CSCI 570 - Fall 2021 - HW 6

Due: October 14

Graded Problems

For the grade problems, you must provide the solution in the form of pseudo-code and also give an analysis on the running time complexity.

Problem 1

[**10 points**] Suppose you have a rod of length N, and you want to cut up the rod and sell the pieces in a way that maximizes the total amount of money you get. A piece of length i is worth p_i dollars. Devise a **Dynamic Programming** algorithm to determine the maximum amount of money you can get by cutting the rod strategically and selling the cut pieces.

Problem 2

[**10 points**] Tommy and Bruiny are playing a turn-based game together. This game involves N marbles placed in a row. The marbles are numbered 1 to N from the left to the right. Marble i has a positive value m_i. On each player's turn, they can remove either the leftmost marble or the rightmost marble from the row and receive points equal to the sum of the remaining marbles' values in the row. The winner is the one with the higher score when there are no marbles left to remove.

Tommy always goes first in this game. Both players wish to maximize their score by the end of the game.

Assuming that both players play optimally, devise a **Dynamic Programming** algorithm to return the difference in Tommy and Bruiny's score once the game has been played for any given input.

Your algorithm **must** run in $O(N^2)$ time.

Problem 3

[**10 points**] The Trojan Band consisting of n band members hurries to lined up in a straight line to start a march. But since band members are not positioned by height the line is looking very messy. The band leader wants to pull out the minimum number of band members that will cause the line to be in a *formation*

(the remaining band members will stay in the line in the same order as they were before). The formation refers to an ordering of band members such that their heights satisfy $r_1 < r_2 < ... < r_i > ... > r_n$, where $1 \leq i \leq n$.

For example, if the heights (in inches) are given as

$$R = (67, 65, 72, 75, 73, 70, 70, 68)$$

the minimum number of band members to pull out to make a formation will be 2, resulting in the following formation:

$$(67, 72, 75, 73, 70, 68)$$

Give an algorithm to find the minimum number of band members to pull out of the line.

Note: you do not need to find the actual formation. You only need to find the minimum number of band members to pull out of the line, but you need to find this minimum number in $O(n^2)$ time.

For this question, you must write your algorithm using pseudo-code.

Problem 4

[**15 points**] You've started a hobby of retail investing into stocks using a mobile app, RogerGood. You magically gained the power to see N days into the future and you can see the prices of one particular stock. Given an array of prices of this particular stock, where $prices[i]$ is the price of a given stock on the ith day, find the maximum profit you can achieve through various buy/sell actions. RogerGood also has a fixed *fee* per transaction. You may complete as many transactions as you like, but you need to pay the transaction fee for each transaction.

Note: You may not engage in multiple transactions simultaneously (i.e., you must sell the stock before you buy again).

Practice Problems

Problem 5

[**10 points**] From the lecture, you know how to use dynamic programming to solve the 0-1 knapsack problem where each item is unique and only one of each kind is available. Now let us consider knapsack problem where you have infinitely many items of each kind. Namely, there are n different types of items. All the items of the same type i have equal size w_i and value v_i. You are offered with infinitely many items of each type. Design a dynamic programming algorithm to compute the optimal value you can get from a knapsack with capacity W.

Problem 6

Given n balloons, indexed from 0 to $n-1$. Each balloon is painted with a number on it represented by array nums. You are asked to burst all the balloons. If the you burst balloon i you will get $nums[left] * nums[i] * nums[right]$ coins. Here left and right are adjacent indices of i. After the burst, the left and right then becomes adjacent. You may assume $nums[-1] = nums[n] = 1$ and they are not real therefore you can not burst them. Design an dynamic programming algorithm to find the maximum coins you can collect by bursting the balloons wisely. Analyze the running time of your algorithm.

Here is an example. If you have the nums arrays equals $[3, 1, 5, 8]$. The optimal solution would be 167, where you burst balloons in the order of 1, 5 3 and 8. The left balloons after each step is:

$$[3, 1, 5, 8] \rightarrow [3, 5, 8] \rightarrow [3, 8] \rightarrow [8] \rightarrow []$$

And the coins you get equals:

$$167 = 3 * 1 * 5 + 3 * 5 * 8 + 1 * 3 * 8 + 1 * 8 * 1.$$

Problem 7

Solve Kleinberg and Tardos, Chapter 6, Exercise 5.

Problem 8

Solve Kleinberg and Tardos, Chapter 6, Exercise 6.