CHAPTER 10

Ordinary Generating Functions

Introduction

We'll begin this chapter by introducing the notion of ordinary generating functions and discussing the basic techniques for manipulating them. These techniques are merely restatements and simple applications of things you learned in algebra and calculus. You *must* master these basic ideas before reading further.

In Section 2, we apply generating functions to the solution of simple recursions. This requires no new concepts, but provides practice manipulating generating functions. In Section 3, we return to the manipulation of generating functions, introducing slightly more advanced methods than those in Section 1. If you found the material in Section 1 easy, you can skim Sections 2 and 3. If you had some difficulty with Section 1, those sections will give you additional practice developing your ability to manipulate generating functions.

Section 4 is the heart of this chapter. In it we study the Rules of Sum and Product for ordinary generating functions. Suppose that we are given a combinatorial description of the construction of some structures we wish to count. These two rules often allow us to write down an equation for the generating function directly from this combinatorial description. Without such tools, we may get bogged down in lengthy algebraic manipulations.

10.1 What are Generating Functions?

In this section, we introduce the idea of ordinary generating functions and look at some ways to manipulate them. This material is essential for understanding later material on generating functions. Be sure to work the exercises in this section before reading later sections!

Definition 10.1 Ordinary generating function (OGF) *Suppose we are given a sequence $a_0, a_1, \ldots$. The* **ordinary generating function** *(also called* **OGF***) associated with this sequence is the function whose value at x is $\sum_{i=0}^{\infty} a_i x^i$. The sequence $a_0, a_1, \ldots$ is called the coefficients of the generating function.*

People often drop "ordinary" and call this the *generating function* for the sequence. This is also called a "power series" because it is the sum of a series whose terms involve powers of x. The summation is often written $\sum_{i\geq 0} a_i x^i$ or $\sum a_i x^i$.

If your sequence is finite, you can still construct a generating function by taking all the terms after the last to be zero. If you have a sequence that starts at a_k with $k > 0$, you can define $a_0, \ldots, a_{k-1}$ to be any convenient values. "Convenient values" are ones that make equations nicer in some sense. For example, if $H_{n+1} = 2H_n + 1$ for $n > 0$ and $H_1 = 1$. It is convenient to let $H_0 = 0$ so that the recursion is valid for $n \geq 0$. (H_n is the number of moves required for the Tower of Hanoi puzzle. See Exercise 7.3.9 (p. 218).) On the other hand, if $b_1 = 1$ and $b_n = \sum_{k=1}^{n-1} b_k b_{n-k}$ for $n > 1$, it's convenient to define $b_0 = 0$ so that we have $b_n = \sum_{k=0}^{n} b_k b_{n-k}$ for $k \neq 1$. (The latter sum is a "convolution", which we will define in a little while.)

To help us keep track of which generating function is associated with which sequence, we try to use lower case letters for sequences and the corresponding upper case letters for the generating functions. Thus we use the function A as generating function for a sequence of a_n's and B as the generating function for b_n's. Sometimes conventional notation for certain sequences make this upper and lower case pairing impossible. In those cases, we improvise.

You may have noticed that our definition is incomplete because we spoke of a function but did not specify its domain or range. The domain will depend on where the power series converges; however, for combinatorial applications, there is usually no need to be concerned with the convergence of the power series. As a result of this, we will often ignore the issue of convergence. In fact, we can treat the power series like a polynomial with an infinite number of terms. The domain in which the power series converges does matter when we study asymptotics, but that is still several sections in the future.

If we have a doubly indexed sequence $b_{i,j}$, we can extend the definition of a generating function:

$$B(x,y) \;=\; \sum_{j\geq 0}\sum_{i\geq 0} b_{i,j} x^i\, y^j \;=\; \sum_{i,j=0}^{\infty} b_{i,j} x^i\, y^j .$$

Clearly, we can extend this idea to any number of indices—we're not limited to just one or two.

Definition 10.2 $[x^n]$ *Given a generating function $A(x)$ we use $[x^n]\,A(x)$ to denote a_n, the coefficient of x^n. For a generating function in more variables, the coefficient may be another generating function. For example $[x^n y^k]\,B(x,y) = b_{n,k}$ and $[x^n]\,B(x,y) = \sum_{i\geq 0} b_{n,i} y^i$.*

Implicit in the preceding definition is the fact that the generating function *uniquely determines* its coefficients. In other words, given a generating function there is just one sequence that gives rise to it. Without this uniqueness, generating functions would be of little use since we wouldn't be able to recover the coefficients from the function alone.

This leads to another question. Given a generating function, say $A(x)$, how can we find its coefficients $a_0, a_1, \ldots$? One possibility is that we might know the sequence already and simply recognize its generating function. Another is Taylor's Theorem. We'll phrase it slightly differently here to avoid questions of convergence. In our form, it is practically a tautology.

Theorem 10.1 Taylor's Theorem *If $A(x)$ is the generating function for a sequence $a_0, a_1, \ldots$, then $a_n = A^{(n)}(0)/n!$, where $A^{(n)}$ is the nth derivative of A and $0! = 1$. (The theorem extends to more than one variable, but we will not state it.)*

We stated this to avoid questions of convergence—but don't we have to worry about convergence of infinite series? Yes and no:

When manipulating generating functions we normally do not need to worry about convergence unless we are doing asymptotics (see Section 11.4) or substituting numbers for the variables (see the next example).

Example 10.1 Binomial coefficients Let's use the binomial coefficients to get some practice. Set $a_{k,n} = \binom{n}{k}$. Remember that $a_{k,n} = 0$ for $k > n$. From the Binomial Theorem, $(1+x)^n = \sum_{k=0}^{n} \binom{n}{k} x^k$. Thus $\sum a_{k,n} x^k = (1+x)^n$ and so

$$A(x,y) \;=\; \sum_{n\ge 0}\sum_{k\ge 0} a_{k,n} x^k y^n \;=\; \sum_{n\ge 0} (1+x)^n\, y^n \;=\; \sum_{n=0}^{\infty} ((1+x)y)^n.$$

From the formula $\sum_{k\ge 0} az^k = a/(1-z)$ for summing a *geometric series*, we have

$$A(x,y) \;=\; \frac{1}{1-(1+x)y} \;=\; \frac{1}{1-y-xy}. \qquad 10.1$$

Let's see what we can get from this.

- From our definitions, $[x^k y^n]\, A(x,y) = \binom{n}{k}$ and $[y^n]\, A(x,y) = (1+x)^n$, which is equivalent to

$$\sum_{k=0}^{n} \binom{n}{k} x^k \;=\; (1+x)^n \qquad 10.2$$

  Of course, this is nothing new — it's what we started out with when we worked out the formula for $A(x,y)$. We just did this to become more familiar with the notation and manipulation.

- Now let's look at $[x^k]\, A(x,y)$. From (10.1) and the formula for a geometric series,

$$\begin{aligned} A(x,y) &= \frac{1}{(1-y)-xy} \;=\; \frac{1/(1-y)}{1-xy/(1-y)} \\ &= \sum_{k\ge 0} \frac{1}{1-y}\left(\frac{xy}{1-y}\right)^k \;=\; \sum_{k\ge 0} \frac{1}{1-y}\left(\frac{y}{1-y}\right)^k x^k. \end{aligned}$$

  Thus $[x^k]\, A(n,k) = \frac{1}{1-y}\left(\frac{y}{1-y}\right)^k$. In other words, we have the generating function

$$\sum_{n\ge 0} \binom{n}{k} y^n \;=\; \frac{y^k}{(1-y)^{k+1}}. \qquad 10.3$$

  This is new and we'll get more in a minute.

- We can replace the x and y in our generating functions by numbers. If we do that in (10.2) it's not very interesting. Let's do it in (10.3). We must be careful: The sum on the left side is infinite and so convergence is an issue. With $y = 1/3$ we have

$$\sum_{n\ge 0} \binom{n}{k} 3^{-n} \;=\; \frac{3^k}{2^{k+1}}, \qquad 10.4$$

  and it can be shown that the sum converges. So this is a new result. On the other hand, if we set $y = 2$ instead the series would have been $\sum \binom{n}{k} 2^n$ which diverges to infinity. The right side of (10.3) is not infinity but $(-1)^{k+1} 2^k$, which is nonsensical for a sum of positive terms. That's a warning that something is amiss, namely a lack of convergence.

- Returning to (10.1), let's set $x = y$. In that case, we obtain

$$\sum_{n,k\ge 0} \binom{n}{k} x^{n+k} \;=\; A(x,x) \;=\; \frac{1}{1-x-x^2}. \qquad 10.5$$

  What is the coefficient of x^m on the left side? You should be able to see that it will be the sum of $\binom{n}{k}$ over all n and k such that $n + k = m$. Thus $n = m - k$ and so

$$\sum_{k\ge 0} \binom{m-k}{k} \;=\; [x^m]\left(\frac{1}{1-x-x^2}\right).$$

In the next section, we will see how to obtain such coefficients, which turn out to be the Fibonacci numbers. Convergence is not an issue: the sum on the left is finite since the binomial coefficients are nonzero only when $m - k \geq k$, that is $k \leq m/2$. $\square$

There are two important differences in the study of generating functions here and in calculus. We've already noted one: convergence is usually not an issue as long as we know the coefficients make sense. The second is that our interest is in the reverse direction: We study generating functions to learn about their coefficients but in calculus one studies the coefficients to learn about the functions. For example, one might use the first few terms of the sum to estimate the value of the function.

The following simple theorem is important in combinatorial uses of generating functions. Some applications can be found in the exercises. It plays a crucial role in the Rule of Product in Section 10.4. Later, we will extend the theorem to generating functions with more than one variable.

Theorem 10.2 Convolution Formula *Let $A(x)$, $B(x)$, and $C(x)$ be generating functions. Then $C(x) = A(x)B(x)$ if and only if*

$$c_n = \sum_{k=0}^{n} a_k b_{n-k} \quad \textit{for all } n \geq 0. \tag{10.6}$$

The sum can also be written $\sum_{k\geq 0} a_{n-k} b_k$ and also as the sum of $a_i b_j$ over all i, j such that $i + j = n$. We call (10.6) a convolution.

Proof: You should have no difficulty verifying that the two other forms given for the sum are in fact the same as $\sum a_k b_{n-k}$.

We first prove that $C(x) = A(x)B(x)$ gives the claimed summation. Since we are not concerning ourselves with convergence, we can multiply generating functions like polynomials:

$$A(x)B(x) = \left(\sum_{k\geq 0} a_k x^k\right)\left(\sum_{j\geq 0} b_j x^j\right) = \sum_{k,j\geq 0} a_k b_j x^{k+j} = \sum_{n\geq 0}\left(\sum_{k=0}^{n} a_k b_{n-k}\right)x^n,$$

where the last equality follows by letting $k+j = n$; that is, $j = n-k$. The sum on k stops at n because $j \geq 0$ is equivalent to $n - k \geq 0$, which is equivalent to $k \leq n$. This proves that $C(x) = A(x)B(x)$ implies (10.6).

Now suppose we are given (10.6). Multiply by x^n, sum over $n \geq 0$, let $j = n - k$ and reverse the steps in the previous paragraph to obtain

$$C(x) = \sum_{n\geq 0} c_n x^n = \sum_{k,j\geq 0} a_k b_j x^{k+j} = A(x)B(x).$$

We've omitted a few computational details that you should fill in. $\square$

Here are a few generating functions that are useful to know about. The first you've already encountered, the second appears in Exercise 10.1.4, the third is an application of the convolution formula (Exercise 10.1.6), and the others are results from calculus.

$$\sum_{k=0}^{\infty} (ar^k)x^k = \frac{a}{1-rx}, \tag{10.7}$$

$$\sum_{k=0}^{\infty} \binom{r}{k} x^k = (1+x)^r \quad \text{where} \quad \binom{r}{k} = \frac{r(r-1)\cdots(r-k+1)}{k!} \quad \text{for all } r, \tag{10.8}$$

$$\sum_{n=0}^{\infty}\left(\sum_{k=0}^{n} a_k\right)x^n = \frac{1}{1-x}\sum_{n\geq 0} a_n x^n, \tag{10.9}$$

$$\sum_{k=0}^{\infty} \frac{a^k x^k}{k!} = e^{ax}, \tag{10.10}$$

$$\sum_{k=1}^{\infty} \frac{a^k x^k}{k} = -\ln(1-ax). \tag{10.11}$$

Exercises

These exercises will give you some practice manipulating generating functions.

10.1.1. Let $p = 1 + x + x^2 + x^3$, $q = 1 + x + x^2 + x^3 + x^4$, and $r = \frac{1}{1-x}$.

(a) Find the coefficient of x^3 in p^2; in p^3; in p^4.

(b) Find the coefficient of x^3 in q^2; in q^3; in q^4.

(c) Find the coefficient of x^3 in r^2; in r^3; in r^4.

(d) Can you offer a simple explanation for the fact that p, q and r all gave the same answers?

(e) Repeat (a)–(c), this time finding the coefficient of x^4. Explain why some are equal and some are not.

10.1.2. Find the coefficient of x^2 in each of the following.

(a) $(2 + x + x^2)(1 + 2x + x^2)(1 + x + 2x^2)$

(b) $(2 + x + x^2)(1 + 2x + x^2)^2(1 + x + 2x^2)^3$

(c) $x(1 + x)^{43}(2 - x)^5$

10.1.3. Find the coefficient of x^{21} in $(x^2 + x^3 + x^4 + x^5 + x^6)^8$.
Hint. If you are clever, you can do this without a lot of calculation.

10.1.4. This exercise explores the general binomial theorem, geometric series and related topics. Part (a) requires calculus.

(a) Let r be any real number. Use Taylor's Theorem without worrying about convergence to prove

$$(1+z)^r = \sum_{k\geq 0} \binom{r}{k} z^k \quad \text{where} \quad \binom{r}{k} = \frac{r(r-1)\cdots(r-k+1)}{k!}.$$

If you're familiar with some form of Taylor's Theorem with remainder, use it to show that, for some $C > 0$, the infinite sum converges when $|z| < C$. (The largest possible value is $C = 1$, but you may find it easier to use a smaller value.)

(b) Use the previous result to obtain the geometric series formula:
$\sum_{k\geq 0} az^k = a/(1-z)$.

(c) Show that $\sum_{k=0}^{n} az^k = (a - az^{n+1})/(1-z)$.

(d) Find a *simple* formula for the coefficient of x^n in $(1-ax)^{-2}$.

10.1.5. In this exercise we'll explore the effect of derivatives. Let $A(x) = \sum_{m=0}^{\infty} a_m x^m$, the ordinary generating function for the sequence a. In each case, first answer the question for $k = 1$ and $k = 2$ and then for general k.

(a) What is $[x^n]\,(x^k A(x))$, that is, the coefficient of x^n in $x^k A(x)$?

(b) Show that $[x^n]\left(\frac{d}{dx}\right)^k A(x) = \frac{(n+k)!\,a_{n+k}}{n!}$. This notation means compute the kth derivative of $A(x)$ and then find the coefficient of x^n in the generating function. It can also be written $[x^n]\,A^{(k)}(x)$.

