CSCI 338—Algorithm Design and Analysis
Programming Assignment 6: Sequence alignment

Due: Friday, March 31, 2017

This is an individual assignment.

This assignment is to implement the sequence alignment algorithm from the textbook. To keep things simple, you don't need to adapt it to support other scoring schemes, but if you get it done quickly and want to improve it, be my guest. (It's a story as old as time.) If you do that, just be sure to save a working copy of the first version.

Here is a pseudo-code version that also makes the table of directions:

```
function editDistance(strX, strY)
input:  strX, strY – two strings to align
output: dist  – a 2D table of the computed edit distances for all prefixes
        direc – a 2D table containing the direction(s) the distance was
                computed from
allocate the two tables with the appropriate size
dist(0, 0)  ← 0
direc(0, 0) ← empty
for i ← 1 … length(strX):
   dist(i, 0)  ← i
   direc(i, 0) ← UP
for j ← 1 … length(strY):
   dist(0, j)  ← j
   direc(0, j) ← LEFT
for i ← 1 … length(strX):
   for j ← 1 … length(strY):
      distUp      ← dist(i-1, j) + 1
      distLeft    ← dist(i, j-1) + 1
      distDiag    ← dist(i-1, j-1) + if strX(i) = strY(j) then: 0 else: 1
      dist(i, j)  ← min(distUp, distLeft, distDiag)
      direc(i, j) ← empty
      if dist(i, j) = distUp   then: add UP   to direc(i, j)
      if dist(i, j) = distLeft then: add LEFT to direc(i, j)
      if dist(i, j) = distDiag then: add DIAG to direc(i, j)
return (dist, direc)
```

A couple of notes on the algorithm:

You will need to code the values in the *direc* table somehow. You could use a bitmap with the logical or operator (|), you could use a string, or you could take a different approach. If you use a string, I suggest you use one character (such as '^', '<', and '\') for each direction. (Unicode arrow characters like the one in the pseudo-code would look nicer but would be more work.)

The method or function for the algorithm will need to return two tables. Java and C++ only allow you to return a single object, so you'll need to figure out a way to bundle the two tables into a single object that you can return.

Write a program to test your algorithm that inputs two strings from the user, runs the algorithm, and prints both of the tables in readable form. It may be helpful to write a method/function or two to print the tables; the columns should be aligned, even if the entries aren't the same length. It's OK if you need to resize the window to display large tables correctly and if very large tables are just too big to display correctly.

Note that you don't have to code the actually backtracking algorithm that finds the alignments. You can do that by hand.

Test your algorithm with several pairs of strings, including identical, similar, and very different strings; strings of the same, nearly the same, and very different lengths; and strings with repeated internal sections.