COMP 282 Summer 2019 **Project 3**

Due Tuesday, July 9th 2019 23:55

Requirements: To complete this project you will write and submit 1 file: `Project3.zip`

Project3.zip is a zip-file archive containing **all of your source code, as .java files**. **If you are doing extra credit portion, you must include your SQLite file as well!** You do not need to include .class files in your Project3.zip. You must also include a plain-text file inside the zip-file named `Readme.txt`

Each of these files will be created as an ASCII file (i.e. a plain text document with a .java or .txt extension). You may create your source code and `Readme.txt` with any editor or IDE but you must ensure that they are plain text files. In other words, they should not contain anything except ASCII characters.

`Readme.txt` must contain your answers to the questions at the bottom of this assignment. See section named "QUESTIONS FOR README.TXT"

NOTE: you may use any standard Java libraries for this assignment, but you must implement the weighted undirected graph data structure and all graph algorithms (BFS, DFS, and MST) on your own. You may not use any Java library, standard or otherwise, nor code from the internet for the graph implementation, although you can use linked list, array, vector, set, queue, and stack data structures in the standard Java library as building blocks for this assignment.

IMPLEMENTATION

Create an undirected, weighted graph as an adjacency list. Every edge is of the form:

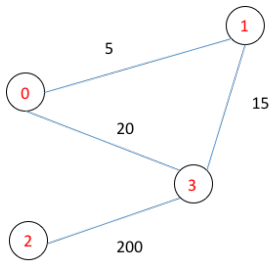
(vertexA, vertexB, weight)

where (12, 15, 25) is an edge between vertex 12 and vertex 15 of weight 25.

Read a graph from a user as a list of edges. This can be read from an input file, from command line, or from a command prompt.

Example input: **(0, 1, 5), (1, 3, 15), (2, 3, 200), (0, 3, 20)**

That would produce this graph…



Execute the following algorithms on the input graph:

- Breadth First Search, starting at first vertex given (in previous example, that was 0, the first vertex of the first edge)
- Depth First Search, starting at first vertex given
- Minimum Spanning Tree, using Prim's algorithm

QUESTIONS FOR README.TXT

1) What, if anything, does not work in this code?
2) What, if anything, is incomplete in this code?
3) Show that this code works across 5 graph inputs by printing, for each of the 5 inputs
    - the given graph (as a list of edges)
    - the Breadth First Search tree, as a list of edges, printed in order
    - the Depth First Search tree, as a list of edges, printed in order
    - the Minimum Spanning Tree, as a list of edges

EXTRA CREDIT (worth 10% of additional points on this Project 3 assignment)

Store all graphs (as given input string) in a local SQLite database, using the JDBC driver for SQLite. You can create the SQLite database file and its schema outside of your Java program, using the sqlite3 app, as shown here: https://www.sqlite.org/quickstart.html

You can use the JDBC to insert/query the graph database, using instructions and sample code provided here: http://www.sqlitetutorial.net/sqlite-java/insert/

If you perform this part, you must package your SQLite database file with your zip file.

You must also modify your application to have a very simple menu, which allows a user to…

- list all of the graphs in the database
- enter a new graph
- save a new graph in the database
- load any graph in the database

Hint: you can store graphs in a table Graphs, which has an auto-incrementing graphId column. When querying this table Graphs, you can order results by graphId. You can then use the graphId to display/fetch a given graph based on the user's choice.

SUBMISSION

Prior to the deadline upload your file `Project3.zip` to Canvas. Do not submit any other documents!

GRADING CRITERIA

10%: Submission instructions followed to the letter (1 zip file named Project3.zip, containing appropriate Readme.txt and all required .java files for compilation – including your SQLite file if performing the extra credit)

20%: Readme.txt answers questions

30%: All source code compiles and executes as specified

40%: Source code looks proper and matches specification

A Word On Cheating: This project is an individual project. You can discuss this project with other students. You can explain what needs to be done and give suggestions on how to do it. You cannot share source code. If two projects are submitted which show significant similarity in source code then both students will receive an F in the course. Note a person who gives his code to another student also fails the class (you are facilitating the dishonest actions of another).