CPS 130 Homework 8 - Solutions

1. (CLRS 9.3-3) Show how QUICKSORT can be made to run in $O(n \log n)$ time in the worst case.

   Solution: Modify the PARTITION procedure of QUICKSORT to use the SELECT algorithm to choose the median of the input array as the pivot element. The worst-case running time of SELECT is linear, so we do not increase the time requirement of PARTITION, and selecting the median as pivot guarantees the input is split into two equal parts, so that we always have the best-case partitioning. The running time of the entire computation is then given by the recurrence $T(n) = 2\,T(n/2) + O(n) = O(n \lg n)$.

2. (CLRS 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.

   Solution: Let $A[1 \dots n]$ denote the given array and denote the order statistic by k. The black-box subroutine on A returns the $(n/2)$ element. If $k = n/2$ then we are done. Else, we scan through A and divide into two groups A_1, A_2 those elements less than $A[n/2]$ and those greater than $A[n/2]$, respectively. If $k < n/2$, we find the order statistic for the k^{th} element in A_1. If $k > n/2$, we find the order statistic for the $(n/2 - k)^{th}$ element in A_2. An example algorithm is as follows:

```
SELECTION($A, k$)

    BLACK-BOX($A$)
    IF $k = n/2$ return $A[n/2]$
    DIVIDE($A$) /* returns $A_1, A_2$ */
    IF $k < n/2$ SELECTION($A_1, k$)
    ELSE SELECTION($A_2, n/2 - k$)

END SELECTION
```

   The cost of computing the median using the black-box subroutine is $O(n)$, and the cost of dividing the array is $O(n)$. Let $T(n)$ be the cost of computing the $k-th$ order statistic using the algorithm described above. Then

$$\begin{aligned} T(n) &\leq cn + T(n/2) \\ &= c(n + n/2 + n/4 + n/8 + \dots + T(1)) \\ &\leq 2cn \\ &= O(n) \end{aligned}$$

3. Let A be a list of n (not necessarily distinct) integers. Describe an $O(n)$-algorithm to test whether any item occurs more than $\lceil n/2 \rceil$ times in A.

   Solution: If an element occurs more than $\lceil n/2 \rceil$ times in a A then it must be the median of A. However, the reverse is not true, so once the median is found, you must check to see how many times it occurs in A. The algorithm takes $O(n)$ time provided you use linear selection and $O(n)$ space.

```
TEST($A, n$)

    1 Use linear selection to find the median $m$ of $A$.

    2 Do one more pass through $A$ and count the number of
      occurences of $m$.

        - if $m$ occurs more than $\lceil n/2 \rceil$ times then return
          YES;

        - otherwise return NO.
```

4. (CLRS 9.3-7) Describe an $O(n)$ 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.

   Solution: Assume for simplicity that n is odd and k is even. If the set S was in sorted order, the median is in position $n/2$ and the k numbers in S that closest to the median are in positions $(n-k)/2$ through $(n+k)/2$. We first use linear time selection to find the $(n-k)/2$, $n/2$, and $(n+k)/2$ elements and then pass through the set S to find the numbers less than $(n+k)/2$ element, greater than the $(n-k)/2$ element, and not equal to the $n/2$ element. The algorithm takes $O(n)$ time as we use linear time selection exactly three times and traverse the n numbers in S once.