Computer Science 501
Data Structures & Algorithms
The College of Saint Rose
Fall 2013

Programming Project 3: Dijkstra's Road Trip

Due: 4:00 PM, Monday, December 9, 2013

For your last programming project, you will be working with some real world data derived from highway systems. A big advantage of working with this kind of data is that it has a connection to reality, and that we can visualize the data and the results of our manipulations of that data with the Google Maps API. This data is collected by the Clinched Highway Mapping (CHM) Project (`http://cmap.m-plex.com/`). I have taken some of the data from the CHM collaborators and converted into a format that is more convenient for us to load into a graph structure and use. Much more about the project is available at `http://courses.teresco.org/chm/`, but everything you need to know should be on this sheet.

You may work independently or in groups of size 2 or 3 for this project.

The Data

The data is in "`.gra`" files which have the following format:

- The first line consists of two numbers: the number of vertices, $|V|$, (we'll call them "waypoints") and the number of edges, $|E|$, (road segments that connect adjacent waypoints).
- The next $|V|$ lines descibe the waypoints. Each line consists of a string describing a waypoint, followed by its latitude and longitude as floating-point numbers.
- The last $|E|$ lines describe the road segments. Each line consists of two numbers specifying the waypoint numbers (0-based and in the order read in from this file) connected by this road segment, followed by a string with the name of the road or roads that form this segment.

You can find a few dozen example graph files linked from `http://courses.teresco.org/chm/graphs.html`. For example, `usai.gra` describes the entire U.S. Interstate Highway system. `canyt.gra` describes a much smaller system: the territorial highway system in the Yukon. The links of most interest to you are the "download" and "view" links.

Programming Part 1

Your initial tasks are designed to familiarize you with the graph data and the structure package's graph implementations.

Your overall approach will be to develop a Java program or programs that can read in graph data, store it appropriately in memory, and perform a variety of operations on that data.

- Come up with an appropriate graph structure (something that holds the appropriate data in the vertex and edge labels) to hold this data and write code to construct one from a given `.gra` data file. You are strongly encouraged, but not required, to use the graph implementations from Java Structures. Big hint: labels need not be simple objects like `String`s or `Integer`s. You can use any object type (including those you define yourself) for those labels. (9 points)
- Print out, in a nice format, a list of all waypoints. (3 points)
- Print out the northernmost, southernmost, easternmost, and westernmost waypoints in the graph. (3 points)
- Print out the shortest and longest road segments. (3 points)

Programming Part 2

Your second major task is to add a capability to your program to be able to find and report the n longest or shortest edges in the graph.

For example, with the `canyt.gra` graph, the 10 shortest edges:

```
YT6@MacBoatLau to YT6@+X646670 via YT6 length 0.0000
YT1@MorLakeRS to YT1@BC/YT via YT1 length 0.0936
YT1@BeaCrkAir to YT1@CanCus via YT1 length 0.1077
YT2@DukeSt to YT2@GeoBlaFry via YT2 length 0.1979
YT4_S/YT6_S to YT4@OldCanRd&YT6@OldCanRd_S via YT4,YT6 length 0.2299
YT6@+x822 to YT6@+x116 via YT6 length 0.2522
YT6@+X297007 to YT6@+X720169 via YT6 length 0.2704
YT4@+x88 to YT4@+x87 via YT4 length 0.2739
YT2@+X456835 to YT2@CliAgaRd via YT2 length 0.3073
YT6@+X401385 to YT6@+x771 via YT6 length 0.3446
```

and the 10 longest edges:

```
YT1@+X372730 to YT1@+X931620 via YT1 length 10.3368
YT1@+X687758 to YT1@+X612218 via YT1 length 9.8789
YT3@+X260467 to YT3@+x898989 via YT3 length 9.0087
YT2@+X558217 to YT2@GraRd via YT2 length 8.2056
YT1@+X336247 to YT1@+X858279 via YT1 length 7.4635
YT4@+x48 to YT4@+X800213 via YT4 length 7.1541
YT5@+X717826 to YT5@+X920982 via YT5 length 7.0805
YT1@CanCus to YT1@+X680719 via YT1 length 6.9887
YT2@HunCrkRd to YT2@BonCrkRd via YT2 length 6.9588
YT6@+X620794 to YT6@LapCanTr via YT6 length 6.8125
```

A format similar to the above would be appropriate for text output. You will earn 12 points for producing correct output similar to the above.

One way to accomplish this is to create a list of all edges and sort them by edge length. But we can do this more efficiently. For 5 points, compute the set of longest/shortest edges in $\Theta(n|E|)$ time, where n is the number of longest/shortest edges you are looking for and $|E|$ is total the number of edges in the graph. See the idea of a "BestOf" structure as described in the lab exercise at the end of Chapter 11 (p. 275) of Bailey.

