Practical No. 5

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

Problem Statement: 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.

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

X=AGCCCTAAGGGCTACCTAGCTT

Y= GACAGCCTACAAGCGTTAGCTTG

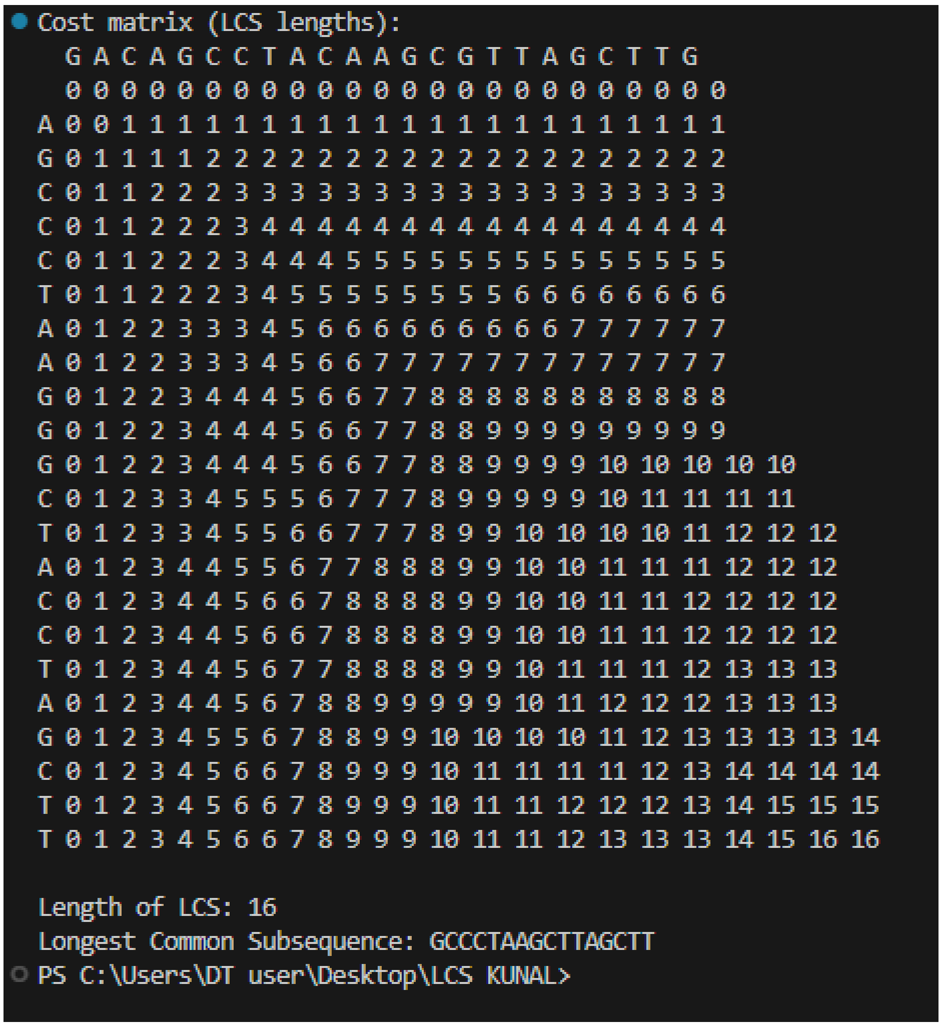
Code :

```
#include <iostream>

#include <string>

#define MAX 100


void printMatrix(int dp[MAX][MAX], const std::string& X, const std::string& Y) {

  int m = X.size();

  int n = Y.size();


  std::cout << "Cost matrix (LCS lengths):\n  ";

  for (int j = 0; j < n; ++j) std::cout << Y[j] << ' ';

  std::cout << '\n';


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

    if (i == 0)

      std::cout << ' ';

    else
```

```
            std::cout << X[i-1];
        std::cout << ' ';

        for (int j = 0; j <= n; ++j) {
            std::cout << dp[i][j] << ' ';
        }
        std::cout << '\n';
    }
}

std::string findLCS(const std::string& X, const std::string& Y, int dp[MAX][MAX]) {
    int i = X.size();
    int j = Y.size();
    int length = dp[i][j];
    std::string lcs(length, ' ');

    while (i > 0 && j > 0) {
        if (X[i-1] == Y[j-1]) {
            lcs[--length] = X[i-1];
            i--; j--;
        } else if (dp[i-1][j] > dp[i][j-1]) {
            i--;
        } else {
            j--;
        }
    }
    return lcs;
}
```

```
int main() {
    std::string X = "AGCCCTAAGGGCTACCTAGCTT";
    std::string Y = "GACAGCCTACAAGCGTTAGCTTG";

    int m = X.size();
    int n = Y.size();

    int dp[MAX][MAX] = {0};

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

    printMatrix(dp, X, Y);

    std::cout << "\nLength of LCS: " << dp[m][n] << '\n';
    std::cout << "Longest Common Subsequence: " << findLCS(X, Y, dp) << '\n';

    return 0;
}
```

