DESIGN AND ANALYSIS OF ALGORITHMS LAB

NAME: Mohammed Saad Belgi

UID: 2021700005

BATCH: A

BRANCH: CSE DS

EXPT. NO.: 4

AIM: Experiment using dynamic programming approach: finding longest common subsequence of two strings

ALGORITHM:

```
LCS-LENGTH(X, Y)
    1. m = X.length
    2. n = Y.length
    3. let c[0 ..m, 0 .. n] and b[1...m, 1...n] be new tables
    4. for i = 1 to m
    5.    c[i, 0] = 0
    6. for j = 0 to n
    7.    c[0, j] = 0
    8. for i = 1 to m
    9.    for j = 1 to n
    10.     if x[i] == y[j]
    11.       c[i, j] = c[i — 1, j — 1] + 1
    12.       b[i, j] = 0
    13.     elseif c [i — 1, j] > c[i, j — 1]
    14.       c[i, j] = c[i — 1, j]
    15.       b[i, j] = 1
    16.     else c[i, j] = c[i, j — 1]
    17.       b[i, j] = 2
    18. return c and b

PRINT-LCS(b, X, i, j)
    1. if i == 0 or j== 0
    2.  return
    3. if b[i, j]== 0
    4.  PRINT-LCS(b, X, i —1, j — 1)
    5.  print x[i]
    6. elseif b[i, j] == 1
    7.  PRINT-LCS(b, X, i -1, j)
    8. else PRINT-LCS(b, X, i, j - 1)
```

CODE:

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

void longestCommonSubsequence(char *str1, char *str2, char
*lcs, int *lcs_len)
{
    *lcs_len = 0;
    int m = strlen(str1);
    int n = strlen(str2);
    int c[m + 1][n + 1];
    // initialising first row to 0
    for (int i = 0; i < n + 1; i++)
        c[0][i] = 0;
    // initialising first column to 0
    for (int i = 0; i < m + 1; i++)
        c[i][0] = 0;
    for (int i = 1; i < m + 1; i++)
    {
        for (int j = 1; j < n + 1; j++)
        {
            if (str1[i - 1] == str2[j - 1])
                c[i][j] = c[i - 1][j - 1] + 1;
            else
            {
                if (c[i - 1][j] > c[i][j - 1])
                    c[i][j] = c[i - 1][j];
                else
                    c[i][j] = c[i][j - 1];
            }
        }
    }
    printf("TABLE:\n");
    printf("0\t0\t");
    for (int i = 0; i < n; i++)
        printf("%c\t", str2[i]);
    printf("\n");
    for (int i = 0; i < m + 1; i++)
    {
        if (i != 0)
            printf("%c\t", str1[i - 1]);
```

```
        else
            printf("0\t");
        for (int j = 0; j < n + 1; j++)
            printf("%d\t", c[i][j]);
        printf("\n");
    }
    *lcs_len = c[m][n];
    lcs[(*lcs_len)] = '\0';
    int u = m, v = n;
    int idx = (*lcs_len) - 1;
    while (idx >= 0)
    {
        if (str1[u - 1] == str2[v - 1])
        {
            lcs[idx--] = str1[u - 1];
            u--;
            v--;
        }
        else if (c[u][v] == c[u][v - 1])
            v--;
        else
            u--;
    }
}

int main()
{
    char a[100], b[100];
    printf("Enter first string: ");
    fgets(a, sizeof(a), stdin);
    int a_size = strlen(a);
    a[--a_size] = '\0';
    printf("Enter second string: ");
    fgets(b, sizeof(b), stdin);
    int b_size = strlen(b);
    b[--b_size] = '\0';
    char lcs[100];
    int lcs_len = 0;
    longestCommonSubsequence(a, b, lcs, &lcs_len);
    printf("Length of longest common subsequence: %d\n",
lcs_len);
    printf("Longest common subsequence: %s\n", lcs);
```

```
}
```

OUTPUT:

```
Enter first string: saadbelgi
Enter second string: advaitsapkal
TABLE:
0       0       a       d       v       a       i       t       s       a       p       k       a       l
0       0       0       0       0       0       0       0       0       0       0       0       0       0
s       0       0       0       0       0       0       0       1       1       1       1       1       1
a       0       1       1       1       1       1       1       1       2       2       2       2       2
a       0       1       1       1       2       2       2       2       2       2       2       3       3
d       0       1       2       2       2       2       2       2       2       2       2       3       3
b       0       1       2       2       2       2       2       2       2       2       2       3       3
e       0       1       2       2       2       2       2       2       2       2       2       3       3
l       0       1       2       2       2       2       2       2       2       2       2       3       4
g       0       1       2       2       2       2       2       2       2       2       2       3       4
i       0       1       2       2       2       3       3       3       3       3       3       3       4
Length of longest common subsequence: 4
Longest common subsequence: saal
(base) PS C:\Users\arifa\Desktop\sem4 work\daa lab\exp4>
```

CONCLUSION:

Longest common subsequence problem has optimal substructure property as well as overlapping subproblems property. Hence, it can be solved in O(mn) time using dynamic programming instead of exponential time required by simple divide and conquer approach.