n_1)$ ways, leaving $n - n_1$ positions free. Then the objects of type two can be placed in $C(n - n_1, n_2)$ ways, leaving $n - n_1 - n_2$ positions free. Continue placing the objects of type three,$\cdots$, type $k - 1$, until at the last stage, n_k objects of type k can be placed in $C(n - n_1 - n_2 - \cdots - n_{k-1}, n_k)$ ways. Hence, by the product rule, the total number of different permutations is

$$\begin{aligned} C(n, n_1) \quad & C(n - n_1, n_2) \cdots C(n - n_1 - \cdots - n_{k-1}, n_k) = \\ & = \tfrac{n!}{n1!(n-n_1)!} \tfrac{(n-n_1)!}{n_2!(n-n_1-n_2)!} \cdots \tfrac{(n?n_1-\cdots?n_{k-1})!}{nk!0!} \quad \blacksquare \\ & = \tfrac{n!}{n_1!n_2!\cdots n_k!} \end{aligned}$$

2.5.5 Distributing Objects into Boxes

Many counting problems can be solved by enumerating the ways objects can be placed into boxes (where the order these objects are placed into the boxes does not matter). The objects can be either *distinguishable*, that is, different from each other, or *indistinguishable*, that is, considered identical. Distinguishable objects are sometimes said to be *labeled*, whereas indistinguishable objects are said to be *unlabeled*. Similarly, boxes can be *distinguishable*, that is, different, or *indistinguishable*, that is, identical. Distinguishable boxes are often said to be *labeled*, while indistinguishable boxes are said to be *unlabeled*. When you solve a counting problem using the model of distributing objects into boxes, you need to determine whether the objects are distinguishable and whether the boxes are distinguishable. Although the context of the counting problem makes these two decisions clear, counting problems are sometimes ambiguous and it may be unclear which model applies. In such a case it is best to state whatever assumptions you are making and explain why the particular model you choose conforms to your assumptions.

Remark! A **closed formula** is an expression that can be evaluated using a finite number of operations and that includes numbers, variables, and values of functions, where the operations and functions belong to a generally accepted set that can depend on the context.

DISTINGUISHABLE OBJECTS AND DISTINGUISHABLE BOXES

We first consider the case when distinguishable objects are placed into distinguishable boxes. Consider Example 8 in which the objects are cards and the boxes are hands of players.

EXAMPLE. 8

How many ways are there to distribute hands of 5 cards to each of four players from the standard deck of 52 cards?

Solution: We will use the product rule to solve this problem. To begin, note that the first player can be dealt 5 cards in $C(52,5)$ ways. The second player can be dealt 5 cards in $C(47,5)$ ways, because only 47 cards are left. The third player can be dealt 5 cards in $C(42,5)$ ways. Finally, the fourth player can be dealt 5 cards in $C(37,5)$ ways. Hence, the total number of ways to deal four players 5 cards each is

$$C(52,5)C(47,5) \quad C(42,5)C(37,5) = \frac{52!}{47!5!} \cdot \frac{47!}{42!5!} \cdot \frac{42!}{37!5!} \cdot \frac{37!}{32!5!} = \\ = \frac{52!}{5!5!5!5!32!}$$ ∎

Remark! The solution to Example 8 equals the number of permutations of 52 objects, with 5 indistinguishable objects of each of four different types, and 32 objects of a fifth type. This equality can be seen by defining a one-to-one correspondence between permutations of this type and distributions of cards to the players. To define this correspondence, first order the cards from 1 to 52. Then cards dealt to the first player correspond to the cards in the positions assigned to objects of the first type in the permutation.

Example 8 is a typical problem that involves distributing distinguishable objects into distinguishable boxes. The distinguishable objects are the 52 cards, and the five distinguishable boxes are the hands of the four players and the rest of the deck. Counting problems that involve distributing distinguishable objects into boxes can be solved using Theorem 2.5.4.

