098. Validate Binary Search Tree

098 Validate Binary Search Tree

- **Depth-first Search** + tree

Description

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.

1. Thought line

2. Depth-first Search + tree

```
/**
 * 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 {
private:
    bool isValid(TreeNode* root, long leftBorder, long rightBorder){
        if (root==nullptr) return true;
        int node = root->val;
        if (node<=leftBorder || node>=rightBorder) return false;
        return isValid(root->left, leftBorder,node) && isValid(root->right, node,rightBorder);
    }
public:
```

```
    bool isValidBST(TreeNode* root) {
        long leftBorder = LONG_MIN;
        long rightBorder = LONG_MAX;
        return isValid(root, leftBorder,rightBorder);
    }
};
```