COMP 3711 – Design and Analysis of Algorithms
2024 Fall Semester – Written Assignment 3
Distributed: 9:00 on October 28, 2024
Due: 23:59 on November 11, 2024

Your solution should contain

(i) your name, (ii) your student ID #, and (iii) your email address

at the top of its first page.

Some Notes:

- Please write clearly and briefly. In particular, your solutions should be written or printed on *clean* white paper with no watermarks, i.e., student society paper is not allowed.
- Please also follow the guidelines on doing your own work and avoiding plagiarism as described on the class home page. ***You must acknowledge individuals who assisted you, or sources where you found solutions.*** Failure to do so will be considered plagiarism.
- The term *Documented Pseudocode* means that your pseudocode must contain documentation, i.e., comments, inside the pseudocode, briefly explaining what each part does.
- Many questions ask you to explain things, e.g., what an algorithm is doing, why it is correct, etc. To receive full points, the explanation must also be *understandable* as well as correct.
- Submit a SOFTCOPY of your assignment to Canvas by the deadline. If your submission is a scan of a handwritten solution, make sure that it is of high enough resolution to be easily read. At least 300dpi and possibly denser.

1. (25 points) You are given the locations of n buildings $B_1, \ldots, B_n$ on the real line. Each location is a coordinate (i.e., a real number). Each building B_i has a WIFI signal receiver with a range of $r_i > 0$. That is, if we place a signal tower at distance r_i or less from B_i, then B_i gets WIFI. Note that the signal receivers of different buildings may have different ranges.

   Describe a greedy algorithm that places the smallest number of signal towers so that every building gets WIFI. Explain the correctness of your algorithm. Derive the running time of your algorithm.

2. (25 pts) Given an array $A[1..n]$ of positive integers and an integer $m \leq n$, we want to split A into at most m non-empty contiguous subarrays so that the largest sum among these subarrays is minimized. Design an algorithm based on a greedy strategy to solve this problem. Analyze the running time of your algorithm. Explain the correctness of your algorithm.

   For example, if the array is $A = [7, 2, 5, 10, 8]$ and $m = 2$, there are several ways to split A: $([7, 2, 4, 10, 8], \emptyset)$, $([7], [2, 5, 10, 8])$, $([7, 2], [5, 10, 8])$, $([7, 2, 5], [10, 8])$, and $([7, 2, 5, 10], [8])$. The best way is $([7, 2, 5], [10, 8])$ because the maximum sum is $10 + 8 = 18$ which is the minimum among all possible ways of splitting.

   Hint: You need to invoke a greedy algorithm multiple times. How many times?

3. (25 points) Consider a two-dimensional array $A[1..n, 1..n]$ of distinct integers. We want to find the *longest increasing path* in A. A sequence of entries $A[i_1, j_1], A[i_2, j_2], \ldots, A[i_k, j_k], A[i_{k+1}, j_{k+1}], \ldots$ is a path in A if and only if every two consecutive entries share a common index and the other indices differ by 1, that is, for all k,

   - either $i_k = i_{k+1}$ and $j_{k+1} \in \{j_k - 1, j_k + 1\}$, or
   - $j_k = j_{k+1}$ and $i_{k+1} \in \{i_k - 1, i_k + 1\}$.

   A path in A is increasing $A[i_1, j_1], A[i_2, j_2], \ldots, A[i_k, j_k], A[i_{k+1}, j_{k+1}], \ldots$ if and only if $A[i_k, j_k] < A[i_{i+1}, j_{k+1}]$. The length of a path is the number of entries in it.

   Design a dynamic programming algorithm to find the longest increasing path in A. Your algorithm needs to output the maximum length as well as the indices of the array entries in the path. Note that there is no restriction on where the longest increasing path may start or end. Define and explain your notations. Define and explain your recurrence and boundary conditions. Write your algorithm in pseudo-code. Derive the running time of your algorithm.

4. (25 points) Let T be a rooted full binary tree of n nodes (not necessarily balanced). Each node v of T is given a positive weight $w(v)$. The depth $d(v)$ of v is the number of edges between v and the root of T. An *ancestor* of v is either v itself or a node u that can be reached from v by following parent pointers.

Let $k \leq n$ be a given positive integer. You are asked to mark exactly k nodes of T as depots such that:

- Every node v in T has a depot as its ancestor. Among the depots that are ancestors of v, the one with the largest depth is the *nearest depot* of v. We denote it by t_v. It is possible that $t_v = v$.
- The sum $\sum_{v \in T} w(v) \cdot (d(v) - d(t_v))$ is minimized.

Design a dynamic programming algorithm for this problem. You only need to output the minimized sum. You need to derive and explain your recurrence and boundary conditions. Analyze the running time of your algorithm.

Hint: Depending on the tree height, k, and whether the subtree root is a depot, you may need many subproblems for a subtree.