CS 61B Spring 2021

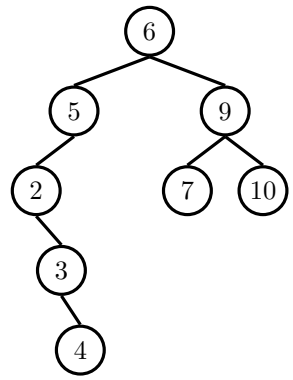
Trees

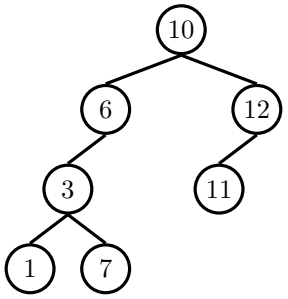
Topical Review Section 5: March 7, 2021

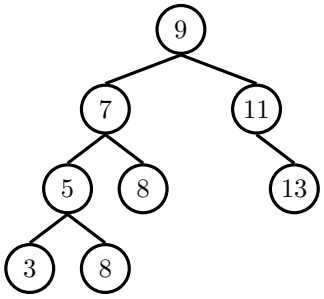
1 Is BST?

(a) Given the following binary trees, determine if each is a binary search tree, and whether the height of the tree is the same as the height of the optimal binary search tree containing the given elements.

Note: Height of a tree is calculated by counting number of edges from the root to the furthest leaf node.







Note: Optimal height of a BST with N nodes is given by floor($\log_2(N)$).

1. Valid but not balanced (Compare height of tree with optimal height)
2. Invalid and not balanced (Node with value 7 should not be to the left of 6 and height is not optimal)
3. Invalid but balanced (No duplicates are allowed in BSTs)

(b) The following method `isBSTGood` is supposed to return `False` if a given tree is not a BST. Unfortunately it is returning the wrong answer when given some binary trees. Think about an example of a binary tree for which `isBSTGood` fails. The `TreeNode` class is defined as follows. Assume that there are never any duplicate elements.

```
class TreeNode {
    int val;
    TreeNode left;
    TreeNode right;
}
```

Hint: You will find `Integer.MIN_VALUE` and `Integer.MAX_VALUE` helpful when writing `isBSTGood`.

```
public static boolean isBSTGood(TreeNode T) {
    if (T == null) {
        return true;
    } else if (T.left != null && T.left.val > T.val) {
        return false;
    } else if (T.right != null && T.right.val < T.val) {
        return false;
    } else {
        return isBSTGood(T.left) && isBSTGood(T.right);
    }
}
```

An example of a binary tree for which the method fails:

```
      10
     / \
    5 15
   / \
  3 12
```

The method fails for some binary trees that are not BSTs since it only checks that the value at a node is greater than its left child and less than its right child, not that its value is greater than every node in the left subtree and less than every node in the right subtree. Above is one example of a tree for which it fails.

By the way, the method does return true for every binary tree that actually is a BST.

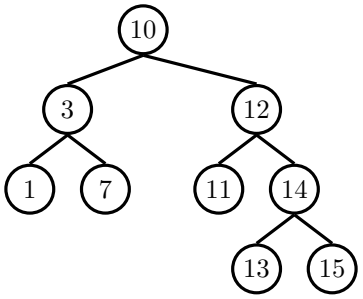
(c) Rewrite `isBSTGood` so that it returns true then the given tree is a BST and false otherwise.

```
public static boolean isBSTGood(TreeNode T) {
    return isBSTHelper(T, Integer.MIN_VALUE, Integer.MAX_VALUE);
}

public static boolean isBSTGoodHelper(TreeNode T, int min, int max) {
```

```
    if (T == null) {
        return true;
    } else if (T.val < min || T.val > max) {
        return false;
    } else {
        return isBST(T.left, min, T.val) && isBST(T.right, T.val, max);
    }
}
```

2 Tree Traversals



Write the pre-order, in-order, post-order, and level-order traversals of the above binary search tree.

Pre-order: 10 3 1 7 12 11 14 13 15

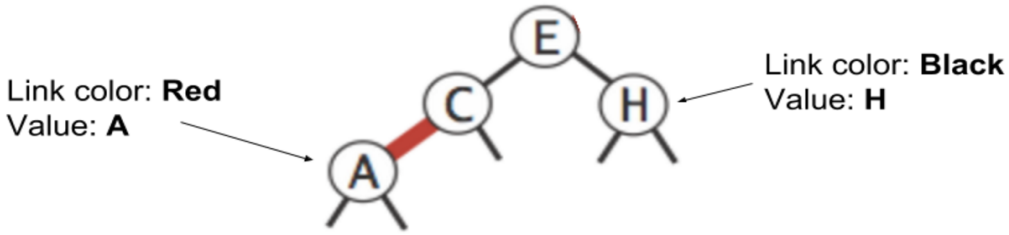
In-order: 1 3 7 10 11 12 13 14 15

Post-order: 1 7 3 11 13 15 14 12 10

Level-order (BFS): 10 3 12 1 7 11 14 13 15

3 What color am I?

For each of the situations below in a valid LLRB tree, indicate whether the node's link to its parent is red, black, or either.



Questions:

(a) _Black_ The largest value in a valid B-tree with more than one node.
The largest value is always a right child, so must be black

(b) _Red_ The smallest value in a valid B-tree with more than one node.
A two node 2-3 tree has a red smallest child. A three node (insert 1,2,3) 2-3 tree has a black smallest child

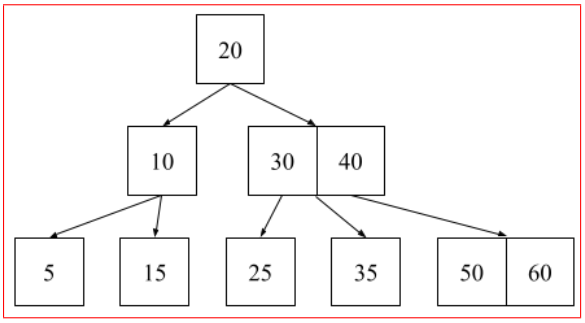
(c) _Black_ A node whose parent is red (parent's link color is red).
We cannot have two consecutive red nodes

(d) _Either_ A node whose children are the same color.
A node with two children of the same (which must be black) can be either red or black

(e) _Either_ A freshly inserted node after the insertion operation is completed.
When the insertion operation for a BST has completed, the node may be either red or black depending on rotations and color flips

4 Balanced Search Tree Mechanisms

(a) Insert the following numbers in order into a 2-3 tree: 20, 10, 35, 40, 50, 5, 25, 15, 30, 60.



(b) Draw a red-black tree that corresponds to the 2-3 tree you drew in (a).