Homework 7

Due: Friday, Oct 24, 11:59 PM PT

For each of the questions below, design a dynamic programming based algorithm by answering the following sub-parts:

a) Define in plain English the **subproblems** that your DP algorithm solves. (4 points)

b) Define a recurrence relation that expresses the value of each subproblem in terms of the smaller subproblems. (6 points)

c) Using the recurrence formula in part b, write pseudocode to find the solution.
Make sure you specify all the base cases and their values, as well as the final answer to the problem. (6 points)

d) What is the time complexity of the solution? Further, determine whether it is an efficient solution or not. (2+2 points)

1. Suppose we are 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 you burst balloon *i*, you will get *nums[left] * nums[i] * nums[right]* coins. Here, *left* and *right* are balloons adjacent to *i* **at the time of bursting**. After bursting any balloon, **its left and right then become adjacent** (thus, there aren't any "gaps" at any point, see example below). Assume *nums[−1] = nums[n] = 1* when determining the coins obtained by bursting the corner balloons, (-1 and n are not indices of real balloons, therefore you can not burst them). Design a dynamic programming algorithm to find the maximum coins you can collect by bursting the balloons. (20 points)

   Here is an example. Suppose you have the *nums* arrays [3, 1, 5, 8]. The optimal solution would yield 167, where you burst balloons in the order of 1, 5, 3 and 8. The balloons left after each step are: [3, **1**, 5, 8] → [3, **5**, 8] → [**3**, 8] → [**8**] → []
   And the coins you get in each step are:
   $(3 * 1 * 5) + (3 * 5 * 8) + (1 * 3 * 8) + (1 * 8 * 1) = 15+120+24+8 = 167$

2. Tommy and Bruiny are playing a turn-based game. It involves *N* marbles placed in a row. Each 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 a score given by the value of that marble. The players' goal is to beat the opponent's score by the highest

margin possible (equivalently, minimizing the losing margin in the case of the player with the lesser score).

Tommy goes first in this game. Devise a Dynamic Programming algorithm to return the maximum difference in score that Tommy can achieve over Bruiny, assuming both players are playing optimally. Your algorithm must run in $O(N^2)$ time.

(**Ungraded** Follow-up: Suppose the score received when removing a marble is changed to 'the sum of the remaining marbles' values left in the row.' How will you modify your solution **without affecting** the runtime complexity?)

3. The Trojan Band consisting of n band members hurries to line up in a straight line to start a march. Due to the height differences, the line is looking very messy. The band leader decides to pull out a few band members so that the line *follows a formation* with the remaining band members staying where they are. We say that k members remaining in the line with heights $r_1, r_2, \ldots, r_k$ are *in formation* if $r_1 < r_2 < \ldots < r_i > \ldots > r_k$, for some $1 \leq i \leq k$.

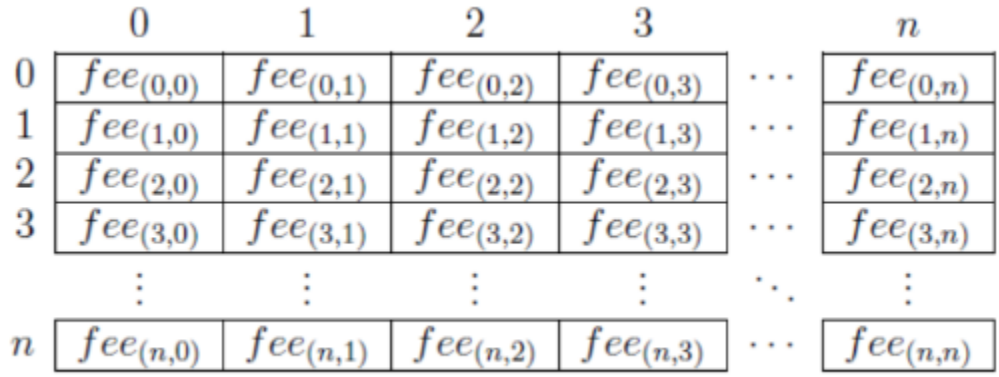
   For example, if the initial sequence of heights in inches is (67, 65, 72, 75, 73, 70, 70, 68),

   then, pulling out member #2 and #6 gives us the formation: (67, 72, **75**, 73, 70, 68).

   Give an algorithm using Dynamic Programming that runs in $O(n^2)$ time to find the minimum number of band members to pull out of the line, so that we are left with a *formation* as described.

Ungraded Problems

1. You are in Downtown of a city where all the streets are one-way streets. At any point, you may go right one block, down one block, or diagonally down and right one block. However, at each city block (i, j) you have to pay the entrance fees fee(i, j). The fees are shown in a grid below:



Your objective is to travel from the starting point at the city's entrance, located at block (0,0), to a specific destination block (n,n) (paying a fee at both these blocks as well).
You would like to get to your destination with the least possible cost.

Formulate the solution to this problem using dynamic programming.

2. Suppose we have N workers to be assigned to work at one of M factories. For each of the M factories, they will produce a different profit depending on how many workers are assigned to that factory. We will denote the profits of factory i with j workers by P(i,j). Develop a dynamic programming solution to find the maximum profit possible by assigning workers to factories.

3. You have two rooms to rent out for a period of D days. There are n customers interested in renting the rooms. The ith customer wishes to rent one room (either room you have) for d[i] days and is willing to pay bid[i] for the entire stay. Customer requests are non-negotiable in that they would not be willing to rent for a shorter or longer duration, but they are indifferent to which days they get room for out of the D days. Devise a dynamic programming algorithm to determine the maximum profit that you can make from the customers over a period of D days.

4. Solve Q4 from exam 1 (non-consecutive subsequence) using Dynamic Programming.