COS 226 Programming Assignment Checklist: Map Routing

Special collaboration policy: for this assignment, you are permitted to work jointly with one classmate. If you choose to do so, you and your partner should submit your code jointly (under one login name only), but each of you should submit your own `readme.txt` file.

Frequently Asked Questions

What's the correct Euclidean update formula again? Update `wt[w]` to be the sum of `wt[v]` plus the distance from `v` to `w` *plus* the Euclidean distance from `w` to `d` *minus* the Euclidean distance from `v` to `d`. See Sedgewick 21.5.

What if there are multiple shortest paths? Print out any one you like.

What's a good way to check my answers? Use the client program `Distances.java` which only prints out the shortest path distances. You can `diff` the results against the ones produced by our bare-bones solution. Because of roundoff error (e.g., when taking and summing square roots), don't be alarmed if you are off by 0.0000000001 or so.

What do you mean by sublinear time? Each shortest path query should take time substantially less than E or V. In particular, you cannot even afford to explicitly re-initialize the V weights or the V entries of the priority queue before each query. Any loop that goes from 1 to V will ruin the sublinearity and be subject to a significant deduction.

Don't the arrays `dist[]` and `pred[]` get re-initialized to 0 each time `dijkstra(s, t)` is called? Yes, each time you call `new`, this is what happens. In the bare-bones version this is done each time `dijkstra(s, t)` is invoked. If you plan to reuse the arrays from one invocation to the next, you must do something else. For example, use a private class instance variable to retain the information from one invocation to the next, and allocate it in the constructor.

How can I profile my code? Use something like `java -Xprof Paths usa.txt < usa-50000short.txt`.

What should I do if the two vertices are not connected? Print that their distance is "infinity" or print out that they are not connected.

The client ShortestPath.java only processes one query pair. Am I doing anything wrong? No, this client is mean for visualization and you don't want to draw dozens of paths on top of each other. Use Paths.java or Distances.java for testing multiple query pairs. Your algorithm's efficiency will only be apparent when you process lots of query pairs simultaneously.

I can't see the blue dots that are supposed to represent the vertices that the algorithm examines. Sorry, we didn't yet implement this. It won't be very interesting for the bare bones implementation sicne it explores all the reachable nodes.

How fast should my algorithm be? Significantly faster than the bare bones implementation.

Input, Output, and Testing

Input and output. Here are a number of sample input files.

Extra credit. Create your own map file. If it's sufficiently interesting, we'll award extra credit and post them for other students to use. It can be of the US, Princeton, or any other recognizable region. For example, delete all roads in Nebraska from the US map (perhaps, for drivers who have an unpaid speeding ticket in the state).

Submission and readme

Use the following template readme.txt file.

Getting started

- Download the directory `map`. This directory is mirrored on arizona at `/u/cos226/pub/map`. It contains a bare bones implementation and sample input files.
- Compile and execute our bare bones implementation with

  ```
  % javac ShortestPath.java
  % java ShortestPath usa.txt < usa-1.txt
  ```

  The client `ShortestPath` peforms a single shortest path query on the given input file using the first pair of integer from standard input. Note that there are two input streams in play: the map comes from the file specified in the command line argument, whereas the queries come from standard input.
- When processing a single shortest path query, the bare bones implementation is essentially optimal since the majority of time is spent reading in the input and plotting it. The client program `Paths.java` processes many shortest path queries and prints the results as text instead of turtle graphics.

  ```
  % javac Paths.java
  % java Paths usa.txt < usa-5000short.txt
  ```

  The program is painfully slow. Your goal is to optimize `Dijkstra.java` so that you get the same output, but faster.