Problem 98: Validate Binary Search Tree

Problem Information

Difficulty: Medium
Acceptance Rate: 35.01%
Paid Only: No
Tags: Tree, Depth-First Search, Binary Search Tree, Binary Tree

Problem Description

Given the `root` of a binary tree, _determine if it is a valid binary search tree (BST)_.

A **valid BST** is defined as follows:

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

Example 1:

Input: root = [2,1,3] **Output:** true

Example 2:

Input: root = [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.

Constraints:

* The number of nodes in the tree is in the range `[1, 104]`. * `-231 <= Node.val <= 231 - 1`

Code Snippets

C++:

```
/**
* Definition for a binary tree node.
* struct TreeNode {
* int val;
* TreeNode *left;
* TreeNode *right;
* TreeNode() : val(0), left(nullptr), right(nullptr) {}
* TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
* TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left),
right(right) {}
* };
*/
class Solution {
public:
bool isValidBST(TreeNode* root) {

}
};
```

Java:

```
/**
* Definition for a binary tree node.
* public class TreeNode {
* int val;
* TreeNode left;
* TreeNode right;
* TreeNode() {}
* TreeNode(int val) { this.val = val; }
* TreeNode(int val, TreeNode left, TreeNode right) {
* this.val = val;
* this.left = left;
* this.right = right;
* }
* }
*/
class Solution {
```

```
public boolean isValidBST(TreeNode root) {

}
}
```

Python3:

```
# Definition for a binary tree node.
# class TreeNode:
# def __init__(self, val=0, left=None, right=None):
# self.val = val
# self.left = left
# self.right = right
class Solution:
def isValidBST(self, root: Optional[TreeNode]) -> bool:
```