K. J. Somaiya College of Engineering

(A Constituent College of Somaiya Vidyavihar University)

Batch: A1 Roll No.: 16010120015

Experiment No. 11

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of the Staff In-charge with date

Title: Implementation of Longest Common Subsequence String Matching Algorithm

Objective: To compute longest common subsequence for the given two strings.

CO to be achieved:

CO 2 Analyze and solve problems for divide and conquer strategy, greedy method, dynamic programming approach and backtracking and branch & bound policies.

CO 3 Analyze and solve problems for different string matching algorithms.

Books/ Journals/ Websites referred:

1. **Ellis horowitz, Sarataj Sahni, S.Rajsekaran," Fundamentals of computer algorithm", University Press**
2. **T.H.Cormen ,C.E.Leiserson,R.L.Rivest and C.Stein," Introduction to algortihtms",2nd Edition ,MIT press/McGraw Hill,2001**
3. **http://www.math.utah.edu/~alfeld/queens/queens.**

Pre Lab/ Prior Concepts:

Data structures, Concepts of algorithm analysis

Historical Profile:

Given 2 sequences, $X = x1, ..., xm$ and $Y = y1, ..., yn$, find a subsequence common to both whose length is longest. A subsequence doesn't have to be consecutive, but it has to be in order.

New Concepts to be learned:

String matching algorithm, Dynamic programming approach for LCS, Applications of LCS.

Recursive **Formulation:**

Define $c[i, j]$ = length of LCS of Xi and $Y j$.
Final answer will be computed with $c[m, n]$.

$c[i, j] = 0$ if i=0 or j=0.

