Institute of Technology, University of Washington Tacoma
TCSS 342 Data Structures
Assignment 5 – Graphs

Due: Thursday 14 March 2019, 23:59:59

Purpose:

This assignment has a number of purposes. It is intended to provide practice with the following:

- terminology used in graph theory
- adjacency matrix and adjacency list representations of Graphs
- depth-first and breadth-first search on graphs
- practice with basic graph algorithms
- familiarization with advanced graph algorithms

Assignment Description:

This assignment is in 2 parts. Starter code for this assignment is provided on Canvas.

Part 1: You will complete the implementation of DisplayCostsMain.java such that it displays some statistics about a graph and the shortest path distance between two cities in the graph. The program builds a graph using data read from a text file and then prompts the user to select start and end points for a path through the graph. Most of the implementation is already provided. You are **NOT** required to write unit tests; do so if it helps you.

Graphs use many data structures internally. For this reason, there are 27 classes included in the starter code (This is NOT code from the textbook we are using this quarter, this code is from another data structures textbook by Simon Gray).
5 of the classes (DisplayCostsMain, FileIO, Tuple, HeapPriorityQueue, and EmptyPriorityQueueException) are provided by the instructor. Note that these classes only generate a few warnings from the Eclipse plugin tools. 3 of the classes (Graph, AdjMatrixDiGraph, WeightedAdjMatrixGraph) are provide by Simon Gray but were modified by the instructor to include new methods. All other classes were written by Simon Gray and are mostly unchanged (and therefore generate MANY warnings). 4 unit test classes are provided. These cumulatively include 63 tests which all currently pass. You should avoid writing code that generates warnings from the Eclipse plugin tools or that causes any of the existing unit tests to fail. You are **NOT** required to correct any of the starter code to avoid warnings.

The program already reads the input file and correctly constructs a graph. The program ***should*** initially display the number of vertices and edges in the graph and the 'diameter' of the graph. (The 'diameter' of the graph is the longest simple path distance between any pair of vertices in the graph. To find the diameter of a graph, first find the shortest path between each pair of vertices. The greatest length of any of these shortest paths is the diameter of the graph.) Currently all of these statistics display zero. **You need to make additions and/or modifications to display the statistics correctly.**

The program then prompts the user to select a start point. After a city is selected, the program ***should*** display the degree of the node representing the selected city. Currently the program displays zero. **You need to make additions and/or modifications to display this statistic correctly.**

Finally, the program prompts the user to select an end point. After a city is selected, the program ***should*** display the distance of the shortest path between the selected start and end cities. Currently the program displays zero. **You need to make additions and/or modifications to display this statistic correctly.**

You should carefully investigate the code that is already provided. You will only need to write a few lines of code (less than 20) to complete the assignment if you properly use code that already exists. **The only file you need to change is DisplayCostsMain.**

Include complete javadoc comments for any classes or methods that you write. Add your @author tag to the class level Javadoc comment of any classes that you modify.

Part 2: You will answer a series of questions about issues relating to the Graph ADT and Simon Gray's implementation. The questions are published in a file called graphQuestions.txt in the Eclipse project.
Edit the file to include your answers to the questions. You could think of this exercise as a take-home quiz on graphs or as preparation for possible final exam questions.
I will publish answers to the questions on Canvas prior to the final exam.

Getting Started:

Create your Eclipse project by downloading the `hw5-project.zip` file from the Assignment 5 page on Canvas, importing it into your workspace. Use "Refactor" to change "USERNAME" in the project name to your UWNetID. Before making any changes I suggest that you make sure that you can run the included JUnit tests. All tests should pass. When you complete the assignment all of these tests must still pass. As you work on this assignment, strive for maximum code reuse. You can significantly reduce the amount of code that you will need to develop if you carefully consider code reuse in your design.

Submission and Grading:

When you have committed the final revision of your code to SVN, make a note of its Subversion revision number. To get the revision number, *perform an update* on the top level of your project; the revision number will then be displayed next to the project name.

Submit (on Canvas) an *executive summary*, containing the Subversion revision number of your submission, an "assignment overview" (1 paragraph, up to about 250 words) explaining what you understand to be the purpose and scope of the assignment, and a "technical impression" section (1-2 paragraphs, about 200-500 words) describing your experiences while carrying out the assignment.

The filename for your executive summary must be "`username-assignment5.txt`", where `username` is your UWNetID. An executive summary template is available on Canvas. Executive summaries will *only* be accepted in plain text format – other file formats (RTF, Microsoft Word, Acrobat PDF, Apple Pages) are *not* acceptable.

The executive summary will account for 10% of this assignment's grade.

Part 1 will account for 50% of this assignment's grade and will be graded on external and internal correctness.

Part 2 (the 20 questions) will account for 40% of this assignment's grade.