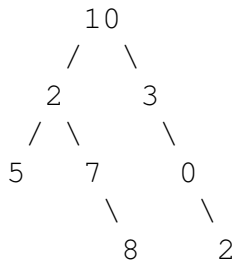
CS 61B Search — Binary Trees and Backtracking — Fall 2014

1 Binary Tree Traversals

Write the preorder, inorder, and postorder traversals of the following binary tree.



Bonus questions: What is the height of a binary tree with N nodes that has the same preorder and inorder traversal? Is this also true for non-binary trees?

2 Is This a BST?

The method `isBST` determines whether or not a given a given TreeNode is the root of a valid BST. Assume that all values in the BST are unique.

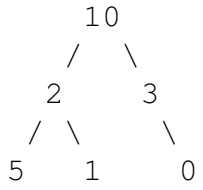
```
public class TreeNode {
    public TreeNode(int val) {
        this.val = val;
    }

    public int val;
    public TreeNode left;
    public TreeNode right;
}

public static boolean isBST(TreeNode n) {
    if (n == null) {
        return true;
    }

    if (n.left != null && n.left.val > n.val) {
        return false;
    }

    if (n.right != null && n.right.val < n.val) {
        return false;
    }

    return isBST(n.left) && isBST(n.right);
}
```

(a) Is the above method correct for all inputs? Why or why not?

(b) If you answered no to the previous part, provide a correction to `isBST` on another sheet of paper.

3 Sum Paths

Define a root-to-leaf path as a sequence of nodes from the root of a tree to one of its leaves. Write a method `printSumPaths(TreeNode root, int n)` that prints out all root-to-leaf paths whose values sum to n. For example, if RootNode is the binary tree rooted in 10 in the diagram below and n is 13, then the program will print out `10 2 1` on one line and `10 3 0` on another.



Provide your solution by filling in the code below:

```
void printSumPaths(Node t, int n) {
    if (t != null) {
        sumPathsHelper(                                    );
    }
}

void sumPathsHelper(                                    ) {









}
```

Bonus question: What is the worst case $\Theta(\cdot)$ runtime of your method in terms of the number of nodes Q in the given tree? The best case?