CSCI-1200 Data Structures — Fall 2019
Lab 10 — Advanced Trees

Checkpoint 1

estimate: 15-30 minutes

Download these files:

http://www.cs.rpi.edu/academics/courses/fall19/csci1200/labs/10_trees_II/ds_set.h
http://www.cs.rpi.edu/academics/courses/fall19/csci1200/labs/10_trees_II/test_ds_set.cpp

Implement and test the decrement operator for `tree_iterator`. Determine an appropriate sequence to *insert* the numbers 1-15 such that the resulting tree is *exactly balanced*. After using the `print_sideways` function to confirm the construction of this tree, test your iterators on the structure. Similarly, create a couple unbalanced trees to demonstrate that both the increment and decrement operators for iterators are debugged. Your decrement operator should correctly decrement the `end()` iterator. You can use the same "trick" we used in Lab 6 to make this work for `ds_list` iterators. Ask a TA if you have any questions.

EDITED TO ADD: Let's monitor the "work" done by the increment and decrement functions. Add two global counter variables named `up_count` and `down_count` to your program. *Yes, global variables are bad in general! It's ok for this quick inspection task. (Alternately you may use static variables within the function or class. Ask a TA or mentor if you'd like to learn how to use them!)*

Initialize these counters to zero. When an iterator moves down one link in the tree, increment the `down_count` variable. When an iterator moves one link up in the tree, increment the `up_count` variable.

Now, iterate from `begin()` to `end()` over your tree, and print out these counts after each step. Do the counts match the distance between the nodes in the diagram? The number of links followed up and down is a good measurement of the total cost / running time of the iteration functions. What is the worst case cost for a call to `operator++` on a tree with n nodes? What is the total cost for iterating over the whole tree? What is the average or *amortized* cost for a call to `operator++` on a tree?

To complete this checkpoint: Show one of the TAs your iterator decrement code and your tests cases. Also be prepared to discuss your measurements of the cost of each step of iteration.

Checkpoint 2

Checkpoint 2 will be available at the start of Wednesday's lab.

Checkpoint 3

estimate: 15-30 minutes

Add a member function called `accumulate` to the public interface of the `ds_set<T>` class, and provide its implementation. The function should take only one argument (of type `T`) and it should return the results of *accumulating* all the data values stored in the tree. The argument is the initial value for the accumulation. The function should only use `operator+=` on type `T`.

Test your code by showing that this works for both a set of ints, where the accumulate function should sum the values in the set (initial value parameter is 0), and a set of strings, where the accumulate function should concatenate the strings in the set (initial value parameter is `""`). Does it matter if the `operator+=` for type `T` is *commutative*? How can you control the result of accumulate if it is *not* commutative?

To complete this checkpoint: Show a TA your completed and tested program.