Exercises

7.3-1
Why do we analyze the expected running time of a randomized algorithm and not its worst-case running time?

7.3-2
When RANDOMIZED-QUICKSORT runs, how many calls are made to the random-number generator RANDOM in the worst case? How about in the best case? Give your answer in terms of Θ-notation.

7.4 Analysis of quicksort

Section 7.2 gave some intuition for the worst-case behavior of quicksort and for why we expect it to run quickly. In this section, we analyze the behavior of quicksort more rigorously. We begin with a worst-case analysis, which applies to either QUICKSORT or RANDOMIZED-QUICKSORT, and conclude with an analysis of the expected running time of RANDOMIZED-QUICKSORT.

7.4.1 Worst-case analysis

We saw in Section 7.2 that a worst-case split at every level of recursion in quicksort produces a $\Theta(n^2)$ running time, which, intuitively, is the worst-case running time of the algorithm. We now prove this assertion.

Using the substitution method (see Section 4.3), we can show that the running time of quicksort is $O(n^2)$. Let $T(n)$ be the worst-case time for the procedure QUICKSORT on an input of size n. We have the recurrence

$$T(n) = \max_{0 \le q \le n-1} (T(q) + T(n-q-1)) + \Theta(n) , \tag{7.1}$$

where the parameter q ranges from 0 to $n-1$ because the procedure PARTITION produces two subproblems with total size $n-1$. We guess that $T(n) \le cn^2$ for some constant c. Substituting this guess into recurrence (7.1), we obtain

$$\begin{aligned} T(n) &\le \max_{0 \le q \le n-1} (cq^2 + c(n-q-1)^2) + \Theta(n) \\ &= c \cdot \max_{0 \le q \le n-1} (q^2 + (n-q-1)^2) + \Theta(n) . \end{aligned}$$

The expression $q^2 + (n-q-1)^2$ achieves a maximum over the parameter's range $0 \le q \le n-1$ at either endpoint. To verify this claim, note that the second derivative of the expression with respect to q is positive (see Exercise 7.4-3). This

observation gives us the bound $\max_{0 \le q \le n-1}(q^2 + (n-q-1)^2) \le (n-1)^2 = n^2 - 2n + 1$. Continuing with our bounding of $T(n)$, we obtain

$$\begin{aligned} T(n) &\le cn^2 - c(2n-1) + \Theta(n) \\ &\le cn^2 , \end{aligned}$$

since we can pick the constant c large enough so that the $c(2n-1)$ term dominates the $\Theta(n)$ term. Thus, $T(n) = O(n^2)$. We saw in Section 7.2 a specific case in which quicksort takes $\Omega(n^2)$ time: when partitioning is unbalanced. Alternatively, Exercise 7.4-1 asks you to show that recurrence (7.1) has a solution of $T(n) = \Omega(n^2)$. Thus, the (worst-case) running time of quicksort is $\Theta(n^2)$.

7.4.2 Expected running time

We have already seen the intuition behind why the expected running time of RANDOMIZED-QUICKSORT is $O(n \lg n)$: if, in each level of recursion, the split induced by RANDOMIZED-PARTITION puts any constant fraction of the elements on one side of the partition, then the recursion tree has depth $\Theta(\lg n)$, and $O(n)$ work is performed at each level. Even if we add a few new levels with the most unbalanced split possible between these levels, the total time remains $O(n \lg n)$. We can analyze the expected running time of RANDOMIZED-QUICKSORT precisely by first understanding how the partitioning procedure operates and then using this understanding to derive an $O(n \lg n)$ bound on the expected running time. This upper bound on the expected running time, combined with the $\Theta(n \lg n)$ best-case bound we saw in Section 7.2, yields a $\Theta(n \lg n)$ expected running time. We assume throughout that the values of the elements being sorted are distinct.

Running time and comparisons

The QUICKSORT and RANDOMIZED-QUICKSORT procedures differ only in how they select pivot elements; they are the same in all other respects. We can therefore couch our analysis of RANDOMIZED-QUICKSORT by discussing the QUICKSORT and PARTITION procedures, but with the assumption that pivot elements are selected randomly from the subarray passed to RANDOMIZED-PARTITION.

The running time of QUICKSORT is dominated by the time spent in the PARTITION procedure. Each time the PARTITION procedure is called, it selects a pivot element, and this element is never included in any future recursive calls to QUICKSORT and PARTITION. Thus, there can be at most n calls to PARTITION over the entire execution of the quicksort algorithm. One call to PARTITION takes $O(1)$ time plus an amount of time that is proportional to the number of iterations of the **for** loop in lines 3–6. Each iteration of this **for** loop performs a comparison in line 4, comparing the pivot element to another element of the array A. Therefore,

if we can count the total number of times that line 4 is executed, we can bound the total time spent in the **for** loop during the entire execution of QUICKSORT.

Lemma 7.1
Let X be the number of comparisons performed in line 4 of PARTITION over the entire execution of QUICKSORT on an n-element array. Then the running time of QUICKSORT is $O(n + X)$.

