Some Techniques for Solving Recurrences

GEORGE S. LUEKER

Department of Information and Computer Science, University of California, Irvine, California 92717

Some techniques for solving recurrences are presented, along with examples of how these recurrences arise in the analysis of algorithms. In addition, probability generating functions are discussed and used to analyze problems in computer science.

Keywords and Phrases: recurrences, characteristic equations, generating functions, probability generating functions, analysis of algorithms, asymptotic analysis

CR Categories: 5.25, 5.30, 5.5

INTRODUCTION

The analysis of a problem in computer science often involves a sequence of numbers. For example, when determining the amount of time used by an algorithm, one may be interested in the sequence $T(1)$, $T(2)$, $T(3)$, ..., where $T(n)$ is the amount of time used on an input of size n. Such a sequence can also arise even if we are considering input of a fixed size; for example, if we are analyzing the average amount of time used by a sorting algorithm we might be interested in a sequence $p_0, p_1, p_2, \ldots$, where p_i is the probability that the algorithm makes i comparisons during the sort. Sometimes instead of knowing a closed-form expression for the nth term in a sequence, we have a *recurrence relation,* which is an equation that expresses the nth term of a sequence as a function of previous terms. (Recurrence relations are sometimes called *difference equations.*)

Consider, for example, the problem of determining the number of comparisons used by Mergesort; it can be viewed as a recursive sorting algorithm that splits the input list of n numbers into two halves, sorts each half, and then merges the results (using at most $n - 1$ comparisons) to yield the final sorted output [AHO74, Chap. 2]. Assuming n is a power of two, an upper bound $T(n)$ on the number of comparisons used by this algorithm to sort n numbers is given by

$$T(n) = 2T(n/2) + n - 1, \quad \text{for} \quad n \geq 2,$$
$$T(1) = 0.$$

The first of these two equations expresses the fact that the number of comparisons required to sort a large list must be, at most, the sum of the number of comparisons required to sort both halves ($2T(n/2)$) and the number of comparisons ($n - 1$) required in the worst case to merge the halves. The second equation expresses the fact that sorting a single element does not require any comparisons. (For more information about sorting algorithms see AHO74, Chaps. 2 and 3; KNUT73, Chap. 5.) As a second example, consider the problem of finding the minimum number of vertices that can be present in an AVL tree of height n; call this number a_n. (An AVL tree is a binary search tree in which the heights of the left and right subtrees of any node differ by at most 1; see KNUT73, Sec. 6.2.3, for a discussion of such trees and how they relate to efficient searching algorithms.) If the AVL tree is to have as few vertices as possible, one subtree must have height $n - 1$ and the other must have height $n - 2$. Both subtrees must also have the minimum number of

CONTENTS

vertices for an AVL tree of that height. We therefore obtain

$$a_n = a_{n-1} + a_{n-2} + 1, \quad \text{for} \quad n \geq 2,$$

$$a_0 = 1,$$

$$a_1 = 2.$$

Note that in each case the recurrence is valid only for n at least as large as some limit. Below that point we specify a few values of the sequence to enable the recurrence to get started; these are called the *boundary conditions.*

This paper is a tutorial on some techniques for solving recurrences. Of course one technique is simply to apply the recurrence over and over again to get successive terms in the sequence. This may not give us much insight into the behavior of the sequence; instead we often seek to discover a closed-form expression for the nth term of the sequence. The first two sections discuss summing factors and characteristic equations, which enable one systematically to solve a fairly large class of equations. The third section discusses domain and range transformations on recurrences, which can significantly extend the class of recurrences solvable by the techniques of Sections 1 and 2. In Section 4 the notion of generating functions is discussed and used to analyze problems which arise in computer science; we also discuss some especially useful properties of generating functions related to probabilities. Finally, we indicate how one may sometimes use generating functions to determine the asymptotic behavior of the solution to a recurrence without having to solve it exactly. This paper is intended for advanced undergraduates and graduate students, but should also be useful to others seeking an introduction to recurrences.

1. SUMMING FACTORS

We begin by considering two easy recurrences which involve counting the number of nodes in a binary tree. Let the *height* of a binary tree be the distance from the root to the furthest leaf, where by *distance* we mean the number of edges on the path joining two nodes. Thus the height of a tree is one less than the number of levels in the tree. Say a binary tree is *perfect* if all nodes have zero or two children and all leaves are at the same distance from the root; in a perfect tree each level either has all possible nodes or no nodes. We now consider the problem of determining the number a_n of nodes in a perfect tree of height n. There are two ways of obtaining a recurrence for a_n. First we could note that a tree of height n can be obtained by adding a new bottom row of leaves to a tree of height $n-1$. Since the number of nodes at each level in a perfect binary tree is twice the number in the next higher row it is not hard to see that the number of leaves in the new bottom row is 2^n. Thus we conclude that

$$a_n = a_{n-1} + 2^n, \quad \text{for} \quad n \geq 1. \quad (1.1)$$

A second way of viewing this problem is to use a different decomposition of a tree of height n. Note that the tree consists of a root and two perfect subtrees of height $n-1$. This leads to the recurrence

$$a_n = 2a_{n-1} + 1, \quad \text{for} \quad n \geq 1. \quad (1.2)$$

In either case the boundary condition is obtained by noting that a tree of height 0 consists of a single node, so

$$a_0 = 1. \quad (1.3)$$

To illustrate the definition and use of summing factors, we compare the problem of

solving eq. (1.1) and of solving eq. (1.2); we should of course get the same answer. Now (1.1) can be rewritten as

$$a_n - a_{n-1} = 2^n. \tag{1.4}$$

Note what happens if we add up eq. (1.4) for n running from 1 to m:

$$\begin{array}{lll} a_1 & - a_0 & = 2 \\ a_2 & - a_1 & = 4 \\ a_3 & - a_2 & = 8 \\ \vdots & \vdots & \vdots \\ a_{m-1} & - a_{m-2} & = 2^{m-1} \\ a_m & - a_{m-1} & = 2^m. \end{array}$$

For $0 < n < m$, a_n occurs positively in one equation but negatively in the next. Thus when we add all of these equations together most of the terms will cancel out; we say that the sum *telescopes*. We are left with

$$a_m - a_0 = \sum_{n=1}^{m} 2^n.$$

The right-hand side is a well-known sum with the value $2^{m+1} - 2$, so we obtain

$$a_m = a_0 + 2^{m+1} - 2 = 2^{m+1} - 1. \tag{1.5}$$

Finding the solution to (1.1) was almost trivial. Sometimes however we have to use a little trickery to make the sum of successive equations telescope. Suppose that our recurrence is

$$\begin{aligned} p_n a_n - q_n a_{n-1} &= r_n, \qquad \text{for} \quad n \geq 1, \\ a_0 &= r_0, \end{aligned} \tag{1.6}$$

where p_n, q_n, and r_n are given and a_n is unknown. If we try to add two successive copies of this equation, for example,

$$p_n a_n - q_n a_{n-1} = r_n$$

and

$$p_{n-1} a_{n-1} - q_{n-1} a_{n-2} = r_{n-1},$$

we note that the a_{n-1} terms do not cancel since p_{n-1} will generally not equal q_n. Note however that we could multiply the equations by some factor which would make the a_{n-1} terms cancel. This motivates the following approach. Multiply the nth equation by some (so far unspecified) factor s_n to obtain

$$s_n p_n a_n - s_n q_n a_{n-1} = s_n r_n$$

and

$$s_{n-1} p_{n-1} a_{n-1} - s_{n-1} q_{n-1} a_{n-2} = s_{n-1} r_{n-1}.$$

In order to guarantee that the a_{n-1} terms cancel, we require that

$$s_n q_n = s_{n-1} p_{n-1},$$

so

$$s_n = s_{n-1} p_{n-1} / q_n.$$

We can force this to be true by letting

$$s_n = \prod_{i=1}^{n-1} p_i \Big/ \prod_{i=1}^{n} q_i \tag{1.7}$$

(or any constant multiple of this). Then telescopy will work when we add up

$$s_n p_n a_n - s_n q_n a_{n-1} = s_n r_n$$

for n running from 1 to m.

