Problems Chapter 9

Exercise 9.2-1

Show that RANDOMIZED-SELECT never makes a recursive call to a 0-length array.

Assume, to the contrary, that RANDOMIZED-SELECT does make a recursive call to a 0-length array. Then, this recursive call must occur on line 8 or 9. If it occurs on line 8, then it must be the case that $q = p$ after the call to RANDOMIZED-PARTITION on line 3. In which case, $k = 1$ and $i \neq 1$, since otherwise we would have returned $A[q]$ on line 6. But since, $k = 1 \neq i$, it cannot be the case that $i < k$ and so the recursive call to RANDOMIZED-SELEECT on line 8 is not executed. But this contradicts our assumption. Otherwise, the recursive call to RANDOMIZED-SELECT to a 0-length array must occur on line 9. In which case we must have $q = r$. By similar reasoning as above, however, if $q = r$, then we must have $i < k$ and so the recursive call on line 9 cannot possibly execute. Therefore, RANDOMIZED-SELECT does not make a recursive call to a 0-length array.

Exercise 9.2-2

Argue that the indicator random variable X_k and the value $T(max(k-1, n-k))$ are independent.

Consider the the expression $max(k-1, n-k)$ which is defined as

$$max(k-1, n-k) = \begin{cases} k-1 & \text{if } k > \lceil n/2 \rceil \\ n-k & \text{if } k \leq \lceil n/2 \rceil \end{cases} \tag{1}$$

and let Y be the random variable that is the value of $max(k-1, n-k)$. Observe, that $P\{Y = k-1\} = P\{k > \lceil n/2 \rceil\} = \frac{n - \lceil n/2 \rceil}{n}$. This probability is independent of the value of k. And since $P\{Y = n-k\} = 1 - P\{Y = k-1\}$ we conclude that the random variable Y is independent of the value of k for all k. And since X_k is entirely dependent on this quantity, it follows that Y is independent of X_k and finally that the random variables X_k and the value $T(max(k-1, n-k))$ are independent.

Exercise 9.3-2

Analyze SELECT to show that if $n \geq 140$, then at least $\lceil n/4 \rceil$ elements are greater than the median-of-medians x and at least $\lceil n/4 \rceil$ elements are less than

x.

The analysis in the text has shown that for any median-of-medians, x, and sufficiently large n, the number of elements greater than x is at least $\frac{3n}{10} - 6$. The same analysis, except considering the elements that are less than x, will show that the lower bound for the number of elements less than x is the same. Using this lower bound, we may write the following inequality

$$\lceil n/4 \rceil \leq \frac{n}{4} + 1 \leq \frac{3n}{10} - 6 \tag{2}$$

whose solution is $n \geq 140$.

Exercise 9.3-3

Show how quicksort can be made to run in $O(n \lg n)$ time in the worst case, assuming that all elements are distinct.

The worst case running time of quicksort of $\Theta(n^2)$ occurs when the pivot element in the call to PARTITION is either the smallest element in the array, or the greatest element in the array. In other words, the worst case occurs when the pivot is either the 1st order statistic, or it is the nth order statistic. By using the technique of finding the median-of-medians introduced in this chapter, we can guarantee that PARTITION will use a good pivot.

We change the call to PARTITION to choose the pivot by first calling SELECT on the lower median of the array, $\lceil (r-p)/2 \rceil$. The call to select occurs only once, and takes time $O(n)$. Thus, the overall running time of partition of $\Theta(n)$ is unchanged. Thus, the sizes of the subproblems produced by PARTITION are $O(n/2)$. Therefore, we may bound the resulting recurrence relation as follows

$$T(n) \leq 2T(n/2) + O(n) \tag{3}$$

whose solution, by the master theorem, is $T(n) = O(n \lg n)$.

Exercise 9.3-4

Suppose that an algorithm uses only comparisons to find the ith smallest element in a set of n elements. Show that it can also find the $i-1$ smaller elements and the $n-i$ larger elements without performing any additional comparisons.

In order to find the ith smallest element, the algorithm must partition the elements into the $i-1$ that are less than the ith smallest element, and the $n-i$ that are greater. Since the algorithm only uses comparisons, it must, in the process of partitioning the elements, identify each one. By keeping track of which partition each element is assigned as they are assigned, the algorithm can find the $i-1$ smaller elements and the $n-i$ larger elements without performing any additional comparisons.

Exercise 9.3-5

Suppose that you have a "black-box" worst-case linear-time median subroutine. Give a simple linear-time algorithm that solves the selection problem for an arbitrary order statistic.

Let A be the input array of n elements, k be the median of n and i be the order statistic we wish to find. The algorithm is as follows. Run the linear-time median subroutine on the input. If $i == k$, then we are done. Otherwise, if $i < k$, discard those elements of A that are greater than or equal to k and recursively call this algorithm on the remaining elements. If $i > k$, discard those elements of A that are less than or equal to k and recursively call the algorithm on the remaining elements, setting $i = i - k$. Assuming that the elements of the array are distinct, each iteration will result in a subproblem of size that is $O(n/2)$. Additionally, the filtering can be done in lienear time. Therefore, the recurrence relation of this algorithm can be bounded by

$$T(n) \leq T(n/2) + O(n) \tag{4}$$

and the solution of this recurrence, by the master theorem, is $T(n) = O(n)$.

Exercise 9.3-6

The kth ***quantiles*** of an n-element set are the $k - 1$ order statistics that divide the sorted set into k equal-sized sets (to within 1). Give an $O(n \lg k)$-time algorithm to list the kth quantiles of a set.