CHAPTER 8
Advanced Counting Techniques

SECTION 8.1 Applications of Recurrence Relations

This section picks up where Section 2.4 left off; recurrence relations were first introduced there. In addition this section is related to Section 5.3, in that recurrence relations are in some sense really recursive or inductive definitions. Many of the exercises in this set provide practice in setting up such relations from a given applied situation. In each problem of this type, ask yourself how the n^{th} term of the sequence can be related to one or more previous terms; the answer is the desired recurrence relation.

Exercise 21 is interesting and challenging, and shows that the inductive step may be quite nontrivial. Exercise 29 deals with onto functions; another—totally different—approach to counting onto functions is given in Section 8.6. Exercises 30–45 deal with additional interesting applications.

1. We want to show that $H_n = 2^n - 1$ is a solution to the recurrence relation $H_n = 2H_{n-1} + 1$ with initial condition $H_1 = 1$. For $n = 1$ (the base case), this is simply the calculation that $2^1 - 1 = 1$. Assume that $H_n = 2^n - 1$. Then by the recurrence relation we have $H_{n+1} = 2H_n + 1$, whereupon if we substitute on the basis of the inductive hypothesis we obtain $2(2^n - 1) + 1 = 2^{n+1} - 2 + 1 = 2^{n+1} - 1$, exactly the formula for the case of $n + 1$. Thus we have shown that if the formula is correct for n, then it is also correct for $n + 1$, and our proof by induction is complete.

3. a) Let a_n be the number of ways to deposit n dollars in the vending machine. We must express a_n in terms of earlier terms in the sequence. If we want to deposit n dollars, we may start with a dollar coin and then deposit $n - 1$ dollars. This gives us a_{n-1} ways to deposit n dollars. We can also start with a dollar bill and then deposit $n - 1$ dollars. This gives us a_{n-1} more ways to deposit n dollars. Finally, we can deposit a five-dollar bill and follow that with $n - 5$ dollars; there are a_{n-5} ways to do this. Therefore the recurrence relation is $a_n = 2a_{n-1} + a_{n-5}$. Note that this is valid for $n \geq 5$, since otherwise a_{n-5} makes no sense.

b) We need initial conditions for all subscripts from 0 to 4. It is clear that $a_0 = 1$ (deposit nothing) and $a_1 = 2$ (deposit either the dollar coin or the dollar bill). It is also not hard to see that $a_2 = 2^2 = 4$, $a_3 = 2^3 = 8$, and $a_4 = 2^4 = 16$, since each sequence of n C's and B's corresponds to a way to deposit n dollars—a C meaning to deposit a coin and a B meaning to deposit a bill.

c) We will compute a_5 through a_{10} using the recurrence relation:

$$a_5 = 2a_4 + a_0 = 2 \cdot 16 + 1 = 33$$
$$a_6 = 2a_5 + a_1 = 2 \cdot 33 + 2 = 68$$
$$a_7 = 2a_6 + a_2 = 2 \cdot 68 + 4 = 140$$
$$a_8 = 2a_7 + a_3 = 2 \cdot 140 + 8 = 288$$
$$a_9 = 2a_8 + a_4 = 2 \cdot 288 + 16 = 592$$
$$a_{10} = 2a_9 + a_5 = 2 \cdot 592 + 33 = 1217$$

Thus there are 1217 ways to deposit \$10.

5. Since this problem concerns a bill of 17 pesos, we can ignore all denominations greater than 17. Therefore we assume that we have coins for 1, 2, 5, and 10 pesos, and bills for 5 and 10 pesos. Then we proceed as in

Exercise 13 to write down a recurrence relation and initial conditions for a_n, the number of ways to pay a bill of n pesos (order mattering). If we want to achieve a total of n pesos, we can start with a 1-peso coin and then pay out $n-1$ pesos. This gives us a_{n-1} ways to pay n pesos. Similarly, there are a_{n-2} ways to pay starting with a 2-peso coin, a_{n-5} ways to pay starting with a 5-peso coin, a_{n-10} ways to pay starting with a 10-peso coin, a_{n-5} ways to pay starting with a 5-peso bill, and a_{n-10} ways to pay starting with a 10-peso bill. This gives the recurrence relation $a_n = a_{n-1} + a_{n-2} + 2a_{n-5} + 2a_{n-10}$, valid for all $n \geq 10$. As for initial conditions, we see immediately that $a_0 = 1$ (there is one way to pay nothing, namely by using no coins or bills), $a_1 = 1$ (use a 1-peso coin), $a_2 = 2$ (use a 2-peso coin or two 1-peso coins), $a_3 = 3$ (use only 1-peso coins, or use a 2-peso coin either first or second), and $a_4 = 5$ (the bill can be paid using the schemes 1111, 112, 121, 211, or 22, with the obvious notation). For $n = 5$ through $n = 9$, we can iterate the recurrence relation $a_n = a_{n-1} + a_{n-2} + 2a_{n-5}$, since no 10-peso bills are involved. This yields:

$$a_5 = a_4 + a_3 + 2a_0 = 5 + 3 + 2 \cdot 1 = 10$$
$$a_6 = a_5 + a_4 + 2a_1 = 10 + 5 + 2 \cdot 1 = 17$$
$$a_7 = a_6 + a_5 + 2a_2 = 17 + 10 + 2 \cdot 2 = 31$$
$$a_8 = a_7 + a_6 + 2a_3 = 31 + 17 + 2 \cdot 3 = 54$$
$$a_9 = a_8 + a_7 + 2a_4 = 54 + 31 + 2 \cdot 5 = 95$$