We now try to use this trick on the recurrence (1.2). This is an example of (1.6) with

$$p_n = 1, \qquad q_n = 2.$$

By (1.7) we may choose

$$s_n = 2^{-n}.$$

Multiplying by s_n gives the new equation

$$2^{-n} a_n - 2^{-(n-1)} a_{n-1} = 2^{-n}.$$

Summing this for n running from 1 to m gives

$$2^{-m} a_m - a_0 = \sum_{n=1}^{m} 2^{-n}.$$

Thus

$$\begin{aligned} a_m &= 2^m \left(\sum_{n=1}^{m} 2^{-n} + a_0 \right) \\ &= \sum_{n=0}^{m-1} 2^n + 2^m \\ &= 2^m - 1 + 2^m \\ &= 2^{m+1} - 1. \end{aligned} \tag{1.8}$$

This of course agrees with (1.5).

For a considerably more challenging application of summing factors, see the analysis of Quicksort in KNUT73, Sec. 5.2.2.

2. CHARACTERISTIC EQUATIONS

2.1 Homogeneous Equations

A recurrence relation is said to be a *homogeneous linear recurrence with constant coefficients* if it has the form

$$p_0 a_n + p_1 a_{n-1} + p_2 a_{n-2} + \cdots + p_k a_{n-k} = 0, \tag{2.1}$$

where each p_i is a constant. It is called a *linear* recurrence because we are specifying a value for some linear combination of the a_i. Since the coefficients p_i are constants, we say the recurrence has *constant coefficients*. The word *homogeneous* refers to the fact that we are requiring that the linear combination be set equal to zero. An example is the recurrence

$$a_n - a_{n-1} - 2a_{n-2} = 0.$$

We now show a general method for solving any such recurrence.

Such recurrences often have solutions which can be expressed as

$$a_n = r^n$$

where r is some constant. Let us plug this into eq. (2.1) and see if we can find a value for r which will work. We obtain

$$\sum_{i=0}^{k} p_i r^{n-i} = 0,$$

which is satisfied provided

$$\sum_{i=0}^{k} p_i r^{k-i} = 0.$$

This kth-degree polynomial in r is called the *characteristic equation* for the recurrence. We need to find the roots of this polynomial, which are called its *characteristic roots*. Since this polynomial has degree k, we expect there will be k roots; temporarily assume they are all distinct and call them $r_1, r_2, \ldots, r_k$. It is not hard to see that any linear combination of solutions to eq. (2.1) is also a solution, so anything of the form

$$a_n = \sum_{i=1}^{k} c_i r_i^n$$

where c_i are arbitrary constants will also be a solution. In fact, as Theorem 1 establishes, if there are k distinct roots, this is a general form for any solution to (2.1). We now need to determine which values of the constants will give the solution we want. In order to do this we use the boundary conditions.

As an example, suppose we are given

$$a_n - 5a_{n-1} + 6a_{n-2} = 0, \quad \text{for} \quad n \geq 2,$$
$$a_0 = 0,$$
$$a_1 = 1.$$

The characteristic equation for this is

$$r^2 - 5r + 6 = 0,$$

whose roots are

$$r_1 = 2, \qquad r_2 = 3.$$

Thus the general solution to the recurrence is

$$a_n = c_1 2^n + c_2 3^n. \tag{2.2}$$

However, we know the values of a_0 and a_1. Thus we may write

$$0 = a_0 = c_1 + c_2,$$
$$1 = a_1 = 2c_1 + 3c_2.$$

Solving for c_1 and c_2 gives

$$c_1 = -1,$$
$$c_2 = 1,$$

so the solution to our recurrence is

$$a_n = 3^n - 2^n.$$

So far we have assumed that the roots of the characteristic equation are distinct. What if there are multiple roots? (For example, $(r-1)^2 = 0$ has 1 as a root with multiplicity 2.) In general, if r is a root of the characteristic equation with multiplicity q, then the equation will have solutions

$$a_n = n^p r^n$$

where p is any integer from 0 to $q-1$. Another way to express this is that any-

thing of the form

$$a_n = (\text{a polynomial in } n \text{ of degree less than } q)\ r^n \tag{2.3}$$

is a solution. One might wonder whether there are also other solutions not of the form shown in (2.3). The answer is no, as the following theorem shows.

Theorem 1

Let $f(x)$ be the characteristic equation of the recurrence

$$p_0 a_n + p_1 a_{n-1} + p_2 a_{n-2} + \cdots + p_k a_{n-k} = 0, \quad \text{for } n \geq k, \tag{2.4}$$

where the p_i are constants. Let the roots of f, over the complex numbers, be r_i, $i = 1, \ldots, m$, and let their respective multiplicities be q_i, $i = 1, \ldots, m$. Then any solution to (2.4) is of the form

$$\sum_{i=1}^{m} \left(r_i^n \sum_{j=0}^{q_i - 1} c_{ij} n^j \right),$$

where the c_{ij} are constants.

We omit the proof here; the interested reader may consult LIU68, App. 3-1. Note that in order for the theorem to be true, it is important that we determine all of the roots, including those that are complex. All of the characteristic equations discussed in this paper have only real roots. Complex roots, which can give rise to solutions with periodic behavior, can be handled by the methods discussed here, but it is often useful to express such solutions in terms of the sine and cosine functions [LIU68, pp. 62–64].

We now give an example of a recurrence whose characteristic equation has a multiple root. Let the *rank* of a node in a tree be one more than its number of descendants. (We count a node as one of its own descendants.) When discussing a type of tree known as a BB(α) tree (see REIN77, Sec. 6.4.3), the notion of the rank of a node can be useful. In particular, sometimes it is useful to find the sum of the ranks of all of the nodes in the tree. This sum is also closely related to the internal and external path length of a tree, discussed in KNUT68, Sec. 2.3.4.5. Here we determine the total rank a_n of the nodes in a perfect binary tree of height n. First note that the rank of the root is 2^{n+1}, since we have already seen that it has $2^{n+1} - 1$ descendants. The total rank of the nodes in each of its subtrees is a_{n-1}. Thus

$$a_n - 2a_{n-1} = 2^{n+1},$$
$$a_0 = 2.$$

Now unfortunately this is not a homogeneous equation because of the 2^{n+1} appearing on the right-hand side. Shortly we will see a systematic way of dealing with such problems, but for the time being we use an ad hoc approach to force the equation to be homogeneous. Note that the right-hand side doubles each time n increases by one. Thus if we take one equation and subtract twice the previous one, we can eliminate right-hand side, as follows:

$$\begin{array}{rl} a_n - 2a_{n-1} & = 2^{n+1} \\ -2(\quad a_{n-1} - 2a_{n-2} & = 2^n) \\ \hline a_n - 4a_{n-1} + 4a_{n-2} & = 0 \end{array}$$

Since this new equation is valid only for $n \geq 2$ we must provide a boundary value for a_1 (which we may determine from the recurrence), so the set of boundary conditions becomes

$$a_0 = 2, \qquad a_1 = 8.$$

The characteristic equation is

$$r^2 - 4r + 4 = 0.$$

This can be rewritten as

$$(r - 2)^2 = 0,$$

so 2 is the only root but it occurs twice. Thus the general solution is

$$a_n = (c_1 + c_2 n)2^n.$$

Using the boundary conditions we obtain

$$2 = a_0 = c_1,$$
$$8 = a_1 = 2(c_1 + c_2).$$

Solving gives

$$c_1 = 2, \qquad c_2 = 2,$$

so the general solution is

$$a_n = (2n + 2)2^n.$$

2.2 Nonhomogeneous Equations

We have now seen how to solve any homogeneous linear recurrence with constant coefficients. Often however the recurrence we wish to solve is not homogeneous, as, for example, in

$$\begin{aligned} a_n - 5a_{n-1} + 6a_{n-2} &= 4, \quad \text{for} \quad n \geq 2, \\ a_0 &= 5, \\ a_1 &= 7. \end{aligned} \tag{2.5}$$

A recurrence which looks like eq. (2.1) but has a nonzero function of n on the right-hand side is called a *nonhomogeneous* linear recurrence with constant coefficients. Any sequence a_n which satisfies the recurrence (but not necessarily the boundary conditions) is called a *particular solution*; any sequence which makes the left-hand side identically zero is called a *homogeneous solution.* Sometimes we can guess a particular solution, but cannot easily find one which makes the boundary conditions hold. In this case the following theorem is very helpful.