THEOREM 2.5.4

The number of ways to distribute n distinguishable objects into k distinguishable boxes so that n_i objects are placed into box i, $i = 1, 2, \cdots, k$, equals

$$\frac{n!}{n1!n2!\cdots n_k!}$$

Theorem 2.5.4 can be proved using the product rule.

INDISTINGUISHABLE OBJECTS AND DISTINGUISHABLE BOXES

Counting the number of ways of placing n indistinguishable objects into k distinguishable boxes turns out to be the same as counting the number of n-combinations for a set with k elements when repetitions are allowed. The reason behind this is that there is a one-to-one correspondence between n-combinations from a set with k elements when repetition is allowed and the ways to place n indistinguishable balls into k distinguishable boxes. To set up this correspondence, we put a ball in the ith bin each time the ith element of the set is included in the n-combination.

EXAMPLE. 9

How many ways are there to place 10 indistinguishable balls into eight distinguishable bins?

Solution: The number of ways to place 10 indistinguishable balls into eight bins equals the number of 10-combinations from a set with eight elements when repetition is allowed. Consequently, there are

$$C(8+10-1,10)=C(17,10)=\frac{17!}{10!7!}=19,448. \quad \blacksquare$$

This means that there are $C(n+r-1,n-1)$ ways to place r indistinguishable objects into n distinguishable boxes.

DISTINGUISHABLE OBJECTS AND INDISTINGUISHABLE BOXES Counting the ways to place n distinguishable objects into k indistinguishable boxes is more difficult than counting the ways to place objects, distinguishable or indistinguishable objects, into distinguishable boxes. We illustrate this with an example.

EXAMPLE. 10

How many ways are there to put four different employees into three indistinguishable offices, when each office can contain any number of employees?

$\boxed{\textit{Solution:}}$ We will solve this problem by enumerating all the ways these employees can be placed into the offices. We represent the four employees by A, B, C, and D. First, we note that we can distribute employees so that all four are put into one office, three are put into one office and a fourth is put into a second office, two employees are put into one office and two put into a second office, and finally, two are put into one office, and one each put into the other two offices. Each way to distribute these employees to these offices can be represented by a way to partition the elements A, B, C, and D into disjoint subsets.

We can put all four employees into one office in exactly one way, represented by $\{\{A, B, C, D\}\}$. We can put three employees into one office and the fourth employee into a different office in exactly four ways, represented by $\{\{A, B, C\}, \{D\}\}$, $\{\{A, B, D\}, \{C\}\}$, $\{\{A, C, D\}, \{B\}\}$, and $\{\{B, C, D\}, \{A\}\}$. We can put two employees into one office and two into a second office in exactly three ways, represented by $\{\{A, B\}, \{C, D\}\}$, $\{\{A, C\}, \{B, D\}\}$, and $\{\{A, D\}, \{B, C\}\}$. Finally, we can put two employees into one office, and one each into each of the remaining two offices in six ways, represented by $\{\{A, B\}, \{C\}, \{D\}\}$, $\{\{A, C\}, \{B\}, \{D\}\}$, $\{\{A, D\}, \{B\}, \{C\}\}$,
$\{\{B, C\}, \{A\}, \{D\}\}$, $\{\{B, D\}\}, \{A\}, \{C\}\}$, and $\{\{C, D\}, \{A\}, \{B\}\}$.

