2.8 Suppose you need to generate a random permutation of the first \( N \) integers. For example, \( \{4, 3, 1, 5, 2\} \) and \( \{3, 1, 4, 2, 5\} \) are legal permutations, but \( \{5, 4, 1, 2, 1\} \) is not, because one number (1) is duplicated and another (3) is missing. This routine is often used in simulation of algorithms. We assume the existence of a random number generator, \( r \), with method \( \text{randInt}(i, j) \), that generates integers between \( i \) and \( j \) with equal probability. Here are three algorithms:

1. Fill the array \( a \) from \( a[0] \) to \( a[N-1] \) as follows: To fill \( a[1] \), generate random numbers until you get one that is not already in \( a[0], a[1], \ldots, a[i-1] \).

2. Same as algorithm (1), but keep an extra array called the used array. When a random number, \( r \), is first put in the array \( a \), set used[\( r \)] = true. This means that when filling \( a[i] \) with a random number, you can test in one step to see whether the random number has been used, instead of the (possibly) \( i \) steps in the first algorithm.

3. Fill the array such that \( a[1] = i+1 \). Then

\[
\text{for}( i = 1; i < n; i++ )
\]

\[
\text{swap}(a[i], a[\text{randInt}(0, i)]);
\]

a. Prove that all three algorithms generate only legal permutations and that all permutations are equally likely.
Chapter 2  Algorithm Analysis

about 20 lines of code. The analysis of Shellsort is still not complete, and the disjoint set algorithm has an analysis that is extremely difficult and requires pages and pages of intricate calculations. Most of the analyses that we will encounter here will be simple and involve counting through loops.

An interesting kind of analysis, which we have not touched upon, is lower-bound analysis. We will see an example of this in Chapter 7, where it is proved that any algorithm that sorts by using only comparisons requires $\Omega(N \log N)$ comparisons in the worst case. Lower-bound proofs are generally the most difficult, because they apply not to an algorithm but to a class of algorithms that solve a problem.

We close by mentioning that some of the algorithms described here have real-life application. The gcd algorithm and the exponentiation algorithm are both used in cryptography. Specifically, a 400-digit number is raised to a large power (usually another 400-digit number), with only the low 400 or so digits retained after each multiplication. Since the calculations require dealing with 400-digit numbers, efficiency is obviously important. The straightforward algorithm for exponentiation would require about $10^{600}$ multiplications, whereas the algorithm presented requires only about 2,600 in the worst case.

Exercises

2.1 Order the following functions by growth rate: $N$, $N^{1.3}$, $N^2$, $N \log N$, $N \log \log N$, $N \log^2 N$, $N \log(N^2)$, $2^N$, $2N^2$, $37$, $N^2 \log N$, $N^3$. Indicate which functions grow at the same rate.

2.2 Suppose $T_1(N) = O(f(N))$ and $T_2(N) = O(f(N))$. Which of the following are true?
   a. $T_1(N) + T_2(N) = O(f(N))$
   b. $T_1(N) - T_2(N) = o(f(N))$
   c. $\frac{T_1(N)}{T_2(N)} = O(1)$
   d. $T_1(N) = O(T_2(N))$

2.3 Which function grows faster: $N \log N$ or $N^{1+\epsilon/\sqrt{\log N}}$, $\epsilon > 0$?

2.4 Prove that for any constant, $k$, $\log^k N = o(N)$.

2.5 Find two functions $f(N)$ and $g(N)$ such that neither $f(N) = O(g(N))$ nor $g(N) = O(f(N))$.

2.6 In a recent court case, a judge cited a city for contempt and ordered a fine of $2 for the first day. Each subsequent day, until the city followed the judge’s order, the fine was squared (that is, the fine progressed as follows: $2$, $4$, $16$, $256$, $65536$, $\ldots$).
   a. What would be the fine on day $N$?
   b. How many days would it take for the fine to reach $D$ dollars (a Big-Oh answer will do)?

2.7 For each of the following six program fragments:
   a. Give an analysis of the running time (Big-Oh will do).
   b. Implement the code in the language of your choice, and give the running time for several values of $N$.
   c. Compare your analysis with the actual running times.

2.8 Suppose $X$ is a random variable that takes on the values from 1 to $n$ with equal probability. Determine the expected value and variance of $X$.