Theorem 2

If we start with any particular solution a_n and add any homogeneous solution, we obtain another particular solution. Moreover the difference between any two particular solutions is always a homogeneous solution.

(The proof is easy; we do not present it here.) This theorem suggests the following approach to solving nonhomogeneous recurrences.

1. Guess a particular solution a_n.
2. Write a formula for a_n plus the general homogeneous solution (with unknown constants).
3. Use the boundary conditions to solve for the constants.

We now demonstrate this technique on the recurrence in (2.5). First we guess a particular solution; after some effort we discover that $a_n = 2$ will work. (This process of trying to guess a solution could be quite frustrating. Fortunately there is a systematic method which covers many cases, and which we discuss in a moment.) We have seen before (2.2) that the homogeneous solution has the form

$$a_n = c_1 2^n + c_2 3^n.$$

Thus by Theorem 2 the solution to (2.5) must have the form

$$a_n = 2 + c_1 2^n + c_2 3^n.$$

Solving for the constants using the boundary conditions we obtain

$$c_1 = 4, \qquad c_2 = -1,$$

so

$$a_n = 2 + 4 \cdot 2^n - 3^n.$$

The method outlined has an obvious disadvantage: It requires that we be able to guess a particular solution. It would seem much more desirable to have a systematic method for producing particular solutions. We now outline a systematic method which applies to many recurrences. Some additional notation is useful. We represent a sequence by simply writing down a formula for its nth element in angle brackets, assuming elements are numbered starting at 0. For example, the sequence 1, 2, 4, ... is represented by $\langle 2^n \rangle$. (Be sure to read this carefully; $\langle a_n \rangle$ denotes a sequence, not the single element a_n.) Let **E** be an operator which transforms a sequence by throwing away its first element. For example, the sequence 1, 2, 4, 8, ... is transformed by **E** into 2, 4, 8, 16, ... This can be abbreviated by writing

$$\mathbf{E}\langle 2^n \rangle = \langle 2^{n+1} \rangle.$$

In general,

$$\mathbf{E}\langle a_n \rangle = \langle a_{n+1} \rangle.$$

We can build new operators by combining **E** with itself and with constants. To do this, for any constant c we define an operator (also denoted by c) by the equation

$$c\langle a_n \rangle = \langle c a_n \rangle.$$

We also define addition and multiplication of operators by

$$(A + B)\langle a_n \rangle = A\langle a_n \rangle + B\langle a_n \rangle,$$

and

$$(AB)\langle a_n \rangle = A(B\langle a_n \rangle).$$

TABLE 1. ANNIHILATORS FOR VARIOUS SEQUENCES

Sequence	*Annihilator*
$\langle c \rangle$	$\mathbf{E} - 1$
$\langle$a polynomial in n of degree $k\rangle$	$(\mathbf{E} - 1)^{k+1}$
$\langle c^n \rangle$	$(\mathbf{E} - c)$
$\langle c^n$ times a polynomial in n of degree $k\rangle$	$(\mathbf{E} - c)^{k+1}$

With these definitions we can easily verify that addition and multiplication of operators are commutative and associative; moreover multiplication distributes over addition. (In fact, the set of all possible operators that can be constructed as described above forms an algebraic structure known as a *Euclidean ring*. See HERS64, p. 104.) Here are two examples of applications of operators.

$$(2 + \mathbf{E})\langle a_n \rangle = \langle 2a_n + a_{n+1} \rangle,$$

$$\mathbf{E}^3\langle a_n \rangle = \langle a_{n+3} \rangle.$$

Such operators enable us to express certain concepts very concisely. Note, for example, that if $f(r)$ is the characteristic polynomial of some recurrence of the form in (2.4), we may write the recurrence as

$$f(\mathbf{E})\langle a_n \rangle = \langle 0 \rangle.$$

As this equation suggests, sometimes a certain operator, when applied to a certain sequence, produces a sequence consisting entirely of zeroes. In this case we say that the operator is an *annihilator* for the sequence. For example, $\mathbf{E} - 2$ is an annihilator for $\langle 2^n \rangle$, since

$$(\mathbf{E} - 2)\langle 2^n \rangle = \langle 2^{n+1} - 2 \cdot 2^n \rangle = \langle 0 \rangle.$$

We can now finally describe a fairly general technique for producing solutions to nonhomogeneous equations with constant coefficients.

1. Apply an annihilator for the right-hand side to both sides of the equation.
2. Solve the resulting homogeneous equation by the methods discussed earlier.

A list of sequences and corresponding annihilators is shown in Table 1. One can prove that this table is correct as follows. It can easily be shown that if we apply $(\mathbf{E} - c)$ to $\langle c^n p(n) \rangle$, where $p(n)$ is a polynomial of positive degree d, we obtain $\langle c^n q(n) \rangle$, where $q(n)$ is a polynomial of degree $d - 1$. Using this and an easy induction, we can establish that the last line in the table is correct. All of the other lines are special cases of the last line.

One can easily prove that if A is an annihilator for $\langle a_n \rangle$ and B is an annihilator for $\langle b_n \rangle$, then the *product* AB is an annihilator for the *sum* $\langle a_n + b_n \rangle$. Thus, for example, an annihilator for $\langle n2^n + 1 \rangle$ is $(\mathbf{E} - 2)^2(\mathbf{E} - 1)$.

We now work out two examples. The first is eq. (2.5), which we solved above by guessing the particular solution. With our new notation we can write this as

$$(\mathbf{E}^2 - 5\mathbf{E} + 6)\langle a_n \rangle = \langle 4 \rangle.$$

Factoring gives

$$(\mathbf{E} - 2)(\mathbf{E} - 3)\langle a_n \rangle = \langle 4 \rangle.$$

Now apply $(\mathbf{E} - 1)$ to annihilate the 4.

$$(\mathbf{E} - 1)(\mathbf{E} - 2)(\mathbf{E} - 3)\langle a_n \rangle = \langle 0 \rangle.$$

This has the characteristic equation

$$(r - 1)(r - 2)(r - 3) = 0,$$

with distinct roots 1, 2, and 3. Thus we must have

$$a_n = c_1 + c_2 2^n + c_3 3^n.$$

We can solve for the constants by noting that

$$a_2 - 5a_1 + 6a_0 = 4,$$

$$a_0 = 5,$$

$$a_1 = 7.$$

The result is

$$a_n = 2 + 4 \cdot 2^n - 3^n,$$

as before.