Next we iterate the full recurrence relation to get up to $n = 17$:

$$a_{10} = a_9 + a_8 + 2a_5 + 2a_0 = 95 + 54 + 2 \cdot 10 + 2 \cdot 1 = 171$$
$$a_{11} = a_{10} + a_9 + 2a_6 + 2a_1 = 171 + 95 + 2 \cdot 17 + 2 \cdot 1 = 302$$
$$a_{12} = a_{11} + a_{10} + 2a_7 + 2a_2 = 302 + 171 + 2 \cdot 31 + 2 \cdot 2 = 539$$
$$a_{13} = a_{12} + a_{11} + 2a_8 + 2a_3 = 539 + 302 + 2 \cdot 54 + 2 \cdot 3 = 955$$
$$a_{14} = a_{13} + a_{12} + 2a_9 + 2a_4 = 955 + 539 + 2 \cdot 95 + 2 \cdot 5 = 1694$$
$$a_{15} = a_{14} + a_{13} + 2a_{10} + 2a_5 = 1694 + 955 + 2 \cdot 171 + 2 \cdot 10 = 3011$$
$$a_{16} = a_{15} + a_{14} + 2a_{11} + 2a_6 = 3011 + 1694 + 2 \cdot 302 + 2 \cdot 17 = 5343$$
$$a_{17} = a_{16} + a_{15} + 2a_{12} + 2a_7 = 5343 + 3011 + 2 \cdot 539 + 2 \cdot 31 = 9494$$

Thus the final answer is that there are 9494 ways to pay a 17-peso debt using the coins and bills described here, assuming that order matters.

7. **a)** Let a_n be the number of bit strings of length n containing a pair of consecutive 0's. In order to construct a bit string of length n containing a pair of consecutive 0's we could start with 1 and follow with a string of length $n-1$ containing a pair of consecutive 0's, or we could start with a 01 and follow with a string of length $n-2$ containing a pair of consecutive 0's, or we could start with a 00 and follow with any string of length $n-2$. These three cases are mutually exclusive and exhaust the possibilities for how the string might start. From this analysis we can immediately write down the recurrence relation, valid for all $n \geq 2$: $a_n = a_{n-1} + a_{n-2} + 2^{n-2}$. (Recall that there are 2^k bit strings of length k.)

b) There are no bit strings of length 0 or 1 containing a pair of consecutive 0's, so the initial conditions are $a_0 = a_1 = 0$. w

c) We will compute a_2 through a_7 using the recurrence relation:

$$a_2 = a_1 + a_0 + 2^0 = 0 + 0 + 1 = 1$$
$$a_3 = a_2 + a_1 + 2^1 = 1 + 0 + 2 = 3$$
$$a_4 = a_3 + a_2 + 2^2 = 3 + 1 + 4 = 8$$
$$a_5 = a_4 + a_3 + 2^3 = 8 + 3 + 8 = 19$$
$$a_6 = a_5 + a_4 + 2^4 = 19 + 8 + 16 = 43$$
$$a_7 = a_6 + a_5 + 2^5 = 43 + 19 + 32 = 94$$

Thus there are 94 bit strings of length 7 containing two consecutive 0's.

9. **a)** This problem is very similar to Example 3, with the recurrence required to go one level deeper. Let a_n be the number of bit strings of length n that do not contain three consecutive 0's. In order to construct a bit string of length n of this type we could start with 1 and follow with a string of length $n-1$ not containing three consecutive 0's, or we could start with a 01 and follow with a string of length $n-2$ not containing three consecutive 0's, or we could start with a 001 and follow with a string of length $n-2$ not containing three consecutive 0's. These three cases are mutually exclusive and exhaust the possibilities for how the string might start, since it cannot start 000. From this analysis we can immediately write down the recurrence relation, valid for all $n \geq 3$: $a_n = a_{n-1} + a_{n-2} + a_{n-3}$.

b) The initial conditions are $a_0 = 1$, $a_1 = 2$, and $a_2 = 4$, since all strings of length less than 3 satisfy the conditions (recall that the empty string has length 0).

$a_0 = 1$
$a_1 = 2$
$a_2 = 4$

c) We will compute a_3 through a_7 using the recurrence relation:

$$a_3 = a_2 + a_1 + a_0 = 4 + 2 + 1 = 7$$
$$a_4 = a_3 + a_2 + a_1 = 7 + 4 + 2 = 13$$
$$a_5 = a_4 + a_3 + a_2 = 13 + 7 + 4 = 24$$
$$a_6 = a_5 + a_4 + a_3 = 24 + 13 + 7 = 44$$
$$a_7 = a_6 + a_5 + a_4 = 44 + 24 + 13 = 81$$

Thus there are 81 bit strings of length 7 that do not contain three consecutive 0's.

