98. Validate Binary Search Tree

Add to List

Question | Editorial Solution | My Submissions (/problems/validate-binary-search-tree/submissions/)

Total Accepted: **136527** Total Submissions: **609084** Difficulty: **Medium** Contributors: **Admin**

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
```

Binary tree `[2,1,3]`, return true.

Example 2:

```
  1
 / \
2   3
```

Binary tree `[1,2,3]`, return false.

Subscribe (/subscribe/) to see which companies asked this question

Hide Tags | Tree (/tag/tree/) | Depth-first Search (/tag/depth-first-search/)

Hide Similar Problems | (M) Binary Tree Inorder Traversal (/problems/binary-tree-inorder-traversal/)

Have you met this question in a real interview? Yes No

Discuss (https://discuss.leetcode.com/category/106) | Top Solutions | Pick One (/problems/random-one-question/)

C++

```cpp
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
class Solution {
public:
    bool isValidBST(TreeNode* root) {
        return helper(root,NULL,NULL);
    }
    bool helper(TreeNode* root,TreeNode* minNode,TreeNode* maxNode){
        if(root==NULL) return true;
        if(minNode&&root->val<=minNode->val||maxNode&&root->val>=maxNode->val) return false;
        return helper(root->left,minNode,root)&&helper(root->right,root,maxNode);
    }
};
```

Custom Testcase

Contribute Testcase

Send Feedback (mailto:admin@leetcode.com?subject=Feedback)