NAME-RUDRA KALE

A8-B4-65

PRACTICAL 5

DAA LAB

Aim: Implement a dynamic algorithm for

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>

void LCS(char A[], char B[])
{
    int m = strlen(A);
    int n = strlen(B);
    int C[m+1][n+1];

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

```
    {
        C[i][j] = 0;
    }
}

for(int i = 1; i <= m; i++)
{
    for(int j = 1; j <= n; j++)
    {
        if(A[i-1] == B[j-1])
        {
            C[i][j] = C[i-1][j-1] + 1;
        }
        else
        {
```

```
                C[i][j] = (C[i-1][j] > C[i][j-
1]) ? C[i-1][j] : C[i][j-1];
            }
        }
    }

    int lcslen = C[m][n];
    printf("LCS length: %d\n",
lcslen);

    char lcs[lcslen + 1];
    int index = lcslen;
    lcs[index] = '\0';

    int i = m, j = n;
```

```
while(i > 0 && j > 0)
{
    if(A[i-1] == B[j-1])
    {
        lcs[index-1] = A[i-1];
        i--;
        j--;
        index--;
    }
    else if(C[i-1][j] > C[i][j-1])
    {
        i--;
    }
    else
    {
```

```
            j--;
        }
    }

    printf("Longest Common
Sequence: %s\n", lcs);
}

int main()
{
    char X[] =
"AGCCCTAAGGGCTACCTAGCTT";
    char Y[] =
"GACAGCCTACAAGCGTTAGCTTG"
;
```

```
    LCS(X, Y);

    return 0;
}
```

OUTPUT

```
LCS length: 16
Longest Common Sequence: 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>

void LRS(char S[])
{
```

```
int n = strlen(S);
int C[n+1][n+1];

for(int i = 0; i <= n; i++)
{
    for(int j = 0; j <= n; j++)
    {
        C[i][j] = 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;
            }
            else
            {
                C[i][j] = (C[i-1][j] > C[i][j-1]) ? C[i-1][j] : C[i][j-1];
            }
        }
    }

    int lrslen = C[n][n];
    printf("LRS length: %d\n", lrslen);
```

```
char lrs[n+1];
int index = lrslen;
lrs[index] = '\0';

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

```
        else if(C[i-1][j] > C[i][j-1])
        {
            i--;
        }
        else
        {
            j--;
        }
    }

    printf("Longest Repeating
Sequence: %s\n\n", lrs);
}

int main()
```

```
{
    char X[] = "AABCBDC";
    char Y[] = "AABEBCDD";

    LRS(X);
    LRS(Y);

    return 0;
}
```

OUTPUT

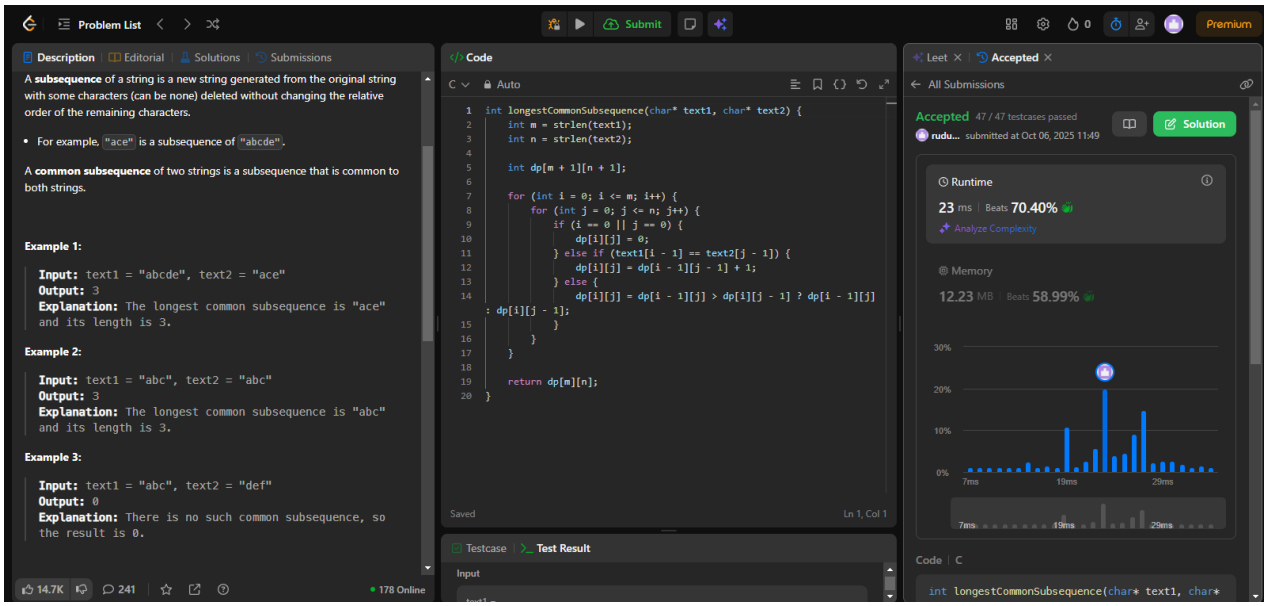
```
LRS length: 3
Longest Repeating Sequence: ABC

LRS length: 3
Longest Repeating Sequence: ABD



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

LEETCODE


Problem List
Submit
Premium
Description
Editorial
Solutions
Submissions
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
Explanation: There is no such common subsequence, so the result is 0.
14.7K
241
178 Online
Code
C
Auto
int longestCommonSubsequence(char* text1, char* text2) {
int m = strlen(text1);
int n = strlen(text2);
int dp[m + 1][n + 1];
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];
}
Saved
Ln 1, Col 1
Testcase
Test Result
Input
Leet
Accepted
All Submissions
Accepted 47 / 47 testcases passed
rudu... submitted at Oct 06, 2025 11:49
Solution
Runtime
23 ms | Beats 70.40%
Analyze Complexity
Memory
12.23 MB | Beats 58.99%
30%
20%
10%
0%
7ms
19ms
29ms
Code | C
int longestCommonSubsequence(char* text1, char*