



# RBU

**RAMDEOBABA UNIVERSITY, NAGPUR**

Formerly Shri Ramdeobaba College of Engineering & Management (RCOEM) Est. 1984

**LEARN | INNOVATE | ACCOMPLISH**

**Name: Nachiket Jagdale**

**Section: A3**

**Batch: B1**

**Roll no. 02**

**Subject: Design and Analysis  
of Algorithms - Lab  
Practical No. 5**

**Aim:- Implement a dynamic algorithm for Longest Common Subsequence (LCS) to find the length and LCS for DNA sequences.**

## **Problem Statement :-**

(i) DNA sequences can be viewed as strings of A, C, G, and T characters, which represent nucleotides. Finding the similarities between two DNA sequences are an important computation performed in bioinformatics.

[Note that a subsequence might not include consecutive elements of the original sequence.]

**TASK 1:** Find the similarity between the given X and Y sequence.

X=AGCCCTAAGGGCTACCTAGCTT

Y= GACAGCCTACAAGCGTTAGCTTG

Output: Cost matrix with all costs and direction, final cost of LCS and the LCS. Length of LCS=16

**TASK-2:** Find the longest repeating subsequence (LRS). Consider it as a variation of the longest common subsequence (LCS) problem. Let the given string be S. You need to find the LRS within S. To use the LCS framework, you effectively compare S with itself. So, consider string1 = S and string2 = S.

Example:

AABCBDC

LRS= ABC or ABD

## **LeetCode Assesment:**

<https://leetcode.com/problems/longest-common-subsequence/description/>

## TASK 1:

### CODE :-

```
#include <stdio.h>
#include <string.h>
int lcsLength(char X[], char Y[])
{
    int m = strlen(X), n = strlen(Y);
    int L[100][100];
    for (int i = 0; i <= m; i++)
    {
        for (int j = 0; j <= n; j++)
        {
            if (i == 0 || j == 0)
                L[i][j] = 0;
            else if (X[i-1] == Y[j-1])
                L[i][j] = L[i-1][j-1] + 1;
            else
                L[i][j] = (L[i-1][j] > L[i][j-1]) ? L[i-1][j] : L[i][j-1];
        }
    }
    return L[m][n];
}
void printLCS(char X[], char Y[])
```

```
{  
int m = strlen(X), n = strlen(Y);  
int L[100][100];  
for (int i = 0; i <= m; i++)  
{  
    for (int j = 0; j <= n; j++)  
    {  
        if (i == 0 || j == 0)  
            L[i][j] = 0;  
        else if (X[i-1] == Y[j-1])  
            L[i][j] = L[i-1][j-1] + 1;  
        else  
            L[i][j] = (L[i-1][j] > L[i][j-1]) ? L[i-1][j] : L[i][j-1];  
    }  
}  
  
int index = L[m][n];  
char lcs[index + 1];  
lcs[index] = '\0';  
int i = m, j = n;  
while (i > 0 && j > 0)  
{  
    if (X[i-1] == Y[j-1])  
    {
```

```
lcs[index-1] = X[i-1];

i--;
j--;
index--;

}

else if (L[i-1][j] > L[i][j-1])

i--;
else

j--;
}

printf("LCS: %s\n", lcs);

}

int main()

{

char X[100], Y[100];

scanf("%s", X);

scanf("%s", Y);

printf("Length of LCS: %d\n", lcsLength(X, Y));

printLCS(X, Y);

return 0;

}
```

## Output:-

The screenshot shows a terminal window from the Programiz C Online Compiler. The code in main.c implements the Longest Common Subsequence (LCS) algorithm. It takes two strings as input and prints their LCS length and the sequence itself. The terminal output shows the input strings "GCCCTAAGGGCTACCTAGCTT" and "GACAGCTTACAAGCGTTAGCTT", the length of LCS (16), and the LCS sequence "GCCCTAAGCTTAGCTT". A message at the bottom indicates successful code execution.

```
Programiz C Online Compiler
main.c
1 #include <stdio.h>
2 #include <string.h>
3
4 #define MAX 100
5
6 int lcsLength(const char X[], const char Y[]) {
7     int m = strlen(X), n = strlen(Y);
8     int L[MAX + 1][MAX + 1];
9
10    for (int i = 0; i <= m; i++) {
11        for (int j = 0; j <= n; j++) {
12            if (i == 0 || j == 0)
13                L[i][j] = 0;
14            else if (X[i - 1] == Y[j - 1])
15                L[i][j] = L[i - 1][j - 1] + 1;
16            else
17                L[i][j] = (L[i - 1][j] > L[i][j - 1]) ? L[i - 1][j] : L[i][j - 1];
18        }
19    }
20
21    return L[m][n];
22 }
23
24 void printLCS(const char X[], const char Y[]) {
25     int m = strlen(X), n = strlen(Y);
26     int L[MAX + 1][MAX + 1];
27
28     for (int i = 0; i <= m; i++) {
29         for (int j = 0; j <= n; j++) {
30             if (i == 0 || j == 0)
31                 L[i][j] = 0;
32             else if (X[i - 1] == Y[j - 1])
33                 printf("%c", X[i - 1]);
34             else
35                 L[i][j] = (L[i - 1][j] > L[i][j - 1]) ? L[i - 1][j] : L[i][j - 1];
36         }
37     }
38
39     printf("\n");
40 }
41
42 int main() {
43     const char X[] = "GCCCTAAGGGCTACCTAGCTT";
44     const char Y[] = "GACAGCTTACAAGCGTTAGCTT";
45
46     int lcsLength = lcsLength(X, Y);
47     printf("Length of LCS: %d\n", lcsLength);
48
49     printLCS(X, Y);
50 }
```