11. **a)** Let a_n be the number of ways to climb n stairs. In order to climb n stairs, a person must either start with a step of one stair and then climb $n-1$ stairs (and this can be done in a_{n-1} ways) or else start with a step of two stairs and then climb $n-2$ stairs (and this can be done in a_{n-2} ways). From this analysis we can immediately write down the recurrence relation, valid for all $n \geq 2$: $a_n = a_{n-1} + a_{n-2}$.

b) The initial conditions are $a_0 = 1$ and $a_1 = 1$, since there is one way to climb no stairs ~~(do nothing)~~ and clearly only one way to climb one stair. Note that the recurrence relation is the same as that for the Fibonacci sequence, and the initial conditions are that $a_0 = f_1$ and $a_1 = f_2$, so it must be that $a_n = f_{n+1}$ for all n.

c) Each term in our sequence $\{a_n\}$ is the sum of the previous two terms, so the sequence begins $a_0 = 1$, $a_1 = 1$, $a_2 = 2$, $a_3 = 3$, $a_4 = 5$, $a_5 = 8$, $a_6 = 13$, $a_7 = 21$, $a_8 = 34$. Thus a person can climb a flight of 8 stairs in 34 ways under the restrictions in this problem.

13. **a)** Let a_n be the number of ternary ~~strings~~ of length n that do not contain two consecutive 0's. In order to construct a bit string ~~of length n~~ of this type we could start with a 1 or a 2 and follow with a string of length $n-1$ not containing two consecutive 0's, or we could start with 01 or 02 and follow with a string of length $n-2$ not containing two consecutive 0's. There are clearly $2a_{n-1}$ possibilities in the first case and $2a_{n-2}$ possibilities in the second. These two cases are mutually exclusive and exhaust the possibilities for how the string might start, since it cannot start 00. From this analysis we can immediately write down the recurrence relation, valid for all $n \geq 2$: $a_n = 2a_{n-1} + 2a_{n-2}$.

b) The initial conditions are $a_0 = 1$ (for the empty string) and $a_1 = 3$ (all three strings of length 1 fail to contain two consecutive 0's).

c) We will compute a_2 through a_6 using the recurrence relation:

$$a_2 = 2a_1 + 2a_0 = 2 \cdot 3 + 2 \cdot 1 = 8$$
$$a_3 = 2a_2 + 2a_1 = 2 \cdot 8 + 2 \cdot 3 = 22$$
$$a_4 = 2a_3 + 2a_2 = 2 \cdot 22 + 2 \cdot 8 = 60$$
$$a_5 = 2a_4 + 2a_3 = 2 \cdot 60 + 2 \cdot 22 = 164$$
$$a_6 = 2a_5 + 2a_4 = 2 \cdot 164 + 2 \cdot 60 = 448$$

Thus there are 448 ternary strings of length 6 that do not contain two consecutive 0's.

15. **a)** Let a_n be the number of ternary strings of length n that do not contain two consecutive 0's or two consecutive 1's. In order to construct a bit string of length n of this type we could start with a 2 and follow with a string of length $n-1$ not containing two consecutive 0's or two consecutive 1's, or we could start with 02 or 12 and follow with a string of length $n-2$ not containing two consecutive 0's or two consecutive 1's, or we could start with 012 or 102 and follow with a string of length $n-3$ not containing two consecutive 0's or two consecutive 1's, or we could start with 0102 or 1012 and follow with a string of length $n-4$ not containing two consecutive 0's or two consecutive 1's, and so on. In other words, once we encounter a 2, we can, in effect, start fresh, but the first 2 may not appear for a long time. Before the first 2 there are always two possibilities—the sequence must alternate between 0's and 1's, starting with either a 0 or a 1. Furthermore, there is one more possibility—that the sequence contains no 2's at all, and there are two cases in which this can happen: 0101... and 1010.... Putting this all together we can write down the recurrence relation, valid for all $n \geq 2$:

$$a_n = a_{n-1} + 2a_{n-2} + 2a_{n-3} + 2a_{n-4} + \cdots + 2a_0 + 2$$

(It turns out that the sequence also satisfies the recurrence relation $\boxed{a_n = 2a_{n-1} + a_{n-2},}$ which can be derived algebraically from the recurrence relation we just gave by subtracting the recurrence for a_{n-1} from the recurrence for a_n. Can you find a direct argument for it?)

b) The initial conditions are that $a_0 = 1$ (the empty string satisfies the conditions) and $a_1 = 3$ (the condition cannot be violated in so short a string).