(c) Show that $[x^n]\left(x\frac{d}{dx}\right)^k A(x) = n^k a_n$. This notation means that you repeat alternately the operations of differentiating and multiplying by x a total of k times each. For example, when $k = 2$ we have $x(xA'(x))'$.

10.1.6. Using Theorem 10.2 or otherwise, do the following.

(a) Prove: If $c_n = a_0 + a_1 + \cdots + a_n$, then $C(x) = A(x)/(1-x)$.

(b) Simplify $\binom{n}{0} - \binom{n}{1} + \cdots + (-1)^k \binom{n}{k}$ when $n > 0$.

(c) Suppose that d_n is the sum of $a_i b_j c_k$ over all $i, j, k \geq 0$ such that $i + j + k = n$. Express $D(x)$ in terms of $A(x)$, $B(x)$, and $C(x)$.

10.1.7. Suppose that $|r| < 1$. Obtain a formula for $\sum_{n \geq 0} \binom{n}{k} r^n$ as a function of k and r. Show that the sum converges by using the ratio test for series.

10.1.8. Note that $(1+x)^{m+n} = (1+x)^m (1+x)^n$. Note that the coefficients of powers of x in $(1+x)^{m+n}$, $(1+x)^m$, and $(1+x)^n$ are binomial coefficients. Use Theorem 10.2 to prove *Vandermonde's formula*:

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

This is one of the many identities that are known for binomial coefficients.
Hint. Remember that n and k in (10.6) can be replaced by other variables. Look at the index and limits on the summation.

10.1.9. Find a *simple* expression for $\sum_i (-1)^i \binom{m}{i} \binom{m}{k-i}$, where the sum is over all values of i for which the binomial coefficients in the sum are defined.

10.1.10. The results given here are referred to as *bisection of series*. Let $A(x) = \sum_{n=0}^{\infty} a_n x^n$.

(a) Show that $(A(x) + A(-x))/2$ is the generating function for the sequence b_n which is zero for odd n and equals a_n for even n.

(b) What is the generating function for the sequence c_n which is zero for even n and equals a_n for odd n?

(c) Evaluate $\sum_{k \geq 0} \binom{n}{2k} x^{2k}$ where x is a real number. In particular, what is $\sum_{k \geq 0} \binom{n}{2k}$?

*10.1.11. Fix $k > 1$ and $0 \leq j < k$. If you are familiar with kth roots of unity, generalize the Exercise 10.1.10 to the sequence b_n which is a_n when $n + j$ is a multiple of k and is zero otherwise:

$$B(x) = \frac{1}{k} \sum_{s=0}^{k-1} \omega^{js} A(\omega^s x),$$

where $\omega = \exp(2\pi i/k)$, a primitive kth root of unity. (The result is called *multisection of series*.)

10.1.12. Evaluate $s_k = \displaystyle\sum_{n=0}^{\infty} \binom{2n}{k} 2^{-n}$.

*10.1.13. Using Exercise 10.1.11, show that

$$\sum_{n=0}^{\infty} \frac{x^{3n}}{(3n)!} = \frac{e^x}{3} + \frac{2\cos(x\sqrt{3}/2)}{3e^{x/2}}$$

and develop similar formulas for $\sum p^{3n+1}/(3n+1)!$ and $\sum p^{3n+2}/(3n+2)!$.

*10.1.14. We use the terminology from the Principle of Inclusion and Exclusion (Theorem 4.1 (p. 95)). Also, let E_k be the number of elements of S that lie in exactly k of the sets $S_1, S_2, \ldots, S_m$.

(a) Using the Rules of Sum and Product (not Theorem 4.1), prove that

$$N_r = \sum_{k\ge 0}\binom{r+k}{r}E_{r+k}.$$

(b) If the generating functions corresponding to $E_0, E_1, \ldots$ and $N_0, N_1, \ldots$ are $E(x)$ and $N(x)$, conclude that $N(x) = E(x+1)$.

(c) Use this to conclude that $E(x) = N(x-1)$ and then deduce the extension of the Principle of Inclusion and Exclusion:

$$E_k = \sum_{i\ge 0}(-1)^i\binom{k+i}{i}N_{k+i}.$$

10.2 Solving a Single Recursion

In this section we'll use ordinary generating functions to solve some simple recursions, including two that we were unable to solve previously: the Fibonacci numbers and the number of unlabeled full binary RP-trees.

Example 10.2 Fibonacci numbers Let F_n be the number of n long sequences of zeroes and ones with no consecutive ones. We can easily see that $F_1 = 2$ and $F_2 = 3$, but what is the general formula?

Suppose that $t_1, \ldots, t_n$ is an arbitrary sequence of desired form. We want to see what happens when we remove the end of the sequence, so we assume that $n > 1$. If $t_n = 0$, then $t_1, \ldots, t_{n-1}$ is also an arbitrary sequence of the desired form. Now suppose that $t_n = 1$. Then $t_{n-1} = 0$ and so, if $n > 2$, $t_1, \ldots, t_{n-2}$ is an arbitrary sequence of the desired form. All this is reversible: Suppose that $n > 2$. The following two operations produce all n long sequences of the desired form *exactly once*.

- Let $t_1, \ldots, t_{n-1}$ be an arbitrary sequence of the desired form. Set $t_n = 0$.
- Let $t_1, \ldots, t_{n-2}$ be an arbitrary sequence of the desired form. Set $t_{n-1} = 0$ and $t_n = 1$.

Since all n long sequences of the desired form are obtained exactly once this way, the Rule of Sum yields the recursion

$$F_n = F_{n-1} + F_{n-2} \quad \text{for} \quad n > 2. \tag{10.12}$$

Here are the first few values.

n	0	1	2	3	4	5	6	7	8	9	10
F_n	1	2	3	5	8	13	21	34	55	89	144

These numbers, called the *Fibonacci numbers*, were studied in Exercise 1.4.10, but we couldn't solve the recursion there. Now we will.

First, we want to adjust (10.12) so that it holds for all $n \ge 0$. To do this we define F_n when n is small and introduce a new sequence c_n to "correct" the recursion for small n;

$$F_n = F_{n-1} + F_{n-2} + c_n, \tag{10.13}$$

where $F_0 = 1$, $F_k = 0$ for $k < 0$, $c_0 = c_1 = 1$, and $c_n = 0$ for $n \geq 2$. This recursion is now valid for $n \geq 0$. Let $F(x)$ be the generating function for $F_0, F_1, \ldots$. In the following series of equations, steps without explanation require only simple algebra.

$$\begin{aligned}
F(x) &= \sum_{n=0}^{\infty} F_n x^n && \text{by defintion} \\
&= \sum_{n=0}^{\infty} (F_{n-1} + F_{n-2} + c_n) x^n && \text{by (10.13)} \\
&= \sum_{n=0}^{\infty} \left(xF_{n-1}x^{n-1} + x^2 F_{n-2} x^{n-2} + c_n x^n \right) \\
&= x \sum_{n=0}^{\infty} F_{n-1} x^{n-1} + x^2 \sum_{n=0}^{\infty} F_{n-2} x^{n-2} + \sum_{n=0}^{\infty} a_n x^n \\
&= x \sum_{i=1}^{\infty} F_i x^i + x^2 \sum_{k=0}^{\infty} F_k x^k + 1 + x && \text{by definition} \\
&= xF(x) + x^2 F(x) + 1 + x.
\end{aligned}$$

In summary, $F(x) = 1 + x + (x + x^2)F(x)$. We can easily solve this equation:

$$F(x) = \frac{1+x}{1-x-x^2}. \tag{10.14}$$

Now what? We want to find a formula for the coefficient of x^n in $F(x)$. We could try using Taylor's Theorem. Unfortunately, $F^{(n)}(x)$ appears to be extremely messy. What alternative do we have?

Remember partial fractions from calculus? If not, you should read Appendix D (p. 387). Using partial fractions, we will be able to write $F(x) = A/(1 - ax) + B/(1 - bx)$ for some constants a, b, A and B. Since the formula for summing geometric series is $1 + ax + (ax)^2 + \cdots = 1/(1 - ax)$, we will have $F_n = Aa^n + Bb^n$. There is one somewhat sneaky point here. We want to factor a polynomial of the form $1 + cx + dx^2$ into $(1 - ax)(1 - bx)$. To do this, let $y = 1/x$ and multiply by y^2. The result is $y^2 + cy + d = (y - a)(y - b)$. Thus a and b are just the roots of $y^2 + cy + d = 0$. In our case we have $y^2 - y - 1 = 0$.

Let's carry out the partial fraction approach. We have

$$1 - x - x^2 = (1 - ax)(1 - bx) \qquad \text{where} \quad a,\, b = \frac{1 \pm \sqrt{5}}{2}.$$

(Work it out.) For definitiveness, let a be associated with the $+$ and b with the $-$. To get some idea of the numbers we are working with, $a = 1.618\cdots$ and $b = -.618\cdots$. By expanding in partial fractions, you should be able to derive

$$F(x) = \frac{1+x}{1-x-x^2} = \frac{1+a}{\sqrt{5}(1-ax)} - \frac{1+b}{\sqrt{5}(1-bx)}.$$

Now use geometric series and the algebraic observations $1 + a = a^2$ and $1 + b = b^2$ to get

$$F_n = \frac{a^{n+2}}{\sqrt{5}} - \frac{b^{n+2}}{\sqrt{5}}. \tag{10.15}$$

It is not obvious that this expression is even an integer, much less equal to F_n. If you're not convinced, you might like to calculate a few values.

Since $|b| < 1$, $\left|b^{n+2}/\sqrt{5}\right| < 1/\sqrt{5} < 1/2$. Thus we have the further observation that F_n is the integer closest to $a^{n+2}/\sqrt{5} = (1.618\cdots)^{n+2}/2.236\cdots$. For example $a^4/\sqrt{5} = 3.065\cdots$ which is close to $F_2 = 3$ and $a^{12}/\sqrt{5} = 144.001\cdots$, which is quite close to $F_{10} = 144$. Of course, the approximations get better as n gets larger since the error is bounded by a large power of b and $|b| < 1$. $\square$

The method that we have just used works for many other recursions, so it is useful to lay it out as a series of steps. Although our description is for a singly indexed recursion, it can be applied to the multiply indexed case as well.

A procedure for solving recursions *Here is a six step procedure for solving recursions. It is not guaranteed to work because it may not be possible to carry out all of the steps. Let the sequence be a_n.*

1. *Adjust the recursion so that it is valid for all n. In particular, a_n should be defined for all n and $a_n = 0$ for $n < 0$. You may need to introduce a "correcting" sequence c_n as in (10.13).*
2. *Introduce the generating function $A(x) = \sum_{n\ge 0} a_n x^n$.*
3. *Substitute the recursion into the summation for $A(x)$.*
4. *Rearrange the result so that you can recognize other occurrences of $A(x)$ and so get rid of summations. (This is not always possible; it depends on what the recursion is like.)*
5. *If possible, solve the resulting equation to obtain an explicit formula for $A(x)$.*
6. *By partial fractions, Taylor's Theorem or whatever, obtain an expression for the coefficient of x^n in this explicit formula for $A(x)$. This is a_n.*

You should go back to the previous example and find out where each step was done.

Example 10.3 Fibonacci numbers continued Setting $y = x$ in (10.1) gives $1/(1-x-x^2)$, which we'll call $H(x)$. This is nearly $F(x) = (1+x)/(1-x-x^2)$ of the previous example, suggesting that there is a connection between binomial coefficients and Fibonacci numbers. Let's explore this.

Writing $F(x)/(1+x) = H(x)$ is not a good idea since the coefficient of x^n on the left side is $F_n - F_{n-1} + F_{n-2} - \cdots$ and we'd like to find a simpler connection if we can. Writing the equation as $(1+x)H(x) = F(x)$ is better since the coefficient of x^n on the left side is just $h_n + h_{n-1}$.

It would be even better if we could avoid the factor of $(1+x)$ and have a monomial instead, since then we would not have to add two terms together. You might like to try to find something like that. After some work, we found that $1 + xF(x) = H(x)$, which is easily verified by using the formulas for $H(x)$ and $F(x)$. You should convince yourself that for $n > 0$ the coefficient of x^n on the left side is F_{n-1} and so $F_n = h_{n+1}$. In fact, some people call $1, 1, 2, 3, \ldots$ the Fibonacci numbers and then h_n is the nth Fibonacci number and $1/(1-x-x^2)$ is the generating function for the Fibonacci numbers. Still others call $0, 1, 1, 2, 3, \ldots$ the Fibonacci numbers and then $x/(1-x-x^2)$ is the generating function for them. Anyway, with $a_{j,i} = \binom{j}{i}$, *our* Fibonacci number F_n is the coefficient of x^{n+1} in $H(x)$. By (10.1),

$$H(x) \;=\; \sum_{j=0}^{\infty}\sum_{i=0}^{\infty}\binom{j}{i}x^i x^j.$$

Note that the coefficient of x^{n+1} on the right side is the sum of $\binom{j}{i}$ over all nonnegative i and j such that $i + j = n + 1$. Hence $F_n = \sum_{i=0}^{n+1}\binom{n+1-i}{i}$. This is such a simple expression that it should have a direct proof. We leave that as an exercise. $\square$

Example 10.4 The worst case time for merge sorting Let $M(n)$ be the maximum number of comparisons needed to merge sort a list of n items. (Merge sorting was discussed in Example 7.13 and elsewhere.) The best way to do a merge sort is to split the list as evenly as possible. If n is even, we can divide the list exactly in half. It takes at most $M(n/2)$ comparisons to merge sort each of the two halves and then at most $n-1<n$ comparisons to merge the two resulting lists. Thus $M(n) < n + 2M(n/2)$. We'd like to use this to define a recursion, but there's a problem: $n/2$ may not be even.

How can we avoid this? We can just look at those values of n which are powers of 2. For example, the fact that $M(1)=0$ gives us

$$\begin{aligned} M(8) &< 8+2M(4) \;<\; 8+2\big(4+2M(2)\big) \\ &< 8+2\Big(4+2\big(2+2M(1)\big)\Big) \;=\; 8+2(4+4) \;=\; 24. \end{aligned}$$

How can we set up a recursion that only looks at values of n which are a power of 2? We let $m_k = M(2^k)$. Then

$$m_0 = M(1) = 0 \quad\text{and}\quad m_k = M(2^k) < 2^k + 2M(2^{k-1}) = 2^k + 2m_{k-1}.$$

So far we have only talked about solving recursive relations that involve equality, but this is an inequality. What can we do about that?

If we define c_k by

$$c_0 = 0 \quad\text{and}\quad c_k = 2^k + 2c_{k-1} \text{ for } k>0, \qquad 10.16$$

then it follows that $m_k \le c_k$. We'll solve (10.16) and so get a bound for $m_k = M(2^k)$.

Before calculating the general solution, it may be useful to use the recursion to calculate a few values. This might lead us to guess what the solution is. Even if we can't guess the solution, we'll have some special cases of the general solution available so that we'll be able to partially check the general solution when we finally get it. It's a good idea to get in the habit of making such checks because it is *very easy* to make algebra errors when manipulating generating functions.

