Due Sunday, April 9, 11.59pm

This project will continue to use the USGS earthquake dataset and the graph you constructed in project 1-1. In part 2 (this project), you will implement the DFS traversal algorithm; you will use the BRIDGES visualizations to show the order in which the nodes are visited by replacing the node labels with numerical labels. Mousing over the nodes should indicate the order.

The DFS traversal algorithm is outlined in the text (Section 3.5). To implement the DFS algorithm, you will need to maintain a visited array for each vertex of the graph. Since each vertex needs to be uniquely identified, use the label strings (combination of magnitude, location, time) that were shown in the nodes in project 1-1 as hash keys in a Java HashMap (`http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html`).

Tasks.

1. **Graph Construction:** Graph construction is similar to project 1-1, with 2 diffeerences:

   - In part 1, it didnt matter what key was used for each vertex; for graph traversal, we need uniquely labeled nodes. You can use a string constructed from the labels used in part 1, i.e., concatenate the magnitude, time and location of the EQ object, to create a unique label. Thus if this string is *s*, then you will add the vertex as g.addVertex(s, ""), where *s* is the unique string that identifies this node.
   - Each time a vertex is inserted into the graph, it will also be kept track of in the hashmap (this is the mark array) with the visited flag (can be simply an integer, 0 or 1). Again, use the string *s*, to map into the hashmap. Use the HashMap class's put() and get() methods to access the mark array.

   Call bridges.visualize() to display the default graph; append the mark array value to the node, this will all be zero to start with.

2. **DFS Traversal:** You will implement the DFS traversal algorithm, outlined in the text (Section 3.5). During the traversal, each time a node is visited, *its label will be replaced by an integer that represents the order in which that node was visited.* Note that you will need to access the adjacency list of graph nodes. See the example on the BRIDGES website (`http://bridgesuncc.github.io/Hello_World_Tutorials/Graph.html`), that walks a user through this process to iterate through adjacency list of any BRIDGES graph node.

3. You will begin your DFS traversal from one of the (magnitude) cluster centers. Show at least 2 examples of the DFS algorithm starting from 2 different magnitude range nodes and generate separate visualizations. Check your traversal by ensuring the node ordering follows a DFS order. *Call bridges.visualize() for each visualization you generate, similar to project 1-1.* Note that all cluster center nodes labeled **0.0-1.0, 1.0-2.0,....** are already unique.

4. **[Extra Credit:]** Use the size or color/opacity coding on the nodes to indicate the traversal order. For instance, you may decrease the size of the nodes as the traversal goes further and further away from the starting vertex. For this you need to maintain the lavel of DFS traversal, with nodes at each level having the same size (or color or opacity). Generate a visualization for each attribute that you manipulate and display.

Evaluation:

- By interactive demo; a schedule will be set up to show your implementation by the teaching assistant.
- You will downnload your submission during the demo and run your program and demonstrate the results.
- Turn in your source code to Canvas by the deadline; **Source code must be well documented for full credit.**