c) We will compute a_2 through a_6 using the recurrence relation:

$$a_2 = a_1 + 2a_0 + 2 = 3 + 2 \cdot 1 + 2 = 7$$
$$a_3 = a_2 + 2a_1 + 2a_0 + 2 = 7 + 2 \cdot 3 + 2 \cdot 1 + 2 = 17$$
$$a_4 = a_3 + 2a_2 + 2a_1 + 2a_0 + 2 = 17 + 2 \cdot 7 + 2 \cdot 3 + 2 \cdot 1 + 2 = 41$$
$$a_5 = a_4 + 2a_3 + 2a_2 + 2a_1 + 2a_0 + 2 = 41 + 2 \cdot 17 + 2 \cdot 7 + 2 \cdot 3 + 2 \cdot 1 + 2 = 99$$
$$a_6 = a_5 + 2a_4 + 2a_3 + 2a_2 + 2a_1 + 2a_0 + 2 = 99 + 2 \cdot 41 + 2 \cdot 17 + 2 \cdot 7 + 2 \cdot 3 + 2 \cdot 1 + 2 = 239$$

Thus there are 239 ternary strings of length 6 that do not contain two consecutive 0's or two consecutive 1's.

17. **a)** Let a_n be the number of ternary strings that do not contain consecutive symbols that are the same. By symmetry we know that $a_n/3$ of these must start with each of the symbols 0, 1, and 2. Now to construct such a string, we can begin with any symbol (3 choices), but we must follow it with a string of length $n-1$ not containing two consecutive symbols that are the same and not beginning with the symbol with which we began ($\frac{2}{3}a_{n-1}$ choices). This tells us that $a_n = 3 \cdot \frac{2}{3}a_{n-1}$, or more simply $a_n = 2a_{n-1}$, valid for every $n \geq 2$.

b) The initial condition is clearly that $\boxed{a_1 = 3.}$ We could also mention that $\boxed{a_0 = 1,}$ but the recurrence only goes one level deep.)

c) Here it is easy to compute the terms in the sequence, since each is just 2 times the previous one. Thus $a_6 = 2a_5 = 2^2a_4 = 2^3a_3 = 2^4a_2 = 2^5a_1 = 2^5 \cdot 3 = 96$.

19. **a)** This problem is really the same as ("isomorphic to," as a mathematician would say) Exercise 11, since a sequence of signals exactly corresponds to a sequence of steps in that exercise. Therefore the recurrence relation is $a_n = a_{n-1} + a_{n-2}$ for all $n \geq 2$.

b) The initial conditions are again the same as in Exercise 11, namely $a_0 = 1$ (the empty message) and $a_1 = 1$.

~~**c)** Continuing~~ where we left off our calculation in Exercise 11, we find that $a_9 = a_8 + a_7 = 34 + 21 = 55$ and then $a_{10} = a_9 + a_8 = 55 + 34 = 89$. (If we allow only part of the time period to be used, and if we rule out the empty message, then the answer will be $1 + 2 + 3 + 5 + 8 + 13 + 21 + 34 + 55 + 89 = 231$.)

21. a) This problem is related to Exercise 62 in Section 5.1. Consider the plane already divided by $n-1$ lines into R_{n-1} regions. The n^{th} line is now added, intersecting each of the other $n-1$ lines in exactly one point, $n-1$ intersections in all. Think of drawing that line, beginning at one of its ends (out at "infinity"). (You should be drawing a picture as you read these words!) As we move toward the first point of intersection, we are dividing the unbounded region of the plane through which it is passing into two regions; the division is complete when we reach the first point of intersection. Then as we draw from the first point of intersection to the second, we cut off another region (in other words we divide another of the regions that were already there into two regions). This process continues as we encounter each point of intersection. By the time we have reached the last point of intersection, the number of regions has increased by $n-1$ (one for each point of intersection). Finally, as we move off to infinity, we divide the unbounded region through which we pass into two regions, increasing the count by yet 1 more. Thus there are exactly n more regions than there were before the n^{th} line was added. The analysis we have just completed shows that the recurrence relation we seek is $R_n = R_{n-1} + n$. The initial condition is $R_0 = 1$ (since there is just one region—the whole plane—when there are no lines). Alternately, we could specify $R_1 = 2$ as the initial condition.

b) The recurrence relation and initial condition we have are precisely those in Exercise 9c, so the solution is $R_n = (n^2 + n + 2)/2$.

$\frac{n(n+1)}{2} + 1$

23. This problem is intimately related to Exercise 50 in the supplementary set of exercises in Chapter 5. It also will use the result of Exercise 21 of the present section.