For a second example, consider the equation

$$a_n - 2a_{n-1} = 2^n - 1, \quad \text{for } n \geq 1,$$

$$a_0 = 0. \tag{2.6}$$

(We show an important application of this recurrence in Section 3.) Using the operator **E** we may write

$$(\mathbf{E} - 2)\langle a_n \rangle = \langle 2^{n+1} - 1 \rangle.$$

The annihilator for the right-hand side is $(\mathbf{E} - 2)(\mathbf{E} - 1)$, so we obtain

$$(\mathbf{E} - 1)(\mathbf{E} - 2)^2\langle a_n \rangle = \langle 0 \rangle.$$

Thus the characteristic roots are 2 (with a multiplicity of 2) and 1, so

$$a_n = (c_1 + c_2 n)2^n + c_3.$$

Solving for the constants we obtain

$$c_1 = -1, \qquad c_2 = 1, \qquad c_3 = 1,$$

so

$$a_n = (n - 1)2^n + 1.$$

3. DOMAIN AND RANGE TRANSFORMATIONS

Sometimes it is useful to apply a transformation to a sequence in order to make it appear in a more desirable form. A sequence can be thought of as a mapping from the integers into the reals; thus we call a transformation on the values of the sequence a *range transformation* and a transformation on the indices a *domain transformation.* (In LEVY61, p. 103, sequences which can be transformed by domain or range transformations into linear recurrences are called "pseudo-non-linear equations.") We begin by showing an example of a range transformation. Suppose

$$a_n = 3a_{n-1}^2, \qquad \text{for} \quad n \geq 1,$$
$$a_0 = 1.$$

This cannot be solved by any method discussed so far, but if we let

$$b_n = \lg a_n$$

(where lg denotes the base-2 logarithm), we may rewrite the recurrence as

$$b_n = 2b_{n-1} + \lg 3,$$
$$b_0 = 0.$$

This can easily be solved by the methods discussed before. The result is

$$b_n = (2^n - 1) \lg 3,$$

so

$$a_n = 2^{(2^n - 1)\lg 3} = 3^{2^n - 1}.$$

Next we show how domain transformations can be useful. Recall the recurrence which we mentioned at the beginning of this paper for Mergesort:

$$T(n) = 2T(n/2) + n - 1, \qquad \text{for} \quad n \geq 2, \qquad (3.1)$$
$$T(1) = 0,$$

where n is required to be a power of 2. (See AHO74, pp. 65-67.) We may be tempted to consider the recurrence

$$a_n = 2a_{n/2} + n - 1,$$

but none of the techniques we have discussed so far are directly applicable to this equation. However if we let

$$n = 2^k \qquad \text{and} \qquad a_k = T(n) = T(2^k),$$

we can write

$$a_k = 2a_{k-1} + 2^k - 1, \qquad \text{for} \quad k \geq 1,$$
$$a_0 = 0.$$

This is a recurrence which can be solved by the methods already described. In fact, it is one of those which we solved as an example in the preceding section, namely (2.6). The solution we found was

$$a_k = (k - 1)\, 2^k + 1.$$

Thus, since $a_k = T(2^k)$, we may conclude that

$$T(n) = (\lg n - 1)\, n + 1.$$

Next we give an example of a more difficult domain transformation. It is possible to multiply two n-bit numbers by doing three multiplications of $(n/2 + 1)$-bit (or shorter) numbers and $O(n)$ additional work. (See AHO74, pp. 62-65; there the algorithm is described in a way which needs only multiply $n/2$-bit numbers, but a more natural implementation yields the recurrence we are about to analyze.) Applying this decomposition recursively (and stopping the recursion when the numbers are of length 3 or less) leads to an algorithm whose complexity is described by

$$T(n) = 3T(n/2 + 1) + O(n), \qquad \text{for} \quad n > 3, \qquad (3.2)$$
$$T(3) = O(1).$$

The O-notation in the recurrence is a little inconvenient. To eliminate it, note that there must be some constant c' such that the extra time represented by the "$O(n)$" in the recurrence is bounded by $c'n$. Similarly, there is a constant c'' such that the time represented by the "$O(1)$" is bounded by c''. Letting c be the larger of these, we conclude that we may bound $T(n)$ by $\hat{T}(n)$, if $\hat{T}(n)$ is the solution to

$$\hat{T}(n) = 3\hat{T}(n/2 + 1) + c\,n, \quad \text{for} \quad n > 3, \tag{3.3}$$
$$\hat{T}(3) = c.$$

Since we only seek the O-notation for the complexity of the algorithm, we can scale this by any constant factor. In particular we may let c be 1, to obtain

$$\hat{T}(n) = 3\hat{T}(n/2 + 1) + n, \quad \text{for} \quad n > 3, \tag{3.4}$$
$$\hat{T}(3) = 1.$$

We would like to be able to choose an index k so that the recurrence could be written as

$$a_k = 3a_{k-1} + (\text{some function of } k). \tag{3.5}$$

Let n_k denote the value of n corresponding to a given k. In order for (3.4) to correspond to (3.5), we must have

$$n_{k-1} = n_k/2 + 1,$$

so

$$n_k = 2n_{k-1} - 2.$$

We call this a *secondary recurrence* for (3.4). If we let $n_0 = 3$, the solution is

$$n_k = 2^k + 2.$$

Now if we let

$$a_k = \hat{T}(n_k),$$

we may rewrite (3.4) as

$$a_k = 3a_{k-1} + 2^k + 2,$$
$$a_0 = 1.$$

Solving this yields

$$a_k = 4\cdot 3^k - 2\cdot 2^k - 1.$$

Since the relation between n and k implies

$$k = \lg(n - 2),$$

we conclude that for any n which appears in the sequence $\langle n_k \rangle$,

$$\begin{aligned}\hat{T}(n) &= 4\cdot 3^{\lg(n-2)} - 2\cdot 2^{\lg(n-2)} - 1\\ &= 4(n-2)^{\lg 3} - 2(n-2) - 1\\ &= 4(n-2)^{\lg 3} - 2n + 3.\end{aligned}$$

Thus this method for multiplying two numbers works in $T(n) = O(n^{\lg 3})$ time. Since $\lg 3$ is about 1.59, this multiplication method is asymptotically faster than the simple $O(n^2)$ approach.

For very large n a much faster way of multiplying n-bit numbers is the Schönhage–Strassen method [AHO74, Sec. 7.5; SCHO71]. In analyzing this method the following recurrence arises.

$$T(n) = \lg n + 2T(4\,n^{1/2}). \tag{3.6}$$

We begin the solution of this recurrence by using a domain transformation. The appropriate secondary recurrence is

$$n_i = 4\,n_{i+1}^{1/2}.$$

To convert this into a more tractable form, we may use the range transformation $m_i = \lg n_i$. Then we have

$$m_i = 2 + m_{i+1}/2.$$

This can be solved by the methods of Section 1 or Section 2. The solution is

$$m_i = 2^i + 4,$$

so

$$n_i = 2^{2^i+4}. \tag{3.7}$$

(Other solutions are also possible since we have not stated the boundary conditions.) If we now let $a_i = T(n_i)$, eq. (3.6) becomes

$$a_i = 2^i + 4 + 2a_{i-1},$$

which is readily solved to yield

$$a_i = i2^i + b2^i - 4,$$

where b is unspecified since we have not given boundary conditions. Then since from (3.7)

$$i = \lg(\lg n_i - 4),$$

it must be that for any n appearing in the sequence $\langle n_i \rangle$,

$$\begin{aligned}M(n) &= \Theta(i2^i)\\ &= \Theta(\lg(\lg n - 4)\, 2^{(\lg(\lg n-4))})\\ &= \Theta(\lg(\lg n - 4)(\lg n - 4))\\ &= \Theta(\lg n \lg\lg n).\end{aligned}$$

TABLE 2. GENERATING FUNCTIONS FOR SOME SEQUENCES

Sequence	*Generating Function*
$1, 1, 1, 1, \ldots$	$1 + z + z^2 + z^3 + \ldots = 1/(1 - z)$
$1, c, c^2, \ldots$	$1 + cz + (cz)^2 + \ldots = 1/(1 - cz)$
$1, 2c, 3c^2, \ldots, (i + 1)c^i, \ldots$	$1/(1 - cz)^2$
$\binom{n}{0}, \binom{n}{1}, \ldots, \binom{n}{i}, \ldots$	$(1 + z)^n$

Using this solution one can show that the Schönhage–Strassen algorithm will multiply two n-bit numbers in $O(n \lg n \lg \lg n)$ time; see AHO74, Sec. 7.5, for more details.