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



```
Cost matrix (LCS lengths):
  G A C A G C C T A C A A G C G T T A G C T T G
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
A 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
G 0 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
C 0 1 1 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
C 0 1 1 2 2 2 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
C 0 1 1 2 2 2 3 4 4 4 5 5 5 5 5 5 5 5 5 5 5 5 5 5
T 0 1 1 2 2 2 3 4 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6
A 0 1 2 2 3 3 3 4 5 6 6 6 6 6 6 6 6 7 7 7 7 7 7
A 0 1 2 2 3 3 3 4 5 6 6 7 7 7 7 7 7 7 7 7 7 7 7
G 0 1 2 2 3 4 4 4 5 6 6 7 7 8 8 8 8 8 8 8 8 8 8
G 0 1 2 2 3 4 4 4 5 6 6 7 7 8 8 9 9 9 9 9 9 9 9
G 0 1 2 2 3 4 4 4 5 6 6 7 7 8 8 9 9 9 9 10 10 10 10 10
C 0 1 2 3 3 4 5 5 5 6 7 7 7 8 9 9 9 9 9 10 11 11 11 11
T 0 1 2 3 3 4 5 5 6 6 7 7 7 8 9 9 10 10 10 10 11 12 12 12
A 0 1 2 3 4 4 5 5 6 7 7 8 8 8 9 9 10 10 11 11 11 12 12 12
C 0 1 2 3 4 4 5 6 6 7 8 8 8 8 9 9 10 10 11 11 12 12 12 12
C 0 1 2 3 4 4 5 6 6 7 8 8 8 8 9 9 10 10 11 11 12 12 12 12
T 0 1 2 3 4 4 5 6 7 7 8 8 8 8 9 9 10 11 11 11 12 13 13 13
A 0 1 2 3 4 4 5 6 7 8 8 9 9 9 9 9 10 11 12 12 12 13 13 13
G 0 1 2 3 4 5 5 6 7 8 8 9 9 10 10 10 10 11 12 13 13 13 13 14
C 0 1 2 3 4 5 6 6 7 8 9 9 9 10 11 11 11 11 12 13 14 14 14 14
T 0 1 2 3 4 5 6 6 7 8 9 9 9 10 11 11 12 12 12 13 14 15 15 15
T 0 1 2 3 4 5 6 6 7 8 9 9 9 10 11 11 12 13 13 13 14 15 16 16

Length of LCS: 16
Longest Common Subsequence: GCCCTAAGCTTAGCTT
PS C:\Users\DT user\Desktop\LCS KUNAL>
```

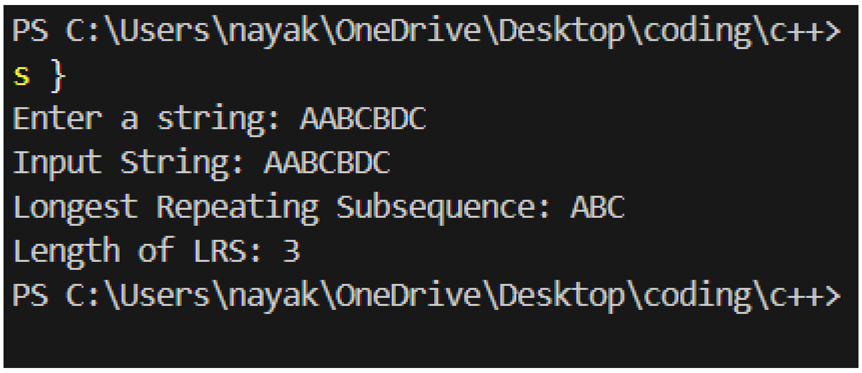
TASK-2: Find the longest repeating subsequence (LRS). Consider it as a variation of the longest common subsequence (LCS) problem. Let the given string be S. You need to find the LRS within S. To use the LCS framework, you effectively compare S with itself. So, consider string1 = S and string2 = S.

