# Programming Homework 3 (120 points)

Data Structures and Algorithms in Java
Shlomo Hershkop
Department of Computer Science
Columbia University
Fall 2013

START EARLY…Due Dec 9

This assignment can be done in a group of 2, but you need to sketch out EXACTLY who is doing what and email me before Nov 10….

You will be coding the backbone of a map travel system. The goal would be to model air travel simulation system which allows you to plan trips between any city in the world based on some cost to travel between nodes. Like in the real world, not every city has a connection to every other city.

As in the real world, you are free to do what you want while meeting the overall goals of the homework. This homework is for practicing algorithms and graph programming, which means you need to create your own graph/graphnode class and anything else to support your program. Comments will count towards your grade…so please make use of them as needed. You can utilitze java's built in DS for hashtables, lists, and arrays, everything else you need to build your own or adopt from the book's code.

You will see a file called worldcities.txt on courseworks which contains the GPS coordinates of 9117 cities around the world. The format of the file is :
N (number of cities)
city comma state
Lat
Long
etc

You will create an air flight path system and costs between cities using the following algorithm:

Once you load up all your cities, for each city, decide how many directed connections it has ( between 2-8 connections (randomly)) and assign it to any other random cities. For example: for city n, we randomly choose 4, which means you add 4 directed edges to 4 randomly chosen cities starting at this one (directed edges). And so on. For each connection choose a random weight between 100-2000. You will need to check for duplicate edges so that for any two city pairs, only one directed edge exists in a specific direction.

Note: the names of the cities might have spaces (and some might be missing state information, in which case just copy from city name). In addition I will provide a short test file called fict100.txt which has a sample of 100 fake cities. (the format is the same). Do not hard code the number of cities, but rather have your program read in the number on the first line to know how many cities to process.

Extra credit for creating a graphical front end to the system (2o points) and generating graphical output (ask me about graphing classes etc). There are many libraries for generating nice graphs.

Your README needs to list all the files involved with notes on each (if you wrote it or took from book etc) and runtimes for all algorithms, which you are running on your graph and in the system.

**Important**: The main java file with main needs to be called **mainf13.java** and the main graph class needs to be renamed **MyGraphMap13.java**

1) You will need to present the user with the following menu items
    a. Load up a city file into the system
    b. Search for state and list all cities there with their in/out counts.
    c. Search for city and display some information about it.
    d. Set current city as the starting point.
    e. Show current city
    f. Find n closest cities to current city using gps distances.
    g. Find n closest cities to current city using directed edge costs
    h. Find shortest path between current and some target city
    i. quit

    Here is an explanation of the above menu choices

    a. Load up file
    will allow the user to add in a file to the current set of data, it should give the user the option of removing all the current information (in case you want to remove old graph) and should detect so that the same city file isn't added twice (need to remember the name of the file not contents). Also some of the cities are missing state info, so treat the single label as both city and state.
    b. Search for state
    will return all cities associated with the specific state. For example if the user enters 'b', prompt for state, and if they enter "New York" show all cities in new york…..along with some ID number (which you create) and in and out counts from the directed graph

c. Search for city
   will return the ID number and some information of the specific city if its in the system

d. Set current city
   will take an ID and remember it as the current city

e. Show current city
   will print the currently set city's information

f. Find n closest cities using gps distances (ignoring directed edge weights but using directed edges to relate cities)
   for calculating gps distances see your favorite search engine
   n is provided by the user

g. Find n closest
   find the n closest cities from the current city, if the current has not been set choose a random city as current, n is a number provided by the user. You will use the randomly generated directed edge weights to measure close-ness.

h. Find shortest path
   will take an ID and calculate all shortest hops from the current city to the destination city, finding the shortest path by the directed weight. Feel free to reuse dijkstra's algorithm from the book. You must credit if you use the books code vs writing it on your own.

i. Quit

   Have fun and please specify in your README and comments which algorithms you are implementing and their expected runtimes….
   note: if you see yourself coding a $n^{10}$ algorithm…find a better one

**IMPORTANT2**:

In order to run tests, your program should also be able to take an input file and process the menu choices from the input file. So if I run "java mainf13" It should present the menu system but if I run "java mainf13 input.txt" it should read the input for the menu's one line at a time and process it as if I typed in the menu's on the command prompt. You will need to check the number of command line args to be able to do it.

For example the input.txt could be:

```
a
worldcity.txt
d
35
f
4
g
20
i
```

(i.e load up worlcity.txt set city 35 as current, show 4 closest gps cities and 20 closest distance cities.)