Solutions for Homework 3

READING: Chapters 6-9, 10, 12.

1. Show that the second largest element can be found with \( n + \lceil \log n \rceil - 2 \) comparisons in the worst case.

   **Solution:** We will first find MAX by using the tournament method. Elements are paired off and compared in rounds. In each round after the first one, the winners from the preceding round are paired off and compared. (If at any round the number of keys is odd, then one of them simply waits for the next round.) We can describe this tournament by a tree, each leaf contains an element, and at each subsequent level the parent of each pair contains the winner. The root contains MAX. We have \( n \) comparisons in total.

   In the process of finding MAX, every element except MAX loses in one comparison. The second largest element must lose directly to MAX. Since MAX is involved in at most \( \lceil \log n \rceil \) comparisons, the second largest must be one of at most \( \lceil \log n \rceil \) elements. We find the maximum of these by \( \lceil \log n \rceil - 1 \) comparisons, that’s the second largest element. Thus the total number of comparisons is \( n + \lceil \log n \rceil - 2 \). (20 points)

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

   **Solution:** We may use the worst-case linear time Selection algorithm to find the (lower) median, and then we modify QUICKSORT by partitioning around this element. Then QUICKSORT will always recurse on subarrays that are at most half the size of the original array, and thus the recurrence for the worst-case running time is

   \[ T(n) \leq 2T(n/2) + \Theta(n) = O(n \log n). \]

   (20 points)
3. Show that $2n - 1$ comparisons are necessary in the worst case to merge two sorted lists containing $n$ elements each.

**Solution:** When the sorted order perfectly interleaves the two lists, each element in the final order must have been compared to its neighbors, which both come from the other list. So consider the two sorted lists $a_1 < \ldots < a_n$ and $b_1 < \ldots < b_n$ such that

$$a_1 < b_1 < a_2 < b_2 < \ldots < a_n < b_n. \quad (1)$$

We claim that in order to correctly merge the two lists, we must make the following $2n - 1$ comparisons

$$(a_1 : b_1), (b_1 : a_2), (a_2 : b_2), \ldots, (a_n : b_n).$$

Indeed, if for example $(b_1 : a_2)$ is not made, then the configuration $a_1 < a_2 < b_1 < b_2 < \ldots < a_n < b_n$ is indistinguishable from (1), since all other results are the same, so our algorithm cannot be correct. (20 points)

4. Describe a $O(n)$ worst-case time algorithm that, given a set $S$ of $n$ distinct numbers and a positive integer $k \leq n$, determines the $k$ numbers in $S$ that are closest to the median of $S$ in the sorted order of $S$ (for simplicity we assume that both $n$ and $k$ are odd, so there is one median).

**Solution:** Let $m = \frac{n+1}{2}$ (the rank of the median). Using the worst-case linear time Selection algorithm twice, we find the two elements $x_1, x_2$ with ranks $m - \frac{k-1}{2}$ and $m + \frac{k-1}{2}$. Then we go through $S$ and find all elements $x$ for which $x_1 \leq x \leq x_2$, these are the solutions. The total worst-case running time is $O(n)$. (20 points)

5. Problem 23-1 a. on page 638.

**Solution:** Let $x$ be a node with two children. Since $x$ has a right child the successor $s$ of $x$ is in the right subtree of $x$. Then $s$ cannot have a left child $y$ because this element would come between $x$ and $s$, and thus $s$ wouldn’t be the successor. (20 points)