

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:



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

Example 2:



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

1. Thought line

2. Depth-first Search + tree

```
1 /**
2  * Definition for a binary tree node.
3  * struct TreeNode {
4  *     int val;
5  *     TreeNode *left;
6  *     TreeNode *right;
7  *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
8  * };
9  */
10
11 class Solution {
12 private:
13     bool isValid(TreeNode* root, long leftBorder, long rightBorder){
14         if (root==nullptr) return true;
15         int node = root->val;
16         if (node<=leftBorder || node>=rightBorder) return false;
17         return isValid(root->left, leftBorder,node) && isValid(root->right, node,rightBorder);
18     }
19 public:
20     bool isValidBST(TreeNode* root) {
21         long leftBorder = LONG_MIN;
22         long rightBorder = LONG_MAX;
23         return isValid(root, leftBorder,rightBorder);
24     }
25 }
```

