

DAA Lab

Practical 5

Name: Madhura Mahatme

Section: A1-B3

Roll no.: 36

Git Link: <https://github.com/mahatmemadhura4-bot/DAA-Lab-Pract5>

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 lcs(char a[], char b[], int c[MAX][MAX], char dir[MAX][MAX], int m, int n) {
    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 (a[i-1] == b[j-1]) {
                c[i][j] = c[i-1][j-1] + 1;
                dir[i][j] = 'D';
            }
            else if(c[i-1][j] >= c[i][j-1]){
                c[i][j] = c[i-1][j];
                dir[i][j] = 'U';
            }
            else{
                c[i][j] = c[i][j-1];
                dir[i][j] = 'L';
            }
        }
    }
}

void print_matrix(int c[MAX][MAX], char dir[MAX][MAX], int m, int n, char a[], char b[])
{
    printf("\nCost Matrix with Directions:\n\n ");
    for(int j = 0; j < n; j++){

```

```

        printf("  %c", b[j]);
    }
    printf("\n");

    for (int i = 0; i <= m; i++){
        if (i==0) printf(" ");
        else printf("%c ", a[i-1]);

        for (int j = 0; j <= n; j++){
            if (i==0 || j==0)
                printf(" %2d ", c[i][j]);
            else
                printf(" %2d%c ", c[i][j], dir[i][j]);
        }
        printf("\n");
    }
}

void print_lcs(char a[], char b[], char dir[MAX][MAX], int i, int j){
    if(i==0 || j==0)
        return;
    if(dir[i][j] == 'D'){
        print_lcs(a, b, dir, i-1, j-1);
        printf("%c", a[i-1]);
    } else if (dir[i][j] == 'U'){
        print_lcs(a, b, dir, i-1, j);
    } else {
        print_lcs(a, b, dir, i, j-1);
    }
}
}

```

```
int main() {  
    char X[MAX], Y[MAX];  
    int c[MAX][MAX];  
    char dir[MAX][MAX];  
  
    printf("Enter first sequence (X): ");  
    scanf("%s", X);  
  
    printf("Enter second sequence (Y): ");  
    scanf("%s", Y);  
  
    int m = strlen(X);  
    int n = strlen(Y);  
  
    lcs(X, Y, c, dir, m, n);  
  
    print_matrix(c, dir, m, n, X, Y);  
  
    printf("\nFinal cost (Length of LCS): %d\n", c[m][n]);  
  
    printf("LCS: ");  
    print_lcs(X, Y, dir, m, n);  
    printf("\n");  
  
    return 0;  
}
```

```

1  ✓ #include <stdio.h>
2  #include <string.h>
3  #define MAX 100
4  ✓ void lcs(char a[], char b[], int c[MAX][MAX], char dir[MAX][MAX], int m, int n) {
5  ✓     for(int i = 0; i <= m; i++){
6      |         c[i][0] = 0;
7      |     }
8  ✓     for(int j = 0; j <= n; j++){
9      |         c[0][j] = 0;
10     |     }
11  ✓     for(int i = 1; i <= m; i++){
12  ✓         for(int j = 1; j <= n; j++){
13  ✓             if(a[i-1] == b[j-1]){
14                 c[i][j] = c[i-1][j-1] + 1;
15                 dir[i][j] = 'D';
16             }
17  ✓             else if(c[i-1][j] >= c[i][j-1]){
18                 c[i][j] = c[i-1][j];
19                 dir[i][j] = 'U';
20             }
21  ✓             else{
22                 c[i][j] = c[i][j-1];
23                 dir[i][j] = 'L';
24             }
25         }
26     }
27 }
28 ✓ void print_matrix(int c[MAX][MAX], char dir[MAX][MAX], int m, int n, char a[], char b[]){
29     printf("\nCost Matrix with Directions:\n\n ");
30  ✓     for(int j = 0; j < n; j++){
31         printf("    %c", b[j]);
32     }
33     printf("\n");
34
35  ✓     for (int i = 0; i <= m; i++){
36         if (i==0) printf(" ");
37         else printf("%c ", a[i-1]);

```

```

38
39     for (int j = 0; j <= n; j++){
40         if (i==0 || j==0)
41             printf(" %2d ", c[i][j]);
42         else
43             printf(" %2d%c ", c[i][j], dir[i][j]);
44     }
45     printf("\n");
46 }
47 }
48 void print_lcs(char a[], char b[], char dir[MAX][MAX], int i, int j){
49     if(i==0 || j==0)
50         return;
51     if(dir[i][j] == 'D'){
52         print_lcs(a, b, dir, i-1, j-1);
53         printf("%c", a[i-1]);
54     }
55     else if(dir[i][j] == 'U'){
56         print_lcs(a, b, dir, i-1, j);
57     }
58     else{
59         print_lcs(a, b, dir, i, j-1);
60     }
61 }
62
63 int main(){
64     char X[MAX], Y[MAX];
65     int c[MAX][MAX];
66     char dir[MAX][MAX];
67
68     printf("Enter first sequence (X): ");
69     scanf("%s", X);
70

```