a) Imagine $n-1$ planes meeting the stated conditions, dividing space into S_{n-1} solid regions. (This may be hard to visualize once n gets to be more than 2 or 3, but you should try to see it in your mind, even if the picture is blurred.) Now a new plane is drawn, intersecting each of the previous $n-1$ planes in a line. Look at the pattern these lines form on the new plane. There are $n-1$ lines, each two of which intersect and no three of which pass through the same point (because of the requirement on the "general position" of the planes). According to the result of Exercise 21b, they form $((n-1)^2 + (n-1) + 2)/2 = (n^2 - n + 2)/2$ regions in the new plane. Now each of these planar regions is actually splitting a former solid region into two. Thus the number of new solid regions this new plane creates is $(n^2 - 2n + 2)/2$. In other words, we have our recurrence relation: $S_n = S_{n-1} + (n^2 - n + 2)/2$. The initial condition is $S_0 = 1$ (if there are no planes, we get one region). Let us verify this for some small values of n. If $n = 1$, then the recurrence relation gives $S_1 = S_0 + (1^2 - 1 + 2)/2 = 1 + 1 = 2$, which is correct (one plane divides space into two half-spaces). Next $S_2 = S_1 + (2^2 - 2 + 2)/2 = 2 + 2 = 4$, and again it is easy to see that this is correct. Similarly, $S_3 = S_2 + (3^2 - 3 + 2)/2 = 4 + 4 = 8$, and we know that this is right from our familiarity with 3-dimensional graphing (space has eight octants). The first surprising case is $n = 4$, when we have $S_4 = S_3 + (4^2 - 4 + 2)/2 = 8 + 7 = 15$. This takes some concentration to see (consider the plane $x + y + z = 1$ passing through space. It splits each octant into two parts except for the octant in which all coordinates are negative, because it does not pass through that octant. Thus 7 regions become 14, and the additional region makes a total of 15.

b) The iteration here gets a little messy. We need to invoke two summation formulae from Table 2 in Section 2.4: $1 + 2 + 3 + \cdots + n = n(n+1)/2$ and $1^2 + 2^2 + 3^2 + \cdots + n^2 = n(n+1)(2n+1)/6$. We proceed as follows:

$$\begin{aligned}
S_n &= \frac{n^2}{2} - \frac{n}{2} + 1 + S_{n-1} \\
&= \frac{n^2}{2} - \frac{n}{2} + 1 + \left(\frac{(n-1)^2}{2} - \frac{(n-1)}{2} + 1\right) + S_{n-2} \\
&= \frac{n^2}{2} - \frac{n}{2} + 1 + \left(\frac{(n-1)^2}{2} - \frac{(n-1)}{2} + 1\right) + \left(\frac{(n-2)^2}{2} - \frac{(n-2)}{2} + 1\right) + S_{n-3} \\
&\;\vdots
\end{aligned}$$

$$= \frac{n^2}{2} - \frac{n}{2} + 1 + \left(\frac{(n-1)^2}{2} - \frac{(n-1)}{2} + 1\right) + \left(\frac{(n-2)^2}{2} - \frac{(n-2)}{2} + 1\right) + \cdots$$
$$+ \left(\frac{1^2}{2} - \frac{1}{2} + 1\right) + S_0$$
$$= \frac{1}{2}\left((n^2 + (n-1)^2 + \cdots + 1^2) - (n + (n-1) + \cdots + 1)\right) + (1 + 1 + \cdots + 1) + 1$$
$$= \frac{1}{2}\left(\frac{n(n+1)(2n+1)}{6} - \frac{n(n+1)}{2}\right) + n + 1$$
$$= \frac{n^3 + 5n + 6}{6} \quad \text{(after a little algebra).}$$

Note that this answer agrees with that given in supplementary Exercise 50 of Chapter 5.

25. The easy way to do this problem is to invoke symmetry. A bit string of length 7 has an even number of 0's if and only if it has an odd number of 1's, since the sum of the number of 0's and the number of 1's, namely 7, is odd. Because of the symmetric role of 0 and 1, there must be just as many 7-bit strings with an even number of 0's as there are with an odd number of 0's, each therefore being $2^7/2$ (since there are 2^7 bit strings altogether). Thus the answer is $2^{7-1} = 64$.

The solution can also be found using recurrence relations. Let e_n be the number of bit strings of length n with an even number of 0's. A bit string of length n with an even number of 0's is either a bit string that starts with a 1 and is then followed by a bit string of length $n-1$ with an even number of 0's (of which there are e_{n-1}), or else it starts with a 0 and is then followed by a bit string of length $n-1$ with an odd number of 0's (of which there are $2^{n-1} - e_{n-1}$). Therefore we have the recurrence relation $e_n = e_{n-1} + (2^{n-1} - e_{n-1}) = 2^{n-1}$. In other words, it is a recurrence relation that is its own solution. In our case, $n = 7$, so there are $2^{7-1} = 64$ such strings. (See also Exercise 31 in Section 6.4.)

✓ **27.** We assume that the walkway is one tile in width and n tiles long, from start to finish. Thus we are talking about ternary sequences of length n that do not contain two consecutive 0's, say. This was studied in Exercise 13, so the answers obtained there apply. We let a_n represent the desired quantity.

$n : 2^{n-1}$ +

a) As in Exercise 13, we find the recurrence relation to be $a_n = 2a_{n-1} + 2a_{n-2}$.

b) As in Exercise 13, the initial conditions are $a_0 = 1$ and $a_1 = 3$.

c) Continuing the computation started in the solution to Exercise 13, we find

$$a_7 = 2a_6 + 2a_5 = 2 \cdot 448 + 2 \cdot 164 = 1224.$$

Thus there are 1224 such colored paths.