From (10.16), the first few values of c_k are

$$c_0 = 0,\quad c_1 = 2,\quad c_2 = 2\cdot 2^2 = 2^3,\quad c_3 = 3\cdot 2^3 \quad\text{and}\quad c_4 = 4\cdot 2^4.$$

This strongly suggests that $c_k = k2^k$. You should verify that this is correct by using (10.16) and induction.

Since we have the answer, why bother with generating functions? We want to study generating function techniques so that you can use them in situations where you can't guess the answer. This problem is a rather simple one, so the algebra won't obscure the use of the techniques.

For Step 1, rewrite (10.16) as

$$c_k \;=\; 2^k + 2c_{k-1} + a_k \quad\text{for}\quad k \ge 0,$$

where $c_k = 0$ for $k<0$, $a_0 = -1$, and $a_n = 0$ for $n>0$. Now

$$\begin{aligned} C(x) &= \sum_{k=0}^{\infty} c_k x^k && \text{This is Step 2.} \\ &= \sum_{k=0}^{\infty} (2^k + 2c_{k-1} + a_k)x^k && \text{This is Step 3.} \\ &= \sum_{k=0}^{\infty} (2x)^k 2x \sum_{k=0}^{\infty} c_{k-1}x^{k-1} - 1 \\ &= \frac{1}{1-2x} + 2xC(x) - 1. && \text{This is Step 4.} \end{aligned}$$

For Step 5 we have $C(x) = 2x/(1-2x)^2$. Partial fractions (Step 6) leads to

$$C(x) \;=\; \frac{1}{(1-2x)^2} - \frac{1}{1-2x} \;=\; \sum \binom{-2}{k}(-2x)^k - \sum (2x)^k.$$

Thus $c_k = 2^k\Big((-1)^k\binom{-2}{k} - 1\Big) = k2^k$. Hence $M(n) \le n\log_2 n$ when n is a power of 2. How good is this bound? What happens when n is not a power of 2? It turns out that $n\log_2 n$ is a fairly good estimate for $M(n)$ for all n, but we won't prove it. $\square$

Perhaps you've noticed that when we obtain a rational function (i.e., a quotient of two polynomials) as a generating function, the denominator is, in some sense, the important part. We can state this more precisely: For rational generating functions, the recursion determines the denominator and the initial conditions interacting with the recursion determine the numerator. No proof of this claim will be given here. A related observation is that, if we have the same denominators for two rational generating functions $A(x)$ and $B(x)$ that have been reduced to lowest terms, then the coefficients a_n and b_n have roughly the same rate of growth for large n; i.e., we usually have $a_n = \Theta(b_n)$.*

Example 10.5 Counting unlabeled full binary RP-trees Let b_n be the number of unlabeled full binary RP-trees with n leaves. By Example 9.4 (p. 251), the number of such trees is the Catalan number C_{n-1}. See Example 1.13 (p. 15) for more examples of things that are counted by the Catalan numbers.

The recursion

$$b_n \;=\; \sum_{k=1}^{n-1} b_k b_{n-k} \quad \text{if } n > 1 \qquad 10.17$$

with $b_1 = 1$ was derived as (9.3). Recall that $b_1 = 1$ and b_0 was not defined. Let's use our procedure to find b_n. Here it is, step by step.

1. Since (10.17) is nearly a convolution, we define $b_0 = 0$ to make it a convolution:

$$b_n \;=\; \sum_{k=0}^{n} b_k b_{n-k} + a_n,$$

where $a_1 = 1$ and $a_n = 0$ for $n \ne 1$.

2. Let $B(x) = \sum_{n\ge 0} b_n x^n$.
3. $B(x) = \sum_{n\ge 0}\sum_{k=0}^{n} b_k b_{n-k} x^n + x$.
4. By the formula for convolutions, we now have

$$B(x) \;=\; B(x)B(x) + x. \qquad 10.18$$

5. The quadratic equation $B = x + B^2$ has the solution $B = (1 \pm \sqrt{1-4x})/2$. Since $B(0) = b_0 = 0$, the minus sign is the correct choice. Thus

$$B(x) \;=\; \frac{1-\sqrt{1-4x}}{2}.$$

6. By Exercise 10.1.4,

$$(1+z)^r \;=\; \sum_{n=0}^{\infty}\binom{r}{n} z^n, \quad \text{where } \binom{r}{n} = \frac{r(r-1)\cdots(r-n+1)}{n!}.$$

* This notation is discussed in Appendix B. It means there exist positive constants A and B such that $Aa_n \le b_n \le Ba_n$.

Now for some algebra. With $n > 0$, $r = 1/2$ and $z = -4x$ we obtain

$$\begin{aligned}
b_n &= -\frac{1}{2}\binom{\frac{1}{2}}{n}(-4)^n \\
&= \left(\frac{1}{2}(2^n)\right)\left(\frac{\frac{1}{2}(\frac{1}{2}-1)(\frac{1}{2}-2)\cdots(\frac{1}{2}-n+1)}{n!}2(-2)^{n-1}\right) \\
&= \left(\frac{2^{n-1}(n-1)!}{(n-1)!}\right)\left(\frac{(-1+2)(-1+4)\cdots(-1+2n-2)}{n!}\right) \\
&= \left(\frac{2\cdot 4\cdots(2n-2)}{(n-1)!}\right)\left(\frac{1\cdot 3\cdots(2n-3)}{n!}\right) \\
&= \frac{(2n-2)!}{(n-1)!n!} \;=\; \frac{1}{n}\binom{2n-2}{n-1}.
\end{aligned}$$

As remarked at the beginning of the example, this number is the Catalan number C_{n-1}. Thus $C_n = \frac{1}{n+1}\binom{2n}{n}$. □

Exercises

10.2.1. Solve the following recursions by using generating functions.

(a) $a_0 = 0$, $a_1 = 1$ and $a_n = 5a_{n-1} - 6a_{n-2}$ for $n > 1$.

(b) $a_0 = a_1 = 1$ and $a_{n+1} = a_n + 6a_{n-1}$ for $n > 0$.

(c) $a_0 = 0$, $a_1 = a_2 = 1$ and $a_n = a_{n-1} + a_{n-2} + 2a_{n-3}$ for $n > 2$.

(d) $a_0 = 0$ and $a_n = 2a_{n-1} + n$ for $n > 0$.

10.2.2. Let $S(n)$ be the number of moves needed to solve the Towers of Hanoi puzzle. In Exercise 7.3.9 you were asked to show that $S(1) = 1$ and $S(n) = 2S(n-1) + 1$ for $n > 1$.

(a) Use this recursion to obtain the generating function for S.

(b) Use the generating function to determine $S(n)$.

10.2.3. Show without generating functions that $\binom{n+1-i}{i}$ is the number of n long sequences of zeroes and ones with exactly i ones, none of them adjacent. Use this result to prove the formula $F_n = \sum_{i\geq 0}\binom{n+1-i}{i}$ that was derived in the Example 10.3 via generating functions.

10.2.4. Let s_n be the number of n long sequences of zeroes, ones and twos with no adjacent ones and no adjacent twos. Let $s_0 = 1$; i.e., there is one empty sequence.

(a) Let k be the position of the last zero in such a sequence. If there is no zero, set $k = 0$. Show that the last $n - k$ elements in the sequence consist of an alternating pattern of ones and twos and that the only restriction on the first $k - 1$ elements in the sequence is that there be no adjacent ones and no adjacent twos.

(b) By considering all possibilities for k in (a), conclude that, for $n > 0$,

$$s_n \;=\; 2 + 2s_0 + 2s_1 + \cdots + 2s_{n-2} + s_{n-1}.$$

(c) Use the convolution formula to deduce

$$S(x) \;=\; (1 + 2x + 2x^2 + 2x^3 + \cdots)(1 + s_0x + s_1x^2 + s_2x^3 + \cdots) \;=\; \left(1 + \frac{2x}{1-x}\right)(1 + xS(x)).$$

(d) Conclude that $S(x) = (1 + x)/(1 - 2x - x^2)$.

(e) Find a formula for s_n and check it for $n = 0, 1, 2$.

(f) Show that s_n is the integer closest to $(1 + \sqrt{2})^{n+1}/2$.

10.2.5. The usual method for multiplying two polynomials of degree $n-1$, say

$$P_1(x) = a_{0,1} + a_{1,1}x + \cdots + a_{n-1,1}x^{n-1} \quad \text{and} \quad P_2(x) = a_{0,2} + a_{1,2}x + \cdots + a_{n-1,2}x^{n-1}$$

requires n^2 multiplications to form the products $a_{i,1}a_{j,2}$ for $0 \le i, j < n$. These are added together in the appropriate way to form the $2n-1$ sums that constitute the coefficients of the product $P_1(x)P_2(x)$. There is a less direct method that requires less multiplications. For simplicity, suppose that $n = 2m$.

- First, split the polynomials in "half": $P_i(x) = L_i(x) + x^m H_i(x)$, where L_i and H_i have degree at most $m-1$.
- Second, let $A = H_1H_2$, $B = L_1L_2$ and $C = (H_1 + L_1)(H_2 + L_2)$.
- Third, note that $P_1P_2 = Ax^{2m} + B + (C - A - B)x^m$.

(a) Prove that the formula for P_1P_2 is correct.

(b) Let $M(n)$ be the least number of multiplications we need in a general purpose algorithm for multiplying two polynomials of degree $n-1$. show that $M(2m) \le 3M(m)$.

(c) Use the previous result to derive an upper bound for $M(n)$ when n is a power of 2 that is better than n^2. (Your answer should be $M(n) \le n^c$ where $c = 1.58\cdots$.) How does this bound compare with n^2 when $n = 2^{10} = 1024$?

Your bound will give a bound for all n since, if $n \le 2^k$, we can fill the polynomials out to degree 2^k by introducing high degree terms with zero coefficients. This gives $M(n) \le M(2^k)$.

(d) Show how the method used to obtain the bound multiplies $1 + 2x - x^2 + 3x^3$ and $5 + 2x - x^3$.

*(e) It may be objected that our method could lead to such a large number of additions and subtractions that the savings in multiplication may be lost. Does this happen? Justify your answer.

10.2.6. Let t_n be the number of n-vertex unlabeled binary RP-trees. (Each vertex has 0, 1 or 2 children.)

(a) Derive the recursion

$$t_1 = 1 \quad \text{and} \quad t_{n+1} = t_n + \sum_{k=1}^{n-1} t_k t_{n-k} \quad \text{for} \quad n > 0.$$

(b) With $t_0 = 0$, derive an equation for the generating function $T(x) = \sum_{n\ge 0} t_n x^n$.

(c) Solve the equation in (b) to obtain

$$T(x) = \frac{1 - x - \sqrt{1 - 2x - 3x^2}}{2x}$$

and explain the choice of sign before the square root.

10.2.7. Let $c_1, \ldots, c_k$ be arbitrary real numbers. If you are familiar with partial fractions, explain why the solution to the recursion $a_n = c_1a_{n-1} + \cdots + c_k a_{k-k}$ has the form $a_n = \sum_{i=1}^m P_i(n) r_i^n$ for all sufficiently large n, where $P_i(n)$ is a polynomial of degree less than d_i, the r_i are all different, and

$$1 - c_1x - \cdots - c_kx^k = \prod_{i=1}^m (1 - r_ix)^{d_i}.$$

How can the polynomials $P_n(n)$ be found without using partial fractions?

10.3 Manipulating Generating Functions

Almost anything we do with generating functions can be regarded as manipulation, so what does the title of this section refer to? We mean the use of tools from algebra and calculus to obtain information from generating functions. We've already seen some examples of one tool being used: partial fractions. In this section we'll focus on two others; (i) the manipulation of generating functions to obtain, when possible, simple recursions and (ii) the interplay of derivatives with generating functions. Some familiarity with calculus is required. The results in this section are used some in later sections, but they are not essential for understanding the concepts introduced there.

Obtaining Recursions

Suppose we have an equation that determines a generating function $B(x)$; for example, $B(x) = \frac{1-\sqrt{1-4x}}{2}$. The basic idea for obtaining a recursion for $B(x)$ is to rewrite the equation so that $B(x)$ appears in expressions that are simple and so that the remaining expressions are easy to expand in power series. Once a simple form has been found, equate coefficients of x^n on the two sides of the equation. We'll explore this idea here.

Example 10.6 Rational functions and recursions Suppose that $B(x) = P(x)/Q(x)$ where $P(x)$ and $Q(x)$ are polynomials. Expressions that involve division are usually not easy to expand unless the divisor is a product of linear factors with integer coefficients. Thus, we would usually rewrite our equation as $Q(x)B(x) = P(x)$ and then equate coefficients. This gives us a recursion for the b_i's which is linear and has constant coefficients.

The description of the procedure is a bit vague, so let's look at an example. When we study systems of recursions in the next chapter, we will show that the number of ways to place nonoverlapping dominoes on a 2 by n board has the generating function

$$C(x) \;=\; \frac{1-x}{1-3x-x^2+x^3}.$$

Thus $P(x) = 1 - x$ and $Q(x) = 1 - 3x - x^2 + x^3$. Using our plan, we have

$$(1-3x-x^2+x^3)C(x) \;=\; 1-x. \qquad 10.19$$

There are now various ways we can proceed:

Keep all subscripts nonnegative: When $n \geq 3$, the coefficient of x^n on the right side is 0 and the coefficient on the left side is $c_n - 3c_{n-1} - c_{n-2} + c_{n-3}$, so all the subscripts are nonnegative. Rearranging this,

$$c_n \;=\; 3c_{n-1} + c_{n-2} - c_{n-3} \quad \text{for } n \geq 3.$$

The values of a_0, a_1 and a_2 are given by initial conditions. Looking at the coefficients of x^0, x^1 and x^2 on both sides of (10.19), we have

$$a_0 \;=\; 1 \qquad a_1 - 3a_0 \;=\; -1 \qquad a_2 - 3a_1 - a_0 \;=\; 0.$$

Solving we have $a_0 = 1$, $a_1 = 2$ and $a_2 = 7$. (You might want to try deriving the recursion directly. It's not easy, but it's not an unreasonable problem for you at this time.)

Allow negative subscripts: We now allow negative subscripts, with the understanding that $a_n = 0$ if $n < 0$. Proceeding as above, we get $c_n - 3c_{n-1} - c_{n-2} + c_{n-3} = 0$ provided $n \geq 2$. Thus we get the same recursion, but now $n \geq 2$ and the initial conditions are only $a_0 = 1$ and $a_1 = 2$ since a_3 is given by the recursion.

Avoid initial conditions: Now we not only allow negative subscripts, we also do not restrict n. From (10.19) we have

$$c_n - 3c_{n-1} - c_{n-2} + c_{n-3} = b_n, \quad \text{where } b_n = [x^n]\,(1-x).$$

Thus we have the recursion

$$c_n \;=\; 3c_{n-1} + c_{n-2} - c_{n-3} + b_n \quad \text{for } n \geq 0,$$

where $b_0 = 1$, $b_1 = -1$ and $b_n = 0$ otherwise. □

The ideas are not limited to ratios of polynomials, but then it's not always clear how to proceed. In the next example, we use the fact that e^{-x} has a simple power series.

