CS 122 – Lab 13 – Graph implementations

The purpose of today's lab is to practice with some Java implementations of graphs. In class, we saw how the adjacency matrix for a graph could be represented as a two-dimensional array. Would it be more efficient if we used an ArrayList? We'll be able to answer that question shortly.

On your account, create a new directory called lab13. The files you need for lab are on the class Web site in a directory called "http://cs.furman.edu/~chealy/cs122/Graph1" . Download these files to your directory.

Look at Graph.java. Right away you notice we have 2 arrays. An array of vertex objects, and an array to hold the adjacency matrix. It's convenient to think of the adjacency matrix as an array, however we suspect it may be inefficient to maintain arrays when it comes time to add vertices.

Copy Graph.java into a new file called ArrayListGraph.java. This is a new version of Graph.java that will employ a ArrayLists instead of arrays. It's not hard to change the declaration of the attributes from arrays to ArrayLists or allocating space in the constructor. The interesting part is addVertex(). To add a vertex, we need to add one new Vertex object to the ArrayList of vertices, and then look at the adjacency list…

Since the adjacency list is an ArrayList of ArrayLists, you will need to add one element to each of the existing (row) ArrayLists, and then add one whole new (row) ArrayList.

Don't forget to change findVertex() and degree() to refer to ArrayLists instead of arrays.

It would be a good idea to run the existing Driver.java using your ArrayListGraph just to make sure your output behavior is the same as with the original Graph. All you need to do is change the declaration of g at the beginning of main().

Finally, time to test the *efficiency* of the two implementations. Write a new file Tester.java that will contain its own main() method. In it, create 2 graphs: a Graph called g1 and an ArrayListGraph called g2. For each graph: Put 1000 vertices in it, add an edge between <u>every</u> pair of vertices. Then add code that will time the 2 portions of the program. The structure of your main() will look like this:

- Check the time (use this function call: System.currentTimeMillis())
- Create a graph g1 from the Graph class
- Check the time
- Create a graph g2 from the ArrayListGraph class
- Check the time.
- Subtract the times to give a comparison of the implementations.

How long did the array implementation take, in seconds? _____

How long did the ArrayList implementation take, in seconds? _____

What do you think is the reason for this difference?