Enter first string: GCCCTAAGGGCTACCTAGCTT  
Enter second string: GACAGCTTACAAGCGTTAGCTT  
Length of LCS: 16  
LCS: GCCCTAAGCTTAGCTT  
== Code Execution Successful ==

## TASK 2:

### CODE :-

```
#include <stdio.h>

#include <string.h>

int lrsLength(char S[])

{

int m = strlen(S);

int L[100][100];

for (int i = 0; i <= m; i++)

{

for (int j = 0; j <= m; j++)

{



}
```

```
if (i == 0 || j == 0)
{
    L[i][j] = 0;
}

else if (S[i-1] == S[j-1] && i != j)
{
    L[i][j] = L[i-1][j-1] + 1;

}
else
{
    L[i][j] = (L[i-1][j] > L[i][j-1]) ? L[i-1][j] : L[i][j-1];
}

return L[m][m];
}

void printLRS(char S[])
{
    int m = strlen(S);

    int L[100][100];

    char lrs[100];

    int index = 0;

    for (int i = 0; i <= m; i++)
    {
```

```

for (int j = 0; j <= m; j++)
{
    if (i == 0 || j == 0)
    {
        L[i][j] = 0;
    }
    else if (S[i-1] == S[j-1] && i != j)
    {
        L[i][j] = L[i-1][j-1] + 1;
    }
    else
    {
        L[i][j] = (L[i-1][j] > L[i][j-1]) ? L[i-1][j] : L[i][j-1];
    }
}
}

int i = m, j = m;
while (i > 0 && j > 0)
{
    if (S[i-1] == S[j-1] && i != j)
    {
        lrs[index++] = S[i-1];
        i--;
        j--;
    }
}

```

```
}

else if (L[i-1][j] > L[i][j-1])

{

i--;

}

else

{

j--;

}

}

for (int k = index - 1; k >= 0; k--)

{

printf("%c", lrs[k]);

}

printf("\n");

}

int main()

{

char S[100];

printf("Enter string: ");

scanf("%s", S);

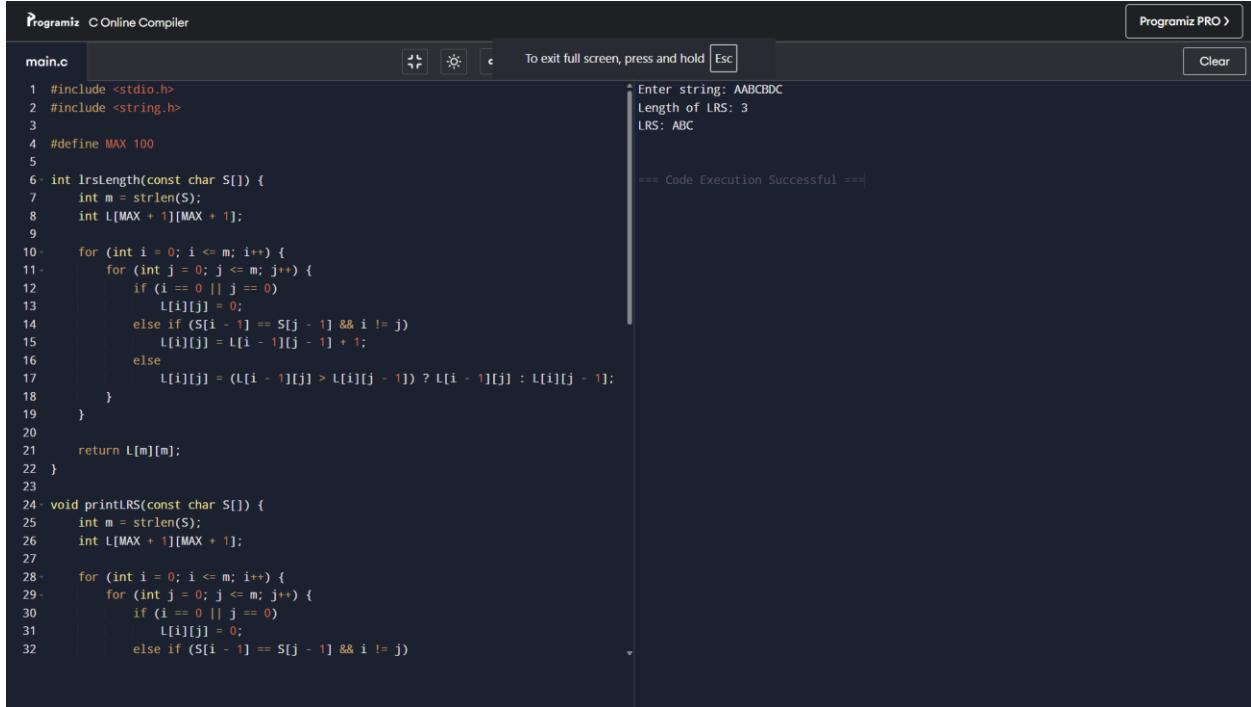
printf("Length of LRS: %d\n", lrsLength(S));

printf("LRS: ");

printLRS(S);
```

```
return 0;  
}  
  
}
```