29. If the codomain has only one element, then there is only one function (namely the function that takes each element of the domain to the unique element of the codomain). Therefore when $n = 1$ we have $S(m,n) = S(m,1) = 1$, the initial condition we are asked to verify. Now assume that $m \geq n > 1$, and we want to count $S(m,n)$, the number of functions from a domain with m elements *onto* a codomain with n elements. The form of the recurrence relation we are supposed to verify suggests that what we want to do is to look at the non-onto functions. There are n^m functions from the m-set to the n-set altogether (by the product rule, since we need to choose an element from the n-set, which can be done in n ways, a total of m times). Therefore we must show that there are $\sum_{k=1}^{n-1} C(n,k)S(m,k)$ functions from the domain to the codomain that are *not* onto. First we use the sum rule and break this count down into the disjoint cases determined by the number of elements—let us call it k—in the range of the function. Since we want the function not to be onto, k can have any value from 1 to $n-1$, but k cannot equal n. Once we have specified k, in order to specify a function we need to first specify the actual range, and this can be done in $C(n,k)$ ways, namely choosing the subset of k elements from the codomain that are to constitute the range; and second choose an *onto* function from the domain to this set of k elements. This latter task can be done in $S(m,k)$ ways, since (and here is the key recursive point) we are defining $S(m,k)$ to be precisely this number. Therefore by the product rule there are $C(n,k)S(m,k)$ different functions with our original domain having a range of k elements, and so by the sum rule there are $\sum_{k=1}^{n-1} C(n,k)S(m,k)$ non-onto functions from our original domain to our original codomain. Note that this two-dimensional recurrence relation can be used to compute $S(m,n)$ for any desired positive integers m and n. Using it is much easier than trying to list all onto functions.

31. We will see that the answer is too large for us to list all the possibilities by hand with a reasonable amount of effort.

a) We know from Example 5 that $C_0 = 1$, $C_1 = 1$, and $C_3 = 5$. It is also easy to see that $C_2 = 2$, since there are only two ways to parenthesize the product of three numbers. We know from Exercise 30 that $C_4 = 14$. Therefore the recurrence relation tells us that $C_5 = C_0C_4 + C_1C_3 + C_2C_2 + C_3C_1 + C_4C_0 = 1 \cdot 14 + 1 \cdot 5 + 2 \cdot 2 + 5 \cdot 1 + 14 \cdot 1 = 42$.

b) Here $n = 5$, so the formula gives $\frac{1}{6}C(10,5) = \frac{1}{6} \cdot 10 \cdot 9 \cdot 8 \cdot 7 \cdot 6/5! = 42$.

33. Obviously $J(1) = 1$. When $n = 2$, the second person is killed, so $J(2) = 1$. When $n = 3$, person 2 is killed off, then person 3 is skipped, so person 1 is killed, making $J(3) = 3$. When $n = 4$, the order of death is 2, 4, 3; so $J(4) = 1$. For $n = 5$, the order of death is 2, 4, 1, 5; so $J(5) = 3$. With pencil and paper (or a computer, if we feel like writing a little program), we find the remaining values:

n	$J(n)$	n	$J(n)$
1	1	9	3
2	1	10	5
3	3	11	7
4	1	12	9
5	3	13	11
6	5	14	13
7	7	15	15
8	1	16	1

35. If the number of players is even (call it $2n$), then after we have gone around the circle once we are back at the beginning, with two changes. First, the number assigned to every player has been changed, since all the even numbers are now missing. The first remaining player is 1, the second remaining player is 3, the third remaining player is 5, and so on. In general, the player in location i at this point is the player whose original number was $2i - 1$. Second, the number of players is half of what it used to be; it's now n. Therefore we know that the survivor will be the player currently occupying spot $J(n)$, namely $2J(n) - 1$. Thus we have shown that $J(2n) = 2J(n) - 1$. The argument when there are an odd number of players is similar. If there are $2n + 1$ players, then after we have gone around the circle once and then killed off player 1, we will have n players left. The first remaining spot is occupied by player 3, the second remaining player is 5, and so on—the i^{th} remaining player is $2i + 1$. Therefore we know that the survivor will be the player currently occupying spot $J(n)$, namely $2J(n) + 1$. Thus we have shown that $J(2n + 1) = 2J(n) + 1$. The base case is clearly $J(1) = 1$.

37. We use the conjecture from Exercise 34: If $n = 2^m + k$, where $k < 2^m$, then $J(n) = 2k + 1$. Thus $J(100) = J(2^6 + 36) = 2 \cdot 36 + 1 = 73$; $J(1000) = J(2^9 + 488) = 2 \cdot 488 + 1 = 977$; and $J(10000) = J(2^{13} + 1808) = 2 \cdot 1808 + 1 = 3617$.

39. It is not too hard to find the winning moves (here $a \xrightarrow{b} c$ means to move disk b from peg a to peg c, where we label the smallest disk 1 and the largest disk 4): $1 \xrightarrow{1} 2$, $1 \xrightarrow{2} 3$, $2 \xrightarrow{1} 3$, $1 \xrightarrow{3} 2$, $1 \xrightarrow{4} 4$, $2 \xrightarrow{3} 4$, $3 \xrightarrow{1} 2$,

$3 \xrightarrow{2} 4$, $2 \xrightarrow{1} 4$. We can argue that at least seven moves are required no matter how many pegs we have: three to unstack the disks, one to move disk 4, and three more to restack them. We need to show that at least two additional moves are required because of the congestion caused by there being only four pegs. Note that in order to move disk 4 from peg 1 to peg 4, the other three disks must reside on pegs 2 and 3. That requires at least one move to restack them and one move to unstack them. This completes the argument.

