03. Red Black Tree Implementation

Previous Work

Complete this week's **RB-Tree Insert Activity (https://canvas.wisc.edu/courses/254888/pages/03-rb-tree-insert-activity?module_item_id=3908013)** and **Rotation Activity (https://canvas.wisc.edu/courses/254888/assignments/1220649)** before working on this implementation.

Work Individually

You must develop the code for this assignment entirely on your own. If you encounter difficulties while working, you are encouraged to discuss general algorithms and debugging strategies with anyone you would like. If you feel that getting assistance will require someone else viewing your code, the course staff are the only people that you are allowed to share or show any part of your code with prior to the hard deadline for this assignment. You may not store or share your code in any way that another students can access. And you are not allowed to view or make use of any Red Black Tree code that is not written entirely by you.

RedBlackTree Specifications

0. Start by making a copy of your RedBlackTree class from this week's Rotation Activity. You can also remove the three JUnit Test methods from this copy, since they may break while changing the mechanics of this new class's insert operation. If additional assistance from the course staff would be helpful for this, please contact us for that help.
1. Add a public boolean isBlack field to the Node inner class. Ensure that every newly instantiated Node object has a false isBlack field (so that all new nodes are red by default).
2. At the end of the insert method, add a statement to always set the root node of your red black tree to be black. No further changes to this method should be made.
3. Add a new private method to your RedBlackTree class called enforceRBTreePropertiesAfterInsert which takes a reference to a newly added red node as its only parameter. Note that this method may also be called recursively, in which case the input parameter may reference a different red node in the tree that potentially has a red parent node. The job of this enforceRBTreePropertiesAfterInsert method is to resolve a red child under a red parent red black tree property violations that are introduced by inserting new nodes into a red black tree. While doing so, all other red black tree properties must also be preserved.
4. The enforceRBTreePropertiesAfterInsert method should be called from insertHelper after adding a new red node to the tree (in both the cases of adding this new node as a left child and as a right child). No further changes to the insertHelper method should be made.
5. Implement the enforceRBTreePropertiesAfterInsert method to recognize and handle each of the possible cases described in your lecture videos and readings. You should make use of the rotate method that you implemented this week, by calling it from your definition of enforceRBTreePropertiesAfterInsert. Implementing this method correctly should ensure that the worst-case height of your tree remains O(log n), where n = the size or number of elements stored in your red black tree.
6. **Notice that you are not being asked to implement the remove operation for this red black tree implementation.**

Assignment Submission

Please submit your progress on this assignment early and often to **gradescope (https://gradescope.com/)** : as a form of back up, and to get feedback from the automated grading tests. The one file that you should include with your submissions is: RedBlackTree.java. Your most recent submission in gradescope will be marked "active" by default, but you can change this to an earlier on-time or less buggy submission prior to the hard deadline. Ensure that your final "active" submission for this assignment is clearly organized, consistently styled, well documented with comments, and includes the following file header information at the top of each source file:

```
// --== CS400 File Header Information ==--
// Name: <your full name>
// Email: <your @wisc.edu email address>
// Notes to Grader: <optional extra notes>
```

Grading Rubric

Criteria	Assignment	Points
Formative Gradescope Tests	Individual	15
Summative Gradescope Tests	Individual	15
Code Clarity (Manual)	Individual	5
Total Points Possible		**35**