CONCORDIA UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE AND SOFTWARE ENGINEERING

COMP 6651: Algorithm Design Techniques

Fall 2019

Quiz # 2

First Name	**Last Name**	**ID#**

Question 1

1. What is the description of the problem solved by the SELECT algorithm?
2. Recall the 5 steps of the SELECT algorithm for computing the k order of a set of n numbers.
3. What is the complexity of each step?
4. What is the overall complexity?

Description of the problem solved by the select algorithm

Selection Problem
Input: A set A of n (distinct) elements and a number i, with $1 \leq i \leq n$
Output: The element $x \in A$ that is larger than exactly $i - 1$ other elements of A
(= find ith **order statistics** of A)
(1 point)

Step 1.

Divide n elements into $\lfloor n/5 \rfloor$ groups of 5 elements. Note that one group may have less than 5 elements.
(.5 point)

Complexity of Step 1.

$O(n)$ **(.5 point)**

Step 2.

Find the median of each group by first insertion sorting the elements of each group, and then picking the median from the sorted list of group elements. $M \leftarrow$ set of medians.
(.5 point)

Complexity of Step 2.

$O(n)$ **(.5 point)**

Step 3.

Use SELECT recursively, i.e., $\textsc{Select}(M, \lfloor \frac{\lceil n/5 \rceil}{2} \rfloor)$, to find the median x of the $\lceil n/5 \rceil$ medians found in Step 2.
(1 point)

Complexity of Step 3.

$T(\lceil n/5 \rceil)$ assuming $T(n)$ denotes the running time of $\textsc{Select}(n)$, i.e., of determining the ith smallest of an input array of $n > 1$ elements.
(1 point)

Step 4.

Partition the input array A around the median-of-medians x using PARTITION. Let i be one more than the number of elements on the low side of the partition, so that x is the ith smallest element and there are $n - i$ elements on the high side of the partition.
(1 point)

Complexity of Step 4.

$O(n)$ using the PARTITION algorithm **(1 point)**

Step 5.

If $i = k$, then return x. Otherwise, use SELECT recursively to find the ith smallest element on the low side if $i < k$
i.e., $\textsc{Select}(A[1..k-1], i))$,
or the $(i - k)$th smallest elements on the high side if $i > k$,
i.e., $\textsc{Select}(A[k+1..n], i-k))$.
(1 point)

Complexity of Step 5.

$\leq T(7n/10 + 6)$ **(1 point)**

Overall Complexity

$$underbrace O(n)_{\text{Step 1}} + \underbrace{O(n)}_{\text{Step 2}} + \underbrace{T(\lceil n/5 \rceil)}_{\text{Step 3}} + \underbrace{O(n)}_{\text{Step 4}} + \underbrace{T(7n/10 + 6)}_{\text{Step 5}} = O(n)$$

(1 point)

Question 2

Recall the quicksort algorithm and its complexity (worst and average case)

For a given subarray $[p..r]$

- **Divide.** Partition (rearrange) the array $A[p..r]$ into two (possibly empty) subarrays $A[p...q-1]$ and $A[q+1..r]$ such that each element of $A[p...q-1]$ is less than or equal to $A[q]$, which is, in turn, less than or equal to each element of $A[q+1..r]$. Compute the index q as part of this partitioning procedure.
- **Conquer.** Sort the two subarrays $A[p...q-1]$ and $A[q+1..r]$ by recursive calls to quicksort.
- **Combine.** Since the subarrays are sorted in place, no work is needed to combine them: the entire array $A[p..r]$ is now sorted.

(4 points) for the correct and complete description of the Quicksort algorithm

Quicksort: The divide-and-conquer paradigm

For a given subarray $[p..r]$ with n elements:
$\textsc{Quicksort}(A, p, r)$ if $p < r$
then
$q \leftarrow \textsc{partition}(A, p, r)$
$\textsc{Quicksort}(A, p, q-1)$
$\textsc{Quicksort}(A, q+1, r)$.

- Worst Case Analysis: $O(n^2)$ **(.5 point)**
- Expected Running Time: $O(n \log n)$ **(.5 point)**

(5 points) for the correct and complete description of the Partitioning algorithm

Partitioning an array

For a given subarray $A = [p..r]$ with n elements

PARTITION(A, p, r) // use $A[r]$ as the pivot for partitioning
// returns location of pivot after partitioning
$x \longleftarrow A[r]$;
$i \longleftarrow p - 1$;
for $j \longleftarrow p$ to $r - 1$
do if $A[j] \leq x$
then $i \longleftarrow i + 1$
exchange $A[i] \leftrightarrow A[j]$
exchange $A[i+1] \leftrightarrow A[r]$
return i+1;