EL9343 Homework 4

Due: Oct. 6nd 11:00 a.m.

1. Demonstrate the operation of HOARE-PARTITION on the array $A =< 14, 12, 14, 19, 5, 3, 4, 14, 7, 22, 16 >$. Show the array after each iteration of the while loop in the lines of 4 to 11 in the code of lecture notes.
2. For the following array: $A =< 3, 9, 5, 8, 15, 7, 4, 10, 6, 12, 16 >$,
   (a) Create a max heap using the algorithm BUILD-MAX-HEAP.
   (b) Remove the largest item from the max heap you created in 2(a), using the HEAP-EXTRACT-MAX function. Show the array after you have removed the largest item.
   (c) Using the algorithm MAX-HEAP-INSERT, insert 11 into the heap that resulted from question 2(b). Show the array after insertion.
3. For an disordered array with n elements, design an algorithm for finding the median of this array. Your algorithm should traverse the array only once.
   Notes: You can imagine the array as a flow which means you can get the data one by one, and you need to do some *cheap* operation at the time you see each element. The size of this array, n, is big and you know n from the start. Please do not sort the array, or you cannot get full mark. A hint to solve this problem is to use heap.
4. Finding the median of an unordered array in $O(n)$ (Part II). In last homework, we looked at an algorithm that tried to find the median in $O(n)$, but it is not correct. This time we are going to fix it.
   Let's consider a more general problem: given an unsorted array L of n elements ($L[1, \ldots, n]$), how to find the k^{th} smallest element in it (and when $k = \lceil \frac{n}{2} \rceil$, this turns out to find the median). We can also do divide-and-conquer to solve it, by the algorithm called QUICKSELECT, as follows.

   QUICKSELECT(L, p, r, k)
   $q \leftarrow$ **PARTITION**(L, p, r)
   $t \leftarrow q - p + 1$
   if $k == t$ **then**
     return $L[q]$
   else if $k < t$ **then**
     QUICKSELECT$(L, p, q - 1, k)$ {Only look at the left part}
   else
     QUICKSELECT$(L, q + 1, r, k - t)$ {Only look at the right part}
   end if

   (a) The pivot selection in PARTITION plays a key role in optimizing the performance of teh QUICKSORT algorithm. If we use HOARE-PARTITION as the PARTITION function, please **solve** for the worst-case running time. And, what is the average running time (just give the answer)?

   (b) The b^* found in the algorithm in last homework may not be the true median, yet it could serve as a good pivot. Let's look at the BFPRT algorithm (a.k.a. median-of-medians), which uses this pivot to do the partition, as follows.

   BFPRT(L, p, r)
   Divide $L[p, \ldots, r]$ into $\frac{r-p+1}{5}$ lists of size 5 each
   Sort each list, let i^{th} list be $a_i \leq a'_i \leq b_i \leq c_i \leq c'_i, i = 1, 2, \ldots, \frac{r-p+1}{5}$
   Recursively find median of $b_1, b_2, \ldots, b_{\frac{r-p+1}{5}}$, call it b^*
   Use b^* as the pivot and reorder the list (do swapping like in HOARE-PARTITION after pivot selection)
   Suppose after reordering, b^* is at $L[q]$, **return** q

   Solve for the worst-time running time of the QUICKSELECT algorithm, if we utilize BFPRT as PARTITION.
   (**Hints**: In QUICKSORT, the worst-time running time occurs when every partition is extremely unbalanced, so does in QUICKSELECT. Therefore, we need to consider how unbalanced this partition could be. Remember that there is a median finding step in BFPRT algorithm. And in this question we only require the solution in big-O notation.)