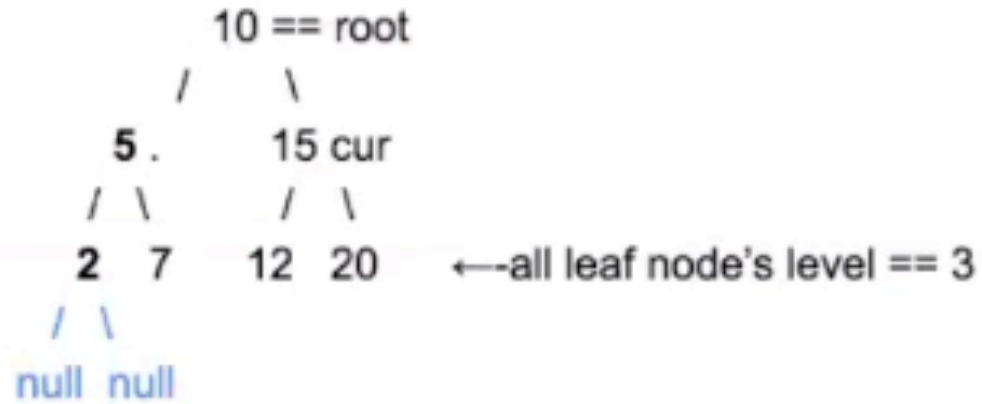
Class 4 Binary Tree & Binary Search Tree

Binary Tree

Definition: at most two children node.

Binary Tree Example:

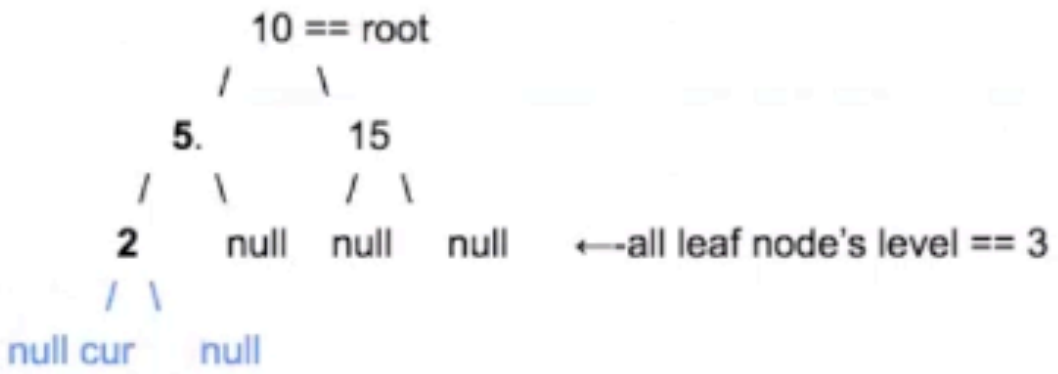



```
ListNode
class TreeNode {
      int value;
      TreeNode* left;
      TreeNode* right;   // 0xFFFF0001
      TreeNode* parent; // point to this node's parent node.
```

Trick: base case for recursion + binary tree related questions usually refers to the null ChildNode below the leaf node.

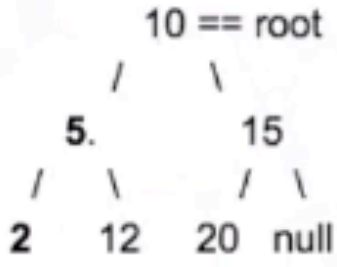
基本概念

- **Balanced binary tree**: is commonly defined as a binary tree in which the depth of the left and right subtrees of **every node** differ by 1 or less

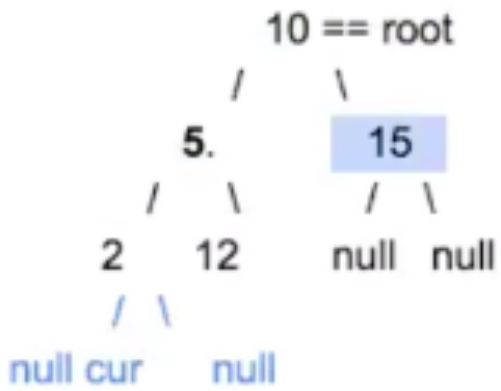


Conclusion 1: If a tree has n number of nodes., and it is balanced, then the height(level) of the tree = O(log_2(n))

