Weekly Assignment 10:
Dynamic Programming

1. A company wants to open a series of shops along a highway. The n possible locations are along a straight line, and the distances of these locations from the start of the highway are, in kilometers and in increasing order $m_1, m_2, \ldots, m_n$. The constraints are as follows:

   - At any location the company may open at most one shop. The expected profit from opening a shop at location i is p_i, where $p_i > 0$ and $i = 1, 2, \ldots, n$.
   - Any two shops should be at least k kilometers apart, where k is a positive integer.

   Use dynamic programming to give an efficient algorithm for computing the maximum expected total profit, subject to the given constraints. Moreover, describe the recurrence equations on which your algorithm is based, explain why these equations hold, and analyze the time complexity of your algorithm.

2. A shuffle of two strings X and Y is formed by interspersing the characters into a new string, keeping the characters of X and Y in the same order. For example, the string BANANAANANAS is a shuffle of the strings BANANA and ANANAS in several different ways:

   BANANAANANAS BANANAANANAS BANANAANANAS

   Similarly, the strings PRODGYRNAMAMMIINCG and DYPRONGARMAMMICING are both shuffles of DYNAMIC and PROGRAMMING:

   PRODGYRNAMAMMIINCG DYPRONGARMAMMICING

   Given three strings $A[1..m]$, $B[1..n]$, and $C[1..m+n]$, describe and analyze (i.e. provide a correctness argument and time complexity) a dynamic programming algorithm to determine whether C is a shuffle of A and B. Specify the recursion equations on which your algorithm is based.

3. A palindrome is a non-empty string over some alphabet that reads the same forward and backward. Examples of palindromes are: all strings of length 1, civic, ABBA, racecar, and aibohphobia (fear of palindromes). A *subsequence* of a string s is a string that can be derived from s by deleting some elements without changing the order of the remaining elements. E.g., `acd` is a subsequence of `abcde`. We want to solve the following problem: given a string s of length n, find the **length** of the longest palindrome that is a subsequence of s. For example, given the input `character`, the output should be 5, i.e., the length of `carac`.

   (a) What is the simplest algorithm you can think of? What is its complexity? (writing the algorithm explicitly is not required).

   (b) Let $L[i, j]$ be the length of the longest palindrome of the sub-string $s[i, \ldots, j]$. Explain how $L[i, j]$ can be recursively computed.

   Hint: If $s[i] = s[j]$, we have found a palindrome subsequence of length 2, then we have to look for palindromes in $s[i+1, \ldots, j-1]$.

   (c) We want to use bottom-up dynamic programming to solve the problem in $\mathcal{O}(n^2)$ time. Give a non-recursive, bottom-up algorithm that returns the length of the longest palindrome subsequence of a given string s. The algorithm should rely on the bottom-up computation of $L[i, j]$ of point b).