98. Validate Binary Search Tree

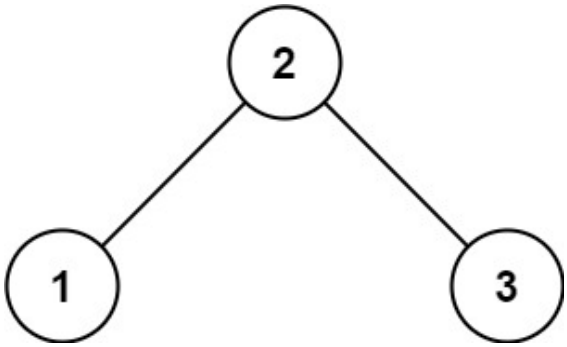
Created	@October 28, 2021 9:38 PM
Difficulty	Medium
LC Url	https://leetcode.com/problems/validate-binary-search-tree/
Importance	
Tag	BFS NEET Tree
Video	https://www.youtube.com/watch?v=ewrpOMA6LrY

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

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

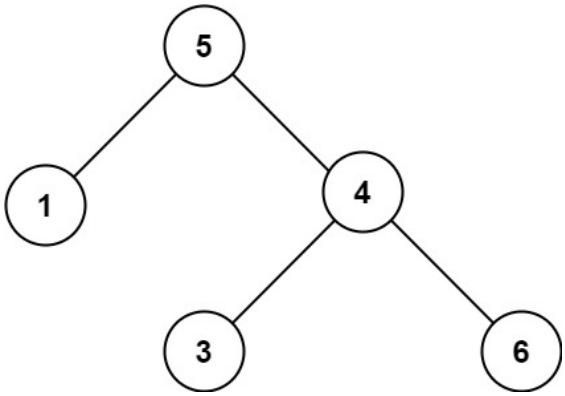
- The left subtree of a node contains only nodes with keys **less than** the node's key.
- The right subtree of a node contains only nodes with keys **greater than** the node's key.
- Both the left and right subtrees must also be binary search trees.

Example 1:




```
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, 10 4 ]`.
- `2 31  <= Node.val <= 2 31  - 1`

Solution

初学者做这题很容易有误区：BST 不是左小右大么，那我只要检查 `root.val > root.left.val` 且 `root.val < root.right.val` 不就行了？

这样是不对的，因为 BST 左小右大的特性是指 `root.val` 要比左子树的所有节点都更大，要比右子树的所有节点都小，你只检查左右两个子节点当然是不够的。

正确解法是通过使用辅助函数，增加函数参数列表，在参数中携带额外信息，将这种约束传递给子树的所有节点，这也是二叉搜索树算法的一个小技巧吧。

```
# 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:
        return self.check(root, None, None)

    # 限定以 root 为根的子树节点必须满足 max.val > root.val > min.val
    def check(self, root, min_node, max_node):
```

```
        if root is None:
            return True
        # 若 root.val 不符合 max 和 min 的限制，说明不是合法 BST
        if min_node and root.val <= min_node.val:
            return False
        if max_node and root.val >= max_node.val:
            return False
        # 限定左子树的最大值是 root.val，右子树的最小值是 root.val
        return self.check(root.left, min_node, root) and self.check(root.right, root, max_node)
```

使用DFS遍历树，遍历的时候向下传上下界，根据上下界判断是否符合要求

```
# 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:
        def check(node, left, right):
            if not node:
                return True

            val = node.val

            if val <= left or val >= right:
                return False

            return check(node.left, left, val) and check(node.right, val, right)

        return check(root, float('-inf'), float('inf'))
```