Complete binary tree: is a binary tree in which every level, *except possibly the last*, is completely filled, and all nodes are as far left as possible

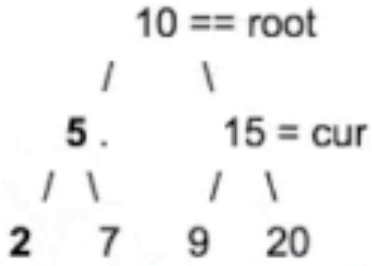


Complete binary tree: is a binary tree in which every level, *except possibly the last*, is completely filled, and all nodes are as far left as possible



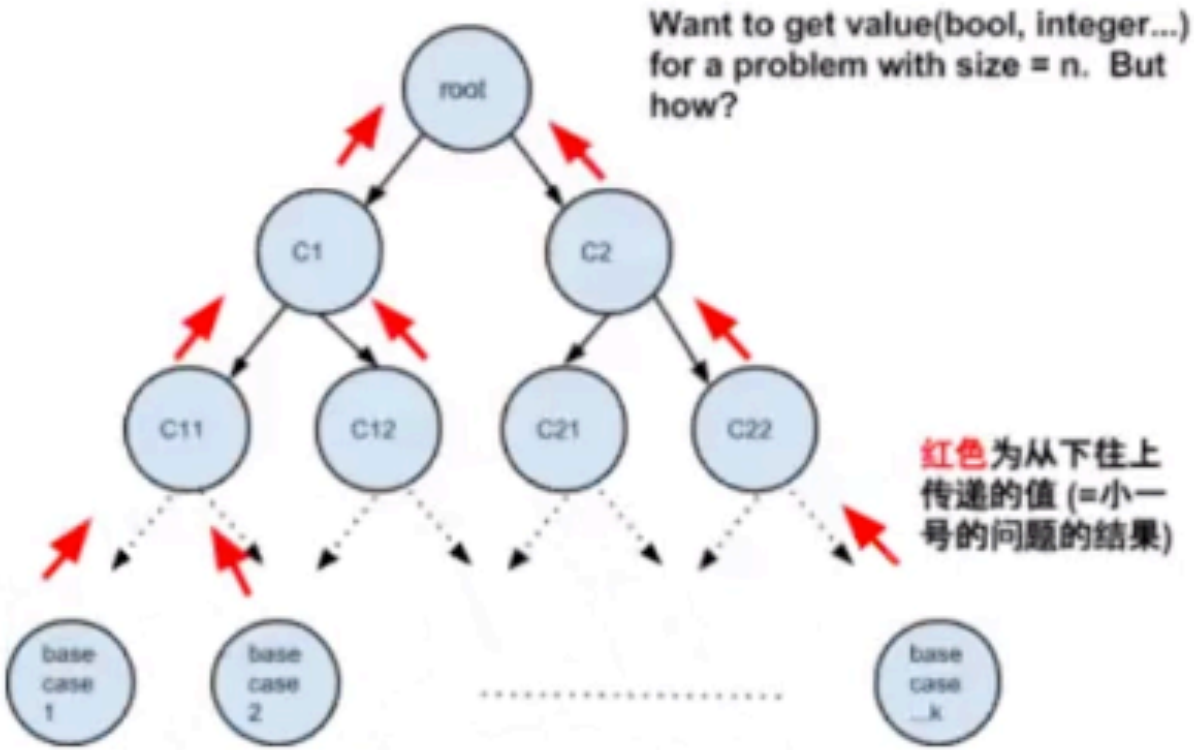
Conclusion 2: If a tree is a complete tree, then it must be a balanced tree.

Binary Search Tree (BST): for every single node in the tree, the values in its left subtree are all smaller than its value, and the values in its right subtree are all larger than its value.

