CSC 205

Fundamentals of Computer Science II

Spring 2017

Project 2

Due by noon on Thursday, May 11.

For this project you will develop methods to manipulate a Binary Search Tree. Binary Search Trees are a fundamental data structure in computing, and are the underlying structure of the TreeSet and TreeMap classes in the Java Collection Frameworks. As before, we will be using the software of the BRIDGES project, which allows us to visualize the shape of the Binary Search Tree, and to see how it is altered by your methods.

The given source code includes a class called **BinarySearchTree_Bridges**. This class implements a Binary Tree, using the **BSTElement** class from Bridges to represent the nodes in the tree. Similar to the TreeMap in the Java Collection Framework, each element in the tree will store both a **key** and the **value** associated with that key. The given Driver source code uses Student objects as the test data, where the key associated with each Student is the **name** field. But any Object that is Comparable could also be stored in this Binary Search Tree.

The given BinarySearchTree_Bridges class has a method to search (i.e., "get") the tree for a given target key.

For this assignment you must complete the methods in the **BinarySearchTree_Bridges** class to perform the following operations.

- public void resetColor (Color c)
  This method traverses the entire tree and sets the color of each node to the specified color **c**.

- public E last (Color c)
  This method locates and returns the **maximum** value in the tree, and also changes the color of every node along the path from the root to the maximum value to be the specified color **c**.

- public E first (Color c)
  This method locates and returns the **minimum** value in the tree, and also changes the color of every node along the path from the root to the minimum value to be the specified color **c**.

- public int countAndColorMajors (Color c , String theMajor)
  This method traverses the entire tree. Each node is examined, and if the value stored at that node is a Student object, and if the major field of that Student object equals the parameter **theMajor**, then that node is set to the specified color **c**. The total number of nodes that are colored by this method is returned. Note that this is the only method in this assignment that assumes that the data stored in the tree are Student objects. Your code can always check whether a value stored in the tree is a Students object using the instanceof operator in Java. For example:

  ```
  if ( foo  instanceof  Student ) {    ....    }
  ```

- public void colorRange (Color c, K minVal, K maxVal)
  This method resets the color of every item in the tree whose key is greater-than-or-equal-to minVal, and less-than maxVal, to be the specified color **c**.

- public void deleteSmall (K key)
  This method removes from the tree every item whose key is less-than-or-equal-to the given parameter **key**. To remove these keys you should directly alter the pointers in the tree using setLeft and setRight methods to "splice out" the small values.

- public void deleteLarge (K key)
  This method removes from the tree every item whose key is greater-than the give parameter **key**.

- public E colorLower (K key, Color c)
  This method locates and returns the largest value in the tree that is strictly less than the given parameter **key**. If no such item exists, then the value null is returned. If the item exists, then the method changes the color of that node to be the specified color **c**.

- public E colorHigher (K key, Color c)
  This method locates and returns the smallest value in the tree that is strictly greater than the given parameter **key**. If no such item exists, then the value null is returned. If the item exists, then the method changes the color of that node to be the specified color **c**.

As always, you should implement all these methods in the most efficient manner possible. Your code should not traverse more nodes in the tree than is necessary.

The given Driver code will test your methods and display the tree that is produced by your code. You should create additional test cases to insure that your code is correct, instead of relying only on the given test cases.

Deliverables:

- Complete the Bridges survey at https://unccpsych.az1.qualtrics.com/jfe/form/SV_dcK4tmEmIQ68hDv and take a screen shot of the survey after you have completed it.

- Upload a **zipped** file of the **BinarySearchTree_Bridges.java** file and your survey screenshot to Blackboard

- Hand in a **paper copy** of your **BinarySearchTree_Bridges.java** file. Be sure that your code is readable (the code must not spill over the line onto the next line). Your code must use good programming style (proper indentation, good use of comments, well chosen, meaningful variable names, etc.)

If you have difficulty with the BRIDGES software, or seeing your visualization, then you can contact tues-bridges-group@uncc.edu (use the Subject line: **Bridges Query**).