CSE 4810 – Algorithm Engineering Lab

Lab 4 | Tasks | Topic: Dynamic Programming

1. This problem is called **the Rod-Cutting Problem**. The formulation of the problem is as follows: Given a rod of length n inches and a table of prices p_i for $i = 1, 2, \ldots, n$, determine the maximum revenue r_n obtainable by cutting up the rod and selling the pieces. Note that if the price p_n for a rod of length n is large enough, an optimal solution may require no cutting at all.

   a) Formulate this problem using sub problems.

   b) Analyse the sub-problems and comment on whether the sub-sub-problems of those sub-problems overlap or not.

length, i	1	2	3	4	5	6	7	8	9	10
price, p_i	1	5	8	10	14	8	9	7	4	13

Table 1: Length and associated price for an instance of a rod cutting problem

   c) For the instance of the rod-cutting problem given in Table 1, can you find an optimal solution? Build a dynamic-programming table and analyse that table to find the optimal solution. You can write necessary code to build the table. Include the table in your report.

   d) Does the following "greedy" strategy work for the instance of the rod-cutting problem given in Table 1?
   Define the density of a rod of length i to be p_i/i, that is, its value per inch. The greedy strategy for a rod of length n cuts off a first piece of length i, where $1 \leq i \leq n$, having maximum density. It then continues by applying the greedy strategy to the remaining piece of length $n - i$.

2. Another classic problem regarding dynamic problem is 'Coin Change'. In this problem, there are n types of coins of value $A_1, A_2, \ldots, A_n$. You have to find if it is possible to make a total of Sum value using these coins. You can take each coin as many times as you want.

   a) One way to think about this problem is to notice that to make a total of Sum value using using coins of value $A_1, A_2, \ldots, A_n$, there are 2 scenarios:

   - Not using the coin A_n even once. In that case, the total Sum now has to be constructed using coins of value $A_1, A_2, \ldots, A_{(n-1)}$.
   - Using the coin A_n at least once. In that case, the total $Sum - A_n$ now has to be constructed using coins of value $A_1, A_2, \ldots, A_n$

   i. Can this way of thinking be used to engineer a dynamic-programming solution to solve this problem? If so, please come up with an equation to solve the problem using sub-problems. Show how sub-sub-problems may overlap.

   ii. If you had to make a DP table, what would the rows and columns of that table correspond to?

   iii. Imagine, the coins you are provided are of values $2, 5, 10, 50, 100, 200, 501, 997$. Can you make a DP table to find out if the value 252 can be constructed using these coins or not? You can write code in your computer to generate this DP table.

   iv. Using the same sets of coins $(2, 5, 10, 50, 100, 200, 501, 997)$, can the value 10 be constructed? Use the DP table you already constructed in the previous answer.

   v. What is the asymptotic run-time (**big-O**) of this approach?

vi. Construct another DP table using these sets of coin values $(1, 2, 5, 10, 50, 100, 200, 500, 1000)$. Now try to construct these total values: $37, 72, 101, 7$.

vii. Does the greedy approach of taking the biggest valued coin as many times as possible work for problem (vi)? Does it work for problem (iii)? What could be the reason behind this difference?

3. You will be given a string **S**. Your task is to find out the largest contiguous group of characters in the string that remains the same when it is reversed. Here are some examples for better understanding.

Input: s = "hannah"

Output: "hannah"

Input: s = "mosadasno"

Output: s = "sadas"

Input: s = "tddt"

Output: "dd"

Input: s = "abdc"

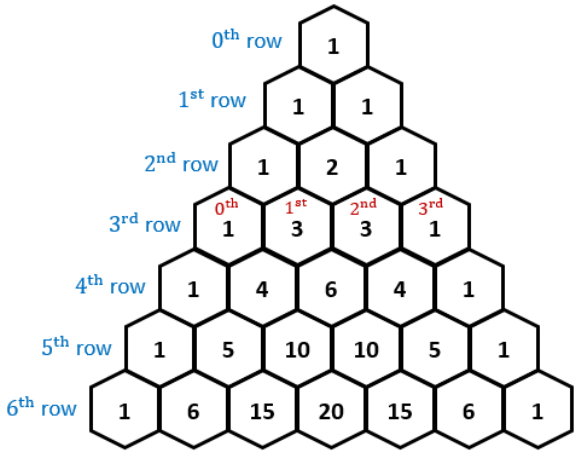
Output: "a" or "b" or "c" or "d"

- Now, construction an algorithm for this problem utilizing dynamic programming.
- Show the DP table for the second input. (Hint: somewhat similar to LCS problem)

4. Construct a dynamic programming solution for the following problem.

Given an integer **R** find the R-th row of a 0-indexed Pascal's triangle.

Pascal's triangle:



Input: 3

Output: 1, 3, 3, 1

Input: 6

Output: 1, 6, 15, 20, 15, 6, 1