$c[i, j] = c[i - 1, j - 1] + 1$ if i,j>0 and xi=yj

$c[i, j] = \max(c[i - 1, j], c[i, j - 1])$ if i, j > 0 and xi <> y j

Algorithm: Longest Common Subsequence

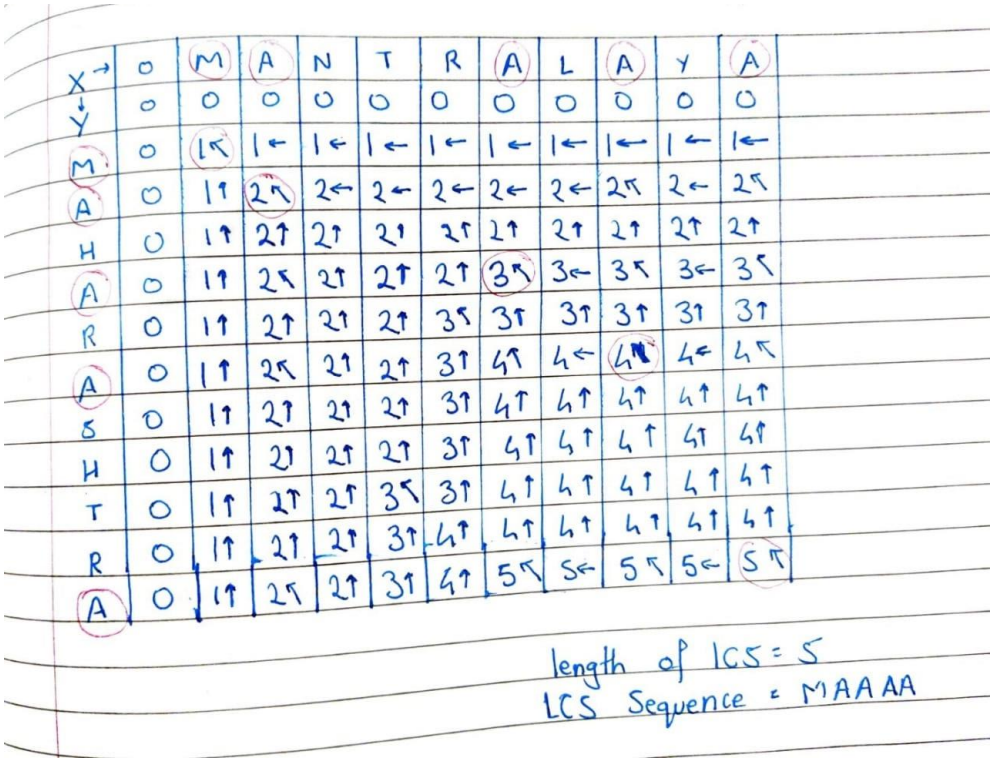
Compute length of optimal solution-

LCS-LENGTH *(X , Y, m, n)*
for $i \leftarrow 1$ **to** m
do $c[i, 0] \leftarrow 0$
for $j \leftarrow 0$ **to** n
do $c[0, j] \leftarrow 0$
for $i \leftarrow 1$ **to** m
do for $j \leftarrow 1$ **to** n
do if $xi = y j$
then $c[i, j] \leftarrow c[i - 1, j - 1] + 1$
$b[i, j] \leftarrow$ "≈"
else if $c[i - 1, j] \geq c[i, j - 1]$
then $c[i, j] \leftarrow c[i - 1, j]$
$b[i, j] \leftarrow$ "↑"
else $c[i, j] \leftarrow c[i, j - 1]$
$b[i, j] \leftarrow$ "←"
return *c* and *b*

Print the solution-
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*
elseif $b[i, j] =$ "↑"
then PRINT-LCS*(b, X ,* $i - 1, j$ *)*
else PRINT-LCS*(b, X , i,* $j - 1$*)*

- Initial call is PRINT-LCS*(b, X , m, n).*
- *b*[*i, j*] points to table entry whose subproblem we used in solving LCS of *Xi* and *Y j* .
- When *b*[*i, j*] = ≈, we have extended LCS by one character. So longest com− mon subsequence = entries with ≈ in them.

Example: LCS computation



Analysis of LCS computation

M= length of string x

N= length of string y

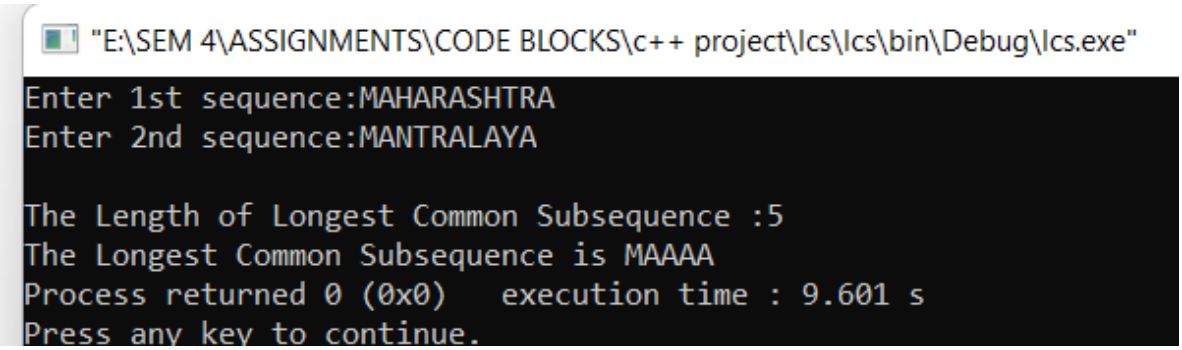
```
#include<stdio.h>

#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;
        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 Length of Longest Common Subsequence :");
```

```
        printf("\nThe Longest Common Subsequence is ");
        lcs();
        print(m,n);
return 0;
}
```

OUTPUT:



0(n*m)

Complexity:

Best case , worst case Time complexity: 0(n*m)

Space complexity: 0(n*m)

Algorithm:

```
LCS-LENGTH ( X , Y, m, n)
for i ← 1 to m
    do c[i, 0] ← 0
for j ← 0 to n
    do c[0, j ] ← 0
for i ← 1 to m
    do for j ← 1 to n
                do if xi  = y j
                 then c[i, j ] ← c[i − 1, j − 1] + 1
                            b[i, j ] ← "≈"
                   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 ] ← "←"
return c and b
```

Conclusion:

The longest common subsequence (LCS) is the problem of finding the longest subsequence common to all sequences in a set of 2 strings.

In this experiment we have explored and implemented the concept of finding the longest common subsequence of two strings with the help of dynamic programming in c.