

CS 5050 - Program #5 - 30 points

Edit Distance as Shortest Path Problem

Part 1 Edit Distance (15 points)

Problems in molecular biology involve finding the minimum number of edit steps which are required to change one string into another.

There are four types of edit steps: insert, delete, replace, and match. The costs of each operation can change according to the specifications for the problem.

Example: abbc → babb

abbc → bbc → bbb → babb (3 steps)

abbc → babbc → babb (2 steps)

We are trying to minimize the number of steps. Assume matching has a cost of 0, insert is 1, delete is 1, and replace is 1, we can change “do” into “redo” in 2 steps as shown below in the array set up for dynamic programming.

	*	r	e	d	o
*	I-0	I-1	I-2	I-3	I-4
d	D-1	R-1	R-2	M-2	I-3
o	D-2	R-2	R-2	R-3	M-2

Do the following:

1. Allow the user to input the cost of replacement
2. Input two strings
3. Display the edit distance table of results (similar to the one above)
4. Display the series of operations that should be performed. For the above example, the operations would be:
 - insert “r”
 - insert “e”
 - match “d”
 - match “o”

Part 2 Edit Distance as Shortest Path Problem (15 points)

Now solve the same problem another way. There is an interesting way of mapping the edit distance problem into a single source shortest path problem. (Note, a single source shortest path problem is quicker to solve than an all points shortest path.) You think of the problem laid out in a matrix in which the cell $C[i,j]$ represents the problem of converting the first i characters of string1 to the first j characters of string2. In the shortest path version, the edges represent the cost of going from one cell of the matrix to the next.

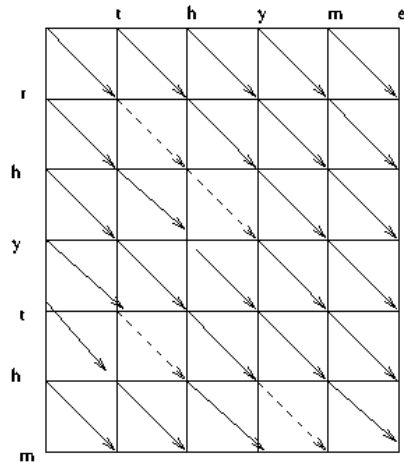
We are trying to convert string1 to string2 for the cheapest cost. Allow the user to input his/her costs for each of the edit steps. For the example shown below, we will use an insertion cost of 1, a deletion cost of 1, and a replacement cost of 2.

The picture below may be helpful:

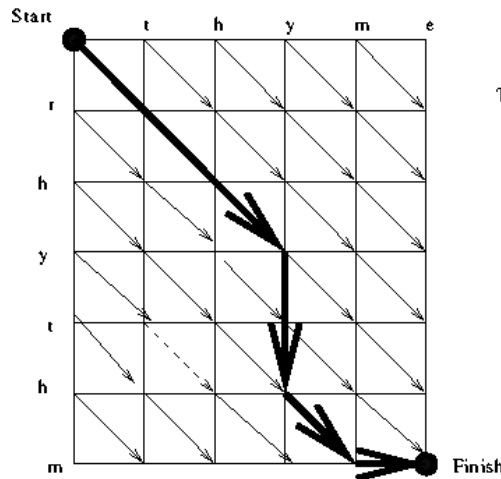
Write the edit distance program using a shortest path solution. To save you a little time, the code to do a shortest path is included. The code `SP.java/Node.java` is an implementation of shortest path which relies on the fact that you have an acyclic graph. Feel free to use a different shortest path algorithm if you desire.

Anytime you convert from one problem to another you have to make sure you do the conversion properly. It may be a little confusing. In a normal shortest path (SP) problem, the nodes are labeled $0, 1, 2, 3, \dots, n$. However, when you do the edit distance problem, your nodes are labeled with $[i][j]$ pairs. I would recommend having a function which maps nodes labels of the form $[i][j]$ into the integer form expected by the SP problem.

Edit distance problem for changing Rhythm to Thyme



Horizontal lines represent insertion and are cost 1
 Vertical lines represent deletion are are cost 1.
 Dotted diagonal lines represent matching and are free
 Solid diagonal lines represent replacement and are cost 2.



The path marked is of cost 5.

Note, you are to use two different algorithms (shortest path and edit distance), **but keep them separate. This is the most important part of making this assignment doable.** In solving a problem which utilizes algorithms A and B, do not attempt to develop a combination of the two (algorithm BA). It is much easier to read, modify, and reuse if the algorithms are kept separate. In other words, utilize the shortest path algorithm in the original way it was written. Do not re-write it (except for possibly trivial ways) to accommodate the edit distance algorithm. The shortest path code should be general purpose, rather than creating a variant of it that only works in this special case.

Input

Try your program on the following sets of data:

dog day
elephant zebra
cat camera
run errand
rigging grinning
really real
integrity category
mycetosma comatosely

Output

Display the edit distance and the series of operations which converts one string to the other. As an aid to the grader (and as a debugging help), print messages as you do the shortest path problem. Every time a node gets a new distance, print an informative message.

Starter code has been provided, but you are not required to use it.

ReadMe File (to be submitted with your code)

1. How many hours did you spend on this assignment?
2. Anything the grader needs to know in running your code?

Point Value

- Correct Output - 15 points (7 points dynamic programming, 8 points shortest path)
- Coding, documentation, design - 15 points
 - insufficient Comments - 1 point
 - insufficient variable explanations - 1 point
 - magic numbers - 1 point
 - awkward, non-extendible, or confusing code - 2-3 points
 - totally missing parts - 1-5 points (You will be docked in two places - once because the output is wrong and again because you didn't even try that part. We will adjust the points so that the total of the two pieces corresponds to the portion of the work you omitted. Thus, if you didn't do a third of the assignment, you will miss a third of the points.)