CSCI 241 Data Structures

Programming Assignment 2

Electronic Turn-in due: 10pm, Monday Nov 16th, 2015.

In the assignment, you will develop a graph implementation and then use it to implement your version of a preferred travel route algorithm. In particular, you will design a travel algorithm that satisfies multiple constraints (e.g., cost of travel, distance, travel time).

Part I. Graph Representation

In this part of the assignment you will implement a graph representation.

The assignment has these files provided:

- Graph.java - Graph interface.
- Vertex.java - Vertex class.
- Edge.java - Edge class. This file will also contain information about the constraints.
- MyGraph.java - Implementation of Graph interface, you will need to fill in code here.

You will add your code to the MyGraph.java file that implements the Graph.java interface. Graph.java will make use of the Vertex.java and Edge.java classes provided for you, although you should feel free to add methods to Vertex.java or Edge.java files. **Please do not modify Graph.java file**. Your code should be correct and efficient and strive for the minimal big-Oh possible for the required operations.

Part II. Graph Algorithms

In this part of the assignment, you will use the graph representation you created to read the description of a graph and answer questions about it. Your task would be to answer the following questions according to user's preferred travel choice:

1. What is the best travel route in terms of distance? (shortest distance is preferable)
2. What is the best travel route in terms of monetary cost? (cheapest route is preferable)
3. What is the best travel route in terms of travel time? (minimum travel time is preferable)

Development and Testing

A program named Routes (Routes.java) will be provided to you. **You need to extend this file to print or display the graph**. This program drives methods written in other files such as reads in

vertex and edge files, creates a graph and prompts user for vertices to find desired travel route between two vertices.
The format of the input files are as follows:

- Vertex: This file has one line per vertex and each line contains a text string with the vertex name.
- Edges: This file has five lines per directed edge.
  - The first line gives the starting vertex.
  - The second line gives the ending vertex (the one being pointed to).
  - The third line gives you the travel cost
  - The fourth line gives you the travel distance
  - The fifth line gives you the travel time

You may assume that every node name that appears in a path (edge) also appears in the vertex list. Note that since data files are expected to represent *directed* graphs, the existence of an edge from a to b does not guarantee the existence of an edge from b to a (unless both edges exist in the data file), nor does it guarantee that the weight of the edge from a to b is the same as the weight of the edge from b to a. You should check for validity of the cost values provides (e.g., negative travel time or distance or cost are not allowed).

Two sample data files will be available to you representing information about airports and the cost of a flight between them. File **vertex.txt** contains a list of 3-letter airport codes, each of which represents a vertex in a directed graph. File **edge.txt** contains information about the directed edges in the graph and different parameters associated with them.

- When the program begins execution, it will read the two data files and create a correct representation for the input graph. You should print or display the graph.

- Once you construct the graph, the program should loop repeatedly and allow the user to ask for information about travel routes. The user should enter two vertex names and one number specifying which parameter s/he wants to optimize for this travel (1 = shortest distance, 2 = cheapest route, 3 = fastest time, 4 = all options). If there is a route from the first to the second vertex, the program should calculate the shortest, cheapest and fastest route, including the vertex names on the path and the total length, cost, and time). If there is no such path, a descriptive message should be printed (i.e., if either name is not a vertex in the graph or if there is no route between the given vertices).
  Examples: If the user enters `BEL HOU 2`, then the output might look like `BEL SEA HOU 750` if the shortest path from `BEL` to `HOU` runs through vertex `SEA` and has a total cost of 750.

Extra Credit

Extra credit will be awarded for the following extensions.

- (5 points) Display a list of top 3 possible routes for all three parameters.
- (5 points) Graphical representation (any of the following):

- Display the graph as a drawing;
- allow the user to select endpoints by clicking on the screen;
- highlight the path visually;
- Design a website,
- Coloring the paths in the graph representation.

The Repository

All code for this program will be developed under your Subversion repository. If you did not create a repository or check out a working copy for Assignment 1, please follow the instructions on the Assignment 1 description pdf to create a repository and check out a working copy. In your working copy, create and add a directory named prog2 (capitalization, spacing and spelling matter!); this is where your work for this assignment will be stored. The first thing you will want to do is to download and svn add the file in the prog2 sub-directory in your working copy.

You must use this repository for the development and submission of your work (failure to have 8 or more non-trivial revisions will result in potentially large penalties). Verify that you are able to access your repository as soon as possible and contact both **cs.support@wwu.edu** and **me** if there is a problem.

Points

This assignment will be scored by taking the points earned and subtracting any deductions. You can earn up to 50 points:

Component	Points
Write Up & Test Cases	5
MyGraph	7
loadVertices	5
loadEdges	5
findAdjacentVertices	5
checkIsAdjacent	5
findRoute	10
Code for Displaying the Graph	8

Submitting Your Work

By 10 PM on the due date, a script will automatically check out the latest committed version of your assignment. (Do not forget to commit the work you want submitted before the due date!) Your repository should have in it, in the prog2 directory, at the least:

1. Vertex.java
   Edges.java
   Graph.java
   MyGraph.java

All other files that are needed to compile and run your program.

2. Your write-up
3. Your test files (at least two different test files)
4. A brief description of work completed for extra credit (optional)

Your repository need not and should not contain your .class files. Upon checking out your files, I will compile all .java files, run it against a series of test graphs, analyze your code, and read your write up.

Write-Up & Test Cases

In one or two pages, provide a write-up of your implementation. Please submit your write-up as a plaintext file named writeup.txt. Your write-up should include the following points:

1. Your name
2. An acknowledgement and discussion of any parts of the program that are not working. Failure to disclose obvious problems will result in additional penalties.
3. An acknowledgment and discussion of any parts of the program that appear to be inefficient (in either time or space complexity).
4. A discussion of the portions of the assignment that were most challenging. What about those portions was challenging?
5. A discussion on how you approached testing that your program was correct and asymptotically efficient. What did test1.txt test? What did test2.txt test?

Academic Honesty

To remind you: you must not share code with your classmates: you must not look at others' code or show your classmates your code. You cannot take, in part or in whole, any code from any outside source, including the internet, nor can you post your code to it. If you need help from other students, all involved should step away from the computer and discuss strategies and approaches, not code specifics. I am also available via email (do not wait until the last minute to email). If you participate in academic dishonesty, you will fail the course.