ESO207A: Data Structures and Algorithms

Practice Set *8: Dynamic Programming* –

Problem 1. Remember the *maximum contiguous subsequence* problem from divide and conquer! We gave an $O(n \log n)$ time algorithm there. You are given an array (or a list) $A[1, \ldots n]$ of n numbers which can be positive or negative. A contiguous subsequence is a segment of the array denoted as $A[i, i+1, \ldots, j]$, for $1 \leq i \leq j \leq n$. The sum of the contiguous subsequence $A[i \ldots j]$ is the sum $S(i,j) = \sum_{k=i}^{j} A[k]$. Give a *linear time* algorithm for finding the contiguous subsequence of maximum sum.

Problem 2. YumDonald (YD) is considering opening a series of restaurants along a national highway. The highway is a straight line and there are n possible locations whose milepost numbers are $m_1, m_2, \ldots, m_n$ in kms (not miles) in increasing order. The constraints are as follows.

1. YumDonald may open at most one restaurant at a location. Moreover, for location i, the expected profit from the restaurant, if opened, is say p_i, $i = 1, \ldots n$.
2. Any two restaurants should be at least k kms apart, for some positive $k > 0$.

Give an efficient algorithm to compute the maximum expected total profit subject to the given constraints. Modify the solution slightly to obtain an optimal assignment of locations for the opening of restaurants.

Problem 3. Given two strings $s = s_1 s_2 \ldots s_m$ and $t = t_1 t_2 \ldots t_m$, design an algorithm to find the longest common substring, that is, the largest K for which there are indices i and j such that $x_i x_{i+1} \ldots x_{i+k-1} = y_j y_{j+1} \ldots y_{j+k-1}$. Show how to do this in time $O(mn)$.

Problem 4. Define a multiplication operation on three symbols a, b, c according to the following table: thus $ab = b, ba = c$, and so on. Note that the multiplication operation defined by the table is neither associative nor commutative.

	a	b	c
a	b	b	a
b	c	b	a
c	a	c	c

Give an efficient algorithm that examines a string of these symbols, e.g., *bbbbac* and decides whether or not it is possible to parenthesize the string in such a way that the value of the resulting expression is a. For example, *bbbbac* can be parenthesized as $((b(bb))(ba)c) = a$ and so the answer should be *yes*.