Code :

```
#include <iostream>

#include <cstring>

using namespace std;
```

```
#define MAX 100
void longestRepeatingSubsequence(char str[]) {
    int n = strlen(str);
    int dp[MAX][MAX];

    for (int i = 0; i <= n; i++) {
        for (int j = 0; j <= n; j++) {
            dp[i][j] = 0;
        }
    }

    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            if (str[i - 1] == str[j - 1] && i != j)
                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];
        }
    }

    int index = dp[n][n];
    char lrs[MAX];
    lrs[index] = '\0';
    int i = n, j = n;

    while (i > 0 && j > 0) {
        if (str[i - 1] == str[j - 1] && i != j) {
            lrs[index - 1] = str[i - 1];
            i--;
            j--;
            index--;
        } else if (dp[i - 1][j] > dp[i][j - 1]) {
            i--;
        } else {
            j--;
        }
    }

    cout << "Input String: " << str << endl;
    cout << "Longest Repeating Subsequence: " << lrs << endl;
    cout << "Length of LRS: " << dp[n][n] << endl;
}

int main() {
    char str[MAX];
    cout << "Enter a string: ";
    cin >> str;
    longestRepeatingSubsequence(str);
    return 0;
}
```

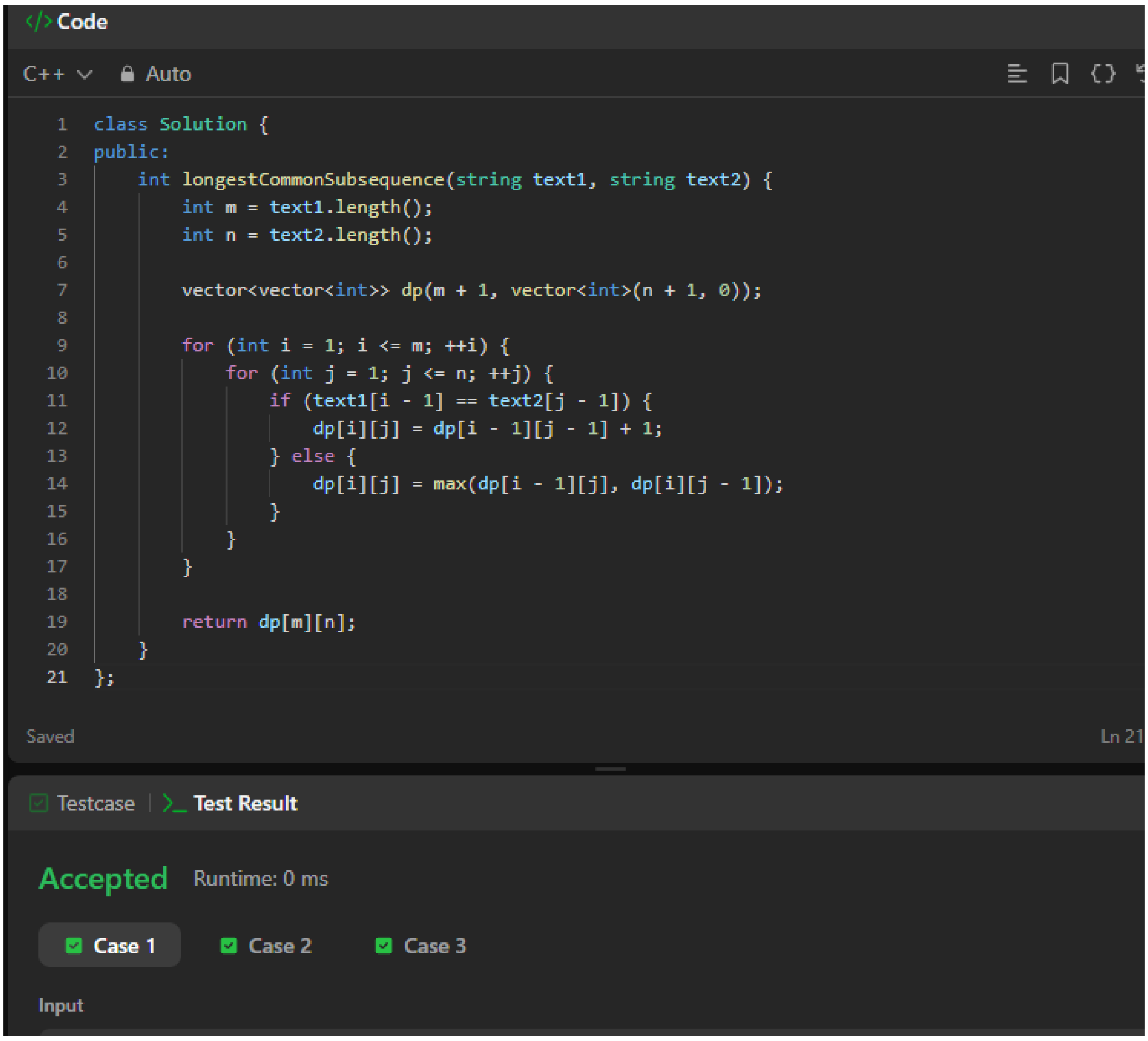
Output :



```
PS C:\Users\nayak\OneDrive\Desktop\coding\c++>
s }
Enter a string: AABCBDC
Input String: AABCBDC
Longest Repeating Subsequence: ABC
Length of LRS: 3
PS C:\Users\nayak\OneDrive\Desktop\coding\c++>
```

