CONCORDIA UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE AND SOFTWARE ENGINEERING

COMP 6651: Algorithm Design Techniques

Winter 2022

Quiz # 2

Question 1

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 (values) in S that are closest to the median of S. Median is not included.

Before describing the algorithm, consider the following example

$$S = \{1, 10, 20, 30, 40, 50, 60, 61, 62, 63, 64, 65, 66\}, n = 13, k = 4,$$

and provide the answer: indicate the median value and the k closest values to that median.

Solution

Example1: if $S = \{1, 3, 5, 9, 13, 21, 101\}$ and $k = 4$, the solution is $\{3, 5, 9, 13\}$ if the median itself is included, but the solution is $\{1, 3, 5, 13\}$ if the median itself is not included. The answer 3, 5, 13, 21 is not correct since 1 is closer to the median (which is 9) than 21. The algorithm should write the output in a separate array, and the numbers do not have to be sorted.

Example2: if $S = \{1, 10, 20, 30, 40, 50, 60, 61, 62, 63, 64, 65, 66\}, n = 13, k = 4$, the solution is $\{61, 62, 63, 64\}$ **1 point**

Note that the k numbers in S that are closest to the median are among the k smallest numbers on either side of the partition (around the median).
We first use linear time selection to find the median.

Then we compute an array B of absolute differences between the ith element in S and the median, $B[i] = |S[i]\text{median}|$. Hence, the k elements closest to the median are the k smallest elements in B (assuming the median has been removed). We use linear time selection to find the $k+1$-th smallest element in B. The first k elements in the resulting partition of B correspond to the k numbers in S closest to the median.

The algorithm takes $O(n)$ time as we use linear time selection exactly two times and traverse the n numbers in S once. The n numbers in the array S are traversed once in order to compute array B. The algorithm time complexity is $O(n)$ **1 point for the complexity analysis**.

Algorithm **2 points for the algorithm**

1. get the median of S
2. compute the distance of S to the median, store in B
3. get the kth order statistics in B. the first k elements in partition of B is the output

Question 2

What is the time complexity of the following algorithm

```
int $i, j$
Initialization: $i, j \leftarrow 0$


for $\{(i = n/2; i \leq n; i++)\{$
        for $(j = 2; j \leq i; j \leftarrow j \times j)\{$
        $\}$
$\}$
```

Solution

j keeps doubling till it is less than or equal to n.
So, j runs for $O(\log n)$ steps.
i runs for $n/2$ steps.
So, total steps $= O(n/2 \times \log(n)) = O(n \times \log n)$.

1 point for the complexity
1 point for its justification

Question 3

Suppose we have a $O(n)$ time algorithm that finds median of an unsorted array. Now consider a QuickSort implementation where we first find median using the above algorithm, then use median as pivot. What will be the worst case time complexity of this modified QuickSort.

You need to justify your answer, write the recurrence relation that defines the complexity of QuickSort in that case, and explain how to solve that recurrence relation.

Solution

When we choose median as pivot element then after the partition algorithm it will go in the middle of the array having half of the elements to left the left and half in the right.

Thus after partition algorithm the array will be divided into two equal parts of $n/2$ elements each.

Hence the resultant recurrence relation would be
$T(n) = O(n)(\text{ for selecting median}) + O(n) \text{ (for partition) } + T(n/2) + T(n/2)$
i.e., $T(n) = O(n) + 2T(n/2)$ **1 point for the recurrence relation**

Solution of the recurrence relation:
Change of variable: $n = 2^k$
$a_k = 2a_{k-1} + O(2^k)$
Homogeneous part: $a_k = 2a_{k-1}$
Characteristic equation: $r = 2 \rightsquigarrow a_k^{(h)} = C2^k$
Particular solution of the type: $a_k^{(p)} = C'k2^k$ as 2 is a root of the characteristic equation
General solution: $a_k = C2^k + C'k2^k = O(2^k) \rightsquigarrow T(n) = O(n \log_2 n)$.

2 points for the solution of the recurrence relation
1 point for the final answer: $O(n \log n)$