98. Validate Binary Search Tree

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

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

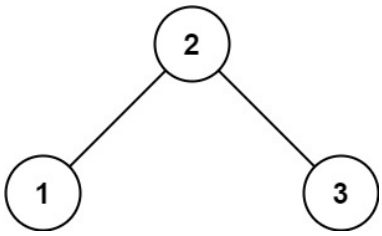
- The left

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.

