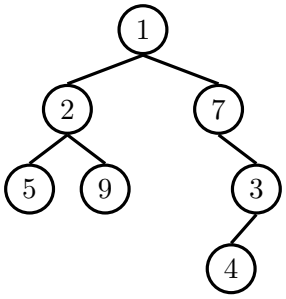
CSM 61B Fall 2020

So Many Trees

Mentoring 8: October 12 - October 16, 2020

1 Binary Trees

```
public class BinaryTree<T> {
    protected Node root;
    protected class Node {
        public T value;
        public Node left;
        public Node right;
    }
}
```



1.1 Define a procedure, `height`, which takes in a `Node` and outputs the height of the tree. Recall that the height of a leaf node is 0.

```
private int height(Node node) {




}
```

What is the runtime of `height`?

1.2 Define a procedure, `isBalanced`, which takes a `Node` and outputs whether or not the tree is balanced. A tree is **balanced** if the left and right branches differ in height by at most one and are themselves balanced.

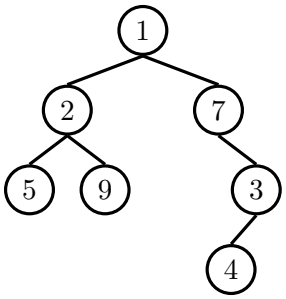
```
private boolean isBalanced(Node node) {




}
```

What is the runtime of `isBalanced`?

2 Traversals

Level-Order Traversals Nodes are visited top-to-bottom, left-to-right.

Depth-First Traversals Visit deep nodes before shallow ones.



2.1 Give the ordering for each depth-first traversal of the tree.

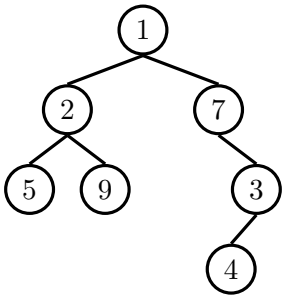
(a) Pre-order

(b) In-order

(c) Post-order

2.2 Give the level-order traversal of the tree.

2.3
```
public void treeTraversal(Fringe<Node> fringe) {
    fringe.add(root);
    while (!fringe.isEmpty()) {
        Node node = fringe.remove();
        System.out.print(node.value);
        if (node.left != null) {
            fringe.add(node.left);
        }
        if (node.right != null) {
            fringe.add(node.right);
        }
    }
}
```



What would Java display?

(a) `tree.traversal(new Queue<Node>());`

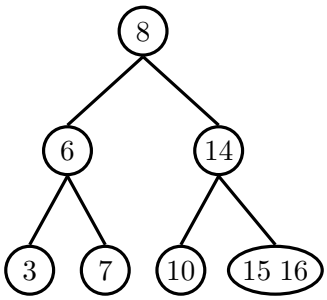
(b) `tree.traversal(new Stack<Node>());`

3 Binary Search Trees

3.1 Implement `fromSortedArray` for binary search trees. Given a sorted **`int[]`** `array`, efficiently construct a balanced binary search tree containing every element of the array.

```
public class BinarySearchTree<T extends Comparable<T>> {
    protected Node root;
    protected class Node {
        public T value;
        public Node left;
        public Node right;
    }
    public static BinarySearchTree<Integer> fromSortedArray(int[] values) {
        BinarySearchTree<Integer> bst = new BinarySearchTree<>();
        bst.root = bst.fromSortedArray(values, 0, values.length - 1);
        return bst;
    }
    private Node fromSortedArray(int[] values, int lower, int upper) {
```

4 2-3 Forever



4.1 Draw what the 2-3 tree would look like after inserting 18, 12, and 13.

4.2 Now, convert the resulting 2-3 tree to a left-leaning red-black tree.