41. It is helpful to do Exercise 40 first to get a feeling for what is going on. The base cases are obvious. If $n > 1$, then the algorithm consists of three stages. In the first stage, by the inductive hypothesis, it takes $R(n-k)$ moves to transfer the smallest $n-k$ disks to peg 2. Then by the usual Tower of Hanoi algorithm, it takes $2^k - 1$ moves to transfer the largest k disks (i.e., the rest of them) to peg 4, avoiding peg 2. Then again by induction, it takes $R(n-k)$ moves to transfer the smallest $n-k$ disks to peg 4; all the pegs are available for this work, since the largest disks, now residing on peg 4, do not interfere. The recurrence relation is therefore established.

43. First write $R(n) = \sum_{j=1}^{n}(R(j) - R(j-1))$, which is immediate from the telescoping nature of the sum (and the fact that $R(0) = 0$. By the result from Exercise 42, this is the sum of $2^{k'-1}$ for this range of values of j (here j is playing the role that n played in Exercise 42, and k' is the value selected by the algorithm for j). But k' is constant (call this constant i) for i successive values of j. Therefore this sum is $\sum_{i=1}^{k} i2^{i-1}$, except that if n is not a triangular number, then the last few values when $i = k$ are missing, and that is what the final term in the given expression accounts for.

45. By Exercise 43, $R(n)$ is no greater than $\sum_{i=1}^{k} i2^{i-1}$. By using algebra and calculus, we can show that this equals $(k+1)2^k - 2^{k+1} + 1$, so it is no greater than $(k+1)2^k$. (The proof is to write the formula for a geometric series $\sum_{i=0}^{k} x^i = (1 - x^{k+1})/(1-x)$, differentiate both sides, and simplify.) Since $n > k(k-1)/2$, we see from the quadratic formula that $k < \frac{1}{2} + \sqrt{2n + \frac{1}{4}} < 1 + \sqrt{2n}$ for all $n > 1$. Therefore $R(n)$ is bounded above by $(1 + \sqrt{2n} + 1)2^{1+\sqrt{2n}} \le 8 \cdot \sqrt{n}\,2^{\sqrt{2n}}$ for all $n > 2$. This shows that $R(n)$ is $O(\sqrt{n}\,2^{\sqrt{2n}})$, as desired.

47. We have to do Exercise 46 before we can do this exercise.
a) We found that the first differences were $\nabla a_n = 0$. Therefore the second differences are given by $\nabla^2 a_n = 0 - 0 = 0$.
b) We found that the first differences were $\nabla a_n = 2n - 2(n-1) = 2$. Therefore the second differences are given by $\nabla^2 a_n = 2 - 2 = 0$.
c) We found that the first differences were $\nabla a_n = n^2 - (n-1)^2 = 2n - 1$. Therefore the second differences are given by $\nabla^2 a_n = (2n-1) - (2(n-1) - 1) = 2$.
d) We found that the first differences were $\nabla a_n = 2^n - 2^{n-1} = 2^{n-1}$. Therefore the second differences are given by $\nabla^2 a_n = 2^{n-1} - 2^{n-2} = 2^{n-2}$.

49. This is just an exercise in algebra. The right-hand side of the given expression is by definition $a_n - 2\nabla a_n + \nabla a_n - \nabla a_{n-1} = a_n - \nabla a_n - \nabla a_{n-1} = a_n - (a_n - a_{n-1}) - (a_{n-1} - a_{n-2})$. Everything in this expression cancels except the last term, yielding a_{n-2}, as desired.

51. In order to express the recurrence relation $a_n = a_{n-1} + a_{n-2}$ in terms of a_n, ∇a_n, and $\nabla^2 a_n$, we use the results of Exercise 48 (that $a_{n-1} = a_n - \nabla a_n$) and Exercise 49 (that $a_{n-2} = a_n - 2\nabla a_n + \nabla^2 a_n$). Thus the given recurrence relation is equivalent to $a_n = (a_n - \nabla a_n) + (a_n - 2\nabla a_n + \nabla^2 a_n)$, which simplifies algebraically to $a_n = 3\nabla a_n - \nabla^2 a_n$.

53. Modify Algorithm 1 so that in addition to storing the value $T(j)$, the maximum number of attendees possible for an optimal schedule of the first j talks, we also store, for each j, a set $S(j)$, which consists of the subscripts of a set of talks among the first j talks that can be scheduled to achieve a total of $T(j)$ attendees. Initially we set $S(0)$ equal to the empty set. At the point at which we compute $T(j)$, we set $S(j)$ equal to $S(j-1)$ if $T(j)$ was set to $T(j-1)$, and we set $S(j)$ equal to $S(p(j)) \cup \{j\}$ otherwise. (If those two values are the same, then we have our choice—from a practical point of view, it would probably be better to see which choice resulted in more talks.)

