MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE

NATIONAL TECHNICAL UNIVERSITY

«KHARKIV POLYTECHNIC INSTITUTE»

Department of Software Engineering and Management Information Technologies

Report from lab № 10

Discipline «Algorithm and Data Structures»

Executed by:

OUSSAMA BARKANE

Student Group:

1. KH201.8ei2

Checked by:

Prof M.O. Bilova

Kharkiv

2019

**THEME :** DYNAMIC PROGRAMMING

**Objective**: learn how to use dynamic programming and assess its complexity.

**Task :**

Develop a program that reads the input and solves the problem using dynamicprogramming. Determine the complexity of the algorithm.

**Variant** : 2) Find the longest common subsequence. Input: positive integers N, M (1<N, M <256) and two sequences X and Y of positive integers of length N and Mrespectively. Output: the dynamic table ( A(i , j) = length for prefixes X i and Yi) and the LCS for X and Y.

**Theory :**

Dynamic Programming is a method for solving a complex problem by breaking it down into a collection of simpler subproblems, solving each of those subproblems just once, and storing their solutions using a memory-based data structure (array, map,etc)

The longest common subsequence (LCS) problem is the problem of finding the longest [subsequence](https://en.wikipedia.org/wiki/Subsequence) common to all sequences in a set of sequences (often just two sequences). It differs from the [longest common substring problem](https://en.wikipedia.org/wiki/Longest_common_substring_problem): unlike substrings, subsequences are not required to occupy consecutive positions within the original sequences. The longest common subsequence problem is a classic [computer science](https://en.wikipedia.org/wiki/Computer_science) problem, the basis of [data comparison](https://en.wikipedia.org/wiki/Data_comparison) programs such as the [diff utility](https://en.wikipedia.org/wiki/Diff_utility), and has applications in [computational linguistics](https://en.wikipedia.org/wiki/Computational_linguistics) and [bioinformatics](https://en.wikipedia.org/wiki/Bioinformatics). It is also widely used by [revision control systems](https://en.wikipedia.org/wiki/Revision_control) such as [Git](https://en.wikipedia.org/wiki/Git_(software)) for [reconciling](https://en.wikipedia.org/wiki/Merge_(revision_control)) multiple changes made to a revision-controlled collection of files.

**Progress of the Lab**

**Code :**

import java.util.HashSet;

import java.util.Scanner;

import java.util.Set;

public class LongestCommonSubsequence {

public static int[][] LCSmatrix(String X, String Y) {

//we ignore the top most row and left most column in this matrix

//so we add 1 and create a matrix with appropriate row and column size

int m = X.length() + 1, n = Y.length() + 1;

int[][] c = new int[m][n];

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

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

//since we added 1 to row size and column size,

// we substract 1 from i,j to find the char at that index

if (X.charAt(i - 1) == Y.charAt(j - 1)) {

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

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

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

} else {

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

}

}

}

printMatrix(c);

return c;

}

public static void printMatrix(int[][] grid) {

for (int r = 0; r < grid.length; r++) {

for (int c = 0; c < grid[r].length; c++) {

System.out.print(grid[r][c] + " ");

}

System.out.println();

}

}

public static void allLCS(int[][] c, String X, String Y, int i, int j, Set<String> setLCS, String s) {

//return when either of the string length is 0

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

setLCS.add(s);

return;

}

//if last characters are equal, they belong in lcs

if (X.charAt(i - 1) == Y.charAt(j - 1)) {

//prepend the char to lcs since, we are going backwards

s = X.charAt(i - 1) + s;

//continue finding lcs in substrings X.substring(0,i-1) and Y.substring(0,j-1)

allLCS(c, X, Y, i - 1, j - 1, setLCS, s);

} // if there is a tie in matrix cells, we backtrack in both ways,

// else one way, which ever is greater

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

//continue finding lcs in substring X.substring(0,i-1)

allLCS(c, X, Y, i - 1, j, setLCS, s);

//continue finding lcs in substring Y.substring(0,j-1)

allLCS(c, X, Y, i, j - 1, setLCS, s);

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

allLCS(c, X, Y, i - 1, j, setLCS, s);

} else {

allLCS(c, X, Y, i, j - 1, setLCS, s);

}

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println(" Enter String X and Y : ");

String X = sc.next();

String Y = sc.next();

sc.close();

Set<String> set = new HashSet<String>();

allLCS(LCSmatrix(X, Y), X, Y, X.length(), Y.length(), set, "");

System.out.println(set.toString());

}

}