Example 10.7 Derangements In the next chapter, we obtain, as (11.17) the formula

$$D(x) \;=\; \sum_{n=0}^{\infty} D_n x^n/n! \;=\; \frac{e^{-x}}{1-x}; \qquad 10.20$$

in other words, $e^{-x}/(1-x)$ is the ordinary generating function for the numbers $d_n = D_n/n!$. We can get rid of fractions in (10.20) by multiplying by $(1-x)$. Since

$$e^{-x} \;=\; \sum_{n=0}^{\infty} \frac{(-1)^n x^n}{n!},$$

equating coefficients of x^n on both sides of $(1-x)D(x) = e^{-x}$ gives us

$$\frac{D_n}{n!} - \frac{D_{n-1}}{(n-1)!} \;=\; \frac{(-1)^n}{n!}.$$

Rearranging leads to the recursion $D_n = nD_{n-1} + (-1)^n$. A direct combinatorial proof of this recursion is known, but it is difficult. □

One method for solving a differential equation is to write the unknown function as a power series $y(x) = \sum a_n x^n$, use the differential equation to obtain a recursion for the a_n, and finally use the recursion to obtain information about the a_n's and hence $y(x)$. Here we proceed differently. Sometimes a recursion may lead to a differential equation which can solved to obtain the generating function. Sometimes a differential equation can be found for a known generating function and then be used to obtain a recursion. We consider the latter approach in the next example. What sort of differential equation should we look for? Linear equations with polynomial coefficients give the simplest recursions.

Example 10.8 A recursion for unlabeled full binary RP-trees In Example 10.5 we found that the generating function for unlabeled full binary RP-trees is $B(x) = \frac{1-\sqrt{1-4x}}{2}$. We then obtained an explicit formula for b_n by expanding $\sqrt{1-4x}$ in a power series. Instead, we could obtain a differential equation which would lead to a recursion.

We can proceed in various ways to obtain a simple differential equation. One is to observe that $2B(x) - 1 = -(1-4x)^{1/2}$ and differentiate both sides to obtain $2B'(x) = 2(1-4x)^{-1/2}$. Multiply by $1-4x$:

$$2(1-4x)B'(x) = 2(1-4x)^{1/2} \;=\; -(2B(x)-1).$$

Thus $2B'(x) - 8xB'(x) + 2B(x) = 1$. Replacing $B(x)$ by its power series we obtain

$$\sum 2nb_nx^{n-1} - \sum 8nb_nx^n + \sum 2b_nx^n \;=\; 1.$$

Replacing the first sum by $\sum 2(n+1)b_{n+1}x^n$ and equating coefficients of x^n gives

$$2(n+1)b_{n+1} - 8nb_n + 2b_n \;=\; 0 \quad \text{for} \quad n > 0.$$

After some rearrangement, $b_{n+1} = (4n-2)b_n/(n+1)$ for $n > 0$. We already know that $b_1 = 1$, so we have the initial condition for the recursion. This recursion was obtained in Exercise 9.3.13 (p. 266) by a counting argument. $\square$

Derivatives, Averages and Probability

The fact that $xA'(x) = \sum na_nx^n$ can be quite useful in obtaining information about averages. We'll explain how this works and then look at some examples.

Let $\mathcal{A}_n$ be a set of objects of size n; for example, some kind of n-long sequences or some kind of n-vertex trees. For each n, make $\mathcal{A}_n$ into a probability space using the uniform distribution:

$$\Pr(\alpha) = \frac{1}{|\mathcal{A}_n|} \text{ for all } \alpha \in \mathcal{A}_n.$$

(Probability is discussed in Appendix C.) Suppose that for each n we have a random variable X_n on $\mathcal{A}_n$ that counts something; for example, the number of ones in a sequence or the number of leaves on a tree. The average value (average number of ones or average number of leaves) is then $\mathbf{E}(X_n)$.

Now let's look at this in generating function terms. Let $a_{n,k}$ be the number of $\alpha \in \mathcal{A}_n$ with $X_n(\alpha) = k$; for example, the number of n-long sequences with k ones or the number of n-vertex trees with k leaves. Let $A(x,y)$ be the generating function $\sum_{n,k} a_{n,k}x^ny^k$. By the definition of expectation and simple algebra,

$$\mathbf{E}(X_n) \;=\; \sum_k k\Pr(X_n{=}k) \;=\; \sum_k k\frac{a_{n,k}}{|\mathcal{A}_n|} \;=\; \frac{\sum_k ka_{n,k}}{|\mathcal{A}_n|} \;=\; \frac{\sum_k ka_{n,k}}{\sum_k a_{n,k}}.$$

Let's look at the two sums in the last fraction.

$$\text{Since } [x^n]\,A(x,y) = \textstyle\sum_k a_{n,k}y^k, \quad \sum_k a_{n,k} = [x^n]A(x,1).$$

$$\text{Since } [x^n]\,\frac{\partial A(x,y)}{\partial y} = \textstyle\sum_k ka_{n,k}y^{k-1}, \quad \sum_k ka_{n,k} = [x^n]A_y(x,1),$$

where A_y stands for $\partial A/\partial y$. Putting this all together,

$$\mathbf{E}(X_n) \;=\; \frac{[x^n]\,A_y(x,1)}{[x^n]\,A(x,1)}. \qquad 10.21$$

We can use the same idea to compute variance. Recall that $\mathbf{var}(X_n) = \mathbf{E}(X_n^2) - \mathbf{E}(X_n)^2$. Since (10.21) tells us how to compute $\mathbf{E}(X_n)$, all we need is a formula for $\mathbf{E}(X_n^2)$. This is just like the previous derivation except we need factors of k^2 multiplying $a_{n,k}$. We can get this by differentiating twice:

$$\sum_k k^2 a_{n,k} \;=\; [x^n]\,\frac{\partial(yA_y(x,y))}{\partial y}\bigg|_{y=1} \;=\; [x^n](A_{yy}(x,1)+A_y(x,1)). \qquad 10.22$$

This discussion has all been rather abstract. Let's apply it.

Example 10.9 Fibonacci sequences What is the average number of ones in an n long sequence of zeroes and ones containing no adjacent ones? We studied these sequences in Example 10.2 (p. 275), where we used the notation F_n. To be more in keeping with the previous discussion, let Let $f_{n,k}$ be the number of n long sequences containing exactly k ones. We need $F(x,y) = \sum_{n,k} f_{n,k}x^n y^k$.

In Example 10.16 we'll see how to compute $F(x,y)$ quickly, but for now the only tool we have is recursions, so it will take a bit longer. You should be able to extend the argument used to derive the recursion (10.12) to show that

$$f_{n,k} \;=\; f_{n-1,k} + f_{n-2,k-1} \quad \text{for} \quad n \geq 2, \qquad 10.23$$

provided we set $f_{n,k} = 0$ when $k < 0$. Let $F_n(y) = \sum_k f_{n,k}y^k$ and sum y^k times (10.23) over all k to obtain

$$F_n(y) \;=\; F_{n-1}(y) + yF_{n-2}(y) \quad \text{for } n \geq 2. \qquad 10.24$$

For $n = 0$ we have only the empty sequence and for $n = 1$ we have the two sequences 0 and 1. Thus, the initial conditions for (10.24) are $F_0(y) = 1$ and $F_1(y) = 1 + y$. Multiplying (10.24) by x^n and summing over $n \geq 2$, we obtain

$$F(x,y) - F_0(y) - xF_1(y) \;=\; x\big(F(x,y) - F_0(y)\big) + x^2yF(x,y).$$

Thus

$$F(x,y) \;=\; \frac{1+xy}{1-x-x^2y}. \qquad 10.25$$

We are now ready to use (10.21). From (10.25),

$$F_y(x,y) \;=\; \frac{x(1-x-x^2y)-(1+xy)(-x^2)}{(1-x-x^2y)^2} \;=\; \frac{x}{(1-x-x^2y)^2}$$

and so

$$F_y(x,1) \;=\; \frac{x}{(1-x-x^2)^2}.$$

Thus

$$[x^n]\,F_y(x,1) \;=\; [x^{n-1}]\,\frac{1}{(1-x-x^2)^2}.$$

This can be expanded by partial fractions in various ways. The easiest method is probably to use the ideas and formulas in Appendix D (p. 387), which we now do. With $a,b = (1 \pm \sqrt{5})/2$, as in Example 10.2, we have

$$\frac{1}{(1-x-x^2)^2} \;=\; \frac{1}{(1-ax)^2(1-bx)^2}.$$

We make use of the relations

$$a+b=1 \quad ab=-1 \quad \text{and} \quad a-b=\sqrt{5}.$$

Here are the calculations

$$\begin{aligned}\frac{1}{(1-ax)^2(1-bx)^2} &= \left(\frac{a/\sqrt{5}}{1-ax} - \frac{b/\sqrt{5}}{1-bx}\right)^2\\ &= \frac{a^2/5}{(1-ax)^2} - \frac{2ab/5}{(1-ax)(1-bx)} + \frac{b^2/5}{(1-bx)^2}\\ &= \frac{a^2/5}{(1-ax)^2} + \frac{2a/5\sqrt{5}}{1-ax} - \frac{2b/5\sqrt{5}}{1-bx} + \frac{b^2/5}{(1-bx)^2}.\end{aligned}$$

Thus

$$\begin{aligned}[x^n]\,F_y(x,1) &= \frac{a^2}{5}\binom{-2}{n-1}(-a)^{n-1} + \frac{2a}{5\sqrt{5}}a^{n-1} - \frac{2b}{5\sqrt{5}}b^{n-1} + \frac{b^2}{5}\binom{-2}{n-1}(-b)^{n-1}\\ &= \frac{na^{n+1}}{5} + \frac{2a^n}{5\sqrt{5}} - \frac{2b^n}{5\sqrt{5}} + \frac{nb^{n+1}}{5}.\end{aligned}$$

Since $|b| < .62$, the last two terms in this expression are fairly small. In fact, we will show that w_n is the integer closest to

$$\frac{a^n}{5}\big(an + 2/\sqrt{5}\big).$$

Using the expression (10.15) for $\sum_k f_{n,k}$, the average number of ones is very close to

$$\frac{n}{a\sqrt{5}} + \frac{2}{5a^2}.$$

We must prove our claim about the smallness of the terms involving b. It suffices to show that their sum is less than 1/2. Since $|b| = (\sqrt{5}-1)/2 < 1$, we have

$$\frac{2|b|^n}{5\sqrt{5}} \;\le\; \frac{2b}{5\sqrt{5}} \;<\; 0.12.$$

The term $n|b|^{n+1}/5$ is a bit more complicated. We study it as a function of n to find its maximum. Its derivative with respect to n is

$$\frac{|b|^{n+1}}{5} + \frac{n\ln|b|\;|b|^{n+1}}{5} \;=\; \frac{|b|^{n+1}}{5}(1+n\ln|b|).$$

Since $-0.25 < \ln|b| < -0.2$, this is positive for $n \le 4$ and negative for $n \ge 5$. It follows that the term achieves its maximum at $n = 4$ or at $n = 5$. The values of these two terms are

$$4|b|^5/5 \;<\; |b|^5 \;<\; 0.1 \quad\text{and}\quad 5|b|^6/5 \;<\; |b|^5 \;<\; 0.1,$$

proving our claim. □

Example 10.10 Leaves in trees What can we say about the number of leaves in n-vertex unlabeled RP-trees? We'll study the average number of leaves and the variance using (10.21) and (10.22).

Let $t_{n,k}$ be the number of unlabeled RP-trees having n vertices and k leaves and let $T(x,y)$ be $\sum_{n,k} t_{n,k}x^ny^k$. Using tools at our disposal, it is not easy to work out the generating function for $T(x,y)$. On the other hand, after you have read the next section, you should be able to show that

$$T(x,y) \;=\; xy + xT(x,y) + x(T(x,y))^2 + \cdots + x(T(x,y))^i + \cdots,$$

where $x(T(x,y))^i$ comes from building trees whose roots have degree i. We'll assume this has been done. Summing the geometric series in (10.25), we have

$$T(x,y) \;=\; xy + \frac{xT(x,y)}{1-T(x,y)}.$$

Clearing of fractions and rearranging:

$$(T(x,y))^2 - (1-x+xy)T(x,y) + xy \;=\; 0,$$

a quadratic equation in $T(,y)$ whose solution is

$$T(x,y) \;=\; \frac{1-x+xy \pm \sqrt{(1-x+xy)^2-4xy}}{2} \;=\; \frac{1-x+xy \pm \sqrt{(1+x-xy)^2-4x}}{2}.$$

Do we use the plus sign or the minus sign? Since there are no trees with no vertices $t_{0,0} = 0$. On the other hand,

$$t_{0,0} \;=\; T(0,0) \;=\; \frac{1 \pm \sqrt{1}}{2}$$

and so we want the minus sign. We finally have $T(x,y)$. Let's multiply be 2 to get rid of the annoying fraction:

$$2T(x,y) \;=\; 1-x+xy - \big((1+x-xy)^2-4x\big)^{1/2}.$$

Differentiating with respect to y, we have

$$2T_y(x,y) \;=\; x + x(1+x-xy)\big((1+x-xy)^2-4x\big)^{-1/2}$$

and

$$2T_{yy}(x,y) \;=\; -x^2\big((1+x-xy)^2-4x\big)^{-1/2} + x^2(1+x-xy)^2\big((1+x-xy)^2-4x\big)^{-3/2}.$$

Thus

$$\begin{aligned} 2T(x,1) &= 1-(1-4x)^{1/2}, \\ 2T_y(x,1) &= x + x(1-4x)^{-1/2}, \\ 2T_{yy}(x,1) &= -x^2(1-4x)^{-1/2} + x^2(1-4x)^{-3/2}. \end{aligned}$$

For $n > 2$ we have

$$\begin{aligned} 2[x^n]\,T(x,1) &= -(-4)^n\binom{1/2}{n}, \\ 2[x^n]\,T_y(x,1) &= [x^{n-1}]\,(1-4x)^{-1/2} \;=\; (-4)^{n-1}\binom{-1/2}{n-1}, \\ 2[x^n]\,T_{yy}(x,1) &= -[x^{n-2}]\,(1-4x)^{-1/2} + [x^{n-2}]\,(1-4x)^{-3/2} \\ &= -(-4)^{n-2}\binom{-1/2}{n-2} + (-4)^{n-2}\binom{-3/2}{n-2}. \end{aligned}$$

Let X_n be the number of leaves in a random n-vertex tree and suppose $n > 2$. Then

$$\begin{aligned} \mathbf{E}(X_n) &= \frac{2[x^n]\,T_y(x,1)}{2[x^n]\,T(x,1)} \;=\; \frac{\binom{-1/2}{n-1}}{-(-4)\binom{1/2}{n}} \\ &= \frac{\dfrac{(-1/2)(-3/2)\cdots(-1/2-(n-2))}{(n-1)!}}{4\dfrac{(1/2)(-1/2)\cdots(1/2-(n-1))}{n!}} \;=\; \frac{n!}{4(1/2)\,(n-1)!} \;=\; \frac{n}{2} \end{aligned}$$

and, recalling (10.22),

$$\begin{aligned}
\mathbf{E}(X_n^2) &= \frac{2[x^n]\,T_{yy}(x,1)}{2[x^n]\,T(x,1)} + \frac{2[x^n]\,T_y(x,1)}{2[x^n]\,T(x,1)} = \left(\frac{\binom{-1/2}{n-2}}{4^2\binom{1/2}{n}} - \frac{\binom{-3/2}{n-2}}{4^2\binom{1/2}{n}}\right) + \frac{n}{2} \\
&= \frac{\dfrac{(-1/2)\cdots(-1/2-(n-3))}{(n-2)!}}{4^2\dfrac{(1/2)\cdots(1/2-(n-1))}{n!}} - \frac{\dfrac{(-3/2)\cdots(-3/2-(n-3))}{(n-2)!}}{4^2\dfrac{(1/2)\cdots(1/2-(n-1))}{n!}} + \frac{n}{2} \\
&= \frac{n!}{4^2(1/2)(1/2-(n-1))\,(n-2)!} - \frac{n!}{4^2(1/2)(-1/2)\,n!} + \frac{n}{2} \\
&= \frac{n(n-1)}{4(3-2n)} + \frac{n(n-1)}{4} + \frac{n}{2} \\
&= \frac{n^2+n}{4} - \frac{n(n-1)}{4(2n-3)}.
\end{aligned}$$

Thus

$$\begin{aligned}
\mathbf{var}(X_n) &= \mathbf{E}(X_n^2) - (\mathbf{E}(X_n))^2 = \left(\frac{n^2+n}{4} - \frac{n(n-1)}{4(2n-3)}\right) - \frac{n^2}{4} \\
&= \frac{n}{4} - \frac{n(n-1)}{4(2n-3)} = \frac{n\big((2n-3)-(n-1)\big)}{4(2n-3)} = \frac{n(n-2)}{4(2n-3)}.
\end{aligned}$$

For large n this is nearly $n/8$.

