Batch: B2 Roll No.: 16010121110

Experiment No. ____

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.Rajasekaran," Fundamentals of computer algorithm", University Press**
2. **T.H.Cormen ,C.E.Leiserson,R.L.Rivest and C.Stein," Introduction to algorithms",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 Yj.
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 $x_i <> y_j$

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

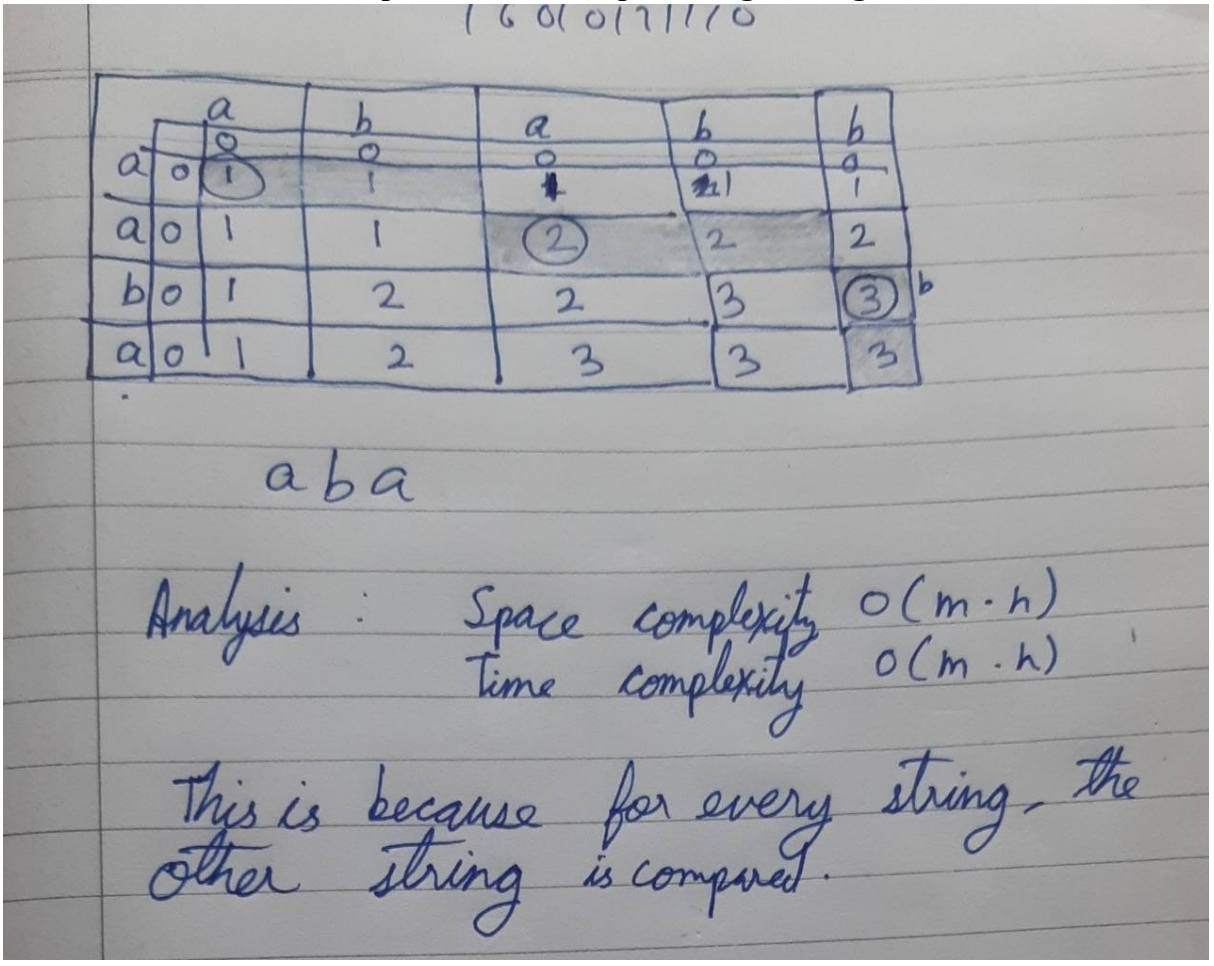
Print the solution-
PRINT-LCS*(b, X , i, j)*

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



Code

```
import java.util.*;
public class Main {
```

```
public static void main(String args[]){
   char [] str1= {'d','b','c','d','e','d'};
   char [] str2 = {'d','b','c','e','d'};
   ArrayList<Character> output= new ArrayList();
   int arr[][]=new int[str1.length+1][str2.length+1];
   for(int i=0;i<str1.length;i++){
      for(int j=0;j<str2.length;j++){
         if(str1[i]==str2[j]){
            arr[i+1][j+1]=arr[i][j]+1;
         }
         else{
            if(arr[i][j+1]<=arr[i+1][j])
            {arr[i+1][j+1]=arr[i+1][j];}

         if(arr[i][j+1]>arr[i+1][j])
            {arr[i+1][j+1]=arr[i][j+1];}
         }
      }
   }
   int p=str1.length-1;
   int q=str2.length-1;

   while(p>=0&q>=0){

      //   System.out.println(str1[p]+","+str2[q]+","+arr[p+1][q+1]+","+p+","+q);
         if(str1[p]==str2[q]){
         //  System.out.println(str1[p]);
           output.add(str1[p]);
           p--;
           q--;
         //  System.out.print(","+p+","+q+"\n");
         }
         else{
            if(arr[p][q-1]<=arr[p-1][q])
            {
             //  System.out.println(arr[p+1][q]);
                p--;
            }

         else if(arr[p][q-1]>arr[p-1][q])
            {
              //  System.out.println(arr[p][q+1]);
               q--;
            }
         }


   }
   for(int i=0;i<arr.length;i++){
```

```
        for(int j=0;j<arr[0].length;j++){
            System.out.print(arr[i][j]);
        }
        System.out.println();
    }
    System.out.print("common string is " );
     for(int j=1;j<=output.size();j++){
            System.out.print(output.get(output.size()-j));
        }
  }

}
```

Output:

```
000000
011111
012222
012333
012334
012344
012345
common string is dbced
```

CONCLUSION:

Thus we have implemented the longest common sub sequence using dynamic programming. This problem occurs in computational linguistics and bioinformatics. We have implemented the problem using the table method.