

Name :- Vrishabh hanmante  
Roll no :-A4 B3 38  
Daa prac 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.

Code:- lcs

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

#define MAX 100

void printLCS(char* X, char* Y, int m, int n, int L[MAX][MAX]) {
    int index = L[m][n];
    char lcs[index+1];
    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 (L[i-1][j] > L[i][j-1]) {
            i--;
        } else {
            j--;
        }
    }
    printf("Longest Common Subsequence: %s\n", lcs);
}

int main() {
    char X[] = "XAGCCCTAACGGCTACCTAGCTT";
    char Y[] = "GACAGCCTACAAGCGTTAGCTTG";
    int m = strlen(X);
    int n = strlen(Y);
    int L[MAX][MAX];
```

```

for (int i = 0; i <= m; i++) {
    for (int j = 0; j <= n; j++) {
        if (i == 0 || j == 0)
            L[i][j] = 0;
        else if (X[i-1] == Y[j-1])
            L[i][j] = L[i-1][j-1] + 1;
        else
            L[i][j] = (L[i-1][j] > L[i][j-1])? L[i-1][j] : L[i][j-1];
    }
}

printf("Length of LCS: %d\n", L[m][n]);
printLCS(X, Y, m, n, L);

return 0;
}

Code:- lrs
#include <stdio.h>
#include <string.h>

#define MAX 100

void printLRS(char* str, int n, int L[MAX][MAX]) {
    int index = L[n][n];
    char lrs[index+1];
    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 (L[i-1][j] > L[i][j-1]) {
            i--;
        } else {
            j--;
        }
    }
    printf("Longest Repeating Subsequence: %s\n", lrs);
}

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

```

```

int L[MAX][MAX];

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];
    }
}

printf("Length of LRS: %d\n", L[n][n]);
printLRS(str, n, L);

return 0;
}

```

output

Parameter	Value
X (Sequence 1)	XAGCCCTAAGGGCTACCTAGCTT
Y (Sequence 2)	GACAGCCTACAAGCGTTAGCTTG
LCS Length	16
LCS Sequence Example	A G C C C T A A G C T T A G C T T

Parameter	Value
S (Input String)	AABCBD
LRS Length	3
LRS Sequence Example	ABC or ABD

The screenshot shows a mobile browser displaying a LeetCode submission page. The URL in the address bar is [leetcode.com/problems/longest-common-subsequence/submissions/18...](https://leetcode.com/problems/longest-common-subsequence/submissions/18...). The page header includes links to Instagram, Classroom, Gmail, YouTube, Maps, News, Translate, and Peazehub - Focus... The top right corner shows a Premium badge.

The main content area shows the submission details:

- Description**: Accepted, 47 / 47 testcases passed.
- Editorial**: Available.
- Solution**: Available.

Performance metrics:

- Runtime**: 24 ms | Beats 50.36%.
- Memory**: 12.15 MB | Beats 77.68%.

A chart below the metrics shows memory usage over time, with a sharp peak around 20-30% usage.

The code editor on the right contains the following C code for the longest common subsequence problem:

```
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];  
}
```

The status bar at the bottom indicates the code was saved on Oct 14 at 9:26.