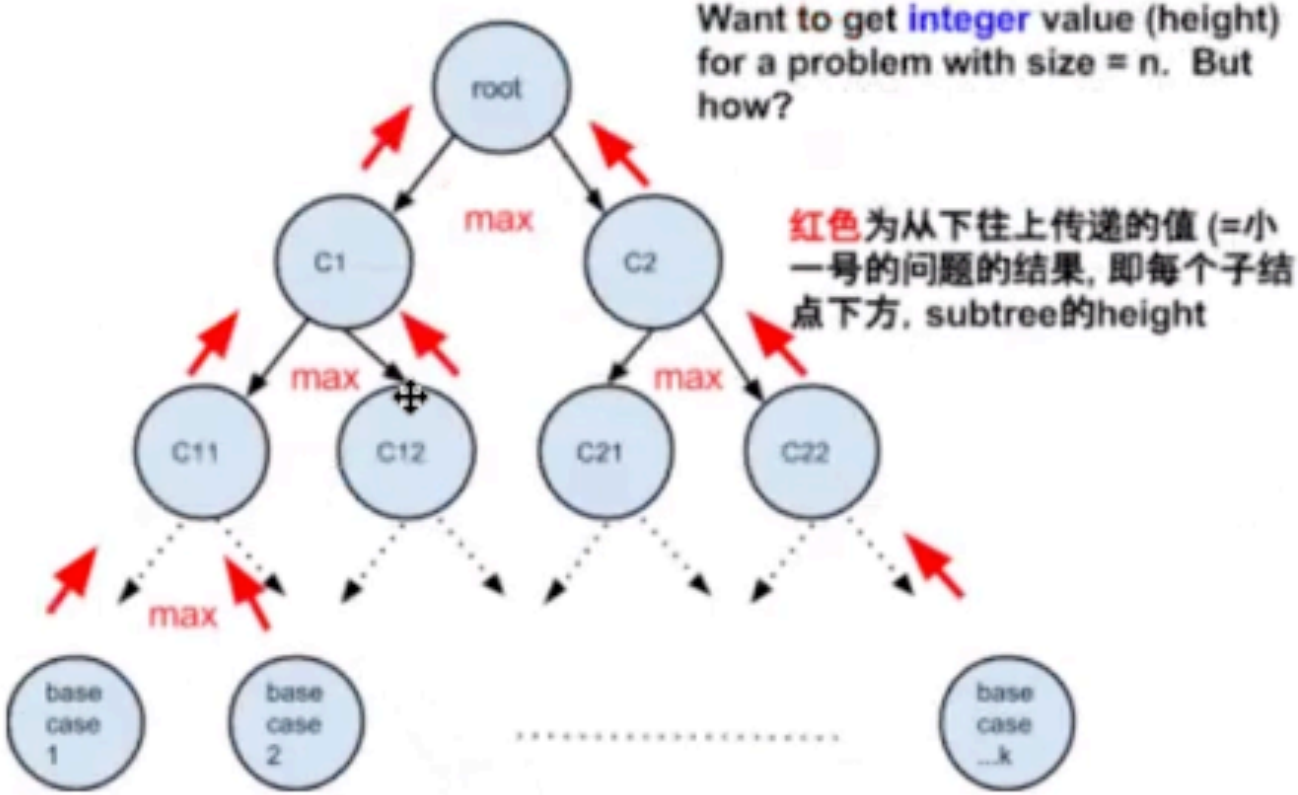


Conclusion 3: If we print the value of the nodes in BST in in-order sequence, then it must form an ascending order.

Discussion (High Level)



- Binary tree 往往是最常见的，和recursion 结合最紧密的面试题目类型。
- Reasons:
  - 每层的node 具备的性质，传递的值和下一层的性质 往往一致。比较容易定义 **recursive rule**.
  - **Base case** (generally): null pointer under the leaf node
  - Example1: **`int getHeight(Node root)`**
  - Example2: 统计tree里边有多少个node?
- Fundamental Knowledge:
  - Traversal of a binary tree
  - Definition



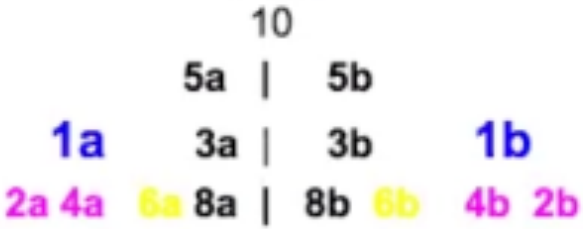

```
int GetHeight (TreeNode *root) {       // O(n), n is the total number of nodes in the subtree
        if (root == NULL) return 0;      // base case;
        int leftHeight  = GetHeight(root->left);
        int rightHeight = GetHeight(root->right);

        return 1 + max(leftHeight,  rightHeight);
}
```

Q1. How to determine whether a binary tree is a **balanced** binary tree?

