Rajiv Gandhi Institute of Petroleum Technology, Jais, Amethi
B. Tech CSE/IT/CSD/IDD Mid Semester Examination 2024 – 25
Subject: CS241 Design and Analysis of Algorithms

Time 2 hrs Full Marks 30

Note: Attempt all questions. The marks of the questions are provided along with the questions. You need to explain your answers. Merely writing the correct answer will get you no marks. Make your bound as tight as possible. Whenever you make any assumption, write it.

1. Suppose to solve a problem P, you are choosing among the following two algorithms proposed by –

   - **Student-1:** Algorithm solves the problem of size n by dividing it into five sub problems (time to divide the problem is constant) of half the size, recursively solving each sub problem, and then combining the solutions in linear time.
   - **Student-2:** Algorithm solves the problem of size n by dividing it into two sub problems (time to divide the problem is constant) of size $n-1$, recursively solving each sub problem, and then combining the solutions in constant time.

   Who is giving you the best algorithm in term of time complexity? Justify the reason. [2+2+1]

2. Suppose that in a $0-1$ knapsack problem, the order of the items when sorted by increasing weight is the same as their order when sorted by decreasing value. Give an $O(n)$ time algorithm to find an optimal solution to this variant of the knapsack problem where there are n items. Justify the correctness of your algorithm. [4+1]

3. Let you are developing a matrix-multiplication algorithm that is asymptotically faster than a algorithm based on divide-and-conquer (say ALGORITHM-1). The time complexity of ALGORITHM-1 is $\Theta(n^{\log 7})$. Your algorithm is using the divide-and-conquer method same as ALGORITHM-1. You are dividing each matrix of size $n \times n$ into pieces of size $n/4 \times n/4$. Let the divide and combine steps together takes $\Theta(n^2)$ time. You need to determine how many sub-problems your algorithm has to create in order to beat ALGORITHM-1. If your algorithm creates a number of sub-problems, then the recurrence for the running time of your algorithm $T(n)$ becomes $T(n) = aT(n/4) + \Theta(n^2)$. What is the largest integer value of a for which your algorithm would be asymptotically faster than ALGORITHM-1? [5]

4. To find k^{th} smallest element in an array A of n elements, suppose we modify the SELECT algorithm, *i.e.*, MEDIAN OF MEDIAN algorithm, by breaking the elements into groups of size m, where $m (m \geq 3)$ is an odd number. Analyze the time complexity of this algorithm where the group size m is 3 and 7. [2.5+2.5]

5. (a) RADIX SORT does not work correctly (*i.e.*, does not produce the correct output) if we sort each individual digit using INSERTION SORT instead of COUNTING SORT. This statement is TRUE/FALSE. Explain with reasoning. Without reasoning no marks. [3]

(b) Given an array of n integers, each belonging to $\{-1, 0, 1\}$, we can sort the array in $\mathcal{O}(n)$ time in the worst case. This statement is TRUE/FALSE. Explain with reasoning. Without reasoning no marks. [2]

6. (a) Let $m[i, j]$ be the minimum number of scalar multiplications needed to compute the matrix $A_i A_{i+1} \ldots A_j$. The matrix A_i has dimension $p_{i-1} \times p_i$. The recursive definition for the minimum cost of parenthesizing the product $A_i A_{i+1} \ldots A_j$ is as follows

$$m[i, j] = \begin{cases} 0 & \text{if } i = j \\ \min_{i \le k \le j} \{m[i, k] + m[k + 1, j] + p_{i-1} p_k p_j\} & \text{if } i < j \end{cases}$$

What is the problem with this recursive definition and how to resolve this problem? [2]

(b) Consider there are N distinct positive integers which are stored in an array A. We want to sort this array in ascending order using Quick sort. However, we cannot spend more than $\mathcal{O}(N \log N)$ time. What changes you will make into the Quick Sort so that the worst case time complexity remains $\mathcal{O}(N \log N)$. Also anlaysze the time complexity of your algorithm, *i.e.*, how you have obtained $\mathcal{O}(N \log N)$ time. [3]