Name: Areeb Ansari Roll No:A3-B1-01

Practical 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

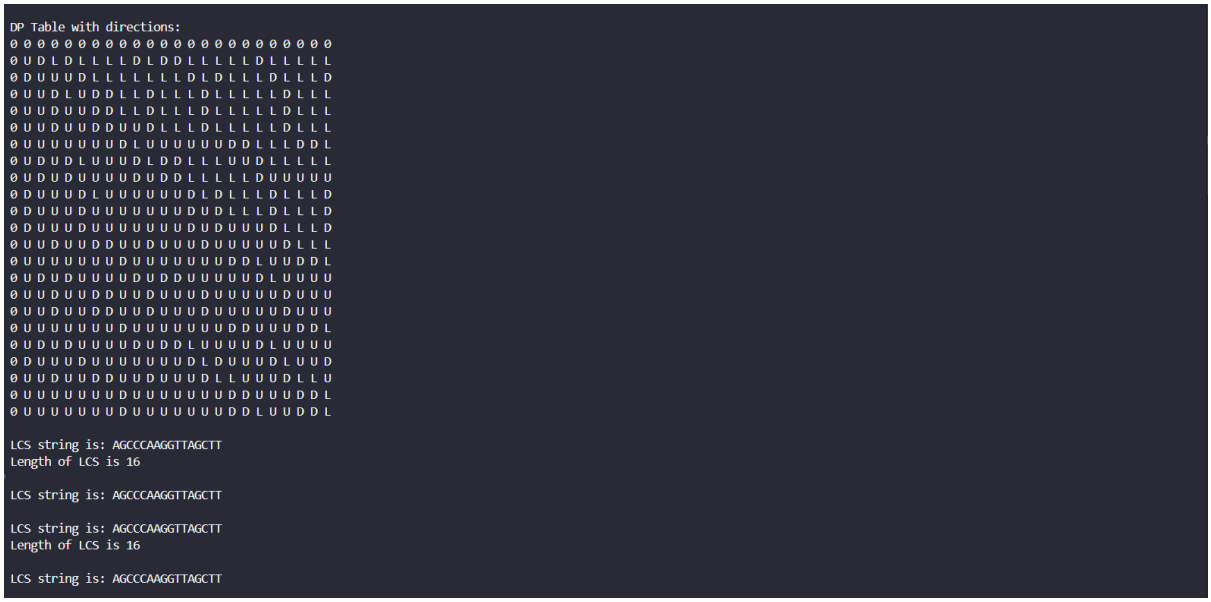
Code:

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

int lcs(char *X, char *Y, int m, int n) {
  int L[m+1][n+1];
  char dir[m+1][n+1];

  for (int i = 0; i <= m; i++) {
    for (int j = 0; j <= n; j++) {
      if (i == 0 || j == 0) {
        L[i][j] = 0;
```

```
            dir[i][j] = '0';
        }
        else if (X[i-1] == Y[j-1]) {
            L[i][j] = L[i-1][j-1] + 1;
            dir[i][j] = 'D'; // Diagonal ↖
        }
        else if (L[i-1][j] >= L[i][j-1]) {
            L[i][j] = L[i-1][j];
            dir[i][j] = 'U'; // Up ↑
        }
        else {
            L[i][j] = L[i][j-1];
            dir[i][j] = 'L'; // Left ←
        }
    }
}

printf("\nDP Table with directions:\n");
for (int i = 0; i <= m; i++) {
    for (int j = 0; j <= n; j++) {
        if (dir[i][j] == 'D') printf("D ");
        else if (dir[i][j] == 'U') printf("U ");
        else if (dir[i][j] == 'L') printf("L ");
        else printf("0 ");
    }
    printf("\n");
}

int index = L[m][n];
char lcsStr[index + 1];
```

```
    lcsStr[index] = '\0';

    int i = m, j = n;
    while (i > 0 && j > 0) {
        if (X[i-1] == Y[j-1]) {
            lcsStr[index-1] = X[i-1];
            i--; j--; index--;
        }
        else if (L[i-1][j] >= L[i][j-1]) {
            i--;
        }
        else {
            j--;
        }
    }

    printf("\nLCS string is: %s\n", lcsStr);
    return L[m][n];
}

int main() {
    char X[] = "AGCCCTAAGGGCTACCTAGCTT";
    char Y[] = "GACAGCCTACAAGCGTTAGCTTG";
    int m = strlen(X);
    int n = strlen(Y);

