UNIVERSITY OF VIRGINIA, DEPARTMENT OF COMPUTER SCIENCE

BSTs - Tree Implementation

Nada Basit and Mark Floryan

February 17, 2022

1 SUMMARY

For this homework, you will be implementing two classes to create a working Binary Search Tree. You will start by implementing a basic Binary Tree with the three tree traversals covered in class. Then, you will implement and test a working Binary Search Tree.

1. Download the provided starter code. You can **ignore the AVLTree** class. You will be using that one next week.

2. Implement some general methods in the BinaryTree class.

3. Implement the BinarySearchTree class

4. Use the provided tester files to verify your implementation works. Note that you should test your code more so than the provided tester does this time. The tester is NOT as thorough as in previous homeworks.

5. **FILES TO DOWNLOAD:** BinarySearchTrees.zip

6. **FILES TO SUBMIT:** BinaryTree.java, BinarySearchTree.java

1.1 BinaryTree.java

To begin, implement the BinaryTree class. These are methods that are useful for ANY binary tree (whether balanced or not). You will need to implement the following methods:

```
public class BinaryTree<T>{
    //Prints the tree using in-order traversal
    //All on one line, space separated
    private void printInOrder(TreeNode<T> curNode);

    //Prints the tree using pre-order traversal
    //All on one line, space separated
    private void printPreOrder(TreeNode<T> curNode);

    //Prints the tree using post-order traversal
    //All on one line, space separated
    private void printPostOrder(TreeNode<T> curNode);
}
```

The Binary Tree contains a few methods that are implemented for you. This includes the main print functions, that simply call the helper functions described above on the root to give off the recursive prints. In addition, a method called printTree() is provided that prints the tree in a somewhat formatted method. This may be useful when debugging your code. Lastly, a simple recursive method that computes the height of the binary tree is provided as an example of recursion in trees.

1.2 BinarySearchTree.java

Next, you will implement a binary search tree. This class will extend the binary tree class from earlier, and thus inherit the print methods defined earlier. Your binary search tree should implement the provided Tree interface, shown below.

```
public interface Tree<T extends Comparable<T>> {

    //remember to IGNORE DUPLICATES
    public void insert(T data);

    public boolean find(T data);

    public void remove(T data);
}
```

```
/* You BST implements the interface above */
public class BinarySearchTree<T extends Comparable<T>>
                    extends BinaryTree<T> implements Tree<T>{

        //TODO: Implement this class
}
```

You may add other supporting methods to your binary search tree if you find that to be helpful.

1.3 Testing your code

Once you are done, you can look at the provided tester file. This tester is VERY minimal and only checks one single test case, providing you with the expected output. You can (and should) create more tests and manually check them to ensure your tree is operating correctly.

You should submit **two files** for this homework: **BinaryTree.java, and BinarySearchTree.java**.

1.4 Gradescope

You should submit your code to *Gradescope*. If you are having trouble with your submission, you should double check the following common problems:

1. Make sure you are only submitting the two requested files, and they are named *BinaryTree.java* and *BinarySearchTree.java* exactly.

2. Make sure you keep any *package* statements in your code before submitting. The autograder expects your files to have the package statements that are provided in the downloaded project.

3. Make sure your output is in the correct format. You should not be printing ANYTHING else or the autograder will think your output is incorrect.