1. Validate Binary Search Tree

Medium

Given a binary tree, determine if it is a valid binary search tree (BST).

Assume a BST is defined as follows:

* The left subtree of a node contains only nodes with keys **less than** the node’s key.
* The right subtree of a node contains only nodes with keys **greater than** the node’s key.
* Both the left and right subtrees must also be binary search trees.

**Example 1:**

2  
 / \  
 1 3  
  
Input: [2,1,3]  
Output: true

**Example 2:**

5  
 / \  
 1 4  
 / \  
 3 6  
  
Input: [5,1,4,null,null,3,6]  
Output: false  
Explanation: The root node's value is 5 but its right child's value is 4.

**Solution**

误区：不能根据 root->val > root->left->val && root->val < root->right->val 判断，需要设定val的上下限

/\*\*  
 \* Definition for a binary tree node.  
 \* struct TreeNode {  
 \* int val;  
 \* TreeNode \*left;  
 \* TreeNode \*right;  
 \* TreeNode(int x) : val(x), left(NULL), right(NULL) {}  
 \* };  
 \*/  
typedef TreeNode\* pnode;  
class Solution {  
public:  
 bool isValidBST(TreeNode\* root) {  
 return judge(root, int(1 << 31 - 1), -int(1 << 31 - 1));  
 }  
 bool judge(pnode root, int lower, int upper){  
 if(root == NULL)return true;  
 if(lower != int(1 << 31 - 1) && root->val <= lower)return false;  
 if(upper != -int(1 << 31 - 1) && root->val >= upper)return false;  
 return judge(root->left, lower, root->val) && judge(root->right, root->val, upper);  
 }  
};