Homework 4: AVL Trees

Due: 6/10/13 11:55 PM

Important

There are a few guidelines you must follow in this homework. If you fail to follow any of the following guidelines you will receive a 0 for the entire assignment.

- All submitted code must compile under JDK 7. This includes unused code, don't submit extra files that don't compile. Java is backwards compatible, so if it compiles under JDK 6 it should compile under JDK 7.
- DO NOT include any package declarations in your classes.
- DO NOT change any existing class headers, constructors, or method signatures. It is fine and encouraged to add extra methods and classes.
- DO NOT import anything that would trivialize the assignment. (e.g. don't import java.util.LinkedList for a Linked List assignment. Ask if you are unsure.)
- You must submit your source code (ie. the .java files), NOT the compiled bytecode (ie. the .class files)

After you submit your files, redownload them and run them to make sure they are what you intended to submit. We are not responsible if you submit the wrong files. DO NOT submit .class files.

Overview

You are implementing an AVL tree, a self-balancing BST. This ensures that practically every operation can be completed in O(log n) instead of O(n). Some of the code from the previous homework will be useful for this assignment. You will need to add helper methods and variables.

Most of the AVL logic is in Node.java. Read the javadocs to understand the requirements for each method. Add() and remove() should rebalance the subtree when necessary. When the user tries to remove a Node that has two children, use the successor to replace it. When trying to find data in your AVL tree, you must use the .equals() method to check for equality. Per the Comparable documentation, we cannot assume that the .compareTo() method is consistent with equals.

OperationResult.java is a simple data-holder class that allows you to return multiple values from the add() and remove() methods. You should not need to modify this file.

The AVLTree.java file that we provided implements the Collection interface. You need to implement the add(), remove(), contains(), inOrder(), size(), isEmpty(), and clear() methods. **You should read Java's specifications for each method before and after you implement it.** The specification can be found here: http://docs.oracle.com/javase/7/docs/api/java/util/Collection.html.

Requirements/Specifications

- Remove() must use the successor if the Node has two children.
- Your AVL tree may assume that no duplicate elements will be added.
- Your AVL tree must always be balanced (ie. |balance_factor| > 1) after every operation.
- Your AVL tree must use .equals() when searching for data. Only using .compareTo() is not enough!
- Your AVL tree must follow the specifications set in the Collection documentation.
- How your AVL tree handles null arguments is undefined. (ie. you can do whatever you want with null arguments because we won't test for them.)

Suggestions

- Start early! This is one of the most difficult homework assignments!
- There are many edge cases, so writing unit tests is very helpful.
- Make sure your code is clean! AVL trees can get very confusing very fast.

Deliverables

You must submit the following files:

- AVLTree.java
- Node.java
- Any additional files that you need.

You may attach them individually or submit them in a .zip folder.