4. GENERATING FUNCTIONS

4.1 Generating Functions for Sequences

Generating functions are an ingenious method by which certain problems can be solved very elegantly. Like Laplace transforms and Fourier transforms, generating functions transform a problem from one conceptual domain into another, in the hope that the problem will be easier to solve in the new domain.

Definition

The *generating function* for the sequence $a_0, a_1, a_2, \ldots$, is the function

$$A(z) = \sum_{i=0}^{\infty} a_i z^i.$$

(The sum in the definition might not always converge; we have more to say about this later. In FELL68, Sec. XI.1, the definition given for generating functions requires that the series converge over some interval of positive length.) At first there seems to be little motivation for the definition of generating functions, but we shall see that certain types of operations on sequences correspond to certain other operations on generating functions. Thus in tackling a problem we can choose whichever domain makes the problem easier. Table 2 provides some examples of sequences and corresponding generating functions. Note that if n is a positive integer, the last generating function is a polynomial. We can extend the usefulness of this generating function by using an extended definition of the binomial coefficients, as in LIU68, Sec. 2-2, and SEDG75, p. 299. For any real n and any integer i define

$$\binom{n}{i} = \text{if } i < 0 \text{ then } 0 \text{ else } \frac{1}{i!} \prod_{j=0}^{i-1} (n - j).$$

Then the last line in the table is valid for any real n and gives a sequence with infinitely many nonzero values if n is negative or nonintegral.

Now we show how some standard operations on functions correspond to operations on sequences. (Further discussion of these operations can be found in KNUT68, Sec. 1.2.9, and REIN77, Sec. 3.3.) Let $A(z)$ and $B(z)$ be the generating functions for $\langle a_i \rangle$ and $\langle b_i \rangle$ respectively. In Table 3 the left column shows a function that can be obtained from A and B by simple operations and the right column shows a formula for the ith element of the corresponding sequence.

We now show how generating functions can be used to solve a simple recurrence. Suppose that

$$a_n = 2a_{n-1} + 1, \quad \text{for} \quad n \geq 1,$$
$$a_0 = 1.$$

Let

$$A(z) = \sum_{n=0}^{\infty} a_n z^n.$$

From the recurrence, this sum can be rewritten as

$$\begin{aligned} A(z) &= 1 + \sum_{n=1}^{\infty} (2a_{n-1} + 1)z^n \\ &= 1 + z \sum_{n=1}^{\infty} 2a_{n-1} + \sum_{n=1}^{\infty} z^n \\ &= 1 + 2zA(z) + \left(\frac{1}{1 - z} - 1\right). \end{aligned}$$

Note what has happened. Initially we had a problem to be solved for $\langle a_n \rangle$. In the left and right sides of the foregoing equation we have a problem which is to be solved for A. This new problem is a simple one. Algebra yields

$$A(z) = \frac{1}{(1 - z)(1 - 2z)}.$$

TABLE 3. OPERATIONS ON GENERATING FUNCTIONS

Generating Function	*Formula for ith Element of Sequence*
cA	ca_i
$A + B$	$a_i + b_i$
AB	$\sum_{j=0}^{i} a_j b_{i-j}$
$z^k A(z)$	if $i < k$ then 0 else a_{i-k}
$\frac{A(z)}{1-z}$	$\sum_{j=0}^{i} a_j$
$zA'(z)$	ia_i
$\int_0^z A(t)\,dt$	if $i = 0$ then 0 else a_{i-1}/i

Now we must return to the original problem domain. Unfortunately $A(z)$ is not in Table 2. However, by use of partial fractions, we may write

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

(We do not discuss here the problem of decomposing a rational expression with partial fractions; the interested reader may consult FADE64, Sec. 13.8.) Using Tables 2 and 3, we now see that the corresponding sequence is given by

$$a_n = 2^{n+1} - 1.$$

This agrees with the solution in eq. (1.8).

At this point a word of caution is in order. We have done some formal manipulations involving sequences but have ignored questions such as convergence. This might lead to extraneous answers in some cases. There are at least two possible ways to avoid difficulty. One approach is to check the answer by some independent method. In our example a simple inductive proof would suffice. A second approach is to check carefully the validity of each step. In our example we could tell by inspection of the recurrence that

$$\lim_{n\to\infty} \frac{a_{n+1}}{a_n} = 2,$$

and hence that the radius of convergence of $A(z)$ is $\frac{1}{2}$ (by BUCK78, Th. 14, p. 240). This could form the basis of a rigorous proof of the validity of our operations. (See also the discussion in KNUT68, Sec. 1.2.9. For a good rigorous discussion of properties of series, see BUCK78, Chaps. 5 and 6.) In the remainder of this paper we do not always perform these verifications.

As a second example of the use of generating functions, we consider the problem of counting the number of distinct binary trees with a given number of nodes. (We assume that the nodes are indistinguishable. Thus we consider two trees identical if they have the same shape. This is sometimes referred to as the *enumeration of planted plane binary trees.*) This example shows that generating functions can be used to solve problems for which the techniques of Sections 1 and 2 do not seem to be of much help. Let b_n be the number of distinct binary trees which can be formed from n nodes; let $B(z)$ be the generating function for $\langle b_n \rangle$.

Before continuing the analysis it is worth noting that the generating function must converge for any z in the open interval $(-\frac{1}{4}, \frac{1}{4})$. To see this let the *type* of a node be 0 if it has no children, 1 if it has only a left child, 2 if it has only a right child, and 3 if it has both children. It is not difficult to show (see STAN80, Sec. 3.5.2) that if we have a list of the types of the nodes in preorder, then we have enough information to reconstruct the tree. Now there are only 4^n strings of n of these four types; thus it must be that $b_n \le 4^n$. Hence the radius of convergence of $B(z)$ must be at least $\frac{1}{4}$.

To get more information about $B(z)$ we determine a recurrence for $\langle b_n \rangle$. A binary tree on one or more vertices has to have a root. Assuming it has $n + 1$ nodes, the remaining n nodes can be distributed arbitrarily between the left and right subtrees. Note that if we put i nodes in one subtree there will be $n - i$ nodes remaining for the other tree. If we choose to partition the nodes this way, we can still build the left subtree in any of b_i ways and the right subtree in any of b_{n-i} ways, for a total of $b_i b_{n-i}$ combinations. Summing over all the possible choices for i, we come up with the following recurrence.

$$b_{n+1} = \sum_{i=0}^{n} b_i b_{n-i}, \qquad \text{for} \quad n \ge 0, \tag{4.1}$$
$$b_0 = 1.$$

The boundary condition may seem a little strange at first. We need this for consistency, however, since if we decide to put

all of the nodes in the one subtree there is exactly one way to build the other subtree, namely, to put nothing there.