We've shown that the average number of leaves in an RP-tree is $n/2$ and the variance in the number of leaves is about $n/8$. By Chebyshev's inequality (C.3) (p. 385), it follows that, in most large RP-trees, about half the vertices are leaves. More precisely:

$$\text{It is unlikely that} \quad \frac{|(\text{number of leaves}) - n/2|}{\sqrt{n/8}} \quad \text{will be large.}$$

By Exercise 5.4.8 (p. 140), every N-vertex full binary tree has exactly $\frac{N+1}{2}$ leaves, very slightly larger than the average over all trees. Since a tree that has many edges out of nonleaf vertices will have more leaves, it would seem that a full binary tree should have relatively few leaves. What is going on? Random RP-trees must have many nonleaf vertices with only one child, counterbalancing those with many children so that the average comes out to be nearly two. □

***Example 10.11 Average distance to a leaf** What is the average distance to the leaf in a random full binary RP-tree?

Before answering this question, we need to say precisely what it means. If T is an unlabeled full binary RP-tree, let $d(T)$ be the sum of the distances from the root to each of the leaves of the tree. (The distance from the root to a leaf is the number of edges on the unique path joining them.) We want the average value of $d(T)/n$ over all unlabeled n leaf full binary RP-trees. This average can be important because many algorithms involve traversing such trees from the root to a leaf and the time required is proportional to the distance.

Let $D(x) = \sum d(T)x^{w(T)}$, where the sum ranges over all unlabeled full binary RP-trees T and $w(T)$ is the number of leaves in T. Let $B(x) = \sum x^{w(T)}$. By Example 10.5

$$B(x) = \frac{1-\sqrt{1-4x}}{2} \quad \text{and} \quad b_n = -\frac{1}{2}\binom{\frac{1}{2}}{n}(-4)^n = \frac{1}{n}\binom{2n-2}{n-1}.$$

Suppose that T has more than one leaf. Let T_1 and T_2 be the two principal subtrees of T; that is, the two trees whose roots are the sons of the root of T. You should be able to show that

$$d(T) = w(T) + d(T_1) + d(T_2).$$

Multiply this by $x^{w(T)}$ and sum over all T with more than one leaf. Since $d(\bullet) = 0$ and $w(T) = w(T_1) + w(T_2)$, we have

$$\begin{aligned} D(x) &= \sum_{n>1} nb_n x^n + \sum_{T_1,T_2} d(T_1) x^{w(T_1)+w(T_2)} + \sum_{T_1,T_2} d(T_2) x^{w(T_1)+w(T_2)} \\ &= xB'(x) - x + D(x)B(x) + B(x)D(x). \end{aligned}$$

Thus

$$D(x) = \frac{xB'(x) - x}{1 - 2B(x)} = \frac{1}{\sqrt{1-4x}}\left(\frac{x}{\sqrt{1-4x}} - x\right) = \frac{x}{1-4x} - \frac{x}{\sqrt{1-4x}}.$$

It follows that

$$d_n = 4^{n-1} - \binom{-\frac{1}{2}}{n-1}(-4)^{n-1} = 4^{n-1} + \frac{n}{2}\binom{\frac{1}{2}}{n-1}(-4)^n = 4^{n-1} - nb_n$$

and so the average distance to a leaf is

$$\frac{4^{n-1}}{nb_n} - 1 = \frac{4^{n-1}}{\binom{2n-2}{n-1}} - 1.$$

Using Stirling's formula, it can be shown that this is asymptotic to $\sqrt{\pi n}$.

This number is fairly small compared to n. We could do much better by limiting ourselves to averaging over certain subclasses of binary RP-trees. For example, we saw in Chapter 8 that if the distances to the leaves of the tree are all about equal, then the average and *largest* distances are both only about $\log_2 n$. Thus, when designing algorithms that use trees as data structures, restricting the shape of the tree could lead to significant savings. Good information storage and retrieval algorithms are designed on this basis. □

***Example 10.12 The average time for Quicksort** We want to find out how long it takes to sort a list using Quicksort. Quicksort was discussed briefly in Chapter 8. We'll review it here. Given a list $a_1, a_2, \ldots, a_n$, Quicksort selects an element x, divides the list into two parts (greater and less than x) and sorts each part by calling itself. There are two problems. First, we haven't been specific enough in our description. Second, the time Quicksort takes depends on the order of the list and the way x is chosen at each call. To avoid the dependence on order, we will average over all possible arrangements. We now give a more specific description using $x = a_1$. Given a list $a_1, a_2, \ldots, a_n$ of distinct elements, we create a new list $s_1, s_2, \ldots, s_n$ with the following properties.

(a) For some $1 \le k \le n$, $s_k = a_1$.

(b) $s_i < a_1$ for $i < k$ and $s_i > a_1$ for $i > k$.

(c) The relative order of the elements in the two sublists is the same as in the original list; i.e., if $s_i = a_p$, $s_j = a_q$ and either $i < j < k$ or $k < i < j$, then $p < q$.

It turns out that this can be done with $n - 1$ comparisons. We now apply Quicksort recursively to $s_1, \ldots, s_{k-1}$ and to $s_{k+1}, \ldots, s_n$.

Let q_n be the average number of comparisons needed to Quicksort an n long list. Thus $q_1 = 0$. We define $q_0 = 0$ for convenience later.

Note that k is the position of a_1 in the sorted list. Since the original list is random, all values of k from 1 to n are equally likely. By analyzing the algorithm carefully, it can be shown that all orderings of $s_1, \ldots, s_{k-1}$ are equally likely as are all orderings of $s_{k+1}, \ldots, s_n$. (We will not do this.) Thus, given k, it follows that the average length of time needed to sort both $s_1, \ldots, s_{k-1}$ and $s_{k+1}, \ldots, s_n$ is $q_{k-1} + q_{n-k}$.

Averaging over all possible values of k and remembering to include the original $n - 1$ comparisons, we obtain

$$q_n = n - 1 + \frac{1}{n}\sum_{k=1}^{n}\bigl(q_{k-1} + q_{n-k}\bigr) = n - 1 + \frac{2}{n}\sum_{j=0}^{n-1} q_j,$$

which is valid for $n > 0$.

To solve this recursion by generating functions, we should let $Q(x) = \sum q_n x^n$ and use the recursion to get a relation for $Q(x)$. If we simply substitute, we obtain

$$Q(x) \;=\; q_0 + \sum_{n=1}^{\infty}\left(n-1+\frac{2}{n}\sum_{j=0}^{n-1} q_j\right)x^n. \qquad 10.26$$

If we try to manipulate this to simplify the double sum over n and j of $2q_j x^n/n$, we will run into problems because of the n in the denominator. How can we deal with this?

One approach would be to multiply the original recursion by n before we use it. Another approach, which it turns out is equivalent, is to differentiate (10.26) with respect to x. Which is better? The latter is easier when we have a denominator as simple as n, but the former may be better when we have more complicated expressions. We use the latter approach. Differentiating (10.26), we have

$$\begin{aligned} Q'(x) \;&=\; \sum_{n=1}^{\infty}\left((n-1)n + 2\sum_{j=0}^{n-1} q_j\right)x^{n-1} \;=\; \sum_{n=1}^{\infty} n(n-1)x^{n-1} + 2\sum_{n=1}^{\infty}\sum_{j=0}^{n-1} q_j x^{n-1} \\ &=\; x\left(\frac{1}{1-x}\right)'' + 2\sum_{k=0}^{\infty}\sum_{j=0}^{k} q_j x^k \;=\; \frac{2x}{(1-x)^3} + 2Q(x)\,\frac{1}{1-x}, \end{aligned}$$

where $Q(x)/(1-x)$ follows either by recognizing that we have a convolution or by applying Exercise 10.1.6 (p. 274).

Rearranging, we see that we must solve the differential equation

$$Q'(x) - 2(1-x)^{-1}Q(x) \;=\; 2x(1-x)^{-3}, \qquad 10.27$$

which is known as a linear first order differential equation. This can be solved by standard methods from the theory of differential equations. We leave it as an exercise to show that the solution is

$$Q(x) \;=\; \frac{-2\ln(1-x) - 2x + C}{(1-x)^2}, \qquad 10.28$$

where the constant C must be determined by an initial condition. Since $Q(0) = q_0 = 0$, we have $C = 0$.

Using the Taylor series

$$-\ln(1-x) \;=\; \sum \frac{x^k}{k}$$

and some algebra, one eventually obtains

$$q_n = 2(n+1)\sum_{k=1}^{n}\frac{1}{k} - 4n. \qquad 10.29$$

Again, details are left as an exercise.

Using Riemann sum approximations, we have

$$\sum_{k=2}^{n}\frac{1}{k} \;<\; \int_1^n \frac{dx}{x} \;<\; \sum_{k=1}^{n-1}\frac{1}{k},$$

from which it follows that the summation in (10.29) equals $\ln n + O(1)$. It follows that

$$q_n \;=\; 2n\ln n + O(n) \quad \text{as } n\to\infty. \qquad 10.30$$

This is not quite as small as the result $n\log_2 n$ that we obtained for worst case merge sorting of a list of length $n = 2^k$; however, merge sorting requires an extra array but Quicksort does not because the array $s_1,\ldots,s_n$ can simply replace the array $a_1,\ldots,a_n$. (Actually, merge sorting can be done "in place" if more time is spent on merging. The Batcher sort is an in place merge sort.) You might like to compare this with Exercise 8.2.10 (p. 238), where we obtained an estimate of $1.78\,n\ln n$ for q_n. □

Exercises

10.3.1. Let $D(x)$ be the "exponential" generating function for the number of derangements as in Example 10.7. You'll use (10.20) to derive a linear differential equation with polynomial coefficients for $D(x)$. Then you'll equate coefficients to get a recursion for D_n.

(a) Differentiate $(1-x)D(x) = e^{-x}$ and the use $e^{-x} = (1-x)D(x)$ to eliminate e^{-x}.

(b) Equate coefficients to obtain $D_{n+1} = n(D_n + D_{n-1})$ for $n > 0$. What are the initial conditions?

10.3.2. A "path" of length n is a sequence $0 = u_0, u_1, \ldots, u_n = 0$ of nonnegative integers such that $u_{k+1} - u_k \in \{-1, 0, 1\}$ for $k < n$. Let a_n be the number of such paths of length n The OGF for a_n can be shown to be $A(x) = (1-2x-3x^2)^{-1/2}$.

(a) Show that $(1-2x-3x^2)A'(x) = (1+3x)A(x)$.

(b) Obtain the recursion

$$(n+1)a_{n+1} = (2n+1)a_n + 3na_{n-1} \quad \text{for} \quad n > 0.$$

What are the initial conditions?

(c) Use the general binomial theorem to expand $(1-(2x+3x^2))^{-1/2}$ and then the binomial theorem to expand $(2x+3x^2)^k$. Finally look at the coefficient of x^n to obtain a_n as a sum involving binomial coefficients.

10.3.3. Fill in the steps in the derivation of the average time formula for Quicksort:

(a) Solve (10.27) to obtain (10.28) by using an integrating factor or any other method you wish.

(b) Obtain (10.29) from (10.28).

10.3.4. In Exercise 10.2.6, you derived the formula

$$T(x) = \frac{1-x-\sqrt{1-2x-3x^2}}{2x}.$$

Use the methods of this section to derive a recursion for t_n that is simpler than the summation in Exercise 10.2.6(a).
Hint. Since the manipulations involve a fair bit of algebra, it's a good idea to check your recursion for t_n by comparing it with actual value for small n. They can be determined by constructing the trees.

10.4 The Rules of Sum and Product

Before the 1960's, combinatorial constructions and generating function equations were, at best, poorly integrated. A common route to a generating function was:

1. Obtain a combinatorial description of how to construct the structures of interest; e.g., the recursive description of unlabeled full binary RP-trees.
2. Translate the combinatorial description into equations relating elements of the sequence that enumerate the objects; e.g., $b_n = \sum_{k=1}^{n-1} b_k b_{n-k}$, for $n > 1$ and $b_1 = 1$.
3. Introduce a generating function for the sequence and substitute the equations into the generating function. Apply algebraic manipulation.
4. The result is a relation for the generating function.

From the 1960's on, various people have developed methods for going directly from a combinatorial construction to a generating function expression, eliminating Steps 2 and 3. These methods often allow us to proceed from Step 1 directly to Step 4. The Rules of Sum and Product for generating functions are basic tools in this approach. We study them in this section.

