Practical 5 (DAA)

Name : Mohammad Ejaz Mehendi

Section : A3-B3

Roll no : 48

**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>

#include <stdlib.h>

typedef struct {

int length;

char\* lcs\_sequence;

} LcsResult;

int max(int a, int b) {

return (a > b) ? a : b;

}

LcsResult find\_lcs(const char\* X, const char\* Y) {

int m = strlen(X);

int n = strlen(Y);

int\*\* dp = (int\*\*)malloc((m + 1) \* sizeof(int\*));

if (dp == NULL) {

LcsResult result = {0, NULL};

return result;

}

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

dp[i] = (int\*)calloc((n + 1), sizeof(int));

if (dp[i] == NULL) {

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

free(dp[k]);

}

free(dp);

LcsResult result = {0, NULL};

return result;

}

}

char\*\* direction = (char\*\*)malloc((m + 1) \* sizeof(char\*));

if (direction == NULL) {

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

free(dp[i]);

}

free(dp);

LcsResult result = {0, NULL};

return result;

}

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

direction[i] = (char\*)calloc((n + 1), sizeof(char));

if (direction[i] == NULL) {

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

free(direction[k]);

}

free(direction);

for (int k = 0; k <= m; k++) {

free(dp[k]);

}

free(dp);

LcsResult result = {0, NULL};

return result;

}

}

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

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

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

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

direction[i][j] = 'D';

} else {

if (dp[i - 1][j] >= dp[i][j - 1]) {

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

direction[i][j] = 'U';

} else {

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

direction[i][j] = 'L';

}

}

}

}

int lcs\_length = dp[m][n];

char\* lcs\_sequence = (char\*)malloc((lcs\_length + 1) \* sizeof(char));

if (lcs\_sequence == NULL) {

lcs\_length = 0;

} else {

lcs\_sequence[lcs\_length] = '\0';

int i = m, j = n;

int k = lcs\_length;

while (i > 0 && j > 0) {

if (direction[i][j] == 'D') {

lcs\_sequence[k - 1] = X[i - 1];

i--;

j--;

k--;

} else if (direction[i][j] == 'U') {

i--;

} else {

j--;

}

}

}

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

free(dp[i]);

free(direction[i]);

}

free(dp);

free(direction);

LcsResult result;

result.length = lcs\_length;

result.lcs\_sequence = lcs\_sequence;

return result;

}

int main() {

const char\* X = "AGCCCTAAGGGCTACCTAGCTT";

const char\* Y = "GACAGCCTACAAGCGTTAGCTTG";

printf("X = %s\n", X);

printf("Y = %s\n\n", Y);

LcsResult result = find\_lcs(X, Y);

if (result.lcs\_sequence) {

printf("Final Cost of LCS (length): %d\n", result.length);

printf("LCS: %s\n", result.lcs\_sequence);

free(result.lcs\_sequence);

} else {

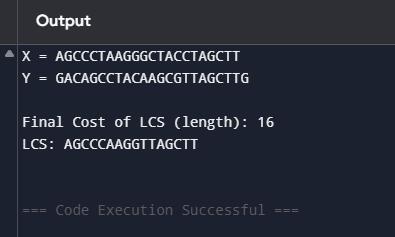
printf("Failed to find LCS due to memory allocation error.\n");

}

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 max(int a, int b) {

return (a > b) ? a : b;

}

void findLRS(char \*s) {

int n = strlen(s);

int dp[n+1][n+1];

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

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

if (i == 0 || j == 0)

dp[i][j] = 0;

else if (s[i-1] == s[j-1] && i != j)

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

else

dp[i][j] = max(dp[i-1][j], dp[i][j-1]);

}

}

int index = dp[n][n];

char lrs[index+1];

lrs[index] = '\0';

int i = n, j = n;

while (i > 0 && j > 0) {

if (dp[i][j] == dp[i-1][j-1] + 1 && s[i-1] == s[j-1] && i != j) {

lrs[index-1] = s[i-1];

i--; j--; index--;

} else if (dp[i-1][j] > dp[i][j-1])

i--;

else

j--;

}

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

}

int main() {

char s[1000];

printf("Enter a string: ");

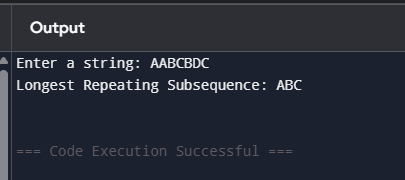
scanf("%s", s);

findLRS(s);

return 0;

}

Output :



**Task-3: LeetCode Assesment:**

**https://leetcode.com/problems/longest-common-subsequence/description/**