    printf("Length of LCS is %d\n", lcs(X, Y, m, n));
    return 0;
}
```

Output:



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>

int lrs(char *str, int n) {
   int L[n+1][n+1];

   for (int i = 0; i <= n; i++) {
      for (int j = 0; j <= n; j++) {
         if (i == 0 || j == 0) {
```

```
            L[i][j] = 0;
        }
        else if (str[i-1] == str[j-1] && i != j) {
            L[i][j] = 1 + L[i-1][j-1];
        }
        else {
            L[i][j] = (L[i-1][j] > L[i][j-1]) ? L[i-1][j] : L[i][j-1];
        }
    }
}

int index = L[n][n];
char lrsStr[index + 1];
lrsStr[index] = '\0';

int i = n, j = n;
while (i > 0 && j > 0) {
    if (L[i][j] == L[i-1][j-1] + 1 && str[i-1] == str[j-1] && i != j) {
        lrsStr[index - 1] = str[i-1];
        i--;
        j--;
        index--;
    }
    else if (L[i-1][j] > L[i][j-1]) {
        i--;
    }
    else {
        j--;
    }
}
```

```
    printf("Longest Repeated Subsequence is: %s\n", lrsStr);
    return L[n][n];
}

int main() {
    char str[] = "AAABCBDC";
    int n = strlen(str);

    printf("Length of LRS is %d\n", lrs(str, n));
    return 0;
}
```

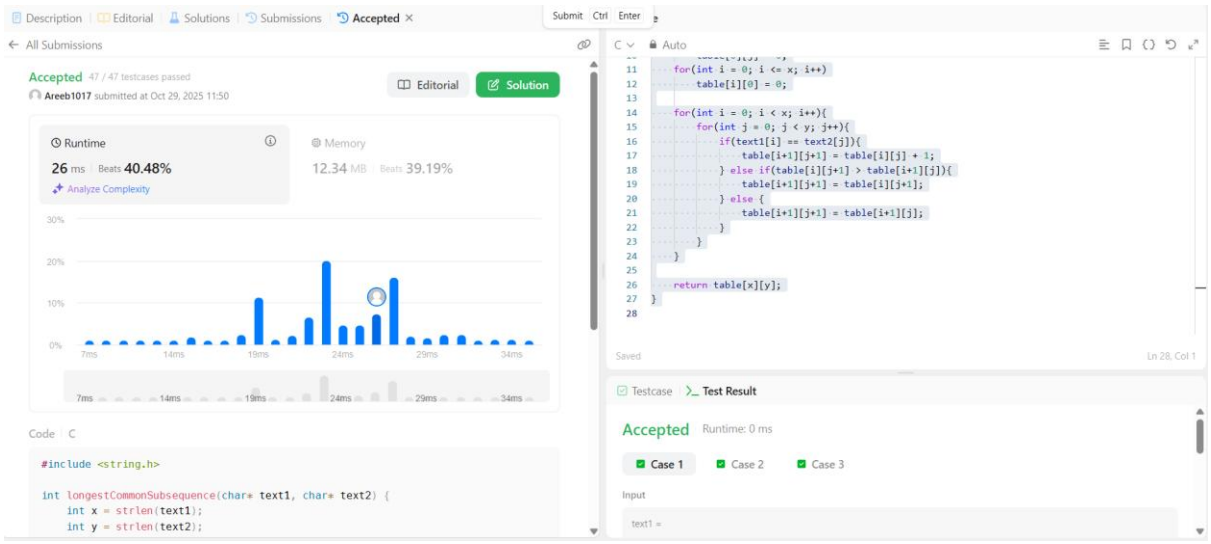
Output:

LeetCode:

```
#include <stdio.h>
#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] = 1 + dp[i-1][j-1];
            }
            else {
                dp[i][j] = (dp[i-1][j] > dp[i][j-1]) ? dp[i-1][j] : dp[i][j-1];
            }
        }
    }
    int result = dp[m][n];

    for (int i = 0; i <= m; i++) {
        free(dp[i]);
    }
    free(dp);

    return result;
}

int main() {
    char text1[] = "abcde";
    char text2[] = "ace";
    printf("Length of LCS = %d\n", longestCommonSubsequence(text1, text2));
    return 0;
```

}


Description
Editorial
Solutions
Submissions
Accepted
Submit Ctrl Enter
All Submissions
Accepted 47 / 47 testcases passed
Areeb1017 submitted at Oct 29, 2025 11:50
Editorial
Solution
Runtime
26 ms | Beats 40.48%
Analyze Complexity
Memory
12.34 MB | Beats 39.19%
30%
20%
10%
0%
7ms
14ms
19ms
24ms
29ms
34ms
Code | C
#include <string.h>
int longestCommonSubsequence(char* text1, char* text2) {
int x = strlen(text1);
int y = strlen(text2);
C
Auto
for(int i = 0; i <= x; i++)
table[i][0] = 0;
for(int i = 0; i < x; i++){
for(int j = 0; j < y; j++){
if(text1[i] == text2[j]){
table[i+1][j+1] = table[i][j] + 1;
} else if(table[i][j+1] > table[i+1][j]){
table[i+1][j+1] = table[i][j+1];
} else {
table[i+1][j+1] = table[i+1][j];
}
}
}
return table[x][y];
}
Saved
Ln 28, Col 1
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Case 3
Input
text1 =