So far we have been thinking of generating functions as being associated with a sequence of numbers $a_0, a_1, \ldots$ which usually happen to be counting something. It is often helpful to think more directly about what is being counted. For example, let $\mathcal{B}$ be the set of unlabeled full binary RP-trees. For $B \in \mathcal{B}$, let $w(B)$ be the number of leaves of B. Then b_n is simply the number of $B \in \mathcal{B}$ with $w(B) = n$ and so

$$\sum_{B\in\mathcal{B}} x^{w(B)} \;=\; \sum_n b_n x^n = B(x). \qquad 10.31$$

We say that $B(x)$ counts unlabeled full binary RP-trees by number of leaves. It is sometimes convenient to refer to the generating function by the set that is associated with it. In this case, the set is $\mathcal{B}$ so we use the notation $\mathsf{G}_{\mathcal{B}}(x)$ or simply $\mathsf{G}_{\mathcal{B}}$. Thus, instead of asking for the generating function for the b_n's, we can just as well ask for the generating function for unlabeled full binary RP-trees (by number of leaves). Similarly, instead of asking for the generating function for F_n, we can ask for the generating function for sequences of zeroes and ones with no adjacent ones (by the length of the sequence). When it is clear, we may omit the phrase "by number of leaves," or whatever it is we are counting things by. We could also keep track of more than one thing simultaneously, like the length of a sequence *and* the number of ones. We won't pursue that now.

As noted above, if $\mathcal{T}$ is some set of structures (e.g., $\mathcal{T} = \mathcal{B}$), we let $\mathsf{G}_{\mathcal{T}}$ be the generating function for $\mathcal{T}$, with respect to whatever we are counting the structures in $\mathcal{T}$ by (e.g., leaves in (10.31)).

The Rule of Sum for generating functions is nothing more than a restatement of the Rule of Sum for counting that we developed in Chapter 1. The Rule of Product is a bit more complex. At this point, you may find it helpful to look back at the Rules of Sum and Product for counting: Theorem 1.2 (p. 6) and Theorem 1.3 (p. 8).

Theorem 10.3 Rule of Sum *Suppose a set $\mathcal{T}$ of structures can be partitioned into sets $\mathcal{T}_1, \ldots, \mathcal{T}_j$ so that each structure in $\mathcal{T}$ appears in exactly one $\mathcal{T}_i$. It then follows that*

$$\mathsf{G}_{\mathcal{T}}(x) \;=\; \mathsf{G}_{\mathcal{T}_1}(x) + \cdots + \mathsf{G}_{\mathcal{T}_j}(x).$$

The Rule of Sum remains valid when the number of blocks in the partition $\mathcal{T}_1, \mathcal{T}_2, \ldots$ is infinite.

Theorem 10.4 Rule of Product *Let w be a function that counts something in structures. Suppose each T in a set $\mathcal{T}$ of structures is constructed from a sequence $T_1, \ldots, T_k$ of k structures such that*

(*i*) *the possible structures T_i for the ith choice may depend on previous choices, but the generating function for them does not depend on previous choices,*

(*ii*) *each structure arises in exactly one way in this process and*

(*iii*) *if the structure T comes from the sequence $T_1, \ldots, T_k$, then*

$$w(T) \;=\; w(T_1) + \ldots + w(T_k).$$

It then follows that

$$\mathsf{G}_{\mathcal{T}}(x) \;=\; \sum_{T\in\mathcal{T}} x^{w(T)} \;=\; \mathsf{G}_1(x) \cdots \mathsf{G}_k(x), \qquad 10.32$$

where G_i is the generating function for the possible choices for the ith structure.

The Rule of Product remains valid when the number of steps is infinite.

As with the Rule of Product for counting, the available choices for the ith step may depend on the previous choices, but the generating function must not. If the choices at the ith step do not depend on the previous choices, we can think of $\mathcal{T}$ as simply a Cartesian product $\mathcal{T}_1 \times \cdots \times \mathcal{T}_k$.

The additivity condition (iii) is needed to insure that multiplication works correctly, namely

$$x^{w(T)} \;=\; x^{w(T_1)} \cdots x^{w(T_k)}.$$

Weights that count things (e.g., leaves in trees, cycles in a permutation, size of set being partitioned) usually satisfy (iii). This is not always the case; for example, counting the number of distinct things (e.g., cycle lengths in a permutation) is usually not additive. Weights dealing with a maximum (e.g., longest path from root to leaf in a tree, longest cycle in a permutation) do *not* satisfy (iii).

Proof: We will prove (10.32) by induction on k, starting with $k = 2$. The induction step is practically trivial—simply group the first $k-1$ choices together as one choice, apply the theorem for $k = 2$ to this grouped choice and the kth choice, and then apply the theorem for $k-1$ to the grouped choice.

The proof for $k = 2$ can be obtained by applying of the Rules of Sum and Product for counting as follows. Let $t_{i,j}$ be the number of ways to choose the ith structure so that it contains exactly j of the objects we are counting; that is, the number of ways to choose T_i so that $w(T_i) = j$. The number of ways to choose T_1 so that it contains j objects AND then choose T_2 so that together T_1 and T_2 contain n objects is $t_{1,j}\, t_{2,n-j}$. Thus, the total number of structures in $\mathcal{T}$ that contain exactly n objects is

$$\sum_{j=0}^{n} t_{1,j}\, t_{2,n-j}.$$

Multiplying by x^n, summing over n and recognizing that we have a convolution, we obtain (10.32) for $k = 2$.

Compare the proof we have just given for $k = 2$ with the following. By hypotheses (ii) and (iii) of the theorem,

$$\sum_{T \in \mathcal{T}} x^{w(T)} \;=\; \sum_{T_1 \in \mathcal{T}_1} x^{w(T_1)} \Bigg(\sum_{T_2 \in \mathcal{T}_2} x^{w(T_2)} \Bigg).$$

By hypothesis (i), the inner sum equals G_2 even though $\mathcal{T}_2$ may depend on T_1. Thus the above expression becomes $\mathsf{G}_1\mathsf{G}_2$. While this might seem almost magical, it's a perfectly valid proof. The lesson here is that it's often easier to sum over structures than to sum over indices.

Passing to the infinite case in the theorems is essentially a matter of taking a limit. We omit the proof. □

Example 10.13 Binomial coefficients Let's apply these theorems to enumerating binomial coefficients. Our structures will be subsets of $\underline{n}$ and we will be keeping track of the number of elements in a subset; i.e., $w(S) = |S|$, the number of elements in S. We form all subsets exactly once by a sequence of n choices. The ith choice will be either $\emptyset$ (the empty set) or the set $\{i\}$. The union of our choices will be a subset. The Rule of Product can be applied. Since $w(\emptyset) = 0$ and $w(\{i\}) = 1$, $\mathsf{G}_i(x) = 1 + x$ by the Rule of Sum. Thus the generating function for subsets of $\underline{n}$ by cardinality is $(1+x)\cdots(1+x) = (1+x)^n$. Compare this with the derivation in Example 1.14 (p. 19). Because this problem is so simple and because you are not familiar with using our two theorems, you may find the derivation in Example 1.14 easier than the one here. Read on. □

Example 10.14 Counting unlabeled RP-trees Let's look at unlabeled RP-trees from this new vantage point. If a tree has more that one vertex, let $s_1, \ldots, s_k$ be the sons of the root from left to right. We can describe such a tree by listing the k subtrees $T_1, \ldots, T_k$ whose roots are $s_1, \ldots, s_k$. This gives us a k-tuple. Note that T has as many leaves as $T_1, \ldots, T_k$ together. In fact, if you look back to the start of Chapter 9, you will see that this is nothing more nor less than the definition we gave there.

Let $B(x)$ be the generating function for unlabeled full binary unlabeled RP-trees by number of leaves. By the previous paragraph, an unlabeled full binary RP-tree is either one vertex OR a 2-tuple of unlabeled full binary RP-trees (joined to a new root). Applying the Rules of Sum and Product with $j = k = 2$, we have

$$\mathsf{G}_{\mathcal{B}}(x) \;=\; x + \mathsf{G}_{\mathcal{B}}(x)\,\mathsf{G}_{\mathcal{B}}(x),$$

which can also be written

$$B(x) \;=\; x + B(x)B(x).$$

This is much easier than deriving the recursion first—compare this derivation with the one in Example 10.5 (p. 279).

Now let's count arbitrary unlabeled RP-trees. In this case, we cannot count them by leaves because there are an infinite number of trees with just one leaf: any path is such a tree. We'll count them by vertices. Let $T(x)$ be the generating function. Proceeding as in the previous paragraph, we say that such a tree is either a single vertex, OR one tree, OR a 2-tuple of trees, OR a 3-tuple of trees, and so on. Thus we (incorrectly) write $T(x) = x + T(x) + T^2(x) + \cdots$. Why is this wrong? We did not apply the Rule of Product correctly. The number of vertices in a tree T is *not* equal to the total number of vertices in the k-tuple $(T_1, \ldots, T_k)$ that comes from the sons of the root: We forgot that there is one more vertex, the root of T.

Let's do this correctly. Instead of a k-tuple of trees, we have a vertex AND a k-tuple of trees. Thus a tree is either a single vertex, OR a single vertex AND a tree, OR a single vertex AND a 2-tuple of trees, and so on. Now we get (correctly)

$$T(x) \;=\; x + xT(x) + xT^2(x) + \cdots \;=\; \frac{x}{1 - T(x)},$$

by the Rules of Sum and Product and the formula for a sum of a geometric series. Multiplying by $1 - T(x)$, we have $T(x) - T^2(x) = x$, which is the same as the equation for $B(x)$. Thus

> **Theorem 10.5** *The number of n vertex unlabeled RP-trees equals the number of n leaf unlabeled full binary RP-trees.*

This was proved in Example 7.9 (p. 206) by showing that the numbers satisfied the same recursion and in Exercise 9.3.12 (p. 266) by giving a bijection.

You should be able to derive $T(x) = x + T(x)^2$ directly from the second definition of RP-trees in Example 7.9 (p. 206) and hence prove the theorem this way.

We've looked at two extremes: full binary trees (all nonleaf vertices have exactly 2 children) and arbitrary trees (nonleaf vertices can have any number of children). We can study trees in between these two extremes. Let D be a set of positive integers. Let $\mathcal{D}$ be those unlabeled RP-trees where the number of children of each vertex lies in D. The two extremes correspond to $D = \{2\}$ and $D = \{1, 2, 3, \ldots\}$. If we count these trees by number of vertices, you should be able to show that

$$\mathsf{G}_{\mathcal{D}}(x) \;=\; x + \sum_{d \in \mathcal{D}} x\mathsf{G}_{\mathcal{D}}(x)^d.$$

In general, we cannot solve this equation; however, we can simplify the sum if the elements of $\mathcal{D}$ lie in an arithmetic progression. Our two extremes are examples of this. For another example, suppose $\mathcal{D}$ is the set of positive odd integers. Then the sum is a geometric series with first term $x\mathsf{G}_{\mathcal{D}}(x)$ and ratio $\mathsf{G}_{\mathcal{D}}(x)^2$. After some algebra, one obtains a cubic equation for $\mathsf{G}_{\mathcal{D}}(x)$. We won't pursue this. □

Example 10.15 Balls in boxes Problems that involve placing unlabeled balls into labeled boxes (or, equivalently, problems that involve compositions of integers), are often easy to do using the Rules of Sum and Product. Let $\mathcal{T}_i$ be the set of possible ways to put things into the ith box. Let $\mathsf{G}_{\mathcal{T}_i}$ be the generating function which is keeping track of the things in the ith box. Suppose that what can be placed into one box is not dependent on what is placed in other boxes. The Rule of Product (in the Cartesian product form), tells us that we can simply multiply the $\mathsf{G}_{\mathcal{T}_i}$'s together.

How many ways can we put unlabeled balls into k labeled boxes so that no box is empty? Since there is exactly one way to place j balls in a box for every $j > 0$ and no ways if $j = 0$ (since the box may not be empty), we have

$$\mathsf{G}_{\mathcal{T}_i}(x) \;=\; 1x^0 + 1x^1 + 1x^2 + \cdots \;=\; \sum_{j=1}^{\infty} x^j = \frac{x}{1-x}$$

for all i. By the Rule of Product, the generating function is

$$\frac{x}{1-x} \cdots \frac{x}{1-x} \;=\; x^k(1-x)^{-k}.$$

Since

$$x^k(1-x)^{-k} \;=\; x^k \sum \binom{-k}{i}(-x)^i \;=\; \sum \binom{k+i-1}{i} x^{k+i},$$

it follows that the number of ways to distribute n unlabeled balls is $\binom{n-1}{n-k} = \binom{n-1}{k-1}$, which you found in Exercise 1.5.4 (p. 38).

How many solutions are there to the equation $z_1 + z_2 + z_3 = n$ where z_1 is an odd positive integer and z_2 and z_3 are nonnegative integers not exceeding 10? We can think of this as placing balls into boxes where z_i balls go into the ith box. Since

$$\mathsf{G}_{\mathcal{T}_1}(x) \;=\; x + x^3 + x^5 + \cdots \;=\; x\big(1 + x^2 + (x^2)^2 + (x^2)^3 + \cdots\big) \;=\; \frac{x}{1-x^2}$$

and

$$\mathsf{G}_{\mathcal{T}_2}(x) \;=\; \mathsf{G}_{\mathcal{T}_3}(x) \;=\; 1 + x + \cdots + x^{10} \;=\; \frac{1-x^{11}}{1-x},$$

it follows that the generating function is

$$\frac{x}{1-x^2}\,\frac{1-x^{11}}{1-x}\,\frac{1-x^{11}}{1-x}.$$

There isn't a nice formula for the coefficient of x^n.

What if we allow positive integer coefficients in our equation? For example, how many solutions are there to $z_1 + 2z_2 + 3z_3 = n$ in nonnegative integers? In this case, put z_1 balls in the first box, $2z_2$ balls in the second and $3z_3$ balls in the third. Since the number of balls in box i is a multiple of i, $\mathsf{G}_{\mathcal{T}_i}(x) = 1/(1-x^i)$. By the Rule of Product $\mathsf{G}_{\mathcal{T}}(x) = 1/((1-x)(1-x^2)(1-x^3))$. This result can be thought of as counting partitions of the number n where z_i is the number of parts of size i. By extending this idea, it follows that, if $p(n)$ is the number of partitions of the integer n, then

$$\sum_{n=0}^{\infty} p(n)x^n \;=\; \frac{1}{1-x}\,\frac{1}{1-x^2}\,\frac{1}{1-x^3}\cdots \;=\; \prod_{i=1}^{\infty}(1-x^i)^{-1}. \quad \square$$

So far we have only used the Rules of Sum and Product for single variable generating functions. We need not limit ourselves in this manner. As we will explain:

Observation *The Rules of Sum and Product apply to generating functions with any number of variables.*

Suppose we are keeping track of m different kinds of things. Replace w by $\mathbf{w}$, an m long vector of integers. Then $\vec{x}^{\vec{w}} = x_1^{w_1} \cdots x_m^{w_m}$. For example, if we count words by the number of vowels, the number of consonants and the length of the word, $\mathbf{w}$ will be a 3 long vector—one component for number of vowels, one for number of consonants and one for total number of letters. In that case, the variables will also form a 3 long vector $\mathbf{x}$. We can replace (10.31) with

$$\sum_{B \in \mathcal{B}} \mathbf{x}^{\mathbf{w}(B)} \;=\; B(\mathbf{x}),$$

where, as we already said, $\mathbf{x}^{\mathbf{w}}$ means $x_1^{w_1} \cdots x_m^{w_m}$. The condition on w in the Rule of Product becomes

$$\mathbf{w}(T) \;=\; \mathbf{w}(T_1) + \ldots + \mathbf{w}(T_k).$$

Of course, we could choose other indices besides $1, \ldots, m$ for our vectors and even replace some of the x_i's with other letters. In the next example, we find it convenient to use $\mathbf{x} = (x_0, x_1)$.