LeetCode Assesment:

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

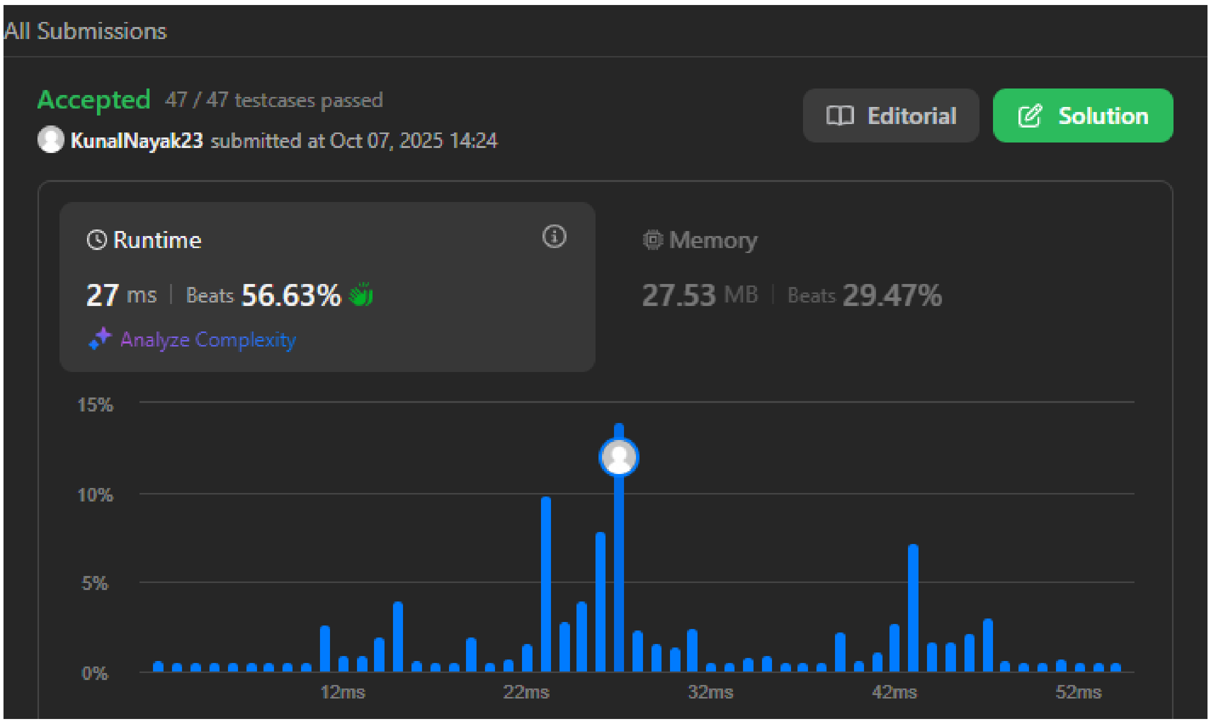


```cpp
class Solution {
public:
    int longestCommonSubsequence(string text1, string text2) {
        int m = text1.length();
        int n = text2.length();

        vector<vector<int>> dp(m + 1, vector<int>(n + 1, 0));

        for (int i = 1; i <= m; ++i) {
            for (int j = 1; j <= n; ++j) {
                if (text1[i - 1] == text2[j - 1]) {
                    dp[i][j] = dp[i - 1][j - 1] + 1;
                } else {
                    dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);
                }
            }
        }

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

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input


All Submissions

Accepted 47 / 47 testcases passed

KunalNayak23 submitted at Oct 07, 2025 14:24

Runtime

27 ms | Beats 56.63%

Memory

27.53 MB | Beats 29.47%