AP Computer Science with Data Structures

Lab: Trees part 1

Collaboration rules apply as per the course policies.

Exercise: Displaying Trees

Download TreeNode.java, TreeUtil.java, and TreeDisplay.java. `TreeNode` is the class you'll be using to represent binary trees in this lab (and on quizzes). In the TreeUtil.java file, you'll be implementing a number of helpful methods (and some fun ones) for operating on binary trees. Later in the lab, you'll use the `TreeDisplay` class to display traversals of binary trees. First, however, you'll need to implement the three methods in `TreeUtil` that `TreeDisplay` calls.

```
public static Object leftmost(TreeNode t)
public static Object rightmost(TreeNode t)
public static int maxDepth(TreeNode t)
```

Be sure to test these methods on some simple trees. Then, go ahead and try out the `TreeDisplay` class. (You can use the `createRandom` method to build a randomly shaped tree of a given maximum depth.)

```
TreeNode tree = TreeUtil.createRandom(6);
TreeDisplay display = new TreeDisplay();
display.displayTree(tree);
```

Exercise: Counting

Go ahead and implement the following standard binary tree methods.

```
public static int countNodes(TreeNode t)
public static int countLeaves(TreeNode t)
```

Test that each of these works correctly by displaying a random tree and printing the output of each of these methods.

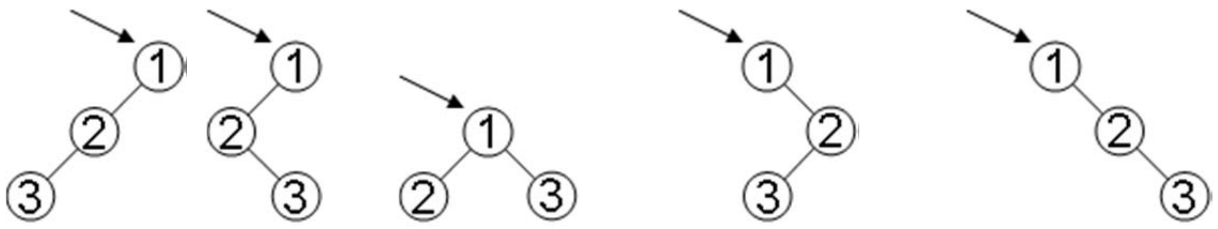
Exercise: Traversals

Implement the three binary tree traversal methods. Notice that each takes in a `TreeDisplay` object, so that we can see the traversal graphically. To light up a node, be sure to call `display.visit(t)` at the appropriate time in your implementation.

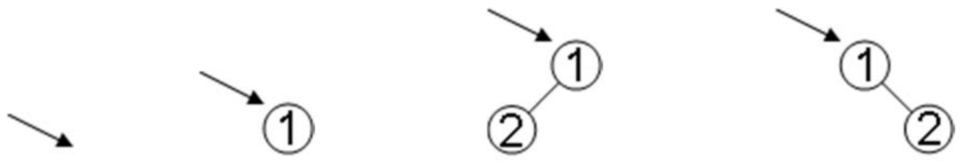
```
public static void preOrder(TreeNode t, TreeDisplay display)
public static void inOrder(TreeNode t, TreeDisplay display)
public static void postOrder(TreeNode t, TreeDisplay display)
```

Exercise: Save the Trees

We'd like to be able to save the contents of a tree to a text file, which is essentially a sequence of text. Suppose we want to save any of the following trees to a file.



We might be tempted to simply follow a preorder traversal and output the sequence of visited nodes to a file. Unfortunately, however, every single one of these trees results in the same preorder traversal: 1, 2, 3. Therefore, if we stored 1, 2, 3 in a file, we would not have enough information to recover the original shape of the tree. One solution to our problem is to output an extra marker every time we reach a `null`. We'll use `$` for this purpose. Below are several simple examples of a tree and the text file produced when that tree is saved to a file.



```
$                   1                   1                   1
                    $                   2                   $
                    $                   $                   2
                                        $                   $
                                        $                   $
```

Before going on, make sure you know what sequence should be written to a text file when saving any of the trees whose preorder traversal gave 1, 2, 3.

Go ahead and implement the method `fillList`, which takes in a `List` of `String`s and fills that list with the contents of the tree `t`, including `$` markers.

```
public static void fillList(TreeNode t, List<String> list)
```

Now download `FileUtil.java`, and use its `saveFile` method to implement the `saveTree` method below.

```
public static void saveTree(String fileName, TreeNode t)
```

Test that you can correctly save a randomly shaped tree by opening the text file created when you call `saveTree`.

Exercise: Load the Trees

Implement the method `buildTree`, which takes in an `Iterator` of `String`s (including `$` markers) and returns the tree represented by the `String`s returned from the `Iterator`.

```
    public static TreeNode buildTree(Iterator<String> it)
```

Finally, use `FileUtil`'s `loadFile` method to implement the `loadTree` method below.

```
    public static TreeNode loadTree(String fileName)
```

Test that you can now correctly load back and display a saved tree.

Turn in your work

Upload your lab work to Athena2 in accordance with the turn-in instructions given in the course policies.