To begin to convert eq. (4.1) into a statement about $B(z)$, we multiply by z^n and sum as n goes from 0 to infinity.

$$\sum_{n=0}^{\infty} b_{n+1}z^n = \sum_{n=0}^{\infty} \sum_{i=0}^{n} b_i b_{n-i} z^n. \quad (4.2)$$

The left side of (4.2) is

$$b_1 + b_2 z + b_3 z^2 + b_4 z^3 + \cdots.$$

This is just the same as $B(z)$, except that we have deleted the first term (b_0) and removed one factor of z. Therefore, since b_0 is 1, the left side is

$$(B(z) - 1)/z.$$

The right side, from the third line of Table 3, is

$$B(z)^2.$$

Thus (4.2) becomes

$$(B(z) - 1)/z = B(z)^2,$$

which can be rearranged as

$$zB(z)^2 - B(z) + 1 = 0,$$

and solved by the quadratic formula to obtain

$$B(z) = \frac{1 \pm \sqrt{1 - 4z}}{2z}. \quad (4.3)$$

At this point we might be a little dismayed, since the ± sign gives two solutions, and it is not apparent which is correct. However note that

$$B(0) = b_0 + b_1 z + b_2 z^2 + \cdots |_{z=0} = b_0.$$

Thus $B(0)$ must be 1. Now if we choose the − sign in (4.3), $B(0)$ will indeed be 1; if we choose the + sign, $B(z)$ approaches infinity as z approaches 0. Thus we may finally conclude that

$$B(z) = \frac{1 - \sqrt{1 - 4z}}{2z}. \quad (4.4)$$

A number of texts have shown how to obtain an exact formula for b_n from this generating function [KNUT68, Sec. 2.3.4.4; STAN80, Sec. 3.3.2; REIN77, Sec. 3.3]. It is shown that

$$b_n = \frac{1}{n+1}\binom{2n}{n}. \quad (4.5)$$

We do not repeat this part of the solution here. However, at the end of the third part of our discussion of generating functions, we show how to obtain an asymptotic description of b_n very easily.

4.2 Probability Generating Functions

Let X be a random variable which assumes nonnegative integer values and let p_i be the probability that $X = i$. Then the *probability generating function* (*pgf*) for X is the generating function for $\langle p_i \rangle$. Note that its radius of convergence must be at least 1, since probabilities lie in the range [0, 1]. Such functions have some particularly nice properties, which we explore in this section.

Suppose X and Y are two independent nonnegative integer random variables, with probability generating functions P and Q respectively. Using the third line in Table 3, we can readily establish the rather pleasing fact that the pgf for the sum of X and Y is the product of P and Q. (It must be stressed that this fact is not true if X and Y are not independent. As an extreme example, note that the pgf for $X + X$ is $P(z^2)$, not $(P(z))^2$.)

Many of the commonly encountered nonnegative integer random variables have easily expressed generating functions. A few of these are summarized in Table 4. (This table is based on FELL68, Sec. XI.2.) The first two distributions can be described in terms of a sequence of flips of a biased coin, which lands heads up with a probability of p. The binomial distribution tells the number of heads in n flips. The geometric distribution describes the number of tails that occur before the first head. The Poisson distribution can be viewed as a limiting case of the binomial distribution, in which we let $n \to \infty$ and $p \to 0$ in such a way that $p = \lambda/n$. Radioactive decay provides a simple example of such a distribution. If we assume that decays of separate atoms are independent, then the number of flashes of a radioactive watch dial in some time period has very nearly a Poisson distribution. By tak-

TABLE 4. PROBABILITY GENERATING FUNCTIONS FOR SOME RANDOM VARIABLES

Name	*Formula for* p_k	*Probability Generating Function*
Binomial	$\binom{n}{k} p^k(1-p)^{n-k}$	$(1-p+pz)^n$
Geometric	pq^k	$\frac{1-q}{1-qz}$ where $q = 1-p$
Poisson	$\lambda^k e^{-\lambda}/k!$	$e^{\lambda(z-1)}$

ing the limit of the generating function for the binomial distribution and invoking the Continuity Theorem, we can obtain the generating function shown in the table for the Poisson distribution; see FELL68, Sec. XI.6.

Another useful property of probability generating functions is that they enable us to find certain expected values quickly. (See FELL68, Sec. IX.2, for a discussion of expectation.) For example, note that

$$P'(1) = \sum_{i=0}^{\infty} ip_i = E[X].$$

Table 5 presents a number of formulas for expected values; D_z denotes the derivative with respect to z. (The first three lines of the table may be found in FELL68, Secs. XI.1 and XI.2, and KNUT68, Sec. 1.2.10. The fourth line follows easily from line 6 of Table 3.)

Some caution must be exercised when using Table 5. We express the possible problem in terms of the last line, since the first three lines are special cases of it. Let r be the radius of convergence of the generating function. If $|c| > r$, the function in the left column does not have a finite expectation, even though the formula in the right column may seem to give an answer. In the case $c = r$, a finite expectation may or may not exist. In this case if $c > 0$ and

$$\lim_{z \uparrow c} (zD_z)^k P(z)$$

exists, then a finite expectation does exist and is given by the limit. If $|c| < r$, the formula always gives the correct answer. Note that for the binomial and Poisson distributions the generating function converges everywhere, so problems of the sort discussed here do not arise.

As an example of the application of probability generating functions we now analyze the expected complexity of a highly efficient sorting algorithm known as *address calculation sorting* [ISAA56; KNUT73, Sec. 5.2.1]. Suppose we know that the data to be sorted $(x_1, x_2, \ldots, x_n)$ are distributed uniformly and independently over the open interval (0, 1). The following algorithm classifies the data into n buckets, each corresponding to a piece of the interval of length $1/n$. Then it sorts each bucket and concatenates them. The hope is that each bucket will contain so few elements that each sort can be performed in very little time.

```
begin
    for k := 0 until n − 1 do
        L_k := the empty list;
    for i := 1 until n do
        begin
            k := floor(n x_i);
            append x_i to L_k;
        end;
    for i := 0 until n − 1 do
        if L_i is nonempty then sort L_i by
                                insertion sort;
    output L_0 || L_1 || L_2 || ··· || L_{n−1};
end;
```

Here two consecutive vertical bars denote concatenation of lists.

Aside from the time in the sorts, the algorithm clearly uses $O(n)$ time. We show that the expected time for each sort is $O(1)$ and hence the overall time is $O(n)$.

First we determine the generating function for the length X of one of the lists, say L_0. Note that each element, independently, has a probability of $1/n$ of being placed into list L_0. Since there are n trials we obtain a binomial distribution. From Table 4, if we let $p = 1/n$, X has the generating function

$$P(z) = (1-p+pz)^n = \left(1 + \frac{z-1}{n}\right)^n.$$

The time to perform insertion sort is $O(X^2)$.

TABLE 5. EXPECTED VALUES FOR SEVERAL FUNCTIONS OF A RANDOM VARIABLE X WITH PROBABILITY GENERATING FUNCTION P

Function of X	*Expected Value*
X	$P'(1)$
X^2	$P''(1) + P'(1)$
c^X	$P(c)$
$X^k c^X$	$(zD_z)^k P(z) \mid_{z=c}$

But by Table 5,

$$E[X^2] = P''(1) + P'(1)$$
$$= \frac{n-1}{n}\left(1 + \frac{z-1}{n}\right)^{n-2} + \left(1 + \frac{z-1}{n}\right)^{n-1}\Bigg|_{z=1}$$
$$= \frac{n-1}{n} + 1 < 2.$$

Thus the expected time for each sort is indeed $O(1)$, so this sorting algorithm runs in $O(n)$ time under the given assumptions.

This same type of problem arises in a different context in an analysis of the traveling salesman problem [KARP77]. There is a well-known dynamic programming approach which yields an $O(n^2 2^n)$ algorithm for this problem [BELL62, HELD62]. In KARP77, this is used as one of the building blocks for an algorithm which tends, asymptotically, to run quite quickly and give near-optimal results on the average, under certain assumptions about the distribution of inputs. In particular it is assumed that the number of points is drawn from a Poisson distribution with mean n, and that the points are distributed uniformly over the unit square. The square is subdivided into subsquares and the TSP is solved optimally for the set of points within each subsquare. These solutions are then combined to produce an approximate solution for the entire problem. Part of the analysis involves determining the average amount of time spent solving one of these subproblems. If each subsquare has area A, then the number of points within each subsquare has a Poisson distribution and a mean of An. Let X denote a variable with this distribution, and, for convenience, let $m = An$. Thus to evaluate $E[X^2 2^X]$ we write

$$(zD_z)^2 e^{m(z-1)}\Big|_{z=2} = (zD_z) zme^{m(z-1)} \mid_{z=2}$$
$$= (4m^2 + 2m)e^m = \Theta(m^2 e^m).$$

Note that if the number of points were exactly m, the time used would be $\Theta(m^2 2^m)$ rather than $\Theta(m^2 e^m)$. Since the function $m^2 2^m$ grows rapidly, the effect of the fluctuation of X about its mean is to increase the mean time somewhat.