(This is NOT an optimal solution)

```
00 public boolean isBalanced(TreeNode root) {
      // base case
      if (root == null) {
            return true;
      }
      int leftHeight = GetHeight(root.left);
      int rightHeight = GetHeight(root.right);
      if (Math.abs(leftHeight - rightHeight) > 1) {
            return false;
      }
      return isBalance(root.left) && isBalance(root.right);
}
```

Q2 How to judge whether a binary tree is symmetric?



(L.L vs R.R) && (L.R vs R.L)

Q2 How to judge whether a binary tree is symmetric?

10

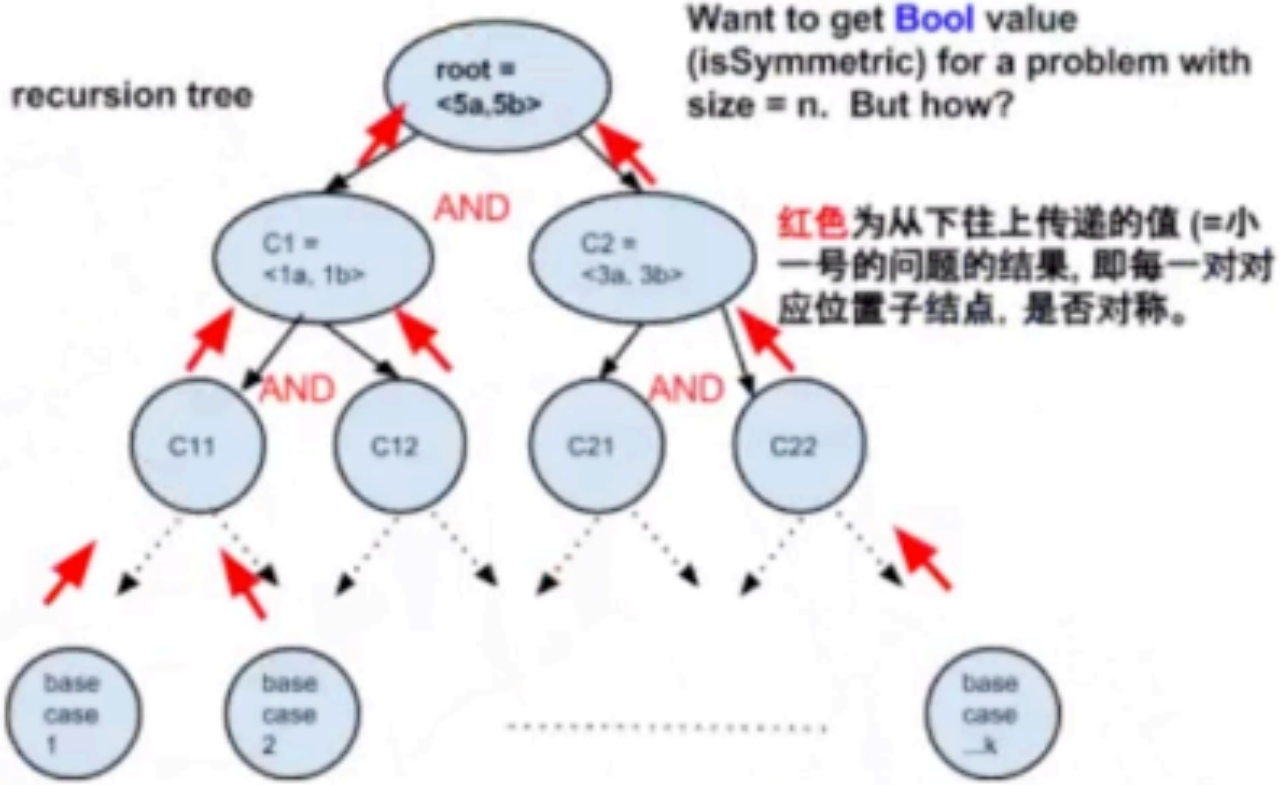
5a | 5b — Time on this level = O(1)

1a 3a | 3b 1b — Time on this level = num of node of this level/2

2a 4a 6a 8a | 8b 6b 4b 2b — Time on this level = num of node of this level/2

....

