CS 331 Lab: AVL Lab
Fall 2010

1 Objectives

This lab is the final project. It is meant to serve as a "final exam" for the lab portion of the course. As such, you will have until the end of finals' week to finish it. You will die a horrible death if you wait until final's week to get started, so start it *now*. Note that there will be (at least) one other lab between now and the end of the semester, as well.

You will implement the AVL tree data structure described in lecture. It is similar to a binary search tree, except there are balancing operations that occur automatically to keep the tree performing well.

2 Your Work

These AVL trees will be dictionary style: the nodes will contain a key used for searching, and a value that is returned when a search succeeds. These types will be referred to as `K` and `V`, respectively.

Your AVL data structure will support the following methods:

constructor

void add(K key, V value) Add the node to the tree, performing whatever rotations are necessary.

V find(K key) Find the node in the tree, returning the associated value. If no value is found, return `null`.

void delete(K key) Delete the item, performing whatever rotations are necessary.

int size()

Iterator<V> mkBFiterator() Return an iterator that traverses the tree in breadth first order.

Iterator<V> mkDFiterator() Return an iterator that traverses the tree in depth first order.

The iterators should support `get`, `isValid`, and `next`, but not `delete`. Furthermore, if the delete method is called on the tree, all iterators must become invalid. You will therefore need to keep track of what iterators are alive.

You will also need to write test cases. The iterators are there in part to assist you in that.

3 Given Files

We will give you `Iterator.java` and a simple `Makefile`. You're on your own for the rest of it. You will need to supply the files `AVL.java` and `TestAVL.java`.

Please be careful not to change the names or types of the classes or methods. It doesn't matter at all if the code compiles in your account. It matters everything if the code compiles in the grading script. If you don't follow the specs, it *will* break.