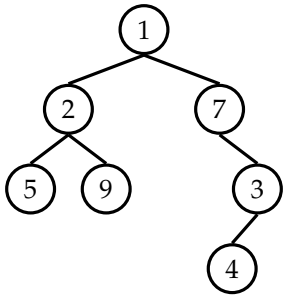
CSM 61B Spring 2017

Binary Trees

Week 9: March 13, 2017

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 Binary Search Trees

2.1 Write the `fromSortedArray` method 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) {
```

3 Balanced Faster *Extra for Experts*

3.1 Design an algorithm to more efficiently compute whether a tree is balanced or not. *Hint: What part of the naive* `isBalanced` *solution was expensive and how can we make it less expensive?*

```
private boolean isBalanced(Node node) {
```

4 Successor *Extra for Experts*

4.1 Given a node in a binary search tree (with parent pointers), write a `successor` method which returns the next node in the in-order traversal of the BST. If there is no successor, simply return null.

```
public class BinarySearchTree<T extends Comparable<T>> {
    protected Node root;
    protected class Node {
        public T value;
        public Node parent;
        public Node left;
        public Node right;
    }
    private Node successor(Node node) {
```