Counting all the possibilities, we find that there are 14 ways to put four different employees into three indistinguishable offices. Another way to look at this problem is to look at the number of offices into which we put employees. Note that there are six ways to put four different employees into three indistinguishable offices so that no office is empty, seven ways to put four different employees into two indistinguishable offices so that no office is empty, and one way to put four employees into one office so that it is not empty. ∎

There is no simple closed formula for the number of ways to distribute n distinguishable objects into j indistinguishable boxes. However, there is a formula involving a summation, which we will now describe. Let $S(n, j)$ denote the number of ways to distribute n distinguishable objects into j indistinguishable boxes so that no box is empty. The numbers $S(n, j)$ are called **Stirling numbers** of the second kind. For instance, Example 10 shows that $S(4, 3) = 6, S(4, 2) = 7$, and $S(4, 1) = 1$. We see that the number of ways to distribute n distinguishable objects into k indistinguishable boxes (where the number of boxes that are nonempty equals $k, k-1, \cdots, 2$, or 1) equals $\sum_{j=1}^{k} S(n, j)$. For instance, following the reasoning in Example 10, the number of ways to distribute four distinguishable objects into three indistinguishable boxes equals $S(4,1) + S(4,2) + S(4,3) = 1 + 7 + 6 = 14$. Using the inclusion–exclusion principle it can be shown that

$$S(n,j) = \frac{1}{j!}\sum_{i=0}^{k}(-1)^i\binom{j}{i}(j-i)^n$$

Consequently, the number ofways to distribute n distinguishable objects into k indistinguishable boxes equals

$$\sum_{j=1}^{k} S(n,j) = \sum_{j=1}^{k}\frac{1}{j!}\sum_{i=0}^{j-1}(-1)^i\binom{j}{i}(j-i)^n$$

Remark! The reader may be curious about the Stirling numbers of the first kind. For the definition of Stirling numbers of the first kind, for more information about Stirling numbers of the second kind, and to learn more about Stirling numbers of the first kind and the relationship between Stirling numbers of the first and second kind, see combinatorics textbooks such as [4], [5], and Chapter 6 in [2].

INDISTINGUISHABLE OBJECTS AND INDISTINGUISHABLE BOXES

Some counting problems can be solved by determining the number of ways to distribute indistinguishable objects into indistinguishable boxes. We illustrate this principle with an example.

EXAMPLE. 11

How many ways are there to pack six copies of the same book into four identical boxes, where a box can contain as many as six books?

Solution: We will enumerate all ways to pack the books. For each way to pack the books, we will list the number of books in the box with the largest number of books, followed by the numbers of books in each box containing at least one book, in order of decreasing number of books in a box. The ways we can pack the books are

6	3,2,1
5,1	3,1,1,1
4,2	2,2,2
4,1,1	2,2,1,1
3,3	

For example, 4, 1, 1 indicates that one box contains four books, a second box contains a single book, and a third box contains a single book (and the fourth box is empty). We conclude that there are nine allowable ways to pack the books, because we have listed them all. ∎

Observe that distributing n indistinguishable objects into k indistinguishable boxes is the same as writing n as the sum of at most k positive integers in nonincreasing order. If $a_1 + a_2 + \cdots + a_j = n$, where $a_1, a_2, \cdots, a_j$ are positive integers with $a_1 \geq a2 \geq \cdots \geq a_j$, we say that $a_1, a_2, \cdots, a_j$ is a **partition** of the positive integer n into j positive integers. We see that if $p_k(n)$ is the number of partitions of n into at most k positive integers, then there are $p_k(n)$ ways to distribute n indistinguishable objects into k indistinguishable boxes. No simple closed formula exists for this number.

2.6 Generating Permutations and Combinations

2.6.1 Introduction

previous sections of this chapter, but sometimes permutations or combinations need to be generated, not just counted. Consider the following three problems. First, suppose that a salesperson must visit six different cities. In which order should these cities be visited to minimize total travel time? One way to determine the best order is to determine the travel time for each of the $6! = 720$ different orders in which the cities can be visited and choose the one with the smallest travel time. Second, suppose we are given a set of six positive integers and wish to find a subset of them that has 100 as their sum, if such a subset exists. One way to find these numbers is to generate all $2^6 = 64$ subsets and check the sum of their elements. Third, suppose a laboratory has 95 employees. A group of 12 of these employees with a particular set of 25 skills is needed for a project. (Each employee can have one or more of these skills.) One way to find such a set of employees is to generate all sets of 12 of these employees and check whether they have the desired skills. These examples show that it is often necessary to generate permutations and combinations to solve problems.

