DAA PRACTICAL NO. 05

NAME: **Mohit Deepak Agrawal**

SECTION: **A2-B2**

ROLL NO: **33**

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. [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

CODE:

```
#include <stdio.h>

#include <string.h>

#define MAX 100

void printMatrix(int dp[][MAX], int m, int n, char *X, char *Y) {

   printf("Cost matrix (LCS lengths):\n    ");

   for (int j = 0; j < n; j++)
```

```
        printf("%c ", Y[j]);
    printf("\n");
    for (int i = 0; i <= m; i++) {
        if (i == 0)
            printf("  ");
        else
            printf("%c ", X[i-1]);
        for (int j = 0; j <= n; j++) {
            printf("%d ", dp[i][j]);
        }
        printf("\n");
    }
}
void findLCS(char *X, char *Y, int m, int n, int dp[][MAX], char *lcs) {
    int index = dp[m][n];
    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 (dp[i-1][j] > dp[i][j-1]) {
            i--;
        } else {
```

```
            j--;
        }
    }
}
int main() {
    char X[] = "AGCCCTAAGGGCTACCTAGCTT";
    char Y[] = "GACAGCCTACAAGCGTTAGCTTG";
    int m = strlen(X);
    int n = strlen(Y);
    int dp[MAX][MAX];
    for (int i = 0; i <= m; i++)
        for (int j = 0; j <= n; j++)
            dp[i][j] = 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, m, n, X, Y);
    printf("\nLength of Longest Common Subsequence: %d\n", dp[m][n]);
    char lcs[MAX];
    findLCS(X, Y, m, n, dp, lcs);
    printf("Longest Common subsequence: %s\n", lcs);
```

```
    return 0;
}
```

Output ScreenShot:

```
Output

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 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 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 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 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 Longest Common Subsequence: 16
Longest Common subsequence: GCCCTAAGCTTAGCTT


=== Code Execution Successful ===
```

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

CODE:

```
#include <stdio.h>
#include <string.h>
#define MAX 100
void printLRS(char str[], int n, int dp[MAX][MAX]) {
    int i = n, j = n;
    char lrs[MAX];
    int index = dp[n][n];
    lrs[index] = '\0';
    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--;
        }
    }
    printf("\nLongest Repeating Subsequence: %s\n", lrs);
    printf("Length of LRS: %d\n", dp[n][n]);
}
void LRS(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];
        }
    }
    printLRS(str, n, dp);
}

int main() {
    char str[] = "AABCBDC";
    LRS(str);

    return 0;
}
```

Output ScreenShot:

Output

```
Longest Repeating Subsequence: ABC
Length of LRS: 3


=== Code Execution Successful ===
```

Leetcode Assesment:

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

Problem List

Submit

Premium

Description | Editorial | Solutions | Submissions

1143. Longest Common Subsequence

Medium | Topics | Companies | Hint

Given two strings `text1` and `text2`, return *the length of their longest* ***common subsequence***. If there is no **common subsequence**, return `0`.

A **subsequence** of a string is a new string generated from the original string with some characters (can be none) deleted without changing the relative order of the remaining characters.

- For example, `"ace"` is a subsequence of `"abcde"`.

A **common subsequence** of two strings is a subsequence that is common to both strings.

Example 1:

```
Input: text1 = "abcde", text2 = "ace"
Output: 3
Explanation: The longest common subsequence is "ace" and its length is 3.
```

Example 2:

```
Input: text1 = "abc", text2 = "abc"
Output: 3
Explanation: The longest common subsequence is "abc" and its length is 3.
```

Example 3:

```
Input: text1 = "abc", text2 = "def"
Output: 0
```

14.7K | 241 | 189 Online

Code

Java | Auto

```java
public class Solution {
public int longestCommonSubsequence(String text1, String text2) {
    int m = text1.length();
     int n = text2.length();

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

Saved | Ln 18, Col 2

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 | Case 2 | Case 3

Input

text1 =

"abcde"

Submit

Accepted 47 / 47 testcases passed

A2_B2_33 Mohit submitted at Oct 07, 2025 14:06

Editorial | Solution

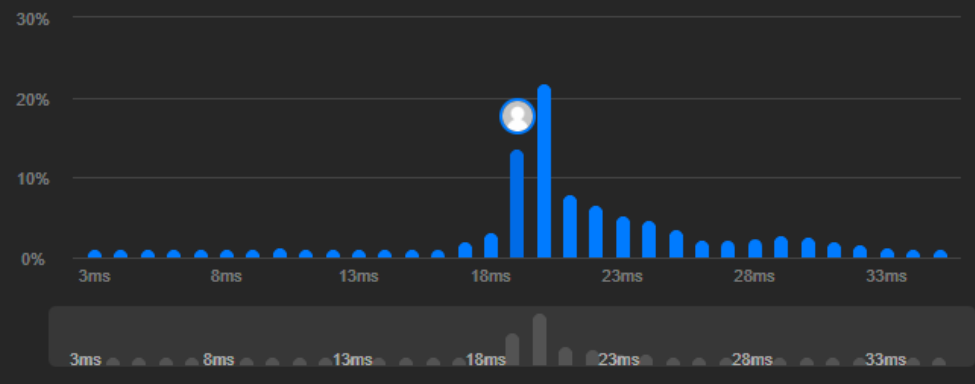
Runtime

19 ms | Beats 88.34%

Analyze Complexity

Memory

50.88 MB | Beats 69.46%



Code | Java

```java
public class Solution {
public int longestCommonSubsequence(String text1, String text2) {
    int m = text1.length();
     int n = text2.length();
```