Ex-no: 7
Date: 25.04.2023

M. Rohith
3122 21 5001 085

Sri Sivasubramaniya Nadar College of Engineering, Kalavakkam - 603 110
(An Autonomous Institution, Affiliated to Anna University, Chennai)

UCS2403: DESIGN & ANALYSIS OF ALGORITHMS

Assignment 7

1. Given two sequences $X = \langle x_1, \ldots, x_m \rangle$ and $Y = \langle y_1, \ldots, y_n \rangle$, the longest common sub-sequence problem (LCS) seeks to find a maximum length common sub-sequence of X and Y.
   For example, if $X = \langle A, B, C, B, D, A, B \rangle$ and $Y = \langle B, D, C, A, B, A \rangle$, then the sequences $\langle B, C, B, A \rangle$ and $\langle B, D, A, B \rangle$ are the longest common sub-sequences, since X and Y have no common sub-sequence of length 5 or greater.

   (a) Obtain a recursive formula for the LCS problem.

   (b) Design a dynamic programming algorithm to solve the LCS problem using the recursive formula in Q 1(a). Write the Python code to implement the same.

   (c) Populate the table using bottom-up approach, starting from the base case(s), for the below example: $X = \langle A, B, A, A, B, A \rangle$ and $Y = \langle B, A, B, B, A, B \rangle$. Find the answer using the data populated in the table.

   (d) Compare the output/answers obtained in Q 1(b) and Q 1(c) for the given example.

Ex-no: 7
Date: 25.04.2023

M. Rohith
3122 21 5001 085

1) Given two sequences X = ⟨x1, . . . , xm⟩ and Y = ⟨y1, . . . , yn⟩, the longest common sub-sequence problem (LCS) seeks to find a maximum length common sub-sequence of X and Y . For example, if X = ⟨A, B, C, B, D, A, B⟩ and Y = ⟨B, D, C, A, B, A⟩, then the sequences ⟨B, C, B, A⟩ and ⟨B, D, A, B⟩ are the longest common sub-sequences, since X and Y have no common sub-sequence of length 5 or greater.

(a) Obtain a recursive formula for the LCS problem.

ALGORITHM IMPLEMENTING RECURSIVE FORMULA:

```
def lcs_length(X, Y):
    if len(X) == 0 or len(Y) == 0:
        return 0
    elif X[-1] == Y[-1]:
        return lcs_length(X[:-1], Y[:-1]) + 1
    else:
        return max(lcs_length(X[:-1], Y), lcs_length(X, Y[:-1]))
```

(b) Design a dynamic programming algorithm to solve the LCS problem using the recursive formula in Q 1(a). Write the Python code to implement the same.

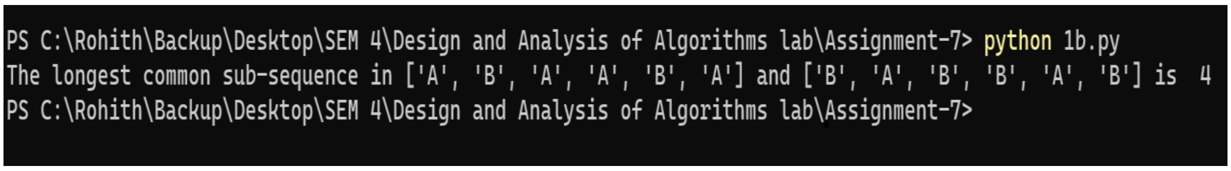
Using recursive function call:

SOURCE CODE:

```python
def lcs_length(X, Y):
    if len(X) == 0 or len(Y) == 0:
        return 0
    elif X[-1] == Y[-1]:
        return lcs_length(X[:-1], Y[:-1]) + 1
    else:
        return max(lcs_length(X[:-1], Y), lcs_length(X, Y[:-1]))

X=['A','B','A','A','B','A']
Y=['B','A','B','B','A','B']
print("The longest common sub-sequence in",X,"and",Y,"is
",lcs_length(X,Y))
```

OUTPUT:



```
PS C:\Rohith\Backup\Desktop\SEM 4\Design and Analysis of Algorithms lab\Assignment-7> python 1b.py
The longest common sub-sequence in ['A', 'B', 'A', 'A', 'B', 'A'] and ['B', 'A', 'B', 'B', 'A', 'B'] is  4
PS C:\Rohith\Backup\Desktop\SEM 4\Design and Analysis of Algorithms lab\Assignment-7>
```

TIME-COMPLEXITY:

The time complexity of the recursive algorithm will be exponential, as it repeatedly computes the same subproblems multiple times, making the same computations over and over again. The exact time complexity of the recursive algorithm is more difficult to determine, as it depends on the structure of the input sequences and the order in which the subproblems are computed. However, in the worst case, the time complexity is O(2^(m+n)), which is very high and makes the algorithm impractical for longer sequences.

