Lab 4: Binary Search Trees

Week of November 14, 2016

Objective

To complete a small program that validates the order of keys in binary search trees.

Exercise

File `Lab4.java` contains a nearly-complete program that constructs a number of binary trees and then tests each to see if the values in the tree were inserted in binary search tree order.

In these trees, no duplicate values are permitted, so binary search tree order can be written as follows: the value at each node n must satisfy

$$\text{values in the left subtree of } n < \text{value at } n < \text{values in the right subtree of } n,$$

where "the left/right subtree of n" is the left/right child of n and all of that child's descendants.

You must write the following methods:

- `getMax`**:** A **recursive** method in the `Node` class with NO parameters that returns the maximum value stored in the subtree that consists of `this` and all the descendants of `this`. The method should assume that the tree is a binary search tree (i.e., that the values in the tree satisfy the binary search tree property). The method should simply use recursion to move to the right child (if there is one) — returning the data stored in `this` if it has no right child, otherwise returning whatever the recursive call returns.

- `getMin`**:** A **recursive** method in the `Node` class with NO parameters that returns the minimum value stored in the subtree that consists of `this` and all the descendants of `this`. The method should assume that the tree is a binary search tree (i.e., that the values in the tree satisfy the binary search tree property). The method should simply use recursion to move to the left child (if there is one) — returning the data stored in `this` if it has no left child, otherwise returning whatever the recursive call returns.

- `isOrdered()`**:** The public driver method in the `BinaryTree` class with NO parameters that returns a `boolean` value. It returns `true` if the calling tree is in BST order, and `false` otherwise. It handles the case when the tree is empty by returning `true`; otherwise it calls a `Node` method, also called `isOrdered()` (see next method), on the root of the tree and returns whatever the `Node` method returns to it.

- `isOrdered()`**:** A **recursive** method in the `Node` class that does most of the work of deciding whether a non-empty tree is ordered or not. It should have NO parameters and return a `boolean` value. It should return `true` if `this` is a leaf. If `this` is not a leaf, it should return `true` only if

  - The left child is in BST order (if there is a left child), AND
  - The data in `this` is greater than the maximum value in the left child (if there is one), AND
  - The right child is in BST order (if there is a right child), AND
  - The data in `this` is less than the minimum value in the right child (if there is one).

  Otherwise, the method should return `false`.

  This method should do its job in a traversal (since it has to visit every `Node` to be sure that the tree is in BST order).

Just add the bodies of the `BinaryTree` and `Node` class `isOrdered` methods and the `Node` class methods `getMax` and `getMin`, and change nothing else in the `Lab4.java` file.