CSE317 Design & Analysis of Algorithms (Spring'19)

Quiz #3

Max Marks: 20 *Time Allowed:* 30 minutes

Name: _____ **ERP:** _____

Question 1: ..... *9 marks*

Suppose you are choosing between the following three algorithms:

- Algorithm A solves problems by dividing them into five subproblems of half the size, recursively solving each subproblem, and then combining the solutions in linear time.
- Algorithm B solves problems of size n by recursively solving two subproblems of size $n-1$ and then combining the solutions in constant time.
- Algorithm C solves problems of size n by dividing them into nine subproblems of size $n/3$, recursively solving each subproblem, and then combining the solutions in $O(n^2)$ time.

What are the running times of each of these algorithms (in big-O notation), and which would you choose?

Question 2: ..... *3 marks*

Recall the Partition subroutine employed by QuickSort. You are told that the following array has just been partitioned around some pivot element:

3	1	2	4	5	8	7	6	9

Which of the elements could have been the pivot element? (List all that apply; there could be more than one possibility.)

.....

.....

Question 3: ..... *3 marks*

Define the recursion depth of QuickSort as the maximum number of successive recursive calls it makes before hitting the base case—equivalently, the largest level of its recursion tree. In randomized QuickSort, the recursion depth is a random variable, depending on the pivots chosen. What is the minimum- and maximum-possible recursion depth of randomized QuickSort?

A. minimum: $\Theta(1)$; maximum: $\Theta(n)$

B. minimum: $\Theta(\log n)$; maximum: $\Theta(n)$

C. minimum: $\Theta(\log n)$; maximum: $\Theta(n \log n)$

D. minimum: $\Theta(\sqrt{n})$; maximum: $\Theta(n)$

Question 4: ..... *5 marks*

Suppose you are given an array $A[1..n]$ of sorted integers that has been *circularly shifted* k positions to the right (k is not given). For example, $[35, 42, 5, 15, 27, 29]$ is a sorted array that has been circularly shifted $k = 2$ positions, while $[27, 29, 35, 42, 5, 15]$ has been shifted $k = 4$ positions.

We can obviously find the largest element in A in $O(n)$ time. Describe an $O(\log n)$ algorithm.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....