CSCI 241 Data Structures
Winter 2017
Programming Assignment 2
Total Point: 40 (15% of course grade)
Due: Friday, Feb 17th 2017, 10 pm
Pair Team Programming
Late work will not be accepted

In the assignment, you will develop a graph implementation and then use it to implement a much simplified version of kayak/Google Flights. In particular, you will design a travel algorithm that satisfies a given constraint (e.g., suggests travel paths based on user preference in terms of shortest distance, travel time, and/or cost).

Part I. Graph Representation

In this part of the assignment, you will implement a graph representation. You will read data about vertices and edges and construct a graph based on that. You will also display the graph.

Part II. Graph Algorithms

In this part of the assignment, you will use the graph representation you created to answer questions about it. Your task would be to answer the following questions:

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

Development and Testing

You will require at least the two following java files:

- MyGraph.java: This file is used to create the graph representation and construct the desired travel path.
- Routes.java: This file will drive methods written in MyGraph.java file. For example, it will call methods to read vertex and edge files, create a graph and prompt 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.
- Edge: 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 distance
    - The fourth line gives you the travel time
    - The fifth line gives you the price of ticket

You may assume that every vertex that appears in an 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 (**vertex.txt and edge.txt**) will be available to you representing information about airports and the cost of travel (distance, time, and price) 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/display the graph.

- Once you construct the graph, the program should loop repeatedly and allow the user to ask for information about desired 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 = fastest time, 3 = cheapest flight, 4 = all options). If there is a route from the first to the second vertex, the program should calculate the shortest, fastest, and/or cheapest routes, including the vertex names on the path and the total cost, length, 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 250 if the fastest path from BEL to HOU runs through SEA and has a total travel time of 250 minutes.

Submitting Your Work

When the clock strikes 10 PM on the due date (**Friday, Feb 17th 2017**), we will check out the latest submitted version of your assignment from Canvas. (Do not forget to submit your latest working version before the due date!) Your submission should include, at the least:

1. MyGraph.java
2. Routes.java
3. All other files that are needed to compile and run your program.
4. Your write-up.
5. Your test files (at least two different test files)

You should not submit your .class files. Upon checking out your files, we will compile all .java files, run it against a series of test graphs, analyze your code, and read your write up.

Points

This assignment will be scored by taking the points earned and subtracting any deductions. You can earn up to 40 points:

Component	**Points**
Write Up & Test Cases	5 (3+2)
Contributions Summary	2
loadVertices	2
loadEdges	2
displayVertices	2
displayEdges	2
displayGraph	2
findAdjacentVertices	4
checkIsAdjacent	4
findShortestRoute	5
findCheapestRoute	5
findFastestRoute	5
Total	40

Your code should be correct, clean and efficient.

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.

Contribution Summary

In an email, provide a two-paragraph write-up of your and your partner's contribution to this assignment. Please submit your write-up as an **email to both instructor and TA.** Your write-up should include the following points:

1. Your names
2. One paragraph explaining your contribution to this assignment. Please include examples of methods you implemented, how you contributed to the design, bug fixing efforts, etc.
3. One paragraph explaining your partners' contribution to this assignment. Please include examples of methods your partner implemented, design ideas, bug fixes, etc.
4. Any collaboration problem experienced and how you approached the problem and solved it.

Academic Honesty

To remind you: aside from your designated partner, 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 and your partner need help from another pair, all involved should step away from the computer and discuss strategies and approaches, not code specifics. I am available for help during office hours, as are department tutors, but you should attend these hours with your partner. I am also available via email (make sure you and your partner are included in the email and do not wait until the last minute to email). If you participate in academic dishonesty, you will fail this course.