|  |  |
| --- | --- |
| Name : | Swati Kundu |
| Section : | A4-B3 |
| Roll No : | 48 |

**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

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>

#define MAX 100

void LCS\_Length(char X[], char Y[], int m, int n, int c[MAX][MAX], char b[MAX][MAX]) {

    for (int i = 0; i <= m; i++)

        c[i][0] = 0;

    for (int j = 0; j <= n; j++)

        c[0][j] = 0;

    for (int i = 1; i <= m; i++) {

        for (int j = 1; j <= n; j++) {

            if (X[i - 1] == Y[j - 1]) {

                c[i][j] = c[i - 1][j - 1] + 1;

                b[i][j] = '\\'; // diagonal

            } else if (c[i - 1][j] >= c[i][j - 1]) {

                c[i][j] = c[i - 1][j];

                b[i][j] = '^'; // up

            } else {

                c[i][j] = c[i][j - 1];

                b[i][j] = '<'; // left

            }

        }

    }

}

void Print\_LCS(char b[MAX][MAX], char X[], int i, int j) {

    if (i == 0 || j == 0)

        return;

    if (b[i][j] == '\\') {

        Print\_LCS(b, X, i - 1, j - 1);

        printf("%c", X[i - 1]);

    } else if (b[i][j] == '^') {

        Print\_LCS(b, X, i - 1, j);

    } else {

        Print\_LCS(b, X, i, j - 1);

    }

}

int main() {

    char X[] = "AGCCCTAAGGGCTACCTAGCTT";

    char Y[] = "GACAGCCTACAAGCGTTAGCTTG";

    int m = strlen(X);

    int n = strlen(Y);

    int c[MAX][MAX];

    char b[MAX][MAX];

    LCS\_Length(X, Y, m, n, c, b);

    printf("Cost Matrix:\n");

    for (int i = 0; i <= m; i++) {

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

            printf("%2d ", c[i][j]);

        }

        printf("\n");

    }

    printf("\nLength of LCS = %d\n", c[m][n]);

    printf("LCS: ");

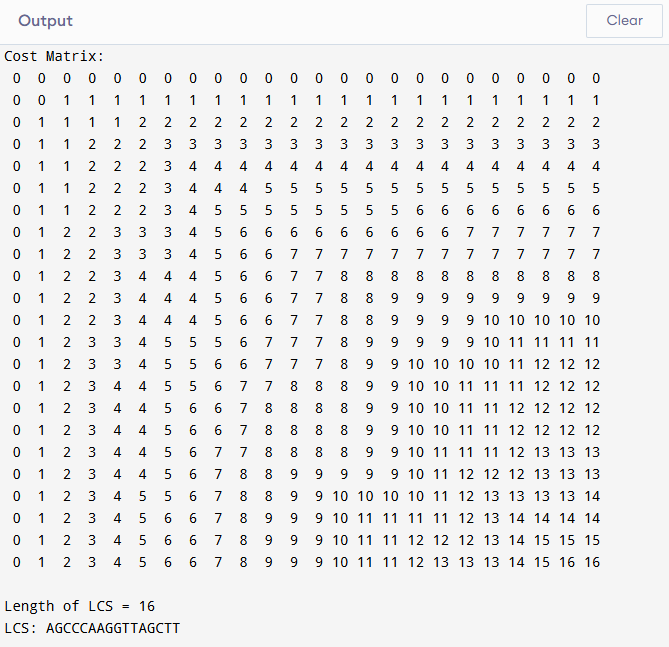
    Print\_LCS(b, X, m, n);

    printf("\n");

    return 0;

}

**Output :**



**Task 2 Code :**

#include <stdio.h>

#include <string.h>

#define MAX 100

void LRS\_Length(char S[], int n, int c[MAX][MAX], char b[MAX][MAX]) {

    for (int i = 0; i <= n; i++)

        c[i][0] = c[0][i] = 0;

    for (int i = 1; i <= n; i++) {

        for (int j = 1; j <= n; j++) {

            if (S[i - 1] == S[j - 1] && i != j) {

                c[i][j] = c[i - 1][j - 1] + 1;

                b[i][j] = '\\'; // diagonal

            } else if (c[i - 1][j] >= c[i][j - 1]) {

                c[i][j] = c[i - 1][j];

                b[i][j] = '^'; // up

            } else {

                c[i][j] = c[i][j - 1];

                b[i][j] = '<'; // left

            }

        }

    }

}

void Print\_LRS(char b[MAX][MAX], char S[], int i, int j) {

    if (i == 0 || j == 0)

        return;

    if (b[i][j] == '\\') {

        Print\_LRS(b, S, i - 1, j - 1);

        printf("%c", S[i - 1]);

    } else if (b[i][j] == '^') {

        Print\_LRS(b, S, i - 1, j);

    } else {

        Print\_LRS(b, S, i, j - 1);

    }

}

int main() {

    char S[] = "AABCBDC";

    int n = strlen(S);

    int c[MAX][MAX];

    char b[MAX][MAX];

    LRS\_Length(S, n, c, b);

    printf("Cost Matrix:\n");

    for (int i = 0; i <= n; i++) {

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

            printf("%2d ", c[i][j]);

        }

        printf("\n");

    }

    // Final cost of LRS

    printf("\nLength of LRS = %d\n", c[n][n]);

    printf("LRS: ");

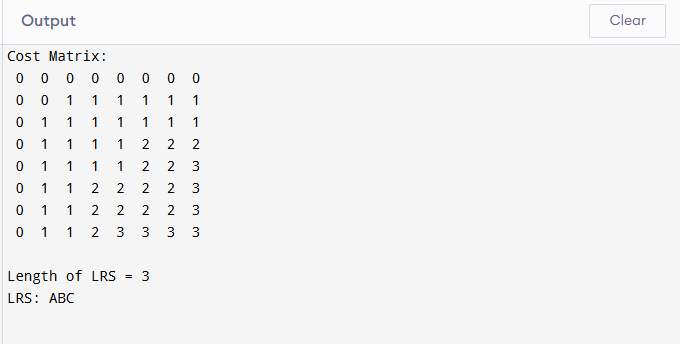
    Print\_LRS(b, S, n, n);

    printf("\n");

    return 0;

}

**Output :**



**Leetcode task :**

int longestCommonSubsequence(char \* text1, char \* text2) {

int len1 = strlen(text1);

int len2 = strlen(text2);

// Create DP table

int dp[len1 + 1][len2 + 1];

// Initialize DP table

for (int i = 0; i <= len1; i++) {

for (int j = 0; j <= len2; j++) {

dp[i][j] = 0;

}

}

// Fill the DP table

for (int i = 1; i <= len1; i++) {

for (int j = 1; j <= len2; j++) {

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];

}

}

}

return dp[len1][len2];

}

**Output:**