(c) Populate the table using bottom-up approach, starting from the base case(s), for the below example:

X = ⟨A, B, A, A, B, A⟩ and Y = ⟨B, A, B, B, A, B⟩.

Find the answer using the data populated in the table.

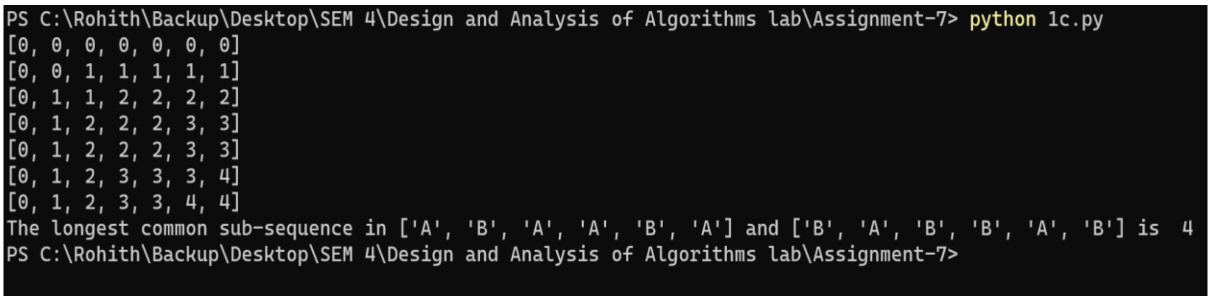
Algorithm Using Bottom-Up Table Approach:

SOURCE CODE:

```python
def lcs_length(X, Y):
    m = len(X)
    n = len(Y)
    # Initialize the table to all zeros
    L = [[0] * (n+1) for i in range(m+1)]
    # Fill in the table bottom-up
    for i in range(1, m+1):
        for j in range(1, n+1):
            if X[i-1] == Y[j-1]:
                L[i][j] = L[i-1][j-1] + 1
            else:
                L[i][j] = max(L[i-1][j], L[i][j-1])
    # The length of the LCS is the value in the bottom-right corner 
of the table or matrix
    for i in range(m+1):
            print(L[i])
    return L[m][n]

X=['A','B','A','A','B','A']
Y=['B','A','B','B','A','B']
print("The longest common sub-sequence in",X,"and",Y,"is
",lcs_length(X,Y))
```

OUTPUT:



```
PS C:\Rohith\Backup\Desktop\SEM 4\Design and Analysis of Algorithms lab\Assignment-7> python 1c.py
[0, 0, 0, 0, 0, 0, 0]
[0, 0, 1, 1, 1, 1, 1]
[0, 1, 1, 2, 2, 2, 2]
[0, 1, 2, 2, 2, 3, 3]
[0, 1, 2, 2, 2, 3, 3]
[0, 1, 2, 3, 3, 3, 4]
[0, 1, 2, 3, 3, 4, 4]
The longest common sub-sequence in ['A', 'B', 'A', 'A', 'B', 'A'] and ['B', 'A', 'B', 'B', 'A', 'B'] is  4
PS C:\Rohith\Backup\Desktop\SEM 4\Design and Analysis of Algorithms lab\Assignment-7>
```

TIME-COMPLEXITY:

We can solve this recurrence relation using dynamic programming, by filling in a table L where L[i][j] represents the length of the LCS of X[1..i] and Y[1..j]. We can fill in the table row by row, from top to bottom and from left to right, using the recurrence relation.

The time complexity of the dynamic programming algorithm is O(mn), where m is the length of sequence X and n is the length of sequence Y. This is because we fill in a table of size (m+1) x (n+1) using a nested loop that iterates over the rows and columns of the table. Each cell of the table takes constant time to compute, since it only depends on the values of adjacent cells and the current characters of the sequences. Therefore, the time complexity of the algorithm is O(mn).

(d) Compare the output/answers obtained in Q 1(b) and Q 1(c) for the given example.

From the output snapshots generated by both algorithms, we can observe that the length of the longest sub-sequence for the same input is equal in both cases.