Sardar Patel Institute of Technology

Name	Bhagya Bijlaney
UID	2021700010
Class	CSE Data Science
Batch	D1
Exp no.	4

Aim – To find the longest common subsequence of 2 strings by using the dynamic programming approach.

Algorithm & Pseudocode –

<u>Algorithm</u>

1. Define MAX_LENGTH constant to specify the maximum length of strings.

2. Define a function lcs that takes in two strings s1 and s2, their lengths m and n, a 2D array LCS and a character array result.

3. Initialize the first row and column of LCS with 0.

4. Loop through the remaining cells of LCS, filling in each cell using the dynamic programming approach:

   a. If s1[i-1] == s2[j-1], set LCS[i][j] = LCS[i-1][j-1] + 1.

   b. Otherwise, set LCS[i][j] = max(LCS[i-1][j], LCS[i][j-1]).

5. Set index to the value in the bottom-right cell of LCS and add a null terminator to the end of the result array.

6. Loop through LCS, tracing back through the cells used to compute the longest common subsequence:

   a. If s1[i-1] == s2[j-1], add s1[i-1] to the result array at index-1 and decrement index, i, and j by 1.

   b. Otherwise, if LCS[i-1][j] > LCS[i][j-1], decrement i by 1.

   c. Otherwise, decrement j by 1.

7. In main function:

   a. Declare character arrays s1, s2, and result, and a 2D integer array LCS with size MAX_LENGTH x MAX_LENGTH.

   b. Prompt the user to enter the two strings.

c. Call the lcs function with s1, s2, their lengths m and n, LCS, and result.

d. Print the length of the longest common subsequence and the longest common subsequence.

End of the program.

Pseudocode

```
1. Constant MAX_LENGTH □ 100
2. Function lcs(s1, s2, m, n, LCS, result)
   a. for i = 0 to m
          for j = 0 to n
              if i == 0 or j == 0
                  LCS[i][j] □ 0

   b. for i = 1 to m
          for j = 1 to n
              if s1[i-1] == s2[j-1]
                  LCS[i][j] □ LCS[i-1][j-1] + 1
              else
                  LCS[i][j] □ max(LCS[i-1][j], LCS[i][j-1])

   c. index □ LCS[m][n] and result[index] □ '\0'
   d. i □ m, j □ n
   e. while i > 0 and j > 0
          if s1[i-1] == s2[j-1]
              result[index-1] □ s1[i-1]
              index--, i-- , j--
```

```
    else if LCS[i-1][j] > LCS[i][j-1]
            i--
    else
            j--
```

3. In main function

```
    a. declare s1[MAX_LENGTH], s2[MAX_LENGTH],
    result[MAX_LENGTH], and
    LCS[MAX_LENGTH][MAX_LENGTH]
    b. prompt the user to enter s1 and s2
    c. set m ← length of s1 and n ← length of s2
    d. call lcs function with s1, s2, m, n, LCS, and result
    e. Print the length of LCS
    f. Print the LCS

4. End of the program
```

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;


// Initialize first row and column with 0
   for (i = 0; i <= m; i++)

   {

for (j = 0; j <= n; j++)

       {

if (i == 0 || j == 0)

           {

LCS[i][j] = 0;

}

}

}


// Fill the remaining cells using dynamic programming
   for (i = 1; i <= m; i++)

   {
```

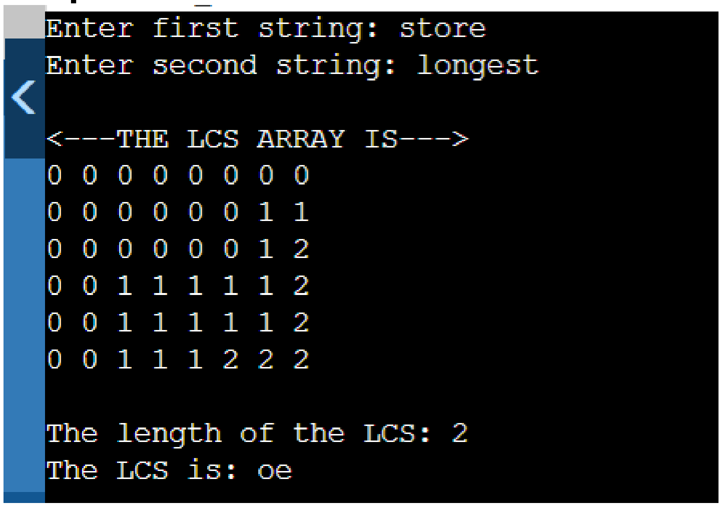
```
for (j = 1; j <= n; j++)

        {

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];

}

}

}


printf ("\n<---THE LCS ARRAY IS--->\n");


for (i = 0; i <= m; i++)

   {

for (j = 0; j <= n; j++)

        {

printf ("%d ", LCS[i][j]);
```

```
}


printf ("\n");

}

printf ("\n");


// Find the longest common subsequence
    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--;

}

}

}



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]);

printf("The LCS is: %s\n", result);

return 0;
}
```

Output –



```
Enter first string: store
Enter second string: longest

<---THE LCS ARRAY IS--->
0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1
0 0 0 0 0 0 1 2
0 0 1 1 1 1 1 2
0 0 1 1 1 1 1 2
0 0 1 1 1 2 2 2

The length of the LCS: 2
The LCS is: oe
```

Obseervations:

The time complexity of the code is O(mn) where m and n are the lengths of the two input strings. This is because we need to fill in an m x n matrix to compute the length of the longest common subsequence.

The space complexity of the code is O(mn) since we use a 2D array of size m x n to store the length of the longest common subsequence.

Conclusion:

In this experiment, we used the dynamic programming approach to find the longest common subsequence of two strings.

We first learned about the algorithm and the two key equations used in the dynamic programming approach. We then implemented the algorithm in C language, and tested it on different strings to find their longest common subsequences.