**Dynamic Programming**

**Hameeduddin Imad Mohammed**

Concordia University

November 25, 2022

CSC- 543

**Self-Evaluation**

**1**. It took me just more than a week to complete the assignment.

**2**. I indulged my entire effort into coding the programs. In this assignment, I will expect an A Grade.

**3**. Every coding solution is accurate. So, I would expect an A grade.

**4**. Learning the fundamentals of C++ produced coding simple. The main issue I encountered was in running the code. As I complete all of the weekly assignments, I am becoming more precise in detecting the errors and executing the program. The overall experience was excellent.

**Dynamic Programming**

#include <iostream>

#include <vector>

#include <algorithm>

#include <time.h>

#include <iomanip>

#include <math.h>

#include <string>

using namespace std;

// g++ -std=c++11 -Wall h4.cpp

// a < h4-test.txt

//global variables

string DNA1;

string DNA2;

int n;

void printLCS(vector<vector<char>> b, string X, int i, int j)

{

// base case for length

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

{

return;

}

//cout << X[i];

if(b[i][j] == '@')

{ //diagonal arrow, MATCH

printLCS(b, X, i - 1, j - 1);

cout << X[i];

}

else if(b[i][j] == '#')

{ //up arrow

printLCS(b, X, i - 1, j);

}

else

{ //left arrow

printLCS(b, X, i, j - 1);

}

}

int longestCommonSubsequence(string X, string Y)

{

int m = X.length();

n = Y.length();

//char b[m+1][n+1];

vector<vector<char>> b(m+1, vector<char>(n+1, 0));

//char c[m+1][n+1];

vector<vector<char>> c(m+1, vector<char>(n+1, 0));

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

{

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

{

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

{ //default fill

c[i][j] = 0;

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

}

else if(X[i] == Y[j])

{

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

b[i][j] = '@'; //using @ to represent diagonal(up/left) arrow.

}

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

{

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

b[i][j] = '#'; //using # to represent the up arrow.

}

else

{

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

b[i][j] = '!'; //using ! to represent the left arrow.

}

}

}

//print the LCS

printLCS(b, X, X.length(), Y.length());

return c[m][n];

}

int main()

{

cin >> DNA1 >> DNA2;

//do stuff

string space = " ";

DNA1.insert(0, space);

DNA2.insert(0, space);

int seqLen = longestCommonSubsequence(DNA1, DNA2);

cout << endl << seqLen - 1 << endl;

}

}

**GitHub repository Link**

<https://github.com/imadmoh/bug-free-train.git>