55. We record the information in tabular form, filling in $T(j)$ and $S(j)$ line by line. For each j, if $w_j + T(p(j)) \le T(j-1)$, then in the j^{th} line we just copy the information from the previous line. If $w_j + T(p(j)) > T(j-1)$, then we put $w_j + T(p(j))$ in the $T(j)$ column and put $S(p(j)) \cup \{j\}$ in the $S(j)$ column.

a)

j	$p(j)$	$w(j)$	$T(j)$	$S(j)$
0	—	—	0	$\emptyset$
1	0	20	20	$\{1\}$
2	0	10	20	$\{1\}$
3	1	50	70	$\{1,3\}$
4	0	30	70	$\{1,3\}$
5	0	15	70	$\{1,3\}$
6	2	25	70	$\{1,3\}$
7	4	40	110	$\{1,3,7\}$

b) This time the answer is not unique. When we had to compute $T(7)$ and $S(7)$ we could achieve an attendance of 140 either with talks 1 and 6 or with talks 1, 3, and 7.

j	$p(j)$	$w(j)$	$T(j)$	$S(j)$
0	—	—	0	$\emptyset$
1	0	100	100	$\{1\}$
2	0	5	100	$\{1\}$
3	1	10	110	$\{1,3\}$
4	0	20	110	$\{1,3\}$
5	0	25	110	$\{1,3\}$
6	2	40	140	$\{1,6\}$
7	4	30	140	$\{1,6\}$

c) Here when we had to compute $T(6)$ and $S(6)$ we could achieve an attendance of 10 either with talks 1 and 3 or with talks 2 and 6. But the final answer is unique.

j	$p(j)$	$w(j)$	$T(j)$	$S(j)$
0	—	—	0	$\emptyset$
1	0	2	2	$\{1\}$
2	0	3	3	$\{2\}$
3	1	8	10	$\{1,3\}$
4	0	5	10	$\{1,3\}$
5	0	4	10	$\{1,3\}$
6	2	7	10	$\{1,3\}$
7	4	10	20	$\{1,3,7\}$

d)

j	$p(j)$	$w(j)$	$T(j)$	$S(j)$
0	—	—	0	$\emptyset$
1	0	10	10	$\{1\}$
2	0	8	10	$\{1\}$
3	1	7	17	$\{1,3\}$
4	0	25	25	$\{4\}$
5	0	20	25	$\{4\}$
6	2	30	40	$\{1,6\}$
7	4	5	40	$\{1,6\}$

57. **a)** By Example 5 the number of ways to parenthesize the product to determine the order in which to perform the multiplications is the Catalan number C_n. By Exercise 41 in Section 8.4, $C_n > 2^{n-1}$. Therefore the brute-force method (to test all possible orders of multiplication) has exponential time complexity.

b) The last step in computing $\mathbf{A}_{ij}$ is to multiply the $m_i \times m_{k+1}$ matrix $\mathbf{A}_{ik}$ by the $m_{k+1} \times m_{j+1}$ matrix $\mathbf{A}_{k+1,j}$ for some k between i and $j-1$ inclusive, which will require $m_i m_{k+1} m_{j+1}$ integer multiplications, independent of the manner in which $\mathbf{A}_{ik}$ and $\mathbf{A}_{k+1,j}$ are computed. Therefore to minimize the total number of integer multiplications, each of those two factors must be computed in the most efficient manner.

c) This follows immediately from part **(b)** and the definition of $M(i,j)$. We just try all $j-i$ possible ways of splitting the product into two factors.

d) In the algorithm shown here, $M(i,j)$ is as defined in part **(b)**, and *where*(i,j) is the value of k such that the calculation of $\mathbf{A}_{ij}$ should be broken into the calculation of $\mathbf{A}_{ik}\mathbf{A}_{k+1,j}$ in order to achieve the fewest possible number of integer multiplications. Once all these values are computed, it is a simple matter to read off, using *where*(i,j), the proper order in which to carry out the matrix multiplications. (First, *where*$(1,n)$ tells us which two subproducts to compute first, say $\mathbf{A}_{1r}$ and $\mathbf{A}_{r+1,n}$. Then *where*$(1,r)$ and *where*$(r+1,n)$ tell us where to break those subproducts, and so on.) Notice that the main loop is indexed by d, which represents $j-i$ in the notation of part **(c)**.

```
procedure matrixorder($m_1, \ldots, m_{n+1}$ : positive integers)
for $i := 1$ to $n$
    $M(i,i) := 0$
for $d := 1$ to $n-1$
    for $i := 1$ to $n-d$
        $min := 0$
        for $k := i$ to $i+d$
            $new := M(i,k) + M(k+1, i+d) + m_i m_{k+1} m_{i+d+1}$
            if $new < min$ then
                $min := new$
                $where(i, i+d) := k$
        $M(i, i+d) := min$
```

e) The work in this algorithm is done by three nested loops, each of which is indexed over at most n values.