**CSC 4740/6740 Data Mining**

**Assignment 2**

**Due Date: 11:59 am, Monday, September 24, 2018**

1. (100 points) Please implement the algorithm for computing edit distance.

Example 1:

sString1 = "kitten"

sString2 = "sitting"

Output in your algorithm:

Edit Distance Matrix

| s | i | t | t | i | n | g

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7

k | 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7

i | 2 | 2 | 1 | 2 | 3 | 3 | 4 | 5

t | 3 | 3 | 2 | 1 | 1 | 2 | 3 | 4

t | 4 | 4 | 3 | 1 | 1 | 2 | 3 | 4

e | 5 | 5 | 4 | 2 | 2 | 2 | 3 | 4

n | 6 | 6 | 5 | 3 | 3 | 3 | 2 | 3

Edit distance is 3

the sequence of the edits:

Step 1 : Move the character s from sitting to k in kitten

Step 2 : Move the character i from sitting to e in kitten

Step 3 : Delete the character g from sitting

Example 2:

sString1 = "GAMBOL"

sString2 = "GUMBO"

Output in your algorithm:

Edit Distance Matrix

| G | U | M | B | O

| 0 | 1 | 2 | 3 | 4 | 5

G | 1 | 0 | 1 | 2 | 3 | 4

A | 2 | 1 | 1 | 2 | 3 | 4

M | 3 | 2 | 2 | 1 | 2 | 3

B | 4 | 3 | 3 | 2 | 1 | 2

O | 5 | 4 | 4 | 3 | 2 | 1

L | 6 | 5 | 5 | 4 | 3 | 2

Edit distance is 2

the sequence of the edits:

Step 1 : Move the character U from GUMBO to A in GAMBOL

Step 2 : Add the character L to GAMBOL

Explain what you observed and whether the output results make sense or not.

The input strings make up a matrix that as they progress from the top left to the bottom right, will change in value as there are changes needed to convert the string at the top to the string on the left. If you need to add characters, moving from left to right you would add 1. If you need to delete a character, moving left to right you would subtract 1. Finally if you need to convert a character or if the character is the same, you wouldn’t change the number moving from left to right. This is done for each row in the matrix. Though, the way the algorithm constructs the matrix is that for any position in the matrix, it find the minimum value between its neighbors above, above to the left and to the left then adds one to it, unless both of the characters at the current position match, then it will not add one.

These values are the “edit distance” for each position in both the strings. The matrix that is constructed is used to find the positions of each letter that must be replaced, added or deleted to convert a string into another string. To find this construction, we start at the bottom right then move up, left or diagonally up and left, whichever value is lowest. For each movement we can tell what change took place by reversing the logic of building the matrix. If we move up, then we added the character from the row that we just moved from, if we move left, we deleted the character from the column we just moved from, and if we move diagonally and to a lower number, we know we replaced the character at the last row with the character at the last column.

As the algorithm to solve the changes is just a reversal of how we built it, and as the printed out steps that instruct the user to make will convert one string into another one, this does make sense.