DESIGN AND ANALYSIS OF ALGORITHMS

EXPERIMENT 4

ANUSHKA ACHARYA

CSE DS D1 BATCH

UID: 2021700001

AIM:
TO FIND LONGEST COMMON SUBSEQUENCE USING DYNAMIC PROGRAMMING

THEORY:

Given two strings, S1 and S2, the task is to find the length of the longest subsequence present in both of the strings.
Note: A subsequence of a string is a sequence that is generated by deleting some characters (possibly 0) from the string without altering the order of the remaining characters. For example, “abc”, “abg”, “bdf”, “aeg”, ‘”acefg”, etc are subsequences of the string “abcdefg”.

Let the input sequences be X[0 . . . m-1] and Y[0 . . . n-1] of lengths m and n respectively. And let L(X[0 . . . m-1], Y[0 . . . n-1]) be the length of the LCS of the two strings X and Y.
Following is the recursive definition of L(X[0 . . . m-1], Y[0 . . . n-1]).

If the last characters of both sequences match (or X[m-1] = Y[n-1]) then
L(X[0 . . . m-1], Y[0 . . . n-1]) = 1 + L(X[0 . . . m-2], Y[0 . . . n-2])

If last characters of both sequences do not match then
L(X[0 . . . m-1], Y[0 . . . n-1]) = MAX (L(X[0 . . . m-2], Y[0 . . . n-1]), L(X[0 . . . m-1], Y[0 . . . n-2]))

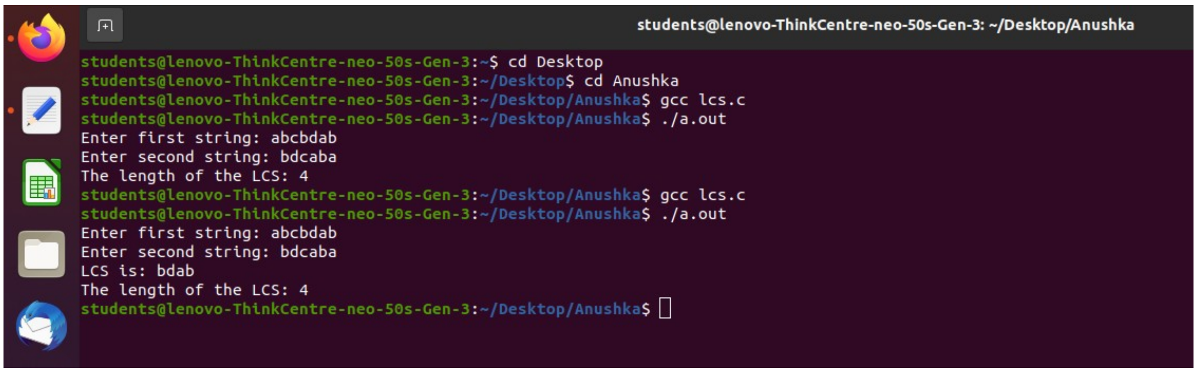
We can use the following steps to implement the dynamic programming approach for LCS.
Create a 2D array dp[][] with rows and columns equal to the length of each input string plus 1 [the number of rows indicates the indices of S1 and the columns indicate the indices of S2].
Initialize the first row and column of the dp array to 0.
Iterate through the rows of the dp array, starting from 1 (say using iterator i).
For each i, iterate all the columns from j = 1 to n:
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).
Else, set the current element of the dp array to the maximum value of dp[i-1][j] and dp[i][j-1].
After the nested loops, the last element of the dp array will contain the length of the LCS.

CODE:

```
#include <stdio.h>
#include <string.h>
#define MAX_LENGTH 100
void lcs(char *s1, char *s2, int m, int n, int LCS[][MAX_LENGTH], char
*result)
{
int i, j, index;
for (i = 0; i <= m; i++)
{
for (j = 0; j <= n; j++)
{
if (i == 0 || j == 0)
{
LCS[i][j] = 0;
}
}
}
for (i = 1; i <= m; i++)
{
for (j = 1; j <= n; j++)
{
if (s1[i - 1] == s2[j - 1])
{
LCS[i][j] = LCS[i - 1][j - 1] + 1;
}
else
{
LCS[i][j] = (LCS[i - 1][j] > LCS[i][j - 1]) ? LCS[i - 1][j] :
LCS[i][j - 1];
}
}
}
index = LCS[m][n];
result[index] = '\0';
i = m;
j = n;
while (i > 0 && j > 0)
{
if (s1[i - 1] == s2[j - 1])
{
result[index - 1] = s1[i - 1];
i--;
j--;
index--;
}
else if (LCS[i - 1][j] > LCS[i][j - 1])
{
```

```
i--;
}
else
{
j--;
}
}
printf("LCS is: %s\n",result);
}
int main()
{
char s1[MAX_LENGTH], s2[MAX_LENGTH], result[MAX_LENGTH];
int LCS[MAX_LENGTH][MAX_LENGTH];
int m, n;
printf("Enter first string: ");
scanf("%s", s1);
printf("Enter second string: ");
scanf("%s", s2);
m = strlen(s1);
n = strlen(s2);
lcs(s1, s2, m, n, LCS, result);
printf("The length of the LCS: %d\n", LCS[m][n]);
}
```

OUTPUT:



```
students@lenovo-ThinkCentre-neo-50s-Gen-3: ~/Desktop/Anushka
students@lenovo-ThinkCentre-neo-50s-Gen-3:~$ cd Desktop
students@lenovo-ThinkCentre-neo-50s-Gen-3:~/Desktop$ cd Anushka
students@lenovo-ThinkCentre-neo-50s-Gen-3:~/Desktop/Anushka$ gcc lcs.c
students@lenovo-ThinkCentre-neo-50s-Gen-3:~/Desktop/Anushka$ ./a.out
Enter first string: abcbdab
Enter second string: bdcaba
The length of the LCS: 4
students@lenovo-ThinkCentre-neo-50s-Gen-3:~/Desktop/Anushka$ gcc lcs.c
students@lenovo-ThinkCentre-neo-50s-Gen-3:~/Desktop/Anushka$ ./a.out
Enter first string: abcbdab
Enter second string: bdcaba
LCS is: bdab
The length of the LCS: 4
students@lenovo-ThinkCentre-neo-50s-Gen-3:~/Desktop/Anushka$ 
```

CONCLUSION:
From this experiment i learnt how to find the longest common subsequence of two sequences using dynamic programming.