Program #4 - Graph Finale

Prof Bill - Feb 2017

Program #4 is all about:

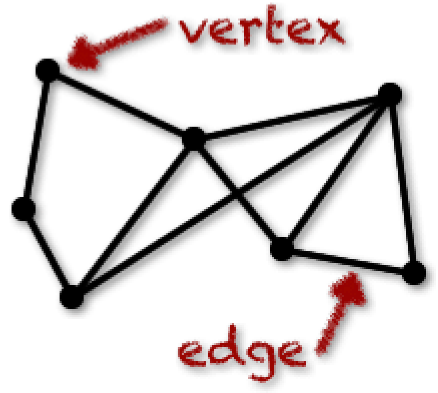
- Emphasis: Chapter 14 graphs and graph algorithms and some javafx gui
- Due: **Fri Mar 10, 2017** at the beginning of class
- Worth: **8 points** (8% of your grade)

1. Description

Program #4 has, ironically, **four** steps:

1. Create a javafx gui to create a graph: vertices and edges.
2. Implement a minimum spanning tree (MST) algorithm for a graph, either Prim's or Kruskal's.
3. Merge gui and graph to create a graph, um, graphically, and then show the MST.
4. Celebrate. This is the last program!

That's it.
thanks… yow, bill



2. Requirements

Here's a more detailed description of P4.

Step #1 - The gui

Your gui should create circles (javafx Circle objects) and connect them with lines (Line objects). You're looking at a flow sort of like this:

- Add a circle to the window
- Add another circle
- Connect the two circles with a line

We're not worried about graph stuff yet. And get as creative as you like. But this is a **Graphical** UI, so please don't make me type in circle names or whatever.
I have some gui code to help you on the k: drive. Use the ideas you like, ditch the rest.

Step #2 - The graph

The P4 graph interface is **Graph210**. It's a simple, tiny interface. It's an undirected graph. There are no names for vertices and edges, only integers. Code's on the k: drive. Some of the classes include:

- ➢ **Graph210** - interface
- ➢ **Edge210** - interface, an edge in a Graph210
- ➢ **GraphHelper** - some helper code for Graph210's

Your code:

- ➢ **MyGraph** - is-a Graph210,; list of edges; list of vertices where each vertex holds a list of its incident edges; remember to add your ctor and toString(); your toString format should match the file format in GraphHelper. So, a good test: read in a graph file and then write it out again using toString()... they should match.
- ➢ **MyEdge** - is-a Edge210; very simple: a number, 2 vertices, and a weight; ctor + toString
- ➢ **Prims** or **Kruskals** - create a class to hold your MST solution; method input is a Graph210 and returns a list of edge numbers

Step #3 - The merge

Once your basic gui and graph are working, then merge the two. Modify your circles to represent vertices. Make your lines edges. And finally, run MST and highlight the edges in your solution.
I recommend you show the edge weights on your graph. That's pretty common. And you should either 1) use random edge weights (from 1 to 20?) or 2) allowing the user to set the weight of an edge.

Step #4 - The celebration

Kool and the Gang can help you here: https://youtu.be/3GwjfUFyY6M

Etc

I'll provide some examples on the k: drive too:

- Input - Some graph files
- Output - Some edge lists for MST solutions

We'll sound off on Piazza and maybe a Helper file.

Grading

Create a **program4** folder on your k: drive.It should contain:

- All your Java source files
- Your program4 executable
- A **README.txt** file where you describe the status of your program and the creative command that you added

Grading will be: gui 3 points, MST 3 points, and merge 2 points.

All your code must follow our class **Coding Guidelines**. Ugly code will be severely penalized. A program that doesn't even compile is probably worth 0 points.

How to **succeed** in coding: 1. Start early! 2. Don't be shy. Ask a question in class. Email me. Come to office hours. 3. Work in small bites. Divide your program into small, manageable tasks. Knock them down one by one. 4. Always be green/working. Your program should always compile and run. Never leave your work in disarray.

Remember our **plagiarism** guidelines as well. Getting help from google or stackoverflow or a friend is OK, but:

1. You must acknowledge any help you receive with a comment in your code. **Exceptions to this: official Java API pages or Javafx pages.**
2. You must understand any code in your solution
3. Get help on program components, not the primary assignment (the tic tac toe philosophy)
4. If you have any questions in this area, contact me **before** you turn in your work, not after (when it's too late)

thanks… yow, bill