## Output:-



The screenshot shows a C online compiler interface. The code in main.c is a solution for finding the longest common subsequence between two strings. It includes functions for calculating the length of the LRS and printing it. The input string 'ABCBCDC' is entered, and the output shows the length of 3 and the LRS 'ABC'. The code execution was successful.

```
Programiz C Online Compiler  
main.c  
1 #include <stdio.h>  
2 #include <string.h>  
3  
4 #define MAX 100  
5  
6 int lrsLength(const char S[]) {  
7     int m = strlen(S);  
8     int L[MAX + 1][MAX + 1];  
9  
10    for (int i = 0; i <= m; i++) {  
11        for (int j = 0; j <= m; j++) {  
12            if (i == 0 || j == 0)  
13                L[i][j] = 0;  
14            else if (S[i - 1] == S[j - 1] && i != j)  
15                L[i][j] = L[i - 1][j - 1] + 1;  
16            else  
17                L[i][j] = (L[i - 1][j] > L[i][j - 1]) ? L[i - 1][j] : L[i][j - 1];  
18        }  
19    }  
20  
21    return L[m][m];  
22 }  
23  
24 void printLRS(const char S[]) {  
25     int m = strlen(S);  
26     int L[MAX + 1][MAX + 1];  
27  
28     for (int i = 0; i <= m; i++) {  
29         for (int j = 0; j <= m; j++) {  
30             if (i == 0 || j == 0)  
31                 L[i][j] = 0;  
32             else if (S[i - 1] == S[j - 1] && i != j)
```

To exit full screen, press and hold Esc

Enter string: ABCBCDC

Length of LRS: 3

LRS: ABC

==== Code Execution Successful ===

## LeetCode Assesment:-

<https://leetcode.com/problems/longest-common-subsequence/description/>

## Leetcode submission :-

The screenshot shows a LeetCode problem page for "Longest Common Subsequence". The top navigation bar includes "Problem List", "Accepted", "Editorial", "Solutions", and "Submissions". The status bar indicates "Premium". The main area shows the code editor with the following C code:

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

int longestCommonSubsequence(char* text1, char* text2) {
    int m = strlen(text1);
    int n = strlen(text2);

    int** dp = (int**)malloc((m + 1) * sizeof(int*));
    for (int i = 0; i <= m; i++) {
        dp[i] = (int*)malloc((n + 1) * sizeof(int));
    }

    for (int i = 0; i <= m; i++) {
        for (int j = 0; j <= n; j++) {
            if (i == 0 || j == 0)
                dp[i][j] = 0;
            else if (text1[i - 1] == text2[j - 1])
                dp[i][j] = dp[i - 1][j - 1] + 1;
            else
                dp[i][j] = (dp[i - 1][j] > dp[i][j - 1]) ? dp[i - 1][j] : dp[i][j - 1];
        }
    }

    return dp[m][n];
}
```

The runtime chart shows performance metrics: 26 ms (Beats 40.51%) and 25.84 MB (Beats 19.89%). The test result section shows "Accepted" with a runtime of 0 ms across three cases.

My leetcode link :-

<https://leetcode.com/u/codd12/>