CSCI261 Analysis of Algorithms, Homework 4

Due Friday, October 23, 2020, 11:59pm

Problem 1

Given is a sequence $a_1, a_2, \ldots, a_n$ of numbers. We say that its subsequence $a_{j_1}, a_{j_2}, \ldots, a_{j_k}$, where $1 \leq j_1 < j_2 < \ldots < j_k \leq n$, is *kind-of-increasing*, if for every three consecutive elements in this subsequence, the average of the first two elements is smaller than the third element. Formally, $(a_{j_i} + a_{j_{i+1}})/2 < a_{j_{i+2}}$ holds for every $i \in \{1, 2, \ldots, k-2\}$. Design an $O(n^3)$ algorithm that finds the length of a longest kind-of-increasing subsequence of $a_1, \ldots, a_n$.

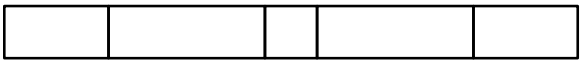
Problem 2

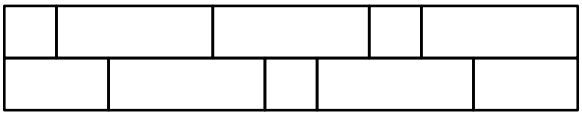
Consider a 2-backpack version of Knapsack. Given are two backpacks of capacities W_1 and W_2. Given are also n items $(w_1, c_1), (w_2, c_2), \ldots, (w_n, c_n)$, where w_i is the weight and c_i the cost of the i-th item. We want to find a set of items that can be split into two parts: one that fits in the first backpack and the other in the second backpack, and the sum of their costs is the largest possible. All the numbers are positive and W_1, W_2, and all the w_i's are integers. Give an $O(nW_1W_2)$ algorithm that finds a set of items to include in the first backpack and a set of items to include in the second backpack such that for each backpack its weight limit is satisfied and the total cost of these items is the largest possible.

Problem 3

This problem is about building a brick wall. We have bricks of lengths 1, 2, and 3. Someone has already built the first brick row of a wall. We want to build the next row. As every builder knows, the bricks in the next row have to "connect" the bricks below — that is for any two neighboring bricks in the first row there has to be a brick above that covers the connection of the two bricks. We have c_1, c_2, and c_3 bricks of lengths 1, 2, and 3, respectively. We also know that the first row consists of n bricks of lengths $\ell_1, \ell_2, \ldots, \ell_n$ in this order, where $\ell_i \in \{1, 2, 3\}$. Design an $O(nc_1c_2c_3)$ algorithm that decides whether it is possible to build the next row with the bricks that we have available.

For example, if the first row consists of bricks of lengths $2, 3, 1, 3, 2$ (as shown below on the left), and we have two bricks of length 1, one brick of length 2, and three bricks of length 3 available, then it is possible to build the next row as shown below on the right (we will leave one brick unused). If, on the other hand, we had one brick of length 1, two of length 2, and three of length 3, it would be impossible to build the next row.





Problem 4

In the Matrix Chain Multiplication problem we are given $a_0, a_1, \ldots, a_n$ that denote the dimensions of n matrices - the i-th matrix A_i is of dimensions $a_{i-1} \times a_i$, and we want to find a parenthesizing that minimizes the number of operations needed to multiply $A_1 A_2 \ldots A_n$. We assume that the number of operations needed to multiply two matrices of dimensions $p \times q$ and $q \times r$ is pqr. In class we will soon discuss how to compute the minimal cost. Give an $O(n^3)$ algorithm that finds a corresponding parenthesizing (more precisely, use $O(n^3)$ time to find the optimal cost, then $O(n)$ time to find the parenthesizing).