

Name	Stephen David Vaz
UID	2021700070
Experiment No.	4

AIM:	Dynamic Programming - Longest Common Subsequence
PROBLEM STATEMENT :	Apply the concept of dynamic programming to solve the problem of finding Longest Common Subsequence
ALGORITHM/ THEORY:	<p>Algorithm:-</p> <ol style="list-style-type: none"> 1. Define two sequences X and Y, with lengths m and n, respectively. 2. Create a matrix of size (m) x (n) and initialize all entries to zero. 3. For i = 1 to m, and for j = 1 to n: <ol style="list-style-type: none"> a. If the i-th character of X is equal to the j-th character of Y, set the i,j entry in the matrix to be the value of the i-1,j-1 entry plus one. b. Otherwise, set the i,j entry in the matrix to be the maximum of the i-1,j and i,j-1 entries. 4. The value of the i,j entry in the matrix represents the length of the longest common subsequence of the first i characters of X and the first j characters of Y. 5. Trace back the matrix starting from the bottom right corner and find the longest common subsequence. <p>The four steps of dynamic programming are:</p> <ol style="list-style-type: none"> 1. Define the problem and identify subproblems. 2. Formulate a recursive relationship between the subproblems. 3. Create a memoization table to store solutions to each subproblem. 4. Solve subproblems in a specific order to obtain the solution to the original problem. <p>Some important applications of LCS include:</p> <ol style="list-style-type: none"> 1. DNA Sequencing 2. Text Comparison 3. Speech Recognition 4. Image Recognition

PROGRAM:

```
#include <stdio.h>
#include <string.h>

void lcs(char str1[], char str2[])
{
    int i, j, m, n, table[20][20];
    m = strlen(str1);
    n = strlen(str2);

    for (i = 0; i <= m; i++)
        table[i][0] = 0;
    for (i = 0; i <= n; i++)
        table[0][i] = 0;

    for (i = 1; i <= m; i++)
        for (j = 1; j <= n; j++)
        {
            if (str1[i - 1] == str2[j - 1])
            {
                table[i][j] = table[i - 1][j - 1] + 1;
            }
            else if (table[i - 1][j] >= table[i][j - 1])
            {
                table[i][j] = table[i - 1][j];
            }
            else
            {
                table[i][j] = table[i][j - 1];
            }
        }

    int index = table[m][n];
    printf("\nLength of LCS: %d\n", index);
    char lcsAlgo[index];
    lcsAlgo[index] = '\0';

    i = m, j = n;
    while (i > 0 && j > 0)
    {
        if (str1[i - 1] == str2[j - 1])
        {
            lcsAlgo[index - 1] = str1[i - 1];
            i--;
            j--;
            index--;
        }
        else if (table[i - 1][j] > table[i][j - 1])
            i--;
        else
            j--;
    }
    printf("\nLCS: %s\n", lcsAlgo);
}
```

```
int main()
{
    char str1[20], str2[20];
    printf("String 1: ");
    scanf("%s", str1);
    printf("String 2: ");
    scanf("%s", str2);
    lcs(str1, str2);
}
```

RESULT:

Using Dynamic Programming,

Time complexity: $O(mn)$

Hirschberg's algorithm uses divide-and-conquer technique

Time complexity: $O(mn \log n)$

Using suffix trees or suffix arrays

Time complexity: $O(m + n)$

where m and n are the lengths of the two input sequences

```
● * Executing task: /usr/bin/clang /Users/stephen03/Dev/repos/step
DAA/exp4/lcs.c -o ../excs/lcs && ../excs/lcs

String 1: abcdef
String 2: abdcf

Length of LCS: 4

LCS: abdf
* Terminal will be reused by tasks, press any key to close it.
```

CONCLUSION:

Successfully understood the concept of dynamic programming and solved the problem of finding out the longest common subsequence for two strings.