Problem Set 6: Dynamic Programming (DP)

1. (EASY) You are given an array A of integers. The goal is to find a contiguous subarray of A such that the sum of its integers is maximized. Design a dynamic programming algorithm to solve this problem with a running time of $O(n)$. Redo the problem if we want to find a subsequence of the array with maximum sum.

2. (EASY) You are given a sequence of items $(h(1), v(1)), ..., (h(n), v(n))$ each with a height $h(i)$ and value $v(i)$. Give an efficient algorithm to compute the subsequence of indices $i_1 < i_2 < ... < i_k$ such that either $h(i_1) < ... < h(i_k)$ or $h(i_1) > ... > h(i_k)$, and whose total value $P_k = \sum_{j=1}^{k} v(i_j)$ is maximized. Design an algorithm for solving this problem in $O(n^2)$ time.

3. (EASY) Let $G = (V, E)$ be an undirected graph with n nodes. A subset of the nodes is called an independent set if no two of them are joined by an edge. G is a path its node can be written as $v_1, v_2, \ldots, v_n$ and for every vertex v_i (which is not the first or the last vertex on the path), v_i is connected to v_{i-1} and v_{i+1}. Each vertex v_i has weight w_i. Design an $O(n)$ time algorithm to find maximum weighted independent set in G.

4. (EASY) Given a triangle array, return the minimum path sum from top to bottom.

   For each step, you may move to an adjacent number of the row below. More formally, if you are on index i on the current row, you may move to either index i or index $i + 1$ on the next row.

   If the triangle is:

   ```
         2
      3     4
   6     5     7
4     1     8     3
   ```

   The minimum path sum from top to bottom is $2 + 3 + 5 + 1 = 11$.

   Given an traingle array with n rows, design an $O(n^2)$ algorithm to find the minimum path.

5. (EASY) Given two string A and B, we wamt to find edit distance between A and B under the following constraints.

   1. A penalty of 2 for deleting a character from A.
   2. A penalty of 3 for inserting a character in A.
   3. A penalty of 4 for substituting a character.

   Modify the edit distance algorithm to satisfy the above constraints.

6. (MEDIUM) Let $p(1), \ldots, p(n)$ be prices of a stock for n consecutive days. A k-block strategy is a collection of m pairs of days $(b_1, s_1), \ldots, (b_m, s_m)$ with $0 \leq m \leq k$ and $1 \leq b_1 < s_1 < \cdots < b_m < s_m \leq n$.

   For each pair of days (b_i, s_i), the investor buys 100 shares of stock on day b_i for a price of $p(b_i)$ and then sells them on day s_i for a price of $p(s_i)$, with a total return of:

   $$100 \sum_{1 \leq i \leq k} (p(s_i) - p(b_i))$$

   Design a dynamic programming (DP) algorithm that takes as input a positive integer k and the prices of the n consecutive days, $p(1), \ldots, p(n)$, and computes the maximum return among all k-block strategies.

7. (MEDIUM) You are given an $n \times 5$ matrix A consisting of integers. For each cell (i, j) in the matrix, its neighbouring cells, if it exists, are defined to be $(i-1, j), (i+1, j), (i, j-1), (i, j+1)$. Your aim is to find a set of cell S such that $\sum_{(i,j) \in S} A[i, j]$ is maximized and for each cell $c \in S$, no neighboring cell of c is in S. Design a DP algorithm for this problem with a running time of $O(n)$.

8. (MEDIUM) You are a manager at a shipping company. On a particular day, you must ship n boxes with weights $w[1...n]$ that are positive integers between 1 and 100. You have three trucks, each with a weight limit of D, which is a positive integer. Design an algorithm to determine if it is possible to pack all the n boxes into the three trucks such that for every truck, the total weight of boxes in the truck is at most D (i.e., the weight limit is not exceeded). Your algorithm should output "yes" if it is possible to pack and "no" otherwise. The running time of your algorithm should be $O(nD^3)$.