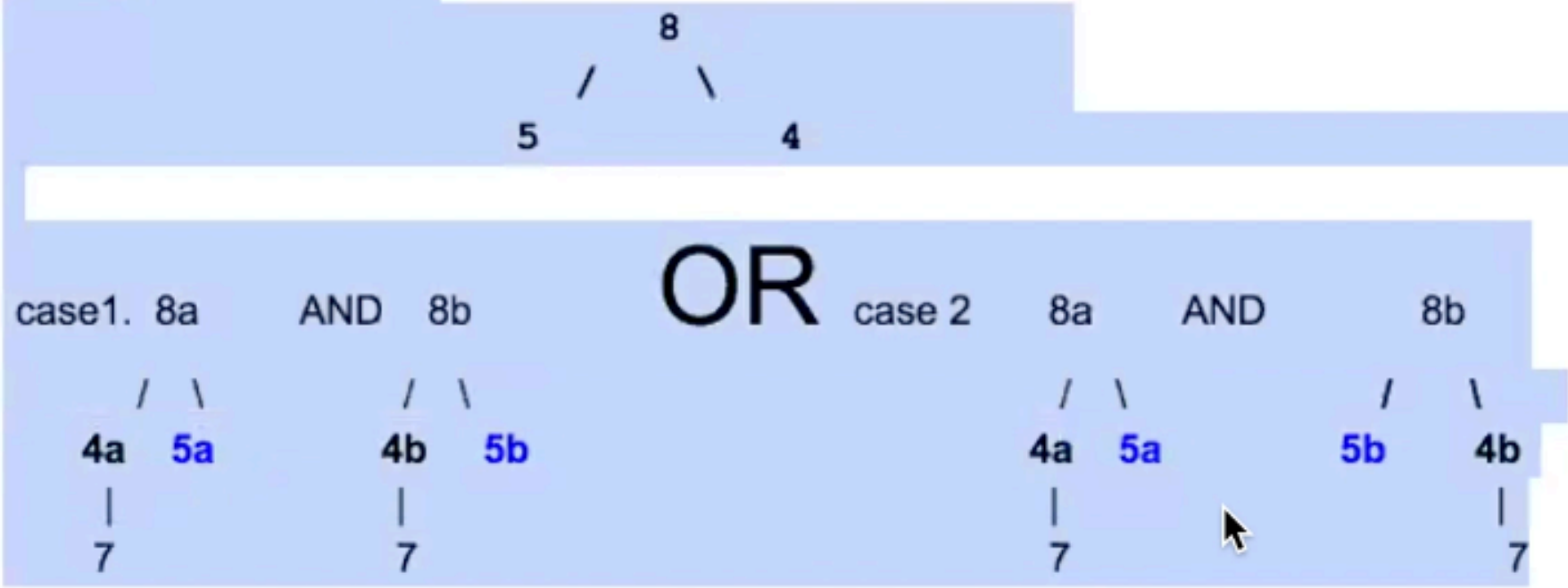
(L.L vs R.R) && (L.R vs R.L)

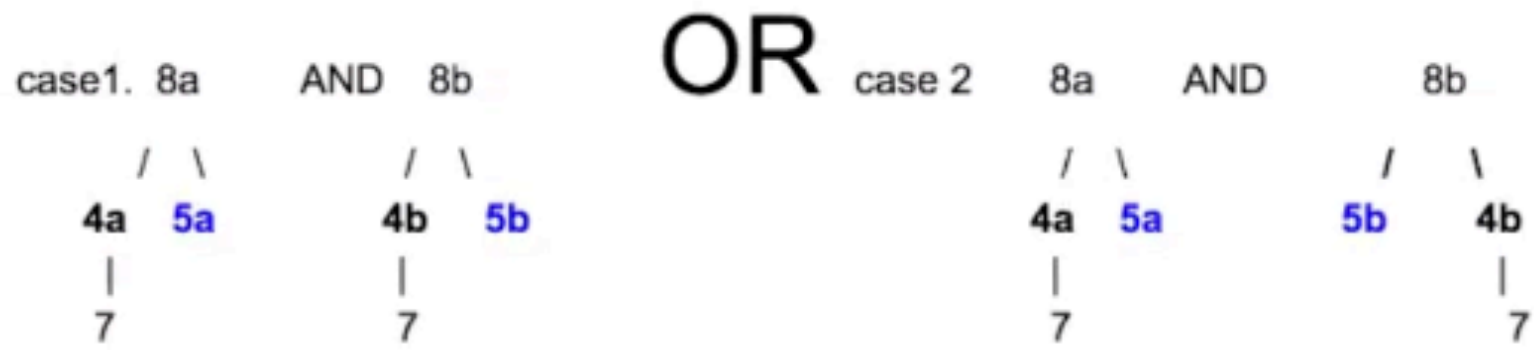


Want to get Bool value (isSymmetric) for a problem with size = n. But how?