Example 10.16 Strings of zeroes and ones Let's look at strings of zeroes and ones. It will be useful to have a shorthand notation for writing down strings. The empty string will be denoted by λ. If s is a string, then $(s)^k$ stands for the string $ss \ldots s$ that consists of k copies of s and $(s)^*$ stands for the set of strings that consist of any number of copies of s, i.e., $(s)^* = \{\lambda, s, (s)^2, (s)^3, \ldots\}$. When s is simply 0 or 1, we usually omit the parentheses. Thus we write 0^* and 1^k instead of $(0)^*$ and $(1)^k$.

The sequences counted by the Fibonacci numbers, namely those which contain no adjacent ones, can be described by

$$\mathcal{F} \;=\; 0^* \cup (0^* \; 1 \; Z^* \; 0^*) \quad \text{where} \quad Z \;=\; 0^* \; 01.$$

This means

(a) any number of zeroes OR

(b) any number of zeroes AND a one AND any number of sequences of the form Z to be described shortly AND any number of zeroes.

A sequence of the form Z is any number of zeroes AND a zero AND a one. You should convince yourself that $\mathcal{F}$ does indeed give exactly those sequences which contain no adjacent ones. As you can guess from the ANDs and ORs above, this is just the right sort of situation for the Rules of Sum and Product.

What good does such a representation do us?

Observation *If this representation gives every pattern in exactly one way, we can mechanically use the Rules of Sum and Product to obtain a generating function.*

For a union (i.e., $\cup$ or $\{\cdots\}$), we are dealing with OR, so the Rule of Sum applies. When symbols appear to be multiplied it means first one thing AND then another, so we can apply the Rule of Product. For a set $\mathcal{S}$, the notation $\mathcal{S}^*$ means any number of copies of things in $\mathcal{S}$. For example, $\{0,1\}^*$ is the set of all strings of zeroes and ones, including the empty string. If there is a *unique* way to decompose elements of $\mathcal{S}^*$ into elements in $\mathcal{S}$, then

$$\mathsf{G}_{\mathcal{S}^*} \;=\; \mathsf{G}_\emptyset + \mathsf{G}_{\mathcal{S}} + \mathsf{G}_{\mathcal{S}\times\mathcal{S}} + \mathsf{G}_{\mathcal{S}\times\mathcal{S}\times\mathcal{S}} + \cdots \;=\; \sum_{k=0}^{\infty}(\mathsf{G}_{\mathcal{S}})^k \;=\; \frac{1}{1-\mathsf{G}_{\mathcal{S}}}.$$

What's "unique" decomposition mean? When $\mathcal{S}=\{0,1\}$, every string of zeroes and ones has a unique decomposition—just look at each element of the string one at a time. When $\mathcal{S}=\{0,01,11\}$ we still have unique decomposition; for example, 110001111101 decomposes uniquely as 11-0-01-11-11-01.

We leave it to you to verify that our representation for $\mathcal{F}$ gives all the patterns exactly once. Let x_0 keep track of zeroes and x_1 keep track of ones; that is, the coefficient of $x_0^n x_1^m$ in $\mathsf{G}_{\mathcal{F}}(x_0,x_1)$ will be the number sequences in $\mathcal{F}$ that have n zeroes and m ones. We have

$$\begin{aligned}
\mathsf{G}_{0^*} &= \frac{1}{1-\mathsf{G}_0} = \frac{1}{1-x_0} = (1-x_0)^{-1}\\
\mathsf{G}_{Z^*} &= \mathsf{G}_{(0^*\,01)^*} = \frac{1}{1-\mathsf{G}_{0^*\,01}} = \frac{1}{1-(1-x_0)^{-1}x_0x_1} = \frac{1-x_0}{1-x_0-x_0x_1}\\
\mathsf{G}_{\mathcal{F}} &= \frac{1}{1-x_0} + \frac{1}{1-x_0}\,x_1\,\frac{1-x_0}{1-x_0-x_0x_1}\,\frac{1}{1-x_0} = \frac{1+x_1}{1-x_0-x_0x_1}.
\end{aligned}$$

We can use this representation to describe and count other sequences; however, the problem can get tricky if we are counting sequences that must avoid patterns more complicated than 11. There are various ways to handle the problem. One method is by the use of sets of recursions, which we'll discuss in the next chapter. Sequences that can be described in this fashion (we haven't said precisely what that means) are called *regular sequences*. They are, in fact, the strings that can be produced by regular grammars, which we saw in Section 9.2 were the strings that can be recognized by finite automata. See Exercise 10.4.19 (p. 304) for a definition and the connection with automata. There is a method for translating finite automata into recursions. We'll explore this in Example 11.2 (p. 310). □

We close this section with an example which combines the Rules of Sum and Product with some techniques for manipulating generating functions.

***Example 10.17 Counting certain spanning trees** Let G be a simple graph with $V=\underline{n}\cup\{0\}$ and the $2n-1$ edges $\{i,i+1\}$ $(1\le i<n)$ and $\{0,j\}$ $(1\le j\le n)$. (Draw a picture!) How many spanning trees does G have? We'll call the number r_n.

To begin with, what does a spanning tree look like? An arbitrary spanning tree can be built as follows. First, select some of the edges $\{i,i+1\}$ $(1\le i<n)$. This gives a graph H with vertex set $\underline{n}$. (Some vertices may not be on any edges.) For each component C of H, select a vertex j in C and add the edge $\{0,j\}$ to our collection. Convince yourself that this procedure gives all the trees.

We can imagine this in a different way. Let $\mathcal{T}$ be the set of rooted trees of the following form. For each $k>0$, let $V=\underline{k}\cup\{0\}$ and let 0 be the root. The tree contains the $k-1$ edges $\{i,i+1\}$ $(1\le i<k)$ and one edge of the form $\{0,j\}$ for some $1\le j\le k$. Join together an ordered list of trees in $\mathcal{T}$ by merging their roots into one vertex and relabeling their nonroot vertices $1,2,\ldots$ in order as shown in Figure 10.1. This process produces each spanning tree exactly once.

What we have just described is the perfect setup for the Rules of Sum (on k) and Product (of $\mathcal{T}$ with itself k times) when we count the number of vertices other than the vertex 0. Thus, recalling the definition of r_n at the start of the example,

$$R \;=\; \sum_{k=1}^{\infty}(\mathsf{G}_{\mathcal{T}})^k \;=\; \sum_{k=1}^{\infty}T^k = \frac{T}{1-T}.$$

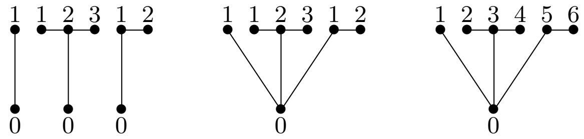

Figure 10.1 Building a spanning tree from pieces. The pieces on the left are assembled to give the middle figure and are then relabeled to give the right-hand figure.

How many trees in $\mathcal{T}$ have k nonroot vertices? Exactly k, one for each choice of a vertex to connect to the root. Thus $T(x) = \sum_{k=1}^{\infty} kx^k$. We can evaluate this sum by using derivatives as discussed in the previous section:

$$\begin{aligned} T(x) &= \sum_{k=1}^{\infty} kx^k = \sum_{k=0}^{\infty} kx^k = \sum_{k=0}^{\infty} x\frac{d(x^k)}{dx} \\ &= x\frac{d}{dx}\left(\sum_{k=0}^{\infty} x^k\right) = x\frac{d\big((1-x)^{-1}\big)}{dx} = \frac{x}{(1-x)^2}. \end{aligned}$$

Combining these results gives us

$$R(x) = \frac{\dfrac{x}{(1-x)^2}}{1-\dfrac{x}{(1-x)^2}} = \frac{x}{1-3x+x^2}. \qquad 10.33$$

What can we do now to get values for r_n? We have two choices: (a) expand by partial fractions to get an exact value or (b) manipulate (10.33) using the ideas of the previous section to obtain a recursion. By partial fractions, r_n is the integer closest to $\alpha^n/\sqrt{5}$, where $\alpha = (3+\sqrt{5})/2$, which gives us a quick, accurate approximation to r_n for large n. We leave the calculations to you and turn our attention to deriving a recursion.

Clearing of fractions in (10.33) and equating the coefficients of x^n on both sides of the resulting equation gives the recursion

$$r_0 = 0, \quad r_1 = 1 \quad \text{and} \quad r_n = 3r_{n-1} - r_{n-2} \text{ for } n \ge 2, \qquad 10.34$$

which makes it fairly easy to build a table of r_n.

Can you prove (10.34) directly; i.e., without using generating functions? It's a bit tricky. With some thought and experimentation, you may be able to discover the argument. □

Exercises

10.4.1. Let $\mathcal{T}$ be a collection of structures. Suppose that $\mathbf{w}(T) \neq \mathbf{0}$ for all $T \in \mathcal{T}$. Prove the following results.

(a) The generating function for k-lists of structures, with repetitions allowed, is $(\mathsf{G}_{\mathcal{T}})^k$.

(b) The generating function for lists of structures, with repetitions allowed, is $(1-\mathsf{G}_{\mathcal{T}})^{-1}$. Here lists of any length are allowed, including the empty list.

(c) If T is a generating function, let $F^{[k]}$ denote the result of replacing all the variables by their kth powers. For example, $F^{[k]}(x,y) = F(x^k, y^k)$. Show that the generating function for sets of structures, where each structure must come from $\mathcal{T}$ is

$$\exp\left(\sum_{k=1}^{\infty}(-1)^{k-1}\big(\mathsf{G}_{\mathcal{T}}\big)^{[k]}/k\right).$$

Hint. Show that the answer is

$$\prod_{T\in\mathcal{T}}(1+\mathbf{x}^{\mathbf{w}(T)}),$$

replace $(1+\mathbf{x}^{\mathbf{w}(T)})$, with $\exp\bigl(\ln(1+\mathbf{x}^{\mathbf{w}(T)})\bigr)$, expand the logarithm by Taylor's Theorem and rearrange the terms.

(d) Show that generating function for multisets of structures is

$$\exp\left(\sum_{k=1}^{\infty}\bigl(\mathsf{G}_{\mathcal{T}}\bigr)^{[k]}/k\right).$$

10.4.2. Return to Exercise 10.2.4 (p. 280). There we counted the number of sequences of zeros, ones and twos with no adjacent ones and no adjacent twos. Show that part (a) of that exercise can be rewritten as follows.
A sequence of the type we want is either

- an alternating sequence of ones and twos OR
- a sequence of the type we want AND a zero AND an alternating sequence of ones and twos.

Here the alternating sequence may be empty. Use this characterization to deduce an equation that can be solved for $S(x)$

10.4.3. Using the notation introduced in Example 10.16, write out expressions for strings satisfying the following properties. Do this in such a way that each string is generated uniquely and then use your representation to get the generating function for the number of patterns of length n. Finally, obtain a recursion from your generating function. Remember to include initial conditions for the recursion.

(a) Strings of zeroes, ones and twos that have do not have the pattern 02 somewhere.
Hint. Except possibly for a run of twos at the very start of the string, every 2 must be preceded by a 1 or a 2.

(b) Strings of zeroes and ones such that each string of ones is followed by a string of at least k zeroes; i.e., if it starts with a string of zeroes, that can be of any length, but every other string of zeroes must have length at least k. Use the notation 0^k to stand for a string of k zeroes.

(c) Strings of zeroes and ones such that each maximal string of ones (i.e., its ends are the ends of the sequence and/or zeroes) has odd length.

10.4.4. Let q_n be the number of partitions of n with no repeated parts allowed and, as usual, let p_n be all partitions of n. Let $q_0 = 1$.

(a) Show that

$$Q(x) = \sum_{n=0}^{\infty} q_n x^n = \prod_{i=1}^{\infty}(1+x^i).$$

(b) Let $P(x)$ be the generating function for partitions of a number. Show that $Q(-x)\,P(x) = 1$. Equate coefficients of x^n for $n > 0$ and then rearrange to avoid subtractions. Interpret the rearranged result combinatorially. Can you give a direct proof of it?

(c) Let $q_{n,k}$ (resp. $p_{n,k}$) be the partitions counted in q_n (resp. p_n) in which no part exceeds k. Obtain formulas for $\sum_{n\ge0} q_{n,k}x^n$ and $\sum_{n\ge0} p_{n,k}x^n$.

10.4.5. Let a "pile" be, roughly, a two dimensional stack of square blocks resting on a flat surface, with each block directly on top of another and each row not having gaps. A more formal definition of a pile of height h is a sequence of $2h$ integers such that

$$0 = a_1 \le a_2 \le \cdots \le a_h < b_h \le \cdots \le b_2 \le b_1.$$

Here a block has width 1 and, counting rows with the first row on the bottom, the left end of row i is at a_i and the right end is at b_i. The number of blocks in the ith row is $b_i - a_i$ and the total number of blocks in the pile is $\sum_{i=1}^{h}(b_i - a_i)$. Let s_n be the number of n-block piles and $s_{n,h}$ the number of those of height h. Obtain a formula for $\sum s_n x^n$ and $\sum_{n\ge 0} s_{n,h} x^n$.
Hint. The generating function for partitions with no part exceeding k will be useful.

10.4.6. Let $a_1 < a_2 < \cdots < a_k$ be a k element subset of $\underline{n} = \{1, 2, \ldots, n\}$. We will study subsets with restrictions on the a_i.

(a) Let $a_0 = 0$. By looking at $a_i - a_{i-1}$, show that there is a bijection between k element subsets of $\underline{n}$ and k long sequences of positive integers with sum not exceeding n.

(b) Let u_n be the number of k element subsets of $\underline{n}$. Use (a) to show that

$$U(x) = \Bigl(\sum_{i\ge 1} x^i\Bigr)^k \Bigl(\sum_{i\ge 0} x^i\Bigr) = \frac{x^k}{(1-x)^{k+1}}.$$

(Do *not* use the fact that $\binom{-k-1}{n-k} = \binom{n}{n-k}$.)

(c) Let t_n be the number of k element subsets $a_1 < a_2 < \cdots$ of $\underline{n}$ such that i and a_i have the same parity. In other words a_{2j} is even and a_{2j+1} is odd. Show that

$$T(x) = \frac{x^k}{(1-x^2)^k}\,\frac{1}{1-x} = \frac{(1+x)x^k}{(1-x^2)^{k+1}}.$$

(d) Let $\lfloor x \rfloor$ be the result of rounding x down; e.g., $\lfloor 3.5 \rfloor = 3$. Show that $t_n = \binom{\lfloor (n+k)/2 \rfloor}{k}$.

