DESIGN AND ANALYSIS OF ALGORITHMS

EXPERIMENT 4

Name: Janhavi R Deshmukh

Branch: CSE(DS) D1

UID: 2021700017

Aim: – Finding longest common subsequence of two strings using dynamic programming approach.

Theory:

The longest common subsequence (LCS) is defined as the longest subsequence that is common to all the given sequences, provided that the elements of the subsequence are not required to occupy consecutive positions within the original sequences.If `S1` and `S2` are the two given sequences then, `Z` is the common subsequence of `S1` and `S2` if `Z` is a subsequence of both `S1` and `S2`. Furthermore, `Z` must be a strictly increasing sequence of the indices of both `S1` and `S2`.

1. Create a 2D array **dp[][]** with rows and columns equal to the length of each input string plus 1 [the number of rows indicates the indices of **S1** and the columns indicate the indices of **S2**].
2. Initialize the first row and column of the dp array to 0.
3. Iterate through the rows of the dp array, starting from 1 (say using iterator **i**).
    a. For each **i**, iterate all the columns from **j = 1 to n**:
        i. If **S1[i-1]** is equal to **S2[j-1]**, set the current element of the dp array to the value of the element to (**dp[i-1][j-1] + 1**).
        ii. Else, set the current element of the dp array to the maximum value of **dp[i-1][j]** and **dp[i][j-1]**.
4. After the nested loops, the last element of the dp array will contain the length of the LCS.

Program:

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

int c[20][20],b[20][20];
char S1[20];
char S2[20];

int max(int a, int b)
{
    if(a>b){
        return a;
```

```
    }
    else
        return b;
}

void print(int i,int j)
{
    if(i==0 || j==0)
        return;
    if(b[i][j]=='c')
    {
        print(i-1,j-1);
        printf("%c",S1[i-1]);
    }
    else if(b[i][j]=='u')
        print(i-1,j);
    else
        print(i,j-1);
}


void lcs(char *S1,char *S2,int m,int n)
{

    for(int i=0;i<=m;i++){
        c[i][0]=0;
        for(int j=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(S1[i-1]==S2[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';
                    }
```

```
                }
            }
        }
    }

    print(m,n);

}


int lcs_length(char* x, char* y,int m,int n)
{
    if (m == 0 || n == 0)
        return 0;
    if (x[m - 1] == y[n - 1])
        return 1 + lcs_length(x, y, m - 1, n - 1);
    else
        return max(lcs_length(x, y, m, n - 1), lcs_length(x, y, m - 1, n));
}


int main()
{

    printf("\nEnter 1st sequence: ");
    scanf("%s",S1);
    printf("Enter 2nd sequence: ");
    scanf("%s",S2);

    int m=strlen(S1);
    int n=strlen(S2);

    printf("\nLength of LCS is %d", lcs_length(S1, S2, m, n));
    printf("\nThe Longest Common Subsequence is: ");
    lcs(S1,S2,m,n);

    return 0;
}
```

Output:

```
  }

Enter 1st sequence: janhavi
Enter 2nd sequence: ruthvi

Length of LCS is 3
The Longest Common Subsequence is: hvi
PS C:\Users\janha\OneDrive\Documents\C programs\arrays>
```

Conclusion:

I understood how to find longest common subsequence of two given strings using dynamic programming approach. Also understood how dynamic programming approach helps to reduce the time complexity to O(m*n) where m and n are length of the strings in comparison to the recursive approach where the time complexity is $O(2^n)$.