Today

• Binary Trees
• Recursion and Trees
• Binary Search Trees

By the end of class
• You will be able to articulate what makes binary search trees so powerfully efficient - including understanding the runtime of the mysterious TreeSet
Binary Tree

IntTreeNode root = null;

public class IntTreeNode {
    public int myValue;
    public IntTreeNode myLeft; // holds smaller tree nodes
    public IntTreeNode myRight; // holds larger tree nodes

    public IntTreeNode(int val) { myValue = val; }
}

Binary Tree

Root: the starting point of the tree

Subtree: any part of the tree is also a tree. This is a “subtree rooted at node 7”

Node 5 is the “parent” of node 2. Node 2 node 5’s “left child”

Leaf: a node that has no child nodes

Internal node: a node that has 1 or 2 children
Today

- Binary Trees
- Recursion and Trees
- Binary Search Trees

By the end of class
- You will be able to articulate what makes binary search trees so powerfully efficient - including understanding the runtime of the mysterious TreeSet
Trees and Recursion

- They go together like PB&J!

- Check current node
  - if no
    - check left subtree
    - check right subtree

```java
public int computeTreeThing(TreeNode current) {
    if (we are at the base case) {
        return obviousValue;
    } else {
        int lResult = computeTreeThing(current.left);
        int rResult = computeTreeThing(current.right);
        int result = //combine those values;
        return result;
    }
}
```
Trees and Recursion

- Code
  - countNodes
  - containsNode
  - findMax

```java
public int computeTreeThing(TreeNode current) {
    if (we are at the base case) {
        return obviousValue;
    } else {
        int lResult = computeTreeThing(current.left);
        int rResult = computeTreeThing(current.right);
        int result = //combine those values;
        return result;
    }
}
```

Trees and Recursion

- What is the running time?
  - countNodes
  - containsNode
  - findMax

```java
public int computeTreeThing(TreeNode current) {
    if (we are at the base case) {
        return obviousValue;
    } else {
        int lResult = computeTreeThing(current.left);
        int rResult = computeTreeThing(current.right);
        int result = //combine those values;
        return result;
    }
}
```
Binary Tree

• A tree is **height-balanced** if
  • left and right subtrees are both height balanced
  • the heights of left and right subtrees do not differ by more than 1

A B C D

Binary Tree

• What is the height of a **height-balanced** tree?

A B C D
Today

- Binary Trees
- Recursion and Trees
- Binary Search Trees

By the end of class
- You will be able to articulate what makes binary search trees so powerfully efficient - including understanding the runtime of the mysterious TreeSet

Binary Search Tree

- Each node has a value
- Nodes with values less than their parent are in the left subtree
- Nodes with values greater than their parent are in the right subtree

![Binary Search Tree Diagram]

- All values < 7
- All values > 7
Binary Search Tree

What is the maximum time to:
- Insert a node?
- Find a node?
Today

• Binary Trees
• Recursion and Trees
• Binary Search Trees

• By the end of class
  • You will be able to articulate what makes binary search trees so powerfully efficient - including understanding the runtime of the mysterious TreeSet