Proof By the discussion above, the algorithm makes at most n calls to PARTITION, each of which does a constant amount of work and then executes the **for** loop some number of times. Each iteration of the **for** loop executes line 4. ■

Our goal, therefore, is to compute X, the total number of comparisons performed in all calls to PARTITION. We will not attempt to analyze how many comparisons are made in *each* call to PARTITION. Rather, we will derive an overall bound on the total number of comparisons. To do so, we must understand when the algorithm compares two elements of the array and when it does not. For ease of analysis, we rename the elements of the array A as $z_1, z_2, \ldots, z_n$, with z_i being the ith smallest element. We also define the set $Z_{ij} = \{z_i, z_{i+1}, \ldots, z_j\}$ to be the set of elements between z_i and z_j, inclusive.

When does the algorithm compare z_i and z_j? To answer this question, we first observe that each pair of elements is compared at most once. Why? Elements are compared only to the pivot element and, after a particular call of PARTITION finishes, the pivot element used in that call is never again compared to any other elements.

Our analysis uses indicator random variables (see Section 5.2). We define

$$X_{ij} = \mathrm{I}\{z_i \text{ is compared to } z_j\} \ ,$$

where we are considering whether the comparison takes place at any time during the execution of the algorithm, not just during one iteration or one call of PARTITION. Since each pair is compared at most once, we can easily characterize the total number of comparisons performed by the algorithm:

$$X = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij} \ .$$

Taking expectations of both sides, and then using linearity of expectation and Lemma 5.1, we obtain

$$\mathrm{E}[X] = \mathrm{E}\left[\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}\right]$$

$$= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \mathrm{E}[X_{ij}]$$

$$= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \Pr\{z_i \text{ is compared to } z_j\} \ . \tag{7.2}$$

It remains to compute $\Pr\{z_i \text{ is compared to } z_j\}$. Our analysis assumes that the RANDOMIZED-PARTITION procedure chooses each pivot randomly and independently.

Let us think about when two items are *not* compared. Consider an input to quicksort of the numbers 1 through 10 (in any order), and suppose that the first pivot element is 7. Then the first call to PARTITION separates the numbers into two sets: $\{1, 2, 3, 4, 5, 6\}$ and $\{8, 9, 10\}$. In doing so, the pivot element 7 is compared to all other elements, but no number from the first set (e.g., 2) is or ever will be compared to any number from the second set (e.g., 9).

In general, because we assume that element values are distinct, once a pivot x is chosen with $z_i < x < z_j$, we know that z_i and z_j cannot be compared at any subsequent time. If, on the other hand, z_i is chosen as a pivot before any other item in Z_{ij}, then z_i will be compared to each item in Z_{ij}, except for itself. Similarly, if z_j is chosen as a pivot before any other item in Z_{ij}, then z_j will be compared to each item in Z_{ij}, except for itself. In our example, the values 7 and 9 are compared because 7 is the first item from $Z_{7,9}$ to be chosen as a pivot. In contrast, 2 and 9 will never be compared because the first pivot element chosen from $Z_{2,9}$ is 7. Thus, z_i and z_j are compared if and only if the first element to be chosen as a pivot from Z_{ij} is either z_i or z_j.

We now compute the probability that this event occurs. Prior to the point at which an element from Z_{ij} has been chosen as a pivot, the whole set Z_{ij} is together in the same partition. Therefore, any element of Z_{ij} is equally likely to be the first one chosen as a pivot. Because the set Z_{ij} has $j-i+1$ elements, and because pivots are chosen randomly and independently, the probability that any given element is the first one chosen as a pivot is $1/(j-i+1)$. Thus, we have

$$\begin{aligned}
\Pr\{z_i \text{ is compared to } z_j\} &= \Pr\{z_i \text{ or } z_j \text{ is first pivot chosen from } Z_{ij}\} \\
&= \Pr\{z_i \text{ is first pivot chosen from } Z_{ij}\} \\
&\quad + \Pr\{z_j \text{ is first pivot chosen from } Z_{ij}\} \\
&= \frac{1}{j-i+1} + \frac{1}{j-i+1} \\
&= \frac{2}{j-i+1} \ .
\end{aligned} \tag{7.3}$$

The second line follows because the two events are mutually exclusive. Combining equations (7.2) and (7.3), we get that

$$\mathrm{E}[X] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} .$$

We can evaluate this sum using a change of variables ($k = j - i$) and the bound on the harmonic series in equation (A.7):

$$\begin{aligned}
\mathrm{E}[X] &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} \\
&= \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{2}{k+1} \\
&< \sum_{i=1}^{n-1} \sum_{k=1}^{n} \frac{2}{k} \\
&= \sum_{i=1}^{n-1} O(\lg n) \\
&= O(n \lg n) . \qquad (7.4)
\end{aligned}$$

Thus we conclude that, using RANDOMIZED-PARTITION, the expected running time of quicksort is $O(n \lg n)$ when element values are distinct.

Exercises

7.4-1
Show that in the recurrence

$$T(n) = \max_{0 \le q \le n-1} (T(q) + T(n-q-1)) + \Theta(n) ,$$

$T(n) = \Omega(n^2)$.

7.4-2
Show that quicksort's best-case running time is $\Omega(n \lg n)$.

7.4-3
Show that the expression $q^2 + (n-q-1)^2$ achieves a maximum over $q = 0, 1, \ldots, n-1$ when $q = 0$ or $q = n-1$.

7.4-4
Show that RANDOMIZED-QUICKSORT's expected running time is $\Omega(n \lg n)$.