Assignment on Dynamic Programming

Data Structures and Algorithms 2 Lab
Section C

1 Subset Sum Problem

Given a set of positive integers (no fraction or real numbers) and a value *sum*, determine if there is a subset of the given set with a sum equal to the given sum.

1.1 Solution Approach:

The Dynamic Programming approach can solve this problem. Please follow the given steps to solve the problem:

1. You are given a set of positive integers in the form of an array, for instance, $A[1, 2, 3]$ (just an example).
2. Create a 2D array DP of size $[size(arr) + 1][sum + 1]$ of type boolean(T/F). The state or each entry of the table $DP[i][j]$ will be **true** if there exists a subset of elements from $A[0...i]$ with sum value $= j$.
3. The approach for the problem is given below:

Listing 1: Subset Sum Pseudo Code

```
 if (A[i-1] > j)
DP[i][j] = DP[i-1][j]
 else
DP[i][j] = DP[i-1][j] OR DP[i-1][j-A[i-1]]
```

4. This means that if the current element on the given array has a value greater than the current sum value on the DP table, we will copy the answer from the previous state.
5. if the current sum value on the DP table is greater than the *ith* element, we will see if any of the previous states have already experienced the $sum = j$ OR any previous states experienced a value j–$A[i]$.

The input-output is shown below:

- Input: set[] = 3, 34, 4, 12, 5, 2, sum = 9 Output: True
- Input: set[] = 3, 34, 4, 12, 5, 2, sum = 30 Output: False

Please refer to this link for a complete understanding of the simulation of the algorithm: DP solution for subset sum. You are to write a separate function to solve this problem.

2 Longest Common Subsequence Problem

Given a sequence, Formally, given a sequence $X =< x_1, x_2, ..., x_m >$, another sequence $Z =< z_1, z_2, ..., z_k >$ is a subsequence of X if there exists a strictly increasing sequence $< i_1, i_2, ..., i_k >$ of indices X such that for all $j = 1, 2, ..., k$ we have $x_{ij} = z_j$. For example, $Z =< B, C, D, B >$ is a subsequence of $X =< A, B, C, B, D, A, B >$ with corresponding index sequence $< 2, 3, 5, 7 >$.

Given two sequences X and Y, we say that a sequence Z is a common subsequence of X and Y if Z is a subsequence of both X and Y.

In the longest-common-subsequence problem, we are given two sequences $X =< x_1, x_2, ..., x_m >$ and $Y =< y_1, y_2, ..., y_m >$ and wish to find a maximum length common subsequence of X and Y.

2.1 Solution Approach:

The Dynamic Programming approach can solve this problem. Please follow the given steps to solve the problem:

1. Let us define $c[i, j]$ to be the length of the LCS of the sequence X_i and Y_j. if either $i = 0$ or $j = 0$, one of the sequences has length 0, so the LCS has length 0. The rest is explained in the recursive formulation given below:

$$c[i,j] = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \text{ ,} \\ c[i-1, j-1] + 1 & \text{if } i, j > 0 \text{ and } x_i = y_j \text{ ,} \\ \max(c[i, j-1], c[i-1, j]) & \text{if } i, j > 0 \text{ and } x_i \neq y_j \text{ .} \end{cases}$$

2. The pseudocode is given below. **NO NEED TO USE THE *b* table in your code. Only find the length of the LCS.**

```
LCS-LENGTH(X, Y)
m = X.length
n = Y.length
let b[1..m, 1..n] and c[0..m, 0..n] be new tables
for i = 1 to m
    c[i, 0] = 0
for j = 0 to n
    c[0, j] = 0
for i = 1 to m
    for j = 1 to n
        if x_i == y_j
            c[i, j] = c[i − 1, j − 1] + 1
            b[i, j] = "↖"
        elseif c[i − 1, j] ≥ c[i, j − 1]
            c[i, j] = c[i − 1, j]
            b[i, j] = "↑"
        else c[i, j] = c[i, j − 1]
            b[i, j] = "←"
return c and b
```

The input-output is shown below:

- Input: X = *abcdaf* Y = *acbcf* Output: Length: 4

Please refer to this link for a complete understanding of the simulation of the algorithm: DP solution for LCS. You are to write a separate function to solve this problem.

3 Marks Distribution and Guidelines

Problems	Marks
Subset Sum	10
Longest Common Subsequence	10
Total	10

Please Follow the guidelines below:

- Please solve the problems in $C++$ language in **separate files**. Name the files in the following way: *studentid_problem1_dp.cpp*. For instance, if your ID is 110202, then for problem 1, it would be: *110202_problem1_dp.cpp*
- Place both files in a folder. Then **zip the folder and put your student ID as the folder's name**.
- **DO NOT PUT *.EXE* or *.OUT* file in the folder. If found, then there will be a marks reduction.**

- **DO NOT COPY from the internet, seniors, batchmates, or any other sources. You are always welcome to discuss and find the solutions independently, but you must write your code. If found out, there will be *-100%* marks reduction.**