(e) We call (a_i, a_{i+1}) a succession if they differ by one. Let $s_{n,j}$ be the number of k element subsets of $\underline{n}$ with exactly j successions. Show that

$$S(x,y) \;=\; \frac{1}{1-x}\,\frac{x}{1-x}\,(xy + x^2 + x^3 + \cdots)^{k-1} \;=\; \frac{x^k(x + y(1-x))^{k-1}}{(1-x)^{k+1}}.$$

(f) Show that $\sum_{n\ge 0} s_{n,j} x^n \;=\; \binom{k-1}{j} x^{2k-j-1}(1-x)^{-(k+1-j)}$.

(g) Express $s_{n,j}$ as a product of two binomial coefficients. Check your result by listing all 4 element subsets of $\{1, \ldots, 6\}$ and determining how many successions they have.

10.4.7. Recall that a binary RP-tree to be an RP-tree where each vertex may have at most two sons. The set $\mathcal{T}$ of such trees was studied in Exercise 10.2.6, where we counted them by number of vertices.

(a) Using the Rules of Sum and Product, derive the relation $T(x) = x + xT(x) + xT(x)^2$ that led to

$$T(x) \;=\; \frac{1 - x - \sqrt{1 - 2x - 3x^2}}{2x}$$

in Exercise 10.2.6

(b) Discuss how you might compute the number of such trees. In particular, can you find a simple expression as a function of n?

10.4.8. Change the definition in Exercise 10.4.7 so that, if a node has just one son, then we distinguish whether or not it is a right or a left son. (This somewhat strange sounding distinction is sometimes important.) How many such trees are there with n internal vertices?

10.4.9. A rooted tree will be called "contractible" if it has a vertex with just one son since one can imagine combining that vertex's information with the information at its son.

(a) Find the generating function for the number of unlabeled noncontractible RP-trees, counting them by number of vertices.

(b) Find the generating function for the number of unlabeled noncontractible RP-trees, counting them by number of leaves.

(c) Obtain a linear differential equation with polynomial coefficients and thence a recursion from each of the generating functions in this problem.
Hint. Solve for the square root, differentiate, multiply by the square of the square root and then replace the square root that remains.

10.4.10. Let $t_{n,k}$ be the number of RP-trees with n leaves and k internal vertices (i.e., nonleaves).

(a) Find a generating function for $T(x, y)$.

(b) Using the previous result, prove that $t_{n,k} = t_{k,n}$ when $n + k > 1$.
Hint. Compare $T(x, y)$ and $T(y, x)$.

*(c) Find a bijection that proves $t_{n,k} = t_{k,n}$ when $n + k > 1$; that is, find a map from RP-trees to RP-trees that carries leaves to internal vertices and vice versa for trees with more than one vertex. Write out your bijection for RP-trees with 5 vertices.
Hint. Describe the construction recursively (or locally).

10.4.11. Let D be a set of nonnegative integers such that $0 \in D$. For this exercise, we'll say that an RP-tree is of outdegree D if the number of sons of each vertex lies in D. Thus, full binary RP-trees are of outdegree $\{0, 2\}$.

(a) Let $T_D(x)$ be the generating function for unlabeled RP-trees of outdegree D by number of vertices. Prove that

$$T_D(x) \;=\; x \sum_{d \in D} T_D(x)^d$$

(b) Show that the previous formula allows us to compute $T_D(x)$ recursively.

(c) Let $L_D(x)$ be the generating function for unlabeled RP-trees of outdegree D by number of leaves. Show that it doesn't make sense to talk about $L_D(x)$ when $1 \in D$, that

$$L_D(x) \;=\; \sum_{d \in D} L_D(x)^d - 1 + x,$$

and that this allows us to compute $L_D(x)$ recursively when $1 \notin D$.

10.4.12. We have boxes labeled with pairs of numbers like $(2, 6)$. The labels have the form (i, j) for $1 \le i \le 3$ and $1 \le j \le k$. Thus we have $3k$ boxes. Unlabeled balls are placed into the boxes. This must be done so that the number of balls in box (i, j) is a multiple of i and, for each j, the total number of balls in boxes $(1, j)$, $(2, j)$ and $(3, j)$ is at most 5. What is the generating function for the number of ways to place n balls?
Hint. Find the generating function for placing balls into $(1, *)$, $(2, *)$ and $(3, *)$ and then use the Rule of Product.

*10.4.13. An unlabeled full binary rooted tree is like the ordered (i.e., plane) situation except that we make no distinction between left and right sons. Let β_n be the number of such trees with n leaves and let $B(x) = \sum \beta_n x^n$. Show that $B(x) = x + (B(x)^2 + B(x^2))/2$.

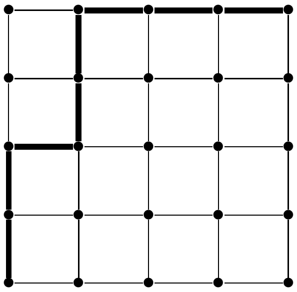

Figure 10.2 A path for $n = 4$ for Exercise 10.4.14.

10.4.14. Imagine the plane with lines like a sheet of graph paper; i.e., the lines $x = k$ and the lines $y = k$ are drawn in for all integers k. Think of the intersections of the lines vertices and the line segments connecting them as edges. The portion of the plane with $0 \le x \le n$ and $0 \le y \le n$ is then a simple graph with $(n+1)^2$ vertices. Let a_n be the number of paths from the vertex $(0,0)$ to the vertex (n,n) that never go down or left and that remain above the line $x = y$ except at $(0,0)$ and (n,n). Figure10.2 shows such a path for $n = 4$. We could describe such a path more formally as a sequence (x_i, y_i) of pairs of nonnegative integers such that $x_0 = y_0 = 0$, $x_{2n} = y_{2n} = n$, $x_i < y_i$ for $0 < i < 2n$ and $(x_i, y_i) - (x_{i-1}, y_{i-1})$ equals either $(1,0)$ or $(0,1)$. Draw some pictures to see what this looks like.

(a) Show that a_n is the number of sequences $s_1, \ldots, s_{2n}$ containing exactly n -1's and n 1's such that $s_1 + \cdots + s_k > 0$ for $0 < k < 2n$.

(b) By looking at $s_2, \ldots, s_{2n-1}$ for $n > 1$, conclude that $A(x) = x + \sum_{k\ge 1} x\, A(x)^k$.

(c) Determine s_n. Note that this number is the same as the number of unlabeled full binary RP-trees with n leaves, which is the same as the number of unlabeled RP-trees with n vertices.

*(d) In the previous part, you concluded that the set $\mathcal{S}_n$ of paths of a certain type from $(0,0)$ to (n,n) has the same size as the set $\mathcal{T}_n$ of unlabeled RP-trees with n vertices. Find a bijection $f_n: \mathcal{S}_n \to \mathcal{T}_n$, and thus prove this equality without the use of generating functions.

10.4.15. Fix a set S of size s. For $n \ge 1$, let $a_{n,k}$ be the number of n long ordered lists that can be made using S so that we never have more than k consecutive identical entries in the list. Thus, with $k \ge n$ there is no restraint while with $k = 1$ adjacent entries must be different. Let $A_k(x) = \sum_{n\ge 0} a_{n,k} x^n$. There are various approaches to $A_k(x)$.

(a) By considering the last run of identical entries in the list and using the Rules of Sum and Product, show that

$$A_k(x) \;=\; s(x + x^2 + \cdots + x^k) + A_k(x)(s-1)(x + x^2 + \cdots + x^k).$$

(b) Find an explicit formula for $A_k(x)$.

(c) Show that $a_{n+1,k} \;=\; s a_{n,k} - (s-1)a_{n-k,k}$ for $n > k$ by using the generating function.

(d) Derive the previous recursion by a direct argument.

10.4.16. We claim that the set of sequences of zeroes and ones that do not contain either 101 or 111 is described by

$$0^* \cup \Big(0^*(1 \cup 11)\big(000^*(1 \cup 11)\big)^* 0^*\Big)$$

and each such sequence has a unique description. You need not verify this. Let a_n be the number of such sequences of length n and let $A(x)$ be the generating function for a_n.

(a) Derive the formula $A(x) = \frac{1+x+2x^2+x^3}{1-x-x^3-x^4}$.

(b) Using $A(x)$, obtain the recursion $a_n = a_{n-1} + a_{n-3} + a_{n-4}$ for $n \geq 4$ and find the initial conditions.

(c) Using $1 - x - x^3 - x^4 = (1 - x - x^2)(1 + x^2)$, derive the formula

$$a_n = \frac{7F_{n+1} + 4F_n - b_n}{5} \quad \text{where} \quad b_n = \begin{cases} 2(-1)^{n/2}, & \text{if } n \text{ is even,} \\ (-1)^{(n-1)/2}, & \text{if } n \text{ is odd,} \end{cases}$$

where the Fibonacci numbers are given by $F_0 = 0$, $F_1 = 1$ and $F_n = F_{n-1} + F_{n-2}$ for $n \geq 2$; that is, their generating function is $\frac{x}{1-x-x^2}$.

(d) Prove that

$$a_{2n} = F_{n+2}^2 \quad \text{and} \quad a_{2n+1} = F_{n+2}F_{n+3} \quad \text{for } n \geq 0.$$

Hint. Show that the recursion and initial conditions are satisfied.

10.4.17. Using partial fractions, obtain a formula for r_n from (10.33).

*10.4.18. Let G be the simple graph with vertex set $\underline{n} \cup \{0\}$ and the $2n$ edges $\{n, 1\}$, $\{i, i+1\}$ $(1 \leq i < n)$ and $\{0, j\}$ $(1 \leq j \leq n)$, except for $n = 1, 2$ where we must avoid adding $\{n, 1\}$ in order to get a simple graph. In other words, G is like the graph in Example 10.17 except that one more edge $\{1, n\}$ has been added so that the picture looks like a wheel with spokes for $n > 2$. We want to know how many spanning trees G has.

(a) Let $\mathcal{T}$ be as in Example 10.17 and let $\mathcal{T}'$ consist of the trees in $\mathcal{T}$ with one of the nonroot vertices marked. Choose one tree from $\mathcal{T}'$ and then a, possibly empty, sequence of trees from $\mathcal{T}$. Suppose we have a total of n nonroot vertices. Merge the root vertices and relabel the nonroot vertices 1 to n, starting with the marked vertex in the tree from $\mathcal{T}'$ and preceding cyclically until all nonroot vertices have been labeled. Explain why this gives all the spanning trees exactly once.

(b) Show that $\mathsf{G}_{\mathcal{T}'}(x) = x(d/dx)(\mathsf{G}_{\mathcal{T}}(x)) = x(1+x)/(1-x)^3$.

(c) Show that generating function for the spanning trees is

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

(d) Show that the number of spanning trees is $2r_{n+1} - 3r_n - 2$, where r_n is given in Example 10.17.

10.4.19. We define a set of *regular sequences* (or regular strings) on the "alphabet" A. (An alphabet is any finite set.) Let R, R_1, and R_2 stand for sets of regular strings on A. The sets of regular strings on A are the empty set, the sets $\{a\}$ where $a \in A$, and the sets that can be built recursively using the following operations:

- union ("or") of sets, i.e., the set of all strings that belong to either R_1 or R_2;
- juxtaposition ("and then"), i.e., the set of all strings r_1r_2 where $r_1 \in R_1$ and $r_2 \in R_2$;
- arbitrary iteration R^*, i.e., for all $n \geq 0$, all strings of the form $r_1r_2\ldots r_n$ where $r_i \in R$. (The empty string is obtained when $n = 0$.)

See Example 10.16 for a specific example of a set of regular sequences The purpose of this exercise is to construct a nondeterministic finite automaton that recognizes any given set of regular strings. Nondeterministic finite automata are defined in Section 6.6 (p. 189). We will build up the machine by following the way the strings are built up.

(a) Let $\mathcal{A}$ be an automaton. Show that there is another automaton $S(\mathcal{A})$ that recognizes the same strings and has no edges leading into the start state.
Hint. Create a new state, let it be the start state and let it have edges to all of the states the old start state did. Remember to specify the accepting states, too.

(b) If $\mathcal{A}$ recognizes the set A and $\mathcal{B}$ recognizes the set B, construct and automaton that recognizes the set $A \cup B$.
Hint. Adjust the idea in (a).

(c) If $\mathcal{A}$ recognizes A, construct an automaton that recognizes A^*.
Hint. Add some edges.

(d) If $\mathcal{A}$ recognizes the set A and $\mathcal{B}$ recognizes the set B, construct and automaton that recognizes AB; i.e., the set $A \times B$.

Notes and References

The classic books on generating functions are those by MacMahon [6] and Riordan [7]. They are quite difficult reading and do not take a "combinatorial" approach to generating functions. There are various combinatorial approaches. Some can be found in the texts by Wilf [10] and Stanley [9, Ch. 3] and in the articles by Bender and Goldman [1] and Joyal [5]. The articles are rather technical.

Parts of the texts by Greene and Knuth [4] and by Graham, Knuth and Patashnik [3] are oriented toward computer science uses of generating functions. See also the somewhat more advanced text by Sedgewick and Flajolet [7]. Wilf [10] gives a nice introduction to generating functions. Goulden and Jackson's book [2] contains a wealth of material on generating functions, but is at a higher level than our text.

We have studied only the simplest sorts of recursions. Recursions that require more sophisticated methods are common as are recursions that cannot be solved exactly. Sometimes approximate solutions are possible. We don't know of any systematic exposition of techniques for such problems.

We have not dealt with the problem of defining formal power series; that is, defining a generating function so that the convergence of the infinite series is irrelevant. An introduction to this can be found in the first few pages of Stanley's text [9].

1. Edward A. Bender and Jay R. Goldman, Enumerative uses of generating functions, *Indiana Univ. Math. J.* **20** (1971), 753–765.
2. Ian P. Goulden and David M. Jackson, *Combinatorial Enumeration*, Dover (2004).
3. Ronald L. Graham, Donald E. Knuth and Oren Patashnik, *Concrete Mathematics*, 2nd ed., Addison-Wesley, Reading (1994).
4. Daniel H. Greene and Donald E. Knuth, *Mathematics for the Analysis of Algorithms*, 3rd ed., Birkhäuser (1990).

5. André Joyal, Une théorie combinatoire des séries formelles, *Advances in Math.* **42** (1981), 1-82.
6. Percy Alexander MacMahon, *Combinatory Analysis*, Chelsea, New York, 1960. Reprint of two volume Cambridge Univ. Press edition (1915, 1916).
7. John Riordan, *An Introduction to Combinatorial Analysis*, Princeton Univ. Press (1980).
8. Robert Sedgewick and Philippe Flajolet, *An Introduction to the Analysis of Algorithms*, Addison-Wesley (1996).
9. Richard P. Stanley, *Enumerative Combinatorics*, vols. 1 and 2. Cambridge Univ. Press (1999, 2001).
10. Herbert S. Wilf, *Generatingfunctionology*, 2nd ed., Academic Press (1993).