9. (MEDIUM) A sequence P exists in a string A if P can obtained by deleting some letters of A. Given two strings A and B (of size m and n respectively), find the longest sequence P that is common to both A and B. The running time of your algorithm should be $O(mn)$.

10. (HARD) Give an efficient algorithm to find the shortest common super-sequence of two strings A and B (of size m and n respectively). Note that C is a super-sequence of A if and only if A is a subsequence of C. The running time of your algorithm should be $O(mn)$.

11. (HARD) A town has n residents labeled 1, ..., n. All n residents live along a single road. The town authorities suspect a virus outbreak and want to set up k testing centers along this road. They want to set up these k testing centers in locations that minimize the total sum of distance that all the residents need to travel to get to their nearest testing center. You have been asked to design an algorithm for finding the optimal locations of the k testing centers.

    Since all residents live along a single road, the location of a resident can be identified by the distance along the road from a single reference point (which can be thought of as the starting point of the town). As input, you are given integer n, integer k, and the location of the residents in an integer array $A[1...n]$ where $A[i]$ denotes the location of resident i. Moreover, $A[1] \leq A[2] \leq A[3] \leq ... \leq A[n]$. Your algorithm should output an integer array $C[1...k]$ of locations such that the following quantity gets minimized:

$$\sum_{i=1}^{n} D(i), \text{ where } D(i) = \min_{j \in \{1,...,k\}} |A[i] - C[j]|$$

    Here $|x - y|$ denotes the absolute value of the difference of numbers x and y. Note that $D(i)$ denotes the distance resident i has to travel to get to the nearest testing center out of centers at $C[1], ..., C[k]$.

    (For example, consider $k = 2$ and $A = [1, 2, 3, 7, 8, 9]$. A solution for this case is $C = [2, 8]$. Note that for testing centers at locations 2 and 8, the total distance traveled by residents will be $(1 + 0 + 1 + 1 + 0 + 1) = 4$.)

    Design a DP algorithm for this problem that outputs the minimum achievable value of the total distance.

12. (MEDIUM) You are walking home from school every day and pass your grandmother's house. On day i, if you visit her, she will give you $L[i]$ lollipops but will not give you any lollipops for the next $k[i]$ days. You are given arrays $L[1, \ldots, n]$ and $k[1, \ldots, n]$ representing the number of lollipops you can get on day i and the subsequent cooldown period, respectively. Design an $O(n)$ algorithm to determine the maximum number of lollipops you can collect over n days.

13. (MEDIUM) The library has n books that must be stored in alphabetical order on adjustable height shelves. The i-th book has height $h[i]$ and thickness $t[i]$. The width of the shelf is fixed at w, and the sum of the thicknesses of books on a single shelf cannot exceed w. The next shelf will be placed atop the tallest book on the shelf. You can assume the shelving takes no vertical space.

    Design an algorithm that minimizes the total height of shelves used to store all the books. You are given the list of books in alphabetical order. Your algorithm should run in time $O(n^2)$.

14. (HARD) Imagine you are playing a game where you are given a long sequence of colors on a string of beads, but there are no clear separations between individual patterns. Your goal is to break this sequence into its distinct color patterns. For example, you might encounter a string of beads with the following color sequence:

`REDGREENBLUEGREENYELLOWREDREDORANGE`

A person familiar with this game's pattern dictionary would parse this sequence as:

`RED GREEN BLUE GREEN YELLOW RED RED ORANGE`

Some sequences can be interpreted in more than one way, but you only care if the sequence can be segmented into valid patterns at all. Given a string S of length n, determine if it can be split into valid patterns. Assume you have access to a subroutine `IsPattern`(i, j) that takes indices i, j with $i \leq j$ as input and checks whether $S[i \ldots j]$ represents a valid pattern, and that it runs in constant time.

Design an algorithm that solves this problem in $O(n^2)$ time.