```

71     printf("Enter second sequence (Y): ");
72     scanf("%s", Y);
73
74     int m = strlen(X);
75     int n = strlen(Y);
76
77     lcs(X, Y, c, dir, m, n);
78
79     print_matrix(c, dir, m, n, X, Y);
80
81     printf("\nFinal cost (Length of LCS): %d\n", c[m][n]);
82
83     printf("LCS: ");
84     print_lcs(X, Y, dir, m, n);
85     printf("\n");
86
87     return 0;
88 }
89
90

```

Output:

```
Enter first sequence (X): AGCCCTAAGGGCTACCTAGCTT
Enter second sequence (Y): GACAGCCTACAAGCGTTAGCTTG

Cost Matrix with Directions:

      G  A  C  A  G  C  C  T  A  C  A  A  G  C  G  T  T  A  G  C  T  T  G
A  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
G  0  0  1D 1U 1U 1U 2D 2L 2L 2L 2L 2L 2L 2D 2L 2D 2L 2L 2L 2D 2L 2L 2L
C  0  1U 1U 1U 2D 2L 2U 3D 3D 3L 3L 3L 3D 3L 3L 3L 3D 3L 3L 3L 3L 3L 3L
C  0  1U 1U 2D 2U 2U 3D 4D 4L 4L 4D 4L 4L 4L 4L 4L 4D 4L 4L 4L 4L 4L 4L
C  0  1U 1U 2D 2U 2U 3D 4D 4U 4U 5D 5L 5L 5L 5L 5D 5L 5L 5L 5L 5L 5L
T  0  1U 1U 2U 2U 2U 3U 4U 5D 5L 5U 5U 5U 5U 5U 5U 6D 6D 6L 6L 6D 6D 6L
A  0  1U 2D 2U 3D 3L 3U 4U 5U 6D 6L 6D 6D 6L 6L 6L 6U 6U 7D 7L 7L 7L 7L
A  0  1U 2D 2U 3D 3U 3U 4U 5U 6D 6U 7D 7D 7L 7L 7L 7L 7L 7D 7U 7U 7U 7U
G  0  1D 2U 2U 3U 4D 4L 4U 5U 6U 6U 7U 7U 8D 8L 8D 8L 8L 8L 8D 8L 8L 8D
G  0  1D 2U 2U 3U 4D 4U 4U 5U 6U 6U 7U 7U 8D 8U 9D 9L 9L 9L 9L 9D 9L 9D
G  0  1D 2U 2U 3U 4D 4U 4U 5U 6U 6U 7U 7U 8D 8U 9D 9U 9U 9U 9U 10D 10L 10L 10D
C  0  1U 2U 3D 3U 4U 5D 5U 6U 7D 7U 7U 8U 8U 9D 9U 9U 9U 9U 9U 10U 11D 11L 11L 11L
T  0  1U 2U 3U 3U 4U 5U 5U 6D 6U 7U 7U 7U 8U 9U 9U 10D 10D 10L 10U 11U 12D 12D 12L 12L
A  0  1U 2D 3U 4D 4U 5U 5U 6U 7D 7U 8D 8D 8U 9U 9U 10U 10U 11D 11L 11U 12U 12U 12U 12U
C  0  1U 2U 3D 4U 4U 5D 6D 6U 7U 8D 8U 8U 8U 9D 9U 10U 10U 11U 11U 12D 12U 12U 12U 12U
C  0  1U 2U 3D 4U 4U 5D 6D 6U 7U 8D 8U 8U 8U 9D 9U 10U 10U 11U 11U 12D 12U 12U 12U 12U
T  0  1U 2U 3U 4U 4U 5U 6U 7D 7U 8U 8U 8U 8U 9U 9U 10D 11D 11U 11U 12U 13D 13D 13L 13L
A  0  1U 2D 3U 4D 4U 5U 6U 7U 8D 8U 9D 9D 9L 9U 9U 10U 11U 12D 12L 12U 13U 13U 13U 13U
G  0  1D 2U 3U 4U 5D 5U 6U 7U 8U 8U 9U 9U 10D 10L 10D 10U 11U 12U 13D 13L 13U 13U 14D 14D
C  0  1U 2U 3D 4U 5U 6D 6D 7U 8U 9D 9U 9U 10U 11D 11L 11L 11U 12U 13U 14D 14L 14L 14L 14U
T  0  1U 2U 3U 4U 5U 6U 6U 7D 8U 9U 9U 9U 10U 11U 11U 12D 12D 12U 13U 14U 15D 15D 15L 15L
T  0  1U 2U 3U 4U 5U 6U 6U 7D 8U 9U 9U 9U 10U 11U 11U 12D 13D 13L 13U 14U 15D 16D 16L 16L

Final cost (Length of LCS): 16
LCS: AGCCCAAGGTTAGCTT
PS C:\Users\Madhura\OneDrive\Desktop\C> 
```

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:

AABCBCDC

LRS= ABC or ABD

Code:

