

EXPERIMENT NO- 07

AIM: Program to implement Longest Common Subsequence algorithm.

PROBLEM STATEMENT : Write a program to find Longest Common Subsequence from the given two sequences.

RESOURCE REQUIRED: Pentium IV, Turbo C, Printer, Printout Stationary

THEORY:

Theory: A Longest Common Subsequence is a common subsequence of maximal length. In the longest common subsequence problem, we are given two sequences $X=$ and $Y=$ and wish to find a maximum-length common subsequence of x and y . The optimal substructure of the LCS problem gives the recursive formula,

Computing the length of an LCS: Procedure LCS-LENGTH takes two sequences $X=$ and $Y=$ as inputs. It stores the $c[i,j]$ values in a table $c[0...m,0...n]$ whose entries are computed in row-major order i.e. the first row of c is filled in from left to right, then second row and so on. It also maintains the table $b[1...m, 1...n]$ to simplify construction of an optimal solution. Initially, $b[i,j]$ points to the table entry corresponding to the optimal sub problem solution. The procedure returns the b and c tables; $c[m,n]$ contains the length of an LCS of X and Y .

ALGORITHM:

LCS - Length (X, Y)

$m \leftarrow \text{length} (X)$

$n \leftarrow \text{length} (Y)$

for ($i \leftarrow 1$ to m) do

$c [i, 0] \leftarrow 0$

for ($j \leftarrow 1$ to n) do

$c [0, j] \leftarrow 0$

for ($i \leftarrow 1$ to m) do

{

 for ($j \leftarrow 1$ to n) do

 {

 if ($x_i = y_j$) then

 {

$c [i, j] \leftarrow c [i-1, j-1] + 1$

$b [i, j] \leftarrow \nwarrow$

 }

 }

```

    else if ( c [ i - 1 , j ] >= c [ i , j - 1 ] ) then
    {
        c [ i , j ] ← c [ i - 1 , j ]
        b [ i , j ] ← " ↑ "
    }
    else
    {
        c [ i , j ] ← c [ i , j - 1 ]
        b [ i , j ] ← " ← "
    }
} // end of inner for loop

} // end of outer for loop
return c and b

```

Constructing an LCS :

The b table returned by LCS_LENGTH can be used to quickly construct an LCS of $X = \langle x_1, x_2, \dots, x_m \rangle$ and $Y = \langle y_1, y_2, \dots, y_n \rangle$. We begin at $b[m, n]$ and trace through the table following the arrows. Whenever we encounter a "↖" in entry $b[i, j]$ it implies that $x_i = y_j$ is an element of the LCS. The elements of the LCS are encountered in reverse order by this method.

PRINT-LCS (b, x, i, j)

```

if i = 0 or j = 0
then return

if b [ i , j ] = " ↖ " then
{
    PRINT - LCS ( b , x , i - 1 , j - 1 )
    print xi
}

else if b [ i , j ] = " ↑ " then
    PRINT - LCS ( b , x , i - 1 , j )

else
    PRINT - LCS ( b , x , i , j - 1 )

```

CODE:

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

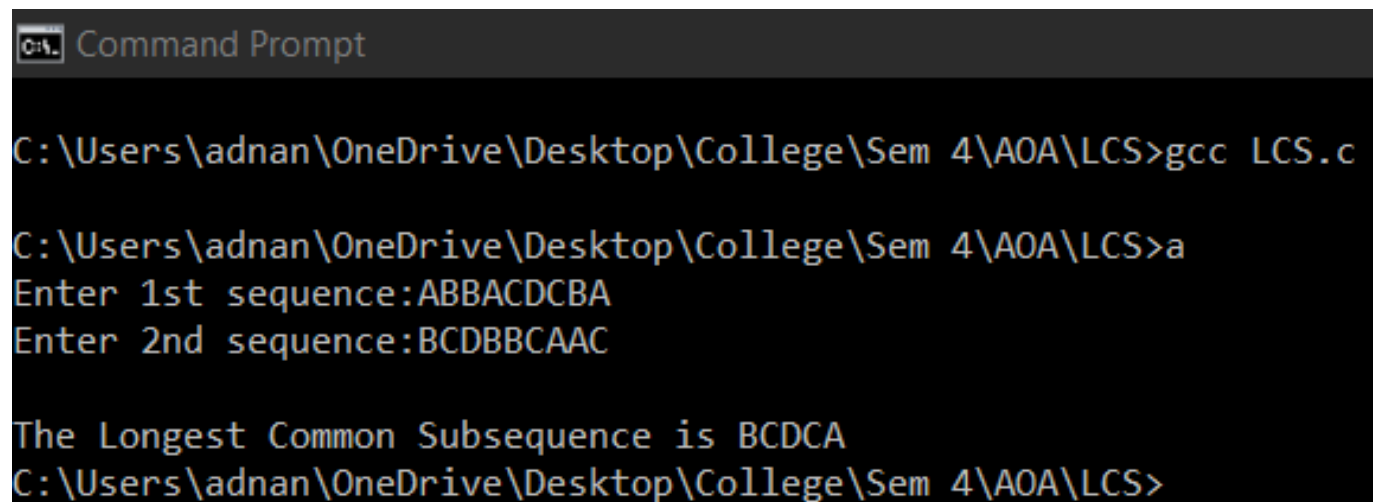
int i,j,m,n,c[20][20];
char x[20],y[20],b[20][20];

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

void lcs()
{
    m=strlen(x);
    n=strlen(y);
    for(i=0;i<=m;i++)
        c[i][0]=0;
    for(i=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(x[i-1]==y[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';
            }
        }
}
```

```
}  
  
int main()  
{  
    printf("Enter 1st sequence:");  
    scanf("%s",x);  
    printf("Enter 2nd sequence:");  
    scanf("%s",y);  
    printf("\nThe Longest Common Subsequence is ");  
    lcs();  
    print(m,n);  
    return 0;  
}
```

OUTPUT:

```
C:\Users\adnan\OneDrive\Desktop\College\Sem 4\AOA\LCS>gcc LCS.c  
  
C:\Users\adnan\OneDrive\Desktop\College\Sem 4\AOA\LCS>a  
Enter 1st sequence:ABBACDCBA  
Enter 2nd sequence:BCDDBCAAC  
  
The Longest Common Subsequence is BCDCA  
C:\Users\adnan\OneDrive\Desktop\College\Sem 4\AOA\LCS>
```

CONCLUSION: In compilation of program, in software design or in system design text processing is a vital activity. While processing the text, string matching is an important activity which is needed most of the time. Least common subsequences is one of the pattern matching algorithm. This algorithm is more efficient than other algorithm and hence implemented successfully.