As a final example of an application of probability generating functions, we consider the problem of solving a random assignment problem. Donath [DONA69] used generating functions in obtaining a lower bound on the average value of the optimum solution. We discuss his argument here. To define the assignment problem assume that a manager must assign n workers to n jobs, and that we have a matrix (c_{ij}) in which c_{ij} indicates how much worker i dislikes job j. The manager wishes to assign workers to jobs so as to minimize the total worker dissatisfaction. More formally, he wishes to choose exactly one element from each row of the matrix in such a way as to minimize the total of the selected elements. We call this minimum total the *value of the optimum solution.* Donath showed how to obtain a lower bound on the average value of the optimum solution for a random matrix. Of course, in order to make this precise, we must state the assumptions to be made about what a "random matrix" looks like. In the result reported here Donath assumed that each row was (independently) a random permutation of the integers from 1 to n; each of the $n!$ permutations could occur with equal probability. (See GONN79 for an interpretation of the assignment problem, with this distribution of inputs, in terms of ideal structuring of hash tables.) To begin, we fix one possible assignment and try to find the pgf for the sum it yields for a random matrix. Note that in each row this assignment yields, with equal probability, a number from 1 to n. Thus the pgf for the number selected from any particular row is

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

To find the pgf for the total cost of the n rows, we merely raise this generating function to the nth power to obtain

$$\frac{z^n(1-z^n)^n}{n^n(1-z)^n}.$$

It will turn out that we are more interested in the cumulative probabilities, that is, the sequence $\langle a_k \rangle$ where a_k is the probability that the total is less than or equal to k. To obtain this, we apply the summation operator found in line 5 of Table 3, namely, $1/(1-z)$, to obtain

$$n^{-n}z^n(1-z)^{-(n+1)}(1-z^n)^n. \quad (4.6)$$

Now, as we verify in a moment, the range of k over which we need the values of a_k is $n < k < 2n$. An inspection of (4.6) shows that for k in this range, the coefficient of z^k must be the coefficient of z^{k-n} in

$$n^{-n}(1-z)^{-(n+1)}.$$

By an application of the binomial theorem we can establish that this coefficient is

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

where the equality follows from SEDG75, identity (16), p. 301.

So far we have been considering the probability that the total cost is bounded by k, given a fixed assignment of columns to rows. To bound the optimum, we must bear in mind that $n!$ assignments are possible. Now by Boole's inequality the probability that any of the assignments yields a cost bounded by k is no more than the sum, over all assignments, of the probability that this particular assignment does. Thus if we let b_k be the probability that the optimum is less than or equal to k we have

$$b_k \le n!a_k = n!n^{-n}\binom{k}{k-n}. \quad (4.7)$$

Using Stirling's approximation one may establish that

$$\ln b_k \le n[\beta \ln \beta - (\beta-1)\ln(\beta-1) - 1] + O(\ln n),$$

where $k = \beta n$. Now let β_0 be the root of

$$\beta \ln \beta - (\beta-1)\ln(\beta-1) - 1 = 0. \quad (4.8)$$

It is not hard to establish that if $\beta < \beta_0$, the right-hand side of (4.7) approaches 0 exponentially as n approaches infinity. From this we may deduce that the average value of the optimum is asymptotically bounded below by $\beta_0 n$. Numerical solution of (4.8) reveals that β_0 is about 1.54221. Simulation of the assignment problem with random data of the type discussed here suggests that actually the optimum tends to be asymptotic to approximately $1.8n$ [DONA69, GONN79]. Thus this lower bound is probably not tight. See WALK79 for a derivation of an upper bound for a very closely related problem.

4.3 Extracting Asymptotic Information from Generating Functions

In this subsection we briefly discuss the problem of determining the asymptotic behavior of the sequence $\langle a_n \rangle$ from its generating function $A(z)$. For this subsection only, the reader needs familiarity with complex variables and with the gamma function. In BEND74 a number of useful techniques are presented, along with numerous interesting applications. Here we discuss one of the theorems from that paper, which can be applied to many generating functions. We apply it to the generating function for the number of binary trees on n nodes, which we showed in (4.4) to be

$$B(z) = \frac{1-\sqrt{1-4z}}{2z}. \quad (4.9)$$

If a function f has a singularity at α, we say it is an *algebraic singularity* if near α we can write f as

$$f(z) = f_0(z) + \frac{g(z)}{(1-z/\alpha)^\omega} \quad (4.10)$$

where f_0 and g are analytic near α, g is nonzero near α, and ω is not $0, -1, -2, \ldots$; we assume here that ω is real. The following theorem is stated (in a slightly different and more general form) in BEND74, Th. 4, p. 498, as a special case of Darboux's Theorem [SZEG59, Th. 8.4, p. 205].

Theorem 3 [BEND74, SZEG59]

Suppose that for some real $r > 0$, $A(z)$ is analytic in the region $|z| < r$, and has a finite number $k > 0$ of singularities on the

circle $|z| = r$, all of which are algebraic. Let α_i, ω_i, and g_i, $i = 1, \ldots, k$, be the values of α, ω, and g in (4.10) corresponding to the i-th such singularity. Then $A(z)$ is the generating function for a sequence $\langle a_n \rangle$ satisfying

$$a_n = \frac{1}{n} \sum_{i=1}^{k} \frac{g_i(\alpha_i) n^{\omega_i}}{\Gamma(\omega_i) \alpha_i^n} + o(r^{-n} n^{w-1}),$$

where w is the maximum of the ω_i, and Γ denotes the gamma function.

Although the statement of this theorem is fairly complex, it can sometimes be applied very easily to give asymptotic information. For example, for the generating function $B(z)$ in (4.9) one easily establishes that

$$r = \frac{1}{4}, \qquad k = 1,$$

$$\alpha_1 = \frac{1}{4}, \qquad \omega_1 = -\frac{1}{2}, \qquad g_1(z) = \frac{-1}{2z}.$$

Hence by the theorem,

$$\begin{aligned} b_n &= \frac{1}{n} \cdot \frac{-2n^{-1/2}}{\Gamma(-\frac{1}{2})(\frac{1}{4})^n} \\ &\quad + o((\tfrac{1}{4})^{-n} n^{-3/2}) \\ &\sim \frac{4^n}{\sqrt{\pi} n^{3/2}}, \end{aligned}$$

which agrees with the result obtained in KNUT68, Sec. 2.3.4.4; STAN80, Sec. 3.3.2; and REIN77, Sec. 3.3, by applying Stirling's approximation to the exact solution (4.5).

Although we do not reproduce the proof of the above theorem, we mention one of the ideas which is close to the heart of the proof, and which is quite interesting in its own right.

Lemma

Suppose that for some real $r > 0$, $A(z)$ is analytic in the region $|z| < r$ and continuous for $|z| \leq r$. Then $A(z)$ is the generating function for a sequence $\langle a_n \rangle$ satisfying

$$a_n = \frac{1}{2\pi i} \int_C \frac{A(z)\, dz}{z^{n+1}},$$

where C is the contour $|z| = r$, the integration is counterclockwise, and i is the square root of -1.

