Due Nov. ??,

This project will use the BRIDGES binary tree API with the Earthquake data that will be retrieved from a source viat Twitter. Instead of the interface from the textbook, you will use an equivalent interface from BRIDGES for binary search tree operations.

The BRIDGES API is accessible from **http://bridgesuncc.github.io/doc/api/0.99.0/**

Dataset:

We will be continue to use tweets of earthquakes as part of this project. The actual tweets will be accessed by BRIDGES from **https://twitter.com/earthquake**. Each tweet contains the quake magnitude, location, date and time. Utilities will be provided to access some of these attributes for use in the tasks described below.

This project will consist of 2 parts: the goal of the first part is to a good and deep understanding of the binary search tree ADT implementation. As such, the task is to embed attribute calls from BRIDGES to highlight aspects of the basic algorithms on binary search trees.

Project Tasks:

1. **BST ADT:** BRIDGES provides the basic algorithms to insert, delete and find elements in a binary search tree. By default, all elements have specific attributes for color, opacity, and size.

   Go through the implementation of the TreeVisualizer(in TreeVisualizer.java) that implements the binary search tree algorithms(these follow from the text with some modifications). Once you understand this class, create a subclass of Visualizer (similar to TreeVisualizer). This will contain the exact same methods, but now these will be modified to include attribute calls to highlight certain aspects of the search tree algorithms.

   Output: Insert at least 50 earthquake tweets into a binary search tree; use the magnitude as the search key and to size the node (similar to project 2). Generate the visualization.

2. **Insert():** You will modify the insert() method, so as to display the path traced out by the insert() function. Use a single color to highlight the relevant nodes and paths, while using a unique color for the newly inserted node. **Output:** Insert 40 earthquake tweets into a binary search tree; use the magnitude as the search key. Generate the visualization. Demonstrate your modified insertion algorithm by inserting the next 3 tweets into the tree and calling the update() method after each insertion. Your 3 visualizations should show the paths traced out by the insertion and the inserted node. You can use the default BRIDGES insert() method to generate the initial tree (in default color) or adapt your implementation.

3. **Find():** Repeat part 2 for the find() algorithm; you will generate 4 visualizations similar to (2). In this case, the element found will be highlighted in a unique color. In case the tweet is not found, only the path is highlighted. Similar to (2), create the default tree and then choose 3 elements within the tree to search for and output the results. Also test one value of the magnitude that does not exist in the tree.

   Output: Similar to (2), you will generate 5 visualizations, 1 for the default tree, 3 for values found in the tree and 1 for a nonexistent quake.

4. **Remove()** Since the node is removed from the tree, you will create before and after snapshots of the tree: (1) once the node to be deleted is found, show the path and the node distinctly(as in the insert() case); once the deletion is complete, show the final tree(for the 2 children case, you can also highlight the path to the replacement node, which will ultimately be removed and will not show in the final tree).

   Output: Similar to (2), you will do three cases: removing a leaf node, removing a node with 1 child, and removing the root node of the tree. You will generate two visualizations (before and after) for each case. Start with a large enough tree and pick suitable values to delete (the last case will remove the root node).

   A simple tree driver will be provided that will illustrate basic tree operations.

   Saving Output.

   You can save your output by selecting unique assignment ids, as part of the Bridge.init() call. Each update() call will save the output within this assignment id. If assignment id is 7, then 7.0, 7.01, 7.02... will be part of the links generated to save the visualization(needed in part 2).

   Evaluation:

   Save all your output in a set of unique links. Once the output is saved, you should not overwrite those pages until grading is completed. Turn in all of your source codes and the output links to Moodle.