Problem Set 10

Question 1. *Rod-cutting Problem*

Find the maximum total sale price that can be obtained by cutting rod that is n units long (n is a positive integer larger than or equal to one) into integer-length pieces if the sale price of a piece i units long is p_i for $i = 1, ..., n$.

(a) State the problem as a recurrence relation.

(b) Design a dynamic programming pseudocode algorithm for the problem.

(c) What is the (worst-case) time and space complexity for your algorithm?

Question 2. Let M be a $m \times n$ matrix of natural numbers and let C be a natural number. A C-path of M is a sequence of the form $M[i_1, j_1], M[i_2, j_2], \ldots, M[i_l, j_l]$, with $l \geq 1$, such that:

1. the sum of its elements is C;
2. for any two consecutive elements $M[i_k, j_k]$ and $M[i_{k+1}, j_{k+1}]$, either $i_{k+1} = i_k + 1$ and $j_{k+1} = j_k$; or $i_{k+1} = i_k$ and $j_{k+1} = j_k + 1$.

In words: a C-path is a non-empty path of sum C through the matrix: at each step it either goes one cell down, or one cell to the right.

We want to write a dynamic programming algorithm that computes the number of C-paths from $M[0, 0]$ to $M[m, n]$. For example, for

$$M = \begin{array}{|c|c|c|} \hline 1 & 2 & 3 \\ \hline 4 & 6 & 5 \\ \hline 3 & 2 & 1 \\ \hline \end{array} \qquad C = 12$$

we have two such C-paths: $1, 2, 6, 2, 1$ and $1, 2, 3, 5, 1$.

(a) Let $P[i, j, k]$ be the number of k-paths from $M[0, 0]$ to $M[i, j]$. Give recurrence equations that can be used to compute $P[i, j, k]$, and explain why these recurrences hold.

(b) Give a bottom-up dynamic programming algorithm in pseudocode that returns the number of C-paths from $M[0, 0]$ to $M[m, n]$. Analyze the time complexity of your algorithm.

(c) Explain why in practice a top-down, recursive implementation (using memoization) might be faster.

Question 3*.

Given an array A of n integers, let $A[i \ldots j]$ denote the sub-array of A from position i to j. We want to find i and j such that $A[i \ldots j]$ has elements with maximum sum, among all sub-arrays of A.

(a) Give an algorithm to compute such an i and j, in at most $O(n^3)$ time.

(b) Consider the following expression, for $1 \leq j \leq n$:

$$M(j) = \max_{0 \leq i < j} \{\text{sum of the elements of } A[i \ldots j]\}$$

Explain how $M(j)$ can be recursively computed from $M(j - 1)$.

Hint: Give a recurrence relation

(c) We want to use bottom-up dynamic programming to solve the problem in $O(n)$ time. Give a non-recursive algorithm that returns i and j such that $A[i \ldots j]$ is a sub-array of A whose elements have maximum sum. The algorithm should rely on the bottom-up computation of $M(j)$ in point b).

Reason why your algorithm is in $O(n)$.