This formula can be established using the Cauchy integral formula [CHUR60, Secs. 51 and 52], the theorem on Taylor series [CHUR60, Sec. 56], and the fact that A must be uniformly continuous on the closed region bounded by C.

For other examples of the extraction of asymptotic information from generating functions, see BEND74 and HARA73, Chap. 9.

5. SUMMARY

We have discussed the application of recurrences to problems in computer science and surveyed some techniques for solving them. We have also seen how generating functions provide a useful technique for manipulating sequences, including some applications other than the solution of recurrences. Many of the techniques used for dealing with recurrences—for example, annihilators—are similar to those used with differential equations [FADE68, Chaps. 9 and 10].

Many authors have written discussions of sequences, recurrences, and generating functions. MILN60 is a useful text entirely on finite differences; in particular, Chapter XIV discusses methods for solving linear recurrences in which the coefficients are rational functions of n. KNUT68, KNUT73, and REIN77 discuss recurrences in the context of algorithm analysis. HALL67, Chap. 3; LIU68, Chaps. 1 and 2; RIOR58, Chaps. 1 and 2; and RIOR68, Chaps. 1 and 4, discuss them in the context of combinatorial analysis. In SEDG75, which performs a very interesting and thorough analysis of a number of variations of Quicksort, Appendix B provides a useful discussion of recurrences and generating functions, especially as they relate to harmonic numbers and binomial coefficients. For more information on generating functions, see BEND74, FLAJ80a, FLAJ80b, LIU68, and STAN78. GONN78 discusses useful methods for obtaining asymptotic estimates of summations. SLOA73 provides a remarkable encyclopedia of various sequences of integers. The reader is encouraged to consult these references for

some fascinating topics which we have not presented here.

ACKNOWLEDGMENTS

This paper was originally written for use in the ICS 161 and ICS 233 classes at the University of California at Irvine. I am grateful to the members of these classes for their comments and suggestions. The referees made many helpful suggestions on the form and content of the paper, and brought to my attention the material on asymptotic analysis of generating functions. I am also grateful to Jon Bentley and Dennis Kibler for their encouragement and suggestions

Preparation of this paper was supported in part by the National Science Foundation under Grant MCS79-04997. It was facilitated by the use of MACSYMA, a large symbolic manipulation program developed at the M.I.T. Laboratory for Computer Science and supported by the National Aeronautics and Space Administration under Grant NSG 1323, by the Office of Naval Research under Grant N00014-77-C-0641. by the U.S. Department of Energy under Grant ET-78-C-02-4687, and by the U.S. Air Force under Grant F49620-79-C-020.

REFERENCES

AHO74 AHO, ALFRED, HOPCROFT, JOHN, AND ULLMAN, JEFFREY *The design and analysis of computer algorithms*, Addison-Wesley, Reading, Mass., 1974.

BELL62 BELLMAN, R. E. "Dynamic programming treatment of the travelling salesman problem," *J. ACM* **9** (1962), 61–63.

BEND74 BENDER, E. A. "Asymptotic methods in enumeration," *SIAM Rev.* **16**, 4 (Oct. 1974), 485–515.

BUCK78 BUCK, R. C. *Advanced calculus*, 3rd ed., McGraw-Hill, New York, 1978.

CHUR60 CHURCHILL, R. V. *Complex variables and applications*, McGraw-Hill, New York, 1960.

DONA69 DONATH, W. E. "Algorithm and average-value bounds for assignment problems," *IBM J. Res. Dev.* **13** (July 1969), 380–386.

FADE64 FADELL, ALBERT G. *Calculus with analytic geometry*, Van Nostrand, Princeton, N.J., 1964.

FADE68 FADELL, ALBERT G. *Vector calculus and differential equations*, Van Nostrand, Princeton, N.J., 1968.

FELL68 FELLER, W. *An introduction to probability theory and its applications, Vol. I*, 3rd ed., Wiley, New York, 1968.

FLAJ80a FLAJOLET, P., FRANÇON, J., AND VUILLEMIN, J. "Sequence of operations analysis for dynamic data structures," *J. Algorith.* 1(1980), 111–141.

FLAJ80b FLAJOLET, P., AND ODLYZKO, A. "Exploring binary trees and other simple trees," in *21st IEEE Symp. Foundations of Computer Science*, Syracuse, N.Y., Oct. 1980, pp. 207–216.

GONN78 GONNET, GASTON H. "Notes on the derivation of asymptotic expressions from summations," *Inf. Process. Lett.* **7**, 4 (June 1978), 165–169.

GONN79 GONNET, GASTON H., AND MUNRO, J. IAN "Efficient ordering of hash tables," *SIAM J. Comput.* **8**, 3 (Aug. 1979), 463–478.

HALL67 HALL, MARSHALL *Combinatorial theory*, Blaisdell, Waltham, Mass., 1967.

HARA73 HARARY, F., AND PALMER, E. M. *Graphical enumeration*, Academic Press, New York, 1973.

HELD62 HELD, M., AND KARP, R. M. "A dynamic programming approach to sequencing problems," *SIAM J.* **10** (1962), 196–210.

HERS64 HERSTEIN, I. N. *Topics in algebra*, Blaisdell, Waltham, Mass., 1964.

ISAA56 ISAAC, E. J., AND SINGLETON, R. C. "Sorting by address calculation," *J. ACM* 3 (1956), 169–174.

KARP77 KARP, R. M. "Probabilistic analysis of partitioning algorithms for the traveling-salesman problem in the plane," *Math. Oper. Res* **2**, 3 (1977), 209–224.

KNUT68 KNUTH, DONALD *The art of computer programming. Vol. 1: Fundamental algorithms*, Addison-Wesley, Reading, Mass., 1968.

KNUT73 KNUTH, DONALD *The art of computer programming. Vol. 3: Sorting and searching*, Addison-Wesley, Reading, Mass., 1973.

LEVY61 LEVY, H., AND LESSMAN, F. *Finite difference equations*, Macmillan, New York, 1961.

LIU68 LIU, C. L. *Introduction to combinatorial mathematics*, McGraw-Hill, New York, 1968.

MILN60 MILNE-THOMSON, L. M. *The calculus of finite differences*, Macmillan, London, 1960.

REIN77 REINGOLD, EDWARD M., NIEVERGELT, JURG, AND DEO, NARSINGH *Combinatorial algorithms: Theory and practice*, Prentice-Hall, Englewood Cliffs, N.J., 1977.

RIOR58 RIORDAN, J. *An introduction to combinatorial analysis*, Wiley, New York, 1958.

RIOR68 RIORDAN, J. *Combinatorial identities*, Wiley, New York, 1968.

SCHO71 SCHONHAGE, A., AND STRASSEN, V "Schnelle Multiplikation Grosser Zahlen," *Computing* **7** (1971), 281–292.

SEDG75 SEDGEWICK, ROBERT "Quicksort," Rep. STAN-CS-75-492, Computer Science Dept., Stanford Univ., Stanford, Calif., May 1975.

SLOA73 SLOANE, N. J. A. *A handbook of integer sequences*, Academic Press, New York, 1973.

STAN78 STANLEY, R. P. "Generating functions," in *MAA studies in mathematics. Vol. 17: Studies in combinatorics*," Gian-Carlo Rota (Ed.), The Mathematical Association of America, 1978, pages 100–141.

STAN80 STANDISH, T. A. *Data structure techniques*, Addison-Wesley, Reading, Mass., 1980.

SZEG59 SZEGO, G. *Orthogonal polynomials*, American Mathematical Society Colloquium Publications, Vol. 23, American Mathematical Society, Washington, D.C., 1959.

WALK79 WALKUP, D. W. "On the expected value of a random assignment problem," *SIAM J. Comput.* **8**, 3 (Aug. 1979), 440–442.

RECEIVED JUNE 1980; FINAL REVISION ACCEPTED SEPTEMBER 1980.