Day 24 – BST Implementations and Performance

Chapter 07, 7.4 – The Implementation Level: Basics , Review Problem(s): ***22***

Chapter 07, 7.5 – Iterative vs. Recursive Method Implementaion

Chapter 07, 7.6 – The Implementation Level: Remaining Observers, Review Problem(s): ***37***

Chapter 07, 7.7 – The Implementation Level: Transformers, Review Problem(s): ***39, 40***

Chapter 07, 7.8 – The Binary Search Performance, Review Problem(s): ***44, 45***

`BSTNode.java` is the node for BSTs, a left pointer, info, and a right pointer.

Chapter 07 PPT, pg 27-31.

Trees are inherently recursive, they consist of subtrees. BSTs can be implemented with recursive or iterative methods.

Chapter 07 PPT, pg 32-40.

We discuss the implementation of the remaining observers, note which methods return an `Iterator`.

Chapter 07 PPT, pg 41-49.

The transformers are very complex. For example, consider deleting a node with the `remove` method. The BST property must be maintained.

In particular:

- Removing a leaf (no children): removing a leaf is simply a matter of setting the appropriate link of its parent to null.
- Removing a node with only one child: make the reference from the parent skip over the removed node and point instead to the child of the node we intend to remove.
- Removing a node with two children: replaces the node's info with the info from another node in the tree so that the search property is retained - then remove this other node. The other node is called the predecessor, which has to have only one child.

Chapter 07 PPT, pg 50-65.

The performance of the BST methods take advantage of the binary tree structure, especially having a balanced tree. We'll discuss balancing BSTs more in CS 315 – Algorithms and Analysis.

Chapter 07 PPT, pg 66-74.