2.6.2 Generating Permutations

Any set with n elements can be placed in one-to-one correspondence with the set $\{1, 2, 3, \cdots, n\}$. We can list the permutations of any set of n elements by generating the permutations of the n smallest positive integers and then replacing these integers with the corresponding elements. Many different algorithms have been developed to generate the $n!$ permutations of this set. We will describe one of these that is based on the lexicographic (or dictionary) ordering of the set of permutations of $\{1, 2, 3, \cdots, n\}$. In this ordering, the permutation $a_1a_2 \cdots a_n$ precedes the permutation of $b_1b_2 \cdots b_n$, if for some k, with $1 \leq k \leq n$, $a_1 = b_1, a_2 = b_2, \cdots, a_{k-1} = b_{k-1}$, and $a_k < b_k$. In other words, a permutation of the set of the n smallest positive integers precedes (in lexicographic order) a second permutation if the number in this permutation in the first position where the two permutations disagree is smaller than the number in that position in the second permutation.

EXAMPLE. 1

The permutation 23415 of the set $\{1, 2, 3, 4, 5\}$ precedes the permutation 23514, because these permutations agree in the first two positions, but the number in the third position in the first permutation, 4, is smaller than the number in the third position in the second permutation, 5. Similarly, the permutation 41532 precedes 52143. ∎

An algorithm for generating the permutations of $\{1, 2, \cdots, n\}$ can be based on a procedure that constructs the next permutation in lexicographic order following a given permutation $a_1a_2\cdots a_n$. We will show how this can be done. First, suppose that $a_{n-1} < a_n$. Interchange a_{n-1} and an to obtain a larger permutation. No other permutation is both larger than the original permutation and smaller than the permutation obtained by interchanging a_{n-1} and a_n. For instance, the next larger permutation after 234156 is 234165. On the other hand, if $a_{n-1} > a_n$, then a larger permutation cannot be obtained by interchanging these last two terms in the permutation. Look at the last three integers in the permutation. If $a_{n-2} < a_{n-1}$, then the last three integers in the permutation can be rearranged to obtain the next largest permutation. Put the smaller of the two integers a_{n-1} and an that is greater than a_{n-2} in position $n-2$. Then, place the remaining integer and a_{n-2} into the last two positions in increasing order. For instance, the next larger permutation after 234165 is 234516.

On the other hand, if $a_{n-2} > a_{n-1}$ (and $a_{n-1} > a_n$), then a larger permutation cannot be obtained by permuting the last three terms in the permutation. Based on these observations, a general method can be described for producing the next larger permutation in increasing order following a given permutation $a_1a_2\cdots a_n$. First, find the integers a_j and a_{j+1} with $a_j < a_{j+1}$ and

$$a_{j+1} > a_{j+2} > \cdots > a_n,$$

that is, the last pair of adjacent integers in the permutation where the first integer in the pair is smaller than the second. Then, the next larger permutation in lexicographic order is obtained by putting in the jth position the least integer among $a_{j+1}, a_{j+2}, \cdots$, and an that is greater than a_j and listing in increasing order the rest of the integers $a_j, a_{j+1}, \cdots$, an in positions $j + 1$ to n. It is easy to see that there is no other permutation larger than the permutation $a_1a_2 \cdots a_n$ but smaller than the new permutation produced. (The verification of this fact is left as an exercise for the reader.)

EXAMPLE. 2

What is the next permutation in lexicographic order after 362541?