```
#include <stdio.h>

#include <string.h>

#define MAX 100

void lrs(char s[], int c[MAX][MAX], int m){
    for (int i=0; i <= m; i++) {
        c[i][0] = 0;
        c[0][i] = 0;
    }
    for (int i = 1; i <= m; i++){
        for (int j = 1; j <= m; 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];
            }
        }
    }
}

void print_lrs(char s[], int c[MAX][MAX], int i, int j){
    if (i==0 || j==0) return;

    if (s[i-1] == s[j-1] && i != j){
        print_lrs(s, c, i-1, j-1);
        printf("%c", s[i-1]);
    }
    else if(c[i-1][j] > c[i][j-1]){
        print_lrs(s, c, i-1, j);
    }
}
```



```

    }
    else{
        print_lrs(s, c, i, j-1);
    }
}

int main(){
    char S[MAX];
    int c[MAX][MAX];
    printf("Enter the string: ");
    scanf("%s", S);

    int m = strlen(S);
    lrs(S, c, m);

    printf("\nLength of Longest Repeating Subsequence: %d\n", c[m][m]);
    printf("LRS: ");
    print_lrs(S, c, m, m);
    printf("\n");

    return 0;
}

```

```

1  #include <stdio.h>
2  #include <string.h>
3  #define MAX 100
4  void lrs(char s[], int c[MAX][MAX], int m){
5      for (int i=0; i <= m; i++) {
6          c[i][0] = 0;
7          c[0][i] = 0;
8      }
9      for (int i = 1; i <= m; i++){
10         for (int j = 1; j <= m; j++){
11             if (s[i-1]==s[j-1] && i != j){
12                 c[i][j]=c[i-1][j-1] + 1;
13             }
14             else{
15                 c[i][j] = (c[i-1][j] > c[i][j-1]) ? c[i-1][j] : c[i][j-1];
16             }
17         }
18     }
19 }
20 void print_lrs(char s[], int c[MAX][MAX], int i, int j){
21     if (i==0 || j==0) return;
22
23     if (s[i-1] == s[j-1] && i != j){
24         print_lrs(s, c, i-1, j-1);
25         printf("%c", s[i-1]);
26     }
27     else if(c[i-1][j] > c[i][j-1]){
28         print_lrs(s, c, i-1, j);
29     }
30     else{
31         print_lrs(s, c, i, j-1);
32     }
33 }
34
35 int main(){
36     char S[MAX];
37     int c[MAX][MAX];
38     printf("Enter the string: ");
39     scanf("%s", S);
40
41     int m = strlen(S);
42     lrs(S, c, m);
43
44     printf("\nLength of Longest Repeating Subsequence: %d\n", c[m][m]);
45     printf("LRS: ");
46     print_lrs(S, c, m, m);
47     printf("\n");
48
49     return 0;
50 }
51

```

Output:

```

Enter the string: AABCBDC

Length of Longest Repeating Subsequence: 3
LRS: ABC
PS C:\Users\Madhura\OneDrive\Desktop\C>

```

LeetCode Assessment:

Problem List

Description

Editorial

Solutions

Submissions

1143. Longest Common Subsequence

Medium

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:

Code

```
1 #include <string.h>
2 int longestCommonSubsequence(char* text1, char* text2) {
3     int m = strlen(text1);
4     int n = strlen(text2);
5     int dp[m + 1][n + 1];
6     for(int i = 0; i <= m; i++){
7         for(int j = 0; j <= n; j++){
8             dp[i][j] = 0;
9         }
10    }
11    for(int i = 1; i <= m; i++){
12        for(int j = 1; j <= n; j++){
13            if(text1[i - 1] == text2[j - 1]){
14                dp[i][j] = dp[i - 1][j - 1] + 1;
15            }
16            else{
17                dp[i][j] = dp[i - 1][j] > dp[i][j - 1] ? dp[i - 1][j] : dp[i][j - 1];
18            }
19        }
20    }
21    return dp[m][n];
22 }
```

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Problem List

Description

Accepted

Editorial

Solutions

Submissions

All Submissions

Accepted 47 / 47 testcases passed

Madhura456Mahatme submitted at Sep 21, 2025 15:29

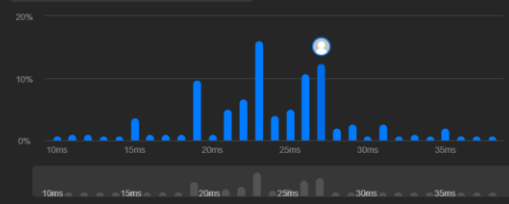
Runtime

27 ms | Beats: 31.44%

Analyze Complexity

Memory

12.52 MB | Beats: 25.42%



Code: C

```
#include <string.h>
int longestCommonSubsequence(char* text1, char* text2) {
    int m = strlen(text1);
    int n = strlen(text2);
    int dp[m + 1][n + 1];
```

Code

```
1 #include <string.h>
2 int longestCommonSubsequence(char* text1, char* text2) {
3     int m = strlen(text1);
4     int n = strlen(text2);
5     int dp[m + 1][n + 1];
```

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

text1 = "abcde"

text2 = "ace"

Output

3

Expected

3

Madhura456Mahatme

Access all features with our Premium subscription!

My Lists

Notebook

Progress

Points

Try New Features

Orders

My Playgrounds

Settings

Appearance

Sign Out