NAME:	Shubham Vishwakarma
UID No.	2021700071
BRANCH:	S.Y CSE-DS
BATCH:	D
SUBJECT	Design and Analysis of Algorithms
EXPERIMENT No.	4
Date of Performance	13/03/2023
Date of Submission	16/03/2023

AIM:

Dynamic Programming -Longest Common Subsequence

Program 1

PROBLEM STATEMENT :

Implement the longest common subsequence algorithm for the given problem statement.

ALGORITHM/ THEORY:

Longest Common Sequence (LCS)

A subsequence of a given sequence is just the given sequence with some elements left out.

Given two sequences X and Y, we say that the sequence Z is a common sequence 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 $Y = (y_1\ y_2\ y_n)$ and wish to find a maximum length common subsequence of X and Y. LCS Problem can be solved using dynamic programming.

Characteristics of Longest Common Sequence

A brute-force approach we find all the subsequences of X and check each subsequence to see if it is also a subsequence of Y, this approach requires exponential time making it impractical for the long sequence.

Given a sequence $X = (x_1\ x_2.....x_m)$ we define the ith prefix of X for i=0, 1, and 2...m as $X_i= (x_1\ x_2.....x_i)$. For example: if X = (A, B, C, B, C, A, B, C) then X_4= (A, B, C, B)

Optimal Substructure of an LCS: Let X = (x_1 x_2....x_m) and Y = (y_1 y_2.....) y_n) be the sequences and let Z = (z_1 z_2......z_k) be any LCS of X and Y.

- If x_m = y_n, then z_k=x_m=y_n and Z_{k-1} is an LCS of $X_{m-1}$and Y_{n-1}
- If $x_m \neq y_n$, then $z_k \neq x_m$ implies that Z is an LCS of $X_{m-1}$and Y.
- If $x_m \neq y_n$, then $z_k \neq y_n$ implies that Z is an LCS of X and Y_{n-1}

Step 2: Recursive Solution: LCS has overlapping subproblems property because to find LCS of X and Y, we may need to find the LCS of X_{m-1} and Y_{n-1}. If $x_m \neq y_n$, then we must solve two subproblems finding an LCS of X and Y_{n-1}.Whenever of these LCS's longer is an LCS of x and y. But each of these subproblems has the subproblems of finding the LCS of X_{m-1} and Y_{n-1}.

Let c [i,j] be the length of LCS of the sequence $X_i$and Y_j.If either i=0 and j =0, one of the sequences has length 0, so the LCS has length 0. The optimal substructure of the LCS problem given the recurrence formula

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

Step3: Computing the length of an LCS: let two sequences X = (x_1 x_2.....x_m) and Y = (y_1 y_2..... y_n) as inputs. It stores the c [i,j] values in the table c [0......m,0..........n].Table b [1..........m, 1..........n] is maintained which help us to construct an optimal solution. c [m, n] contains the length of an LCS of X,Y.

PROGRAM:	**Recursive Approach:**

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

int i, j, m, n, LCS_table[20][20];
char S1[20], S2[20], b[20][20];

void lcsAlgo() {
    m = strlen(S1);
    n = strlen(S2);

    // Filling 0's in the matrix
    for (i = 0; i <= m; i++)
        LCS_table[i][0] = 0;
    for (i = 0; i <= n; i++)
        LCS_table[0][i] = 0;

    // Building the mtrix in bottom-up way
    for (i = 1; i <= m; i++)
        for (j = 1; j <= n; j++) {
        if (S1[i - 1] == S2[j - 1]) {
            LCS_table[i][j] = LCS_table[i - 1][j - 1] + 1;
        } else if (LCS_table[i - 1][j] >= LCS_table[i][j - 1]) {
            LCS_table[i][j] = LCS_table[i - 1][j];
        } else {
            LCS_table[i][j] = LCS_table[i][j - 1];
        }
        }

    int index = LCS_table[m][n];
    char lcsAlgo[index + 1];
    lcsAlgo[index] = '\0';

    int i = m, j = n;
    while (i > 0 && j > 0) {
        if (S1[i - 1] == S2[j - 1]) {
        lcsAlgo[index - 1] = S1[i - 1];
        i--;
        j--;
        index--;
        }

        else if (LCS_table[i - 1][j] > LCS_table[i][j - 1])
        i--;
        else
        j--;
    }
```

```
    // Printing the sub sequences
    printf("S1 : %s \nS2 : %s \n", S1, S2);
    printf("The longest common subsequence is  %s", lcsAlgo);
}

int main()
{
    printf("Enter 1st sequence: ");
    scanf("%s",S1);
    printf("Enter 2nd sequence: ");
    scanf("%s",S2);
    lcsAlgo();
    printf("\n");

return 0;
}
```

Iterative Approach:

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

int i,j,m,n,c[20][20];
char x[20],y[20],b[20][20];

void print(int i,int j)
{
    if(i==0 || j==0)
        return;
    if(b[i][j]=='c')
    {
        print(i-1,j-1);
        printf("%c",x[i-1]);
    }
    else if(b[i][j]=='u')
        print(i-1,j);
    else
        print(i,j-1);
}

void lcs()
{
    m=strlen(x);
```

	`` n=strlen(y); for(i=0;i<=m;i++) c[i][0]=0; for(i=0;i<=n;i++) c[0][i]=0; //c, u and l denotes cross, upward and downward directions respectively for(i=1;i<=m;i++) for(j=1;j<=n;j++) { if(x[i-1]==y[j-1]) { c[i][j]=c[i-1][j-1]+1; b[i][j]='c'; } else if(c[i-1][j]>=c[i][j-1]) { c[i][j]=c[i-1][j]; b[i][j]='u'; } else { c[i][j]=c[i][j-1]; b[i][j]='l'; } } } int main() { printf("Enter 1st sequence:"); scanf("%s",x); printf("Enter 2nd sequence:"); scanf("%s",y); printf("\nThe Longest Common Subsequence is "); lcs(); print(m,n); return 0; } ```

RESULT:

```
PS C:\Users\smsha\Desktop\SEM 4\DAA\Practicals\Exp4\output> & .\'lcs.exe'
Enter 1st sequence: ACADB
Enter 2nd sequence: CBDA
S1 : ACADB
S2 : CBDA
The longest common subsequence is  CB
PS C:\Users\smsha\Desktop\SEM 4\DAA\Practicals\Exp4\output> & .\'lcs.exe'
Enter 1st sequence: BACDB
Enter 2nd sequence: BDCB
S1 : BACDB
S2 : BDCB
The longest common subsequence is  BDB
PS C:\Users\smsha\Desktop\SEM 4\DAA\Practicals\Exp4\output>
```

Self-analysis:

Longest common subsequence

(Q1.) S1 = { A, C, A, D, B }
S2 = { C, B, D, A }

S1\S2		C	B	D	A
	0	0	0	0	0
A	0	0	0	0	1
C	0	1	1	1	1
A	0	1	1	1	2
B	0	1	1	2	2
D	0	1	2	2	2

LCS = { C, A }

(Q2.) X = BACDB
Y = BDCB

X\Y		B	D	C	B
	0	0	0	0	0
B	0	1	1	1	1
A	0	1	1	1	1
C	0	1	1	2	2
D	0	1	2	2	2
B	0	1	2	2	3

LCS = { B, C, B }

CONCLUSION :	In this longest common subsequence problem article, we learned what the LCS problem is with the help of examples. We also discovered the recursive solution to the LCS problem, along with its complexity analysis. After that, we investigated a more optimal approach to implement the LCS solution, which is known as dynamic programming. Finally, we came across the strategy and program to build a bottom-up dynamic programming solution.