Solution: The last pair of integers a_j and a_{j+1} where $a_j < a_{j+1}$ is $a_3 = 2$ and $a_4 = 5$. The least integer to the right of 2 that is greater than 2 in the permutation is $a_5 = 4$. Hence, 4 is placed in the third position. Then the integers 2, 5, and 1 are placed in order in the last three positions, giving 125 as the last three positions of the permutation. Hence, the next permutation is 364125. ∎

To produce the $n!$ permutations of the integers $1, 2, 3, \cdots, n$, begin with the smallest permutation in lexicographic order, namely, $123 \cdots n$, and successively apply the procedure described for producing the next larger permutation of $n! - 1$ times. This yields all the permutations of the n smallest integers in lexicographic order.

EXAMPLE. 3

Generate the permutations of the integers 1, 2, 3 in lexicographic order.

Solution: Begin with 123. The next permutation is obtained by interchanging 3 and 2 to obtain 132. Next, because $3 > 2$ and $1 < 3$, permute the three integers in 132. Put the smaller of 3 and 2 in the first position, and then put 1 and 3 in increasing order in positions 2 and 3 to obtain 213. This is followed by 231, obtained by interchanging 1 and 3, because $1 < 3$. The next larger permutation has 3 in the first position, followed by 1 and 2 in increasing order, namely, 312. Finally, interchange 1 and 2 to obtain the last permutation, 321. We have generated the permutations of 1, 2, 3 in lexicographic order. They are 123, 132, 213, 231, 312, and 321 ∎

Algorithm 1 displays the procedure for finding the next permutation in lexicographic order after a permutation that is not $nn-1n-2\cdots 21$, which is the largest permutation.

ALGORITHM 1 Generating the Next Permutation in Lexicographic Order.

```
procedure next permutation($a_1a_2\cdots a_n$: permutation of
      $\{1,2,\cdots,n\}$ not equal to $nn-1\cdots 21$)
$j := n-1$
while $a_j > a_{j+1}$
   $j := j-1$
$\{j$ is the largest subscript with $a_j < a_{j+1}\}$
$k := n$
while $a_j > a_k$
   $k := k-1$
$\{a_k$ is the smallest integer greater than $a_j$ to the right of $a_j\}$
interchange $a_j$ and $a_k$
$r := n$
$s := j+1$
while $r > s$
    interchange $a_r$ and $a_s$
   $r := r-1$
   $s := s+1$
{this puts the tail end of the permutation after the $j$th position
in increasing order}
$\{a_1a_2\cdots a_n$ is now the next permutation$\}$
```

2.6.3 Generating Combinations

How can we generate all the combinations of the elements of a finite set? Because a combination is just a subset, we can use the correspondence between subsets of $\{a_1, a_2, \cdots, a_n\}$ and bit strings of length n.

Recall that the bit string corresponding to a subset has a 1 in position k if a_k is in the subset, and has a 0 in this position if a_k is not in the subset. If all the bit strings of length n can be listed, then by the correspondence between subsets and bit strings, a list of all the subsets is obtained.

Recall that a bit string of length n is also the binary expansion of an integer between 0 and $2^n - 1$. The 2^n bit strings can be listed in order of their increasing size as integers in their binary expansions. To produce all binary expansions of length n, start with the bit string $000 \cdots 00$, with n zeros. Then, successively find the next expansion until the bit string $111 \cdots 11$ is obtained. At each stage the next binary expansion is found by locating the first position from the right that is not a 1, then changing all the 1s to the right of this position to 0s and making this first 0 (from the right) a 1.

EXAMPLE. 4

Find the next bit string after 10 0010 0111.

Solution: The first bit from the right that is not a 1 is the fourth bit from the right. Change this bit to a 1 and change all the following bits to 0s. This produces the next larger bit string, 10 0010 1000. ∎

The procedure for producing the next larger bit string after $b_{n-1}b_{n-2} \cdots b_1 b_0$ is given as Algorithm 2.

