Bharatiya Vidya Bhavan's

Sardar Patel Institute of Technology

Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai-400058-India
(Autonomous College Affiliated to University of Mumbai)

SE – COMP (SE-A/08) **Sub- DAA Lab**

Name	**Naman Badlani**
UID No.	**2021300008**
Subject	**Design And Analysis Of Algorithm**
Class	**Comps A**
Experiment No.	**6**
AIM	To implement Longest Common Subsequence

Theory –

It involves finding the longest subsequence that is common to two given sequences.

There are several approaches to solving the LCS problem, but one of the most popular is dynamic programming. In this approach, a matrix is constructed to represent the LCS between two sequences. The matrix is filled in iteratively, and each cell in the matrix represents the LCS of a prefix of one sequence and a prefix of the other sequence.

At each step of the iteration, the LCS matrix is updated based on the current characters in the two sequences being compared. If the characters are the same, the LCS at that point is the LCS of the two sequences up to the previous character plus the current character. If the characters are different, the LCS at that point is the longer of the LCS of the first sequence up to the previous character and the LCS of the second sequence up to the previous character.

Once the matrix is fully filled in, the length of the LCS can be found in the bottom right cell of the matrix. The actual LCS can be reconstructed by tracing back through the matrix from the bottom right cell to the top left cell, following the path of arrows that were used to fill in the matrix.

Algorithm –

1. Initialize an (n+1) x (m+1) matrix, called the LCS matrix, with zeros.

2. Loop through each row i of the matrix, from 1 to n+1, and each column j of the matrix, from 1 to m+1.

3. If i or j is 0, set the value of the corresponding cell in the matrix to 0.

4. If the ith character of s1 is the same as the jth character of s2, set the value of the (i,j) cell in the matrix to the value of the (i-1,j-1) cell in the matrix plus 1.

5. If the ith character of s1 is different from the jth character of s2, set the value of the (i,j) cell in the matrix to the maximum of the value of the (i-1,j) cell and the value of the (i,j-1) cell.

6. Once the matrix is fully filled in, the length of the LCS is the value of the bottom-right cell of the matrix.

7. To reconstruct the actual LCS, start at the bottom-right cell of the matrix and follow the path of arrows back to the top-left cell. Whenever there is a diagonal arrow, add the corresponding character to the LCS.

8. Return the LCS.

Program –

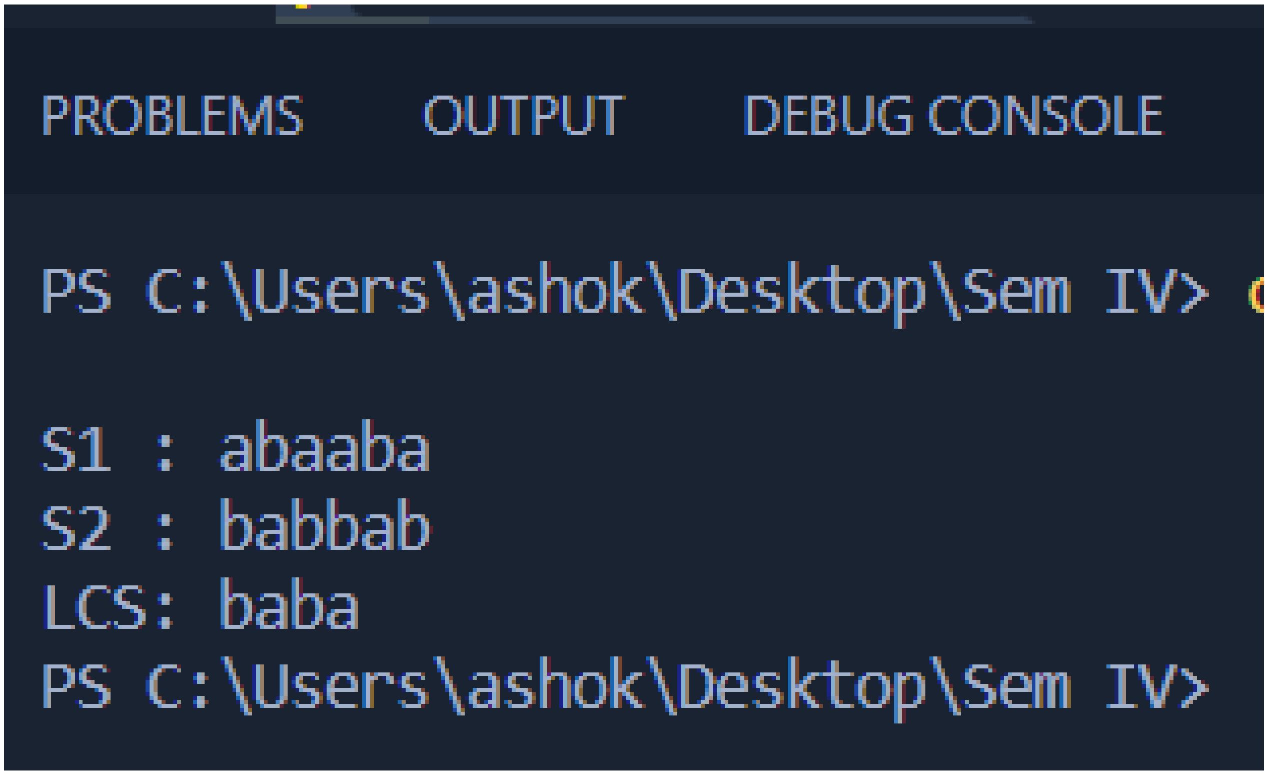
```
#include <stdio.h>
#include <string.h>
int i, j, m, n, LCS_table[20][20];
char S1[20] = "abaaba", S2[20] = "babbab", b[20][20];
void lcsAlgo()
{
    m = strlen(S1);
    n = strlen(S2);
    for (i = 0; i <= m; i++)
        LCS_table[i][0] = 0;
    for (i = 0; i <= n; i++)
        LCS_table[0][i] = 0;
    for (i = 1; i <= m; i++)
```

```c
        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--;
    }
    printf("S1 : %s \nS2 : %s \n", S1, S2);
    printf("LCS: %s", lcsAlgo);
}

int main()
{
    lcsAlgo();
    printf("\n");
```

```
}
```

Result Analysis –



```
PS C:\Users\ashok\Desktop\Sem IV>

S1  : abaaba
S2  : babbab
LCS: baba
PS C:\Users\ashok\Desktop\Sem IV>
```

- The time complexity of the dynamic programming approach to solving the LCS problem is O(mn), where m and n are the lengths of the two sequences being compared. This makes it an efficient approach for finding the LCS of two sequences.