Dynamic Programming Algorithms

Recall that a complete dynamic program is specified by the following four pieces of information:

- The **precise definition** for what cell i or cell (i, j) of your table holds.
- The **formula** for how to fill in the cells of your table.
- The **base cases** for your cells.
- The specific cell that holds the **solution**.

1. Consider the following generalization of the Activity Selection Problem: You are given a set of n activities each with a start time s_i, a finish time f_i, and a weight w_i. Design a dynamic programming algorithm to find the weight of a set of non-conflicting activities with maximum weight.

2. A contiguous subsequence of a list S is a subsequence made up of consecutive elements of S. For instance, if $S = \{5, 15, -30, 10, -5, 40, 10\}$ then $\{15, -30, 10\}$ is a contiguous subsequence but $\{5, 15, 40\}$ is not. Give a dynamic programming algorithm for the following task: You are given a list of numbers, $\{a_1, a_2, \ldots, a_n\}$. You should return the contiguous subsequence of maximum sum (a subsequence of length zero has sum zero). For the preceding example, the answer would be $10, -5, 40, 10$, with a sum of 55.

3. You are going on a long trip. You start on the road at mile post 0. Along the way there are n hotels, at mile posts $a_1 < a_2 < \cdots < a_n$, where each a_i is measured from the starting point. The only places you are allowed to stop are at these hotels, but you can choose which of the hotels you stop at. You must stop at the final hotel (at distance a_n), which is your destination. You would ideally like to travel 300 miles a day, but this may not be possible (depending on the spacing of the hotels). If you travel x miles during a day, the penalty for that day is $(300 - x)^2$. You want to plan your trip so as to minimize the total penalty-that is, the sum, over all travel days, of the daily penalties. Give a dynamic programming algorithm to determine the optimal sequence of hotels at which to stop.

4. Firestones is considering opening a series of restaurants along Highway 1. The n possible locations are along the highway, and the distances of these locations from the downtown San Luis Obispo are, in miles and in increasing order, $m_1, m_2, ..., m_n$. The constraints are:

   - At each location, Firestones may open at most one restaurant. The expected profit from opening a restaurant at location i is p_i, where $p_i > 0$ and $i = 1, 2, \ldots, n$.
   - Any two restaurants must be at least k miles apart, where k is a positive integer.

   Give a dynamic programming algorithm to compute the maximum expected total profit subject to the given constraints.

5. Given two strings $x = x_1 x_2 \cdots x_n$ and $y = y_1 y_2 \cdots y_m$, we wish to find the length of their longest common substring, that is, the largest k for which there are indices i and j with $x_i x_{i+1} \cdots x_{i+k-1} = y_j y_{j+1} \cdots y_{j+k-1}$.

   For example, if $x = dcabagc$ and $y = cbcbcabca$, then the longest common substring is cab with length 3.

   Show how to do this in time $O(mn)$.

6. When a new gene is discovered, a standard approach to understanding its function is to look through a database of known genes and find close matches. The closeness of two genes is measured by the extent to which they are aligned. To formalize this, think of a gene as being a long string over an alphabet $\Sigma = \{A, C, G, T\}$. Consider two genes (strings) $x = ATGCC$ and y = $TACGCA$. An *alignment* of x and y is a way of matching up these two strings by writing them in columns, for instance:

-	A	T	-	G	C	C
T	A	-	C	G	C	A

   Here the "-" indicates a "gap". The characters of each string must appear in order, and each column must contain a character from at least one of the strings. The score of an alignment is specified by a scoring matrix δ of size $(|\Sigma| + 1) \times (|\Sigma| + 1)$, where the extra row and column are to accommodate gaps. For instance the preceding alignment has the following score: $\delta(-, T) + \delta(A, A) + \delta(T, -) + \delta(-, C), +\delta(G, G) + \delta(C, C) + \delta(C, A)$

   Give a dynamic programming algorithm that takes as input two strings $x[1 \dots n]$ and $y[1 \dots m]$ and a scoring matrix δ, and returns the highest scoring alignment.

7. Consider the following task: You are given a set of n positive integers, $\{a_1, a_2, \dots a_n\}$ and some positive integer k. Give a dynamic programming algorithm to determine if there is some subset of the a_i's that adds up to k.

8. You are given a rectangular piece of cloth with dimensions $X \times Y$, where X and Y are positive integers, and a list of n products that can be made using the cloth. For each product $i \in [1, \dots, n]$ you know that a rectangle of cloth of dimensions $a_i \times b_i$ is needed and that the final selling price of the product is c_i. Assume the a_i, b_i, and c_i are all positive integers. You have a machine that can cut any rectangular piece of cloth into two pieces either horizontally or vertically. Design a dynamic programming algorithm that determines the best return on the $X \times Y$ piece of cloth, that is, a strategy for cutting the cloth so that the products made from the resulting pieces give the maximum sum of selling prices. You are free to make as many copies of a given product as you wish, or none if desired.

9. Given an unlimited supply of coins of denominations $x_1, x_2, \dots, x_n$, we wish to make change for a value v; that is, we wish to find a set of coins whose total value is v. This might not be possible: for instance, if the denominations are 5 and 10 then we can make change for 15 but not for 12. Give a dynamic programming algorithm for the following problem: You are given $x_1, \dots, x_n; v$. You must answer the following question: Is it possible to make change for v using coins of denominations $x_1, \dots, x_n$?

10. Consider the following variation on the change-making problem: you are given denominations $x_1, x_2, \dots, x_n$, and you want to make change for a value v, but you are allowed to use each denomination at most once. For instance, if the denominations are 1, 5, 10, 20, then you can make change for 16 = 1 + 15 and for 31 = 1 + 10 + 20 but not for 40 (because you can't use 20 twice).

    You are given positive integers $x_1, x_2, \dots, x_n; v$. Can you make change for v, using each denomination x_i at most once?

    Show how to solve this problem in time $O(nv)$.