One of the major motivations for using the highway graph data is the ability to visualize our results with the Google Maps API. For 5 points, you are to implement a second option, where the lists of longest or shortest edges are written to a file in a particular format. Here, each edge in the set of results should be specified by placing the vertex information for each of its endpoints on consecutive lines of a file. For example, the 3 shortest edges from `canyt.gra` would be specified as:

```
YT6@MacBoatLau 62.86788 -130.82806
YT6@+X646670 62.868138 -130.827901
YT1@MorLakeRS 59.998729 -132.116818
YT1@BC/YT 60.000075 -132.117087
YT1@BeaCrkAir 62.407424 -140.860358
YT1@CanCus 62.40891 -140.85935
```

These files should be given a `.nmp` extension. Once such a file is created, it can be visualized by directing a browser at `http://courses.teresco.org/chm/viewer/` and uploading the `.nmp` file in the file selection box at the top of the page.

Programming Part 3

Your final programming task is to develop a simplified "driving directions" system based on the mapping data you have been working with.

You should use a variant of Dijkstra's Algorithm to compute shortest path from a given starting point (a graph vertex) to a given destination point. The general form of Dijkstra'a Algorithm computes the shortest paths from a starting vertex to all other vertices, but you will be able to stop one you find a shortest path to the specified destination rather than calculating the shortest path to all other places. You will also need to make sure that you can efficiently print/write the computed route in the proper order (starting point to destination point).

Once a shortest path is computed, you will need to be able to output it in a human-readable form or in a form plottable by the Highway Data Examiner.

For example, if you load the `ny-all.gra` file, and compute a shortest path for a few nearby points: `US20@WesAve` (the "Y" intersection at Western and Madison right near campus) and `NY2/US9` (Latham Circle), your path would traverse the following points:

```
US20@WesAve, NY443/US9W@US20&US20@US9W, NY5/US9W, US9/US9W
I-90@6/US9, NY377/US9, NY378/US9, NY155/US9 and NY2/US9.
```

Your "human readable" output might look something like this:

```
Travel from US20@WesAve to NY443/US9W@US20&US20@US9W
 for 1.56 miles along US20, total 1.56
Travel from NY443/US9W@US20&US20@US9W to NY5/US9W
 for 0.37 miles along US9W, total 1.93
Travel from NY5/US9W to US9/US9W
 for 0.28 miles along US9W, total 2.21
Travel from US9/US9W to I-90@6/US9
 for 0.87 miles along US9, total 3.09
Travel from I-90@6/US9 to NY377/US9
 for 0.44 miles along US9, total 3.53
Travel from NY377/US9 to NY378/US9
 for 2.04 miles along US9, total 5.57
Travel from NY378/US9 to NY155/US9
 for 2.24 miles along US9, total 7.81
Travel from NY155/US9 to NY2/US9
 for 0.78 miles along US9, total 8.59
```

Your plottable data for the Highway Data Examiner should be in a "`.pth`" file. This file format must match the following:

```
START US20@WesAve (42.666502,-73.791776)
US20 NY443/US9W@US20&US20@US9W (42.652458,-73.767786)
US9W NY5/US9W (42.656734,-73.763301)
US9W US9/US9W (42.659938,-73.759975)
US9 I-90@6/US9 (42.669562,-73.748817)
US9 NY377/US9 (42.675873,-73.747659)
US9 NY378/US9 (42.704925,-73.754568)
US9 NY155/US9 (42.736832,-73.76225)
US9 NY2/US9 (42.748115,-73.761048)
```

Here, each line describes one "hop" along the route, consisting of the road name of the segment (*i.e.*, your edge label), the waypoint name (*i.e.*, the label in your vertex), and the coordinates of that point. The exception is the first line, where we substitute `START`, since you don't have to take any road to get to your starting point.

These files should be given a `.pth` extension. Once such a file is created, it can be visualized by directing a browser at `http://courses.teresco.org/chm/viewer/` and uploading the `.pth` file in the file selection box at the top of the page.

This part will be graded out of 15 points.

Submission

Before 4:00 PM, Monday, December 9, 2013, submit your Java program(s) for grading. Do this through Submission Box at `http://sb.teresco.org` under assignment "Maps". Since SubmissionBox can only accept one file per assignment, please package up your files in a "zip" or similar archival format first.

Grading

This assignment is graded out of 75 points, broken down as follows:

Grading Breakdown	
Java code style, documentation, and formatting	15 points
Graph data structure construction	9 points
List waypoints	3 points
Print N/S/E/W-most waypoints	3 points
Print single shortest/longest segments	3 points
Print n shortest/longest segments	12 points
Find n shortest/longest in $\Theta(n\|E\|)$ time	5 points
`.nmp` output for shortest/longest segments	5 points
Human readable driving directions from Dijkstra's Algorithm	15 points
Plottable `.pth` of driving directions	5 points