ESO207: Data Structures and Algorithms

Theory Assignment 3

Due Date: 11th April, 2021

Total Number of Pages: 2 Total Points 100

Instructions

1. The assignment contains 2 parts - Part 1 and Part 2. Please submit both parts in separate files titled `part1_roll.pdf` and `part2_roll.pdf` respectively (where `roll` is your roll no). Failure to do so will result in loss of marks.
2. All solutions must be typed on a word processing application such as LaTeX or Word. Handwritten solution will not be accepted.
3. For each question you must give the pseudocode of your algorithm and a brief description of the idea of your algorithm.
4. If an algorithm requires a certain time complexity or space complexity, then you must describe why your algorithm indeed works in that complexity bound.
5. The teaching assistant in charge of Part 1 is Siddharth Srivastava (`sidsri@cse.iitk.ac.in`) and in charge of Part 2 is Vimal Raj Sharma (`vimalraj@cse.iitk.ac.in`). Contact them if you have any doubts.

Part 1

Problem1. (25 points) Electric cars are gaining popularity. This creates the need for charging stations (CS). A single CS is composed of multiple charging points (CP). One CP can charge one car at a time. So the total number of vehicles a CS can handle depends on the number of CPs present in it.

Assume you want to construct a new CS at IITK. There are n electric cars arriving at the CS each day and you are provided with the arrival time (an integer array `arr` of size n) and the charging time (an integer array `time` of size n) of all n cars (for simplicity, assume data is identical throughout the year).

Concerning the above scenario, please perform the following tasks.

1. Design an $O(n \log n)$ time greedy algorithm to find the minimum number of CPs required by the CS so that no car waits in a queue for charging.
2. Explain the designed algorithm.
3. Provide proof of correctness of your proposed algorithm.

Problem2. (25 points) Suppose a Binary Search Tree (BST) is used to store students' names enrolled in the ESO207 course using the format $\{FamilyName : MiddleName : FirstName\}$. [Hint: Please note that a single node of the BST contains three elements i.e. Family name, Middle name and First name separated by ':' (colon).]

1. Design an algorithm that allows user to enter a FamilyName and prints the First and Middle names of all students with the given FamilyName present in the course in $O(k + \log_2 n)$ time. (where k is the number of students whose family name matches the FamilyName inputted by the user).
2. Mention assumptions taken for designing this algorithm.
3. Also prove the correctness of your algorithm.

Part 2

Problem3. (25 points) There are n bags in a sequence and each bag has a certain number of candies. You are given an array B of length n, such that $B[i]$ is the number of candies in the ith bag of the sequence, and a number k. For a contiguous subsequence (say σ) of length k (where $1 \leq k \leq n$) of B, cost of equalization of σ, is the minimum sum of candies that must be added or removed from the bags of σ so that all bags in σ have equal number of candies. The optimal cost of B is the minimum cost of equalization over all the $(n-k+1)$ many contiguous subsequences of B. Given an array B and a number k, design an $O(n \log n)$ time algorithm to compute the optimal cost of B. Prove the correctness of your algorithm.

For example, if $B = [4, 8, 7, 6, 9]$ and $k = 3$, then the three subsequences are $(4, 8, 7)$, $(8, 7, 6)$ and $(7, 6, 9)$. The cost of equalizing $(4, 8, 7)$ is 4, the cost of equalizing $(8, 7, 6)$ is 2 and the cost of equalizing $(7, 6, 9)$ is 3. Hence the optimal cost of B is 2.

Problem4. (25 points) Given two sequences S and S' of length n and m, respectively, design an $O(m(n + m))$ time algorithm to find out the minimum number of elements that you need to add in the beginning and end of S so that S' becomes a subsequence of S. Prove the correctness of your algorithm.

For example, if $S = \{4, 8, 9, 3, 2\}$ and $S' = \{5, 8, 3, 1\}$, then the answer is 2 because you only need to add a 5 in the beginning and a 1 in the end of S in order to make S' a subsequence of S.