Sardar Patel Institute of Technology

SEM IV: DESIGN AND ANALYSIS OF ALGORITHMS.

Name	SMRUTI SONEKAR
UID no.	2021700064
BRANCH:	SE CSE (DATA SCIENCE)
Experiment No.	4

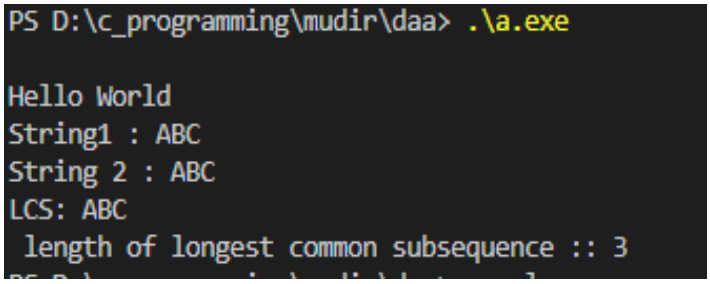
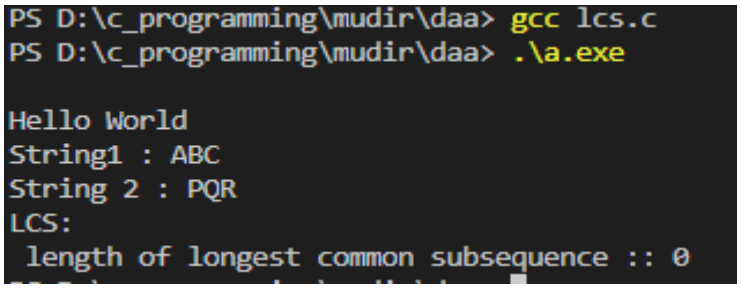
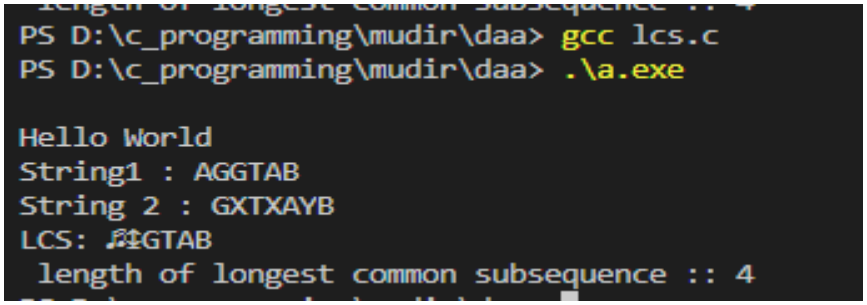
TOPIC:	Dynamic Programming: Longest Common Subsequence.
QUERY:	• Dynamic programming: The method of dynamic programming reduces the number of function calls. It stores the result of each function call so that it can be used in future calls without the need for redundant calls. • LCS: The longest subsequence that is common to all the given sequences, provided that the elements of the subsequence are not required to occupy consecutive positions within the original sequences. Algorithm of LCS: • Create a 2D array dp[][] with rows and columns equal to the length of each input string plus 1 [the number of rows indicates the indices of S1 and the columns indicate the indices of S2]. • Initialize the first row and column of the dp array to 0. • Iterate through the rows of the dp array, starting from 1 (say using iterator i). • For each i, iterate all the columns from j = 1 to n: • If S1[i-1] is equal to S2[j-1], set the current element of the dp array to the value of the element to (dp[i-1][j-1] + 1). • Else, set the current element of the dp array to the maximum value of dp[i-1][j] and dp[i][j-1].

	• After the nested loops, the last element of the dp array will contain the length of the LCS. • Time complexity of LCS: In the worst case we will be solving every subproblem only once and we have (m+1)*(n+1) subproblems and to store this sub problem we create table of (m+1)*(n+1) . • space complexity is O(m*n) and time complexity is O(m*n).
PROGRAM:	

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

int lcs(char X[], char Y[], int m, int n);
int max(int a, int b);

int lcs(char X[], char Y[], int m, int n)
{
    if(m==0 || n==0)
    {
        return 0;
    }

    if(X[m-1]== Y[n-1])
    {
        //printf("%c ",X[m-1]);
        return 1+ lcs(X,Y,m-1,n-1);
    }
    else
    {
        //printf("%c",max(lcs(X,Y,m-1,n),lcs(X,Y,m,n-1)))
        return max(lcs(X,Y,m-1,n),lcs(X,Y,m,n-1));
    }


}

void lcsAlgo(char X[], char Y[], int m, int n)
{
```

```
 int table[20][20];
 int i,j;
 for(i=0;i<=m;i++)
 {
     table[i][0]=0;
 }
  for(i=0;i<=n;i++)
 {
     table[0][i]=0;
 }

 for(i=1;i<=m;i++)
 {
      for(int j=1;j<=n;j++)
      {
          if(X[m-1]==Y[n-1])
          {
              table[i][j]= table[i-1][j-1] + 1;
          }
          else if (table[i-1][j] >= table[i][j-1])
          {
              table[i][j]=table[i-1][j];
          }
          else
          {
              table[i][j]= table[i][j-1];
          }
      }
 }

 int num = table[m][n];
 char result[num + 1];
result[num]='\0';

 int p=m,q=n;
 while(p>0 && q>0)
 {
     if(X[p-1]==Y[q-1])
     {
       result[num-1] = X[p-1];
       //result[num-1] = Y[q-1];
       p--;
       q--;
      num--;
```

```
        }
        else if(table[p-1][q]> table[p][q-1])
        {
            p--;
        }
        else
        {
            q--;
        }

    }

    printf("String1 : %s \nString 2 : %s \n", X, Y);
     printf("LCS: %s", result);
}

int max(int a, int b)
{
    if(a>b)
    {
        return a;
    }
    else
    {
        return b;
    }
}



int main()
{
    printf("\nHello World\n");
    char S1[20] = "AGGTAB";
    char S2[20] = "GXTXAYB";
    int m = strlen(S1);
    int n = strlen(S2);

   lcsAlgo(S1,S2,m,n);
   printf("\n length of longest common subsequence ::
%d\n",lcs(S1,S2,m,n));

    return 0;
}
```

RESULT:	• When same string length is equal and string is equal:  • When same string length is equal and string is not equal:  • When string length is not equal : 
CONCLUSION:	Dynamic programming approach was implemented on longest common subsequence algorithm by C and executed successfully.