ALGORITHM 2 Generating the Next Larger Bit String

```
**procedure** *next bit string*($b_{n-1}b_{n-2} \cdots b_1 b_0$ : bit string not equal to $11 \cdots 11$)
$i := 0$
**while** $b_i = 1$
    $b_i := 0$
    $i := i + 1$
$b_i := 1$
{$b_{n-1}b_{n-2} \cdots b_1 b_0$ is now the next bit string}
```

Next, an algorithm for generating the r-combinations of the set $\{1, 2, 3, \cdots, n\}$ will be given. An r-combination can be represented by a sequence containing the elements in the subset in increasing order. The r-combinations can be listed using lexicographic order on these sequences. In this lexicographic ordering, the first r-combination is $\{1, 2, \cdots, r-1, r\}$ and the last r-combination is $\{n-r+1, n-r+2, \cdots, n-1, n\}$. The next r-combination after $a_1a_2\cdots a_r$ can be obtained in the following way: First, locate the last element a_i in the sequence such that $a_i \neq n-r+i$. Then, replace a_i with a_i+1 and a_j with $a_i+j-i+1$, for $j = i+1, i+2, \cdots, r$. It is left for the reader to show that this produces the next larger r-combination in lexicographic order. This procedure is illustrated with Example 5.

EXAMPLE. 5

Find the next larger 4-combination of the set $\{1, 2, 3, 4, 5, 6\}$ after $\{1, 2, 5, 6\}$.

Solution: The last term among the terms a_i with $a_1 = 1, a_2 = 2, a_3 = 5$, and $a_4 = 6$ such that $a_i \neq 6-4+i$ is $a_2 = 2$. To obtain the next larger 4-combination, increment a_2 by 1 to obtain $a_2 = 3$. Then set $a_3 = 3+1 = 4$ and $a_4 = 3+2 = 5$. Hence the next larger 4-combination is $\{1, 3, 4, 5\}$. ∎

Algorithm 3 displays pseudocode for this procedure.

ALGORITHM 3 Generating the Next r-Combination in Lexicographic Order.

```
procedure next r-combination($\{a_1, a_2, \cdots, a_r\}$ : proper subset of $\{1, 2, \cdots, n\}$ not equal to $\{n-r+1, \cdots, n\}$ with $a_1 < a_2 < \cdots < a_r$)
$i := r$
while $a_i = n-r+i$
   $i := i-1$
$a_i := a_i + 1$
for $j := i+1$ to $r$
   $a_j := a_i + j - i$
{$\{a_1, a_2, \cdots, a_r\}$ is now the next combination}
```

References

1. Rosen K. H. Discrete mathematics and its applications / K. H. Rosen. — McGraw-Hill Education, 2018. — 942 p.

2. J. G. Michaels and K. H. Rosen, Applications of Discrete Mathematics, McGraw-Hill, New York, 1991.

3. Ronald L. Graham, Bruce L. Rothschild, and Joel H. Spencer, Ramsey Theory, 2d ed., Wiley, New York, 1990.

4. M. Bona, Enumerative Combinatorics, McGraw-Hill, New York, 2007.

5. R. A. Brualdi, Introductory Combinatorics, 5th ed., Prentice-Hall, Englewood Cliffs, NJ, 2009.

Електронне навчальне видання

Дворниченко Аліна Василівна,
Лисенко Олександр Володимирович

ДИСКРЕТНА МАТЕМАТИКА ТА ТЕОРІЯ АЛГОРИТМІВ
Конспект лекцій
для студентів спеціальності *113 "Прикладна математика"*
денної форми навчання

У чотирьох частинах
Частина II

(Англійською мовою)

Відповідальний за випуск І. В. Коплик
Редактор А. В. Дворниченко
Комп'ютерне верстання А. В. Дворниченко

Формат 60x84/16. Ум. друк. арк. 8,60. Обл.-вид. арк. 8,42.

Видавець і виготовлювач
Сумський державний університет,
вул. Римського-Корсакова, 2, м. Суми, 40007
Свідоцтво суб'єкта видавничої справи ДК № 3062 від 17.12.2007.