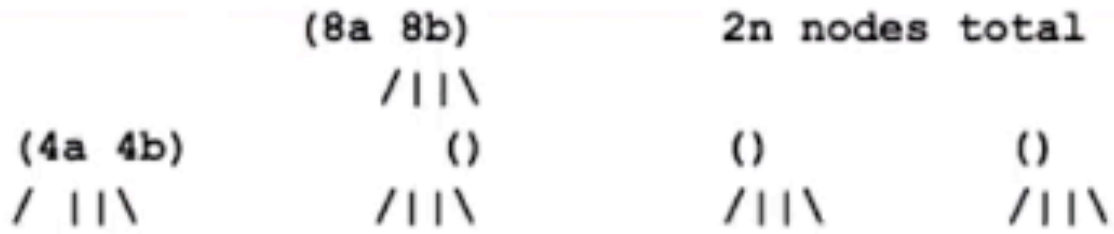
红色为从下往上传递的值 (=小一号的问题的结果, 即每一对对应位置子结点, 是否对称。

Q3. Let's assume if we tweak the lchild with rchild of an arbitrary node in a binary tree, then the "**structure**" of the tree are not changed. Then how can we determine whether two binary trees' structures are identical.





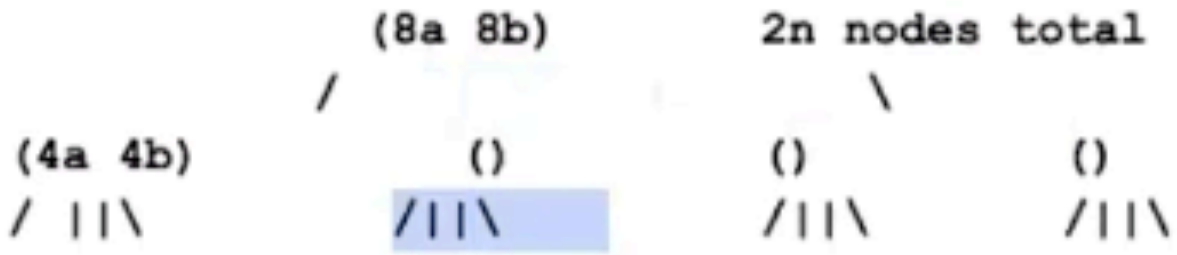

```
boolean  IsSymmetric(TreeNode left,  TreeNode right) {
     if(left == null && right == null){
          Return true;              // case 1, both are null
     }else if(left == null || right == null){
          Return false;             // case 2, one side is null
     }else if(left.val != right.val){
          Return false;             // case 3, not null but value is
different
     }
     Return IsSymmetric(left.left, right.right) &&
IsSymmetric(left.right, right.left) ||              // case2
     IsSymmetric(left.left, right.left) && IsSymmetric(left.right,
right.right);                                       // Case1
}
```



HOw many levels in this recursion tree = O(log_2(n)) because the input tree has log_2(n) (We assume the input tree is balanced.)

Time complexity == total number of nodes in this quadral tree?

Total nodes = 4^(log_2(n)) = 2^ (2* log_2(n)) = 2^(log_2(n^2)) = **O(n^2)**



HOw many levels in this recursion tree = O(log_2(n)) because the input tree has log_2(n) **(We assume the input tree is balanced.)**

Time complexity == total number of nodes in this quadral tree?

Total nodes = 4^(log_2(n)) = 2^ (2* log_2(n)) = 2^(log_2(n^2)) = **O(n^2)**

Follow up: what if the input tree is NOT balanced.
(Ye Wang)

Discussion

Recursion在tree题目的基本应用大致分为2类用法

1. 把value从上往下传递 (then 从下往上)的题目
   1.1. BST 判定方法
   1.2.
2. 只把value 从下往上传递 （更为常见，必须熟练掌握）
   2.1. getHeight(Node* root) 是经典的把 **integer** value 从下往上传递的题目
   2.2. isBalanced(Node* root) 是把 **boolean** value 从下往上传递的题目
   2.3. isSymmetric(Node* root1, Node* root2) 是把 **boolean** value 从下往上传递的题目
   2.4. Assign the value of each node to be the total number of nodes that belong to its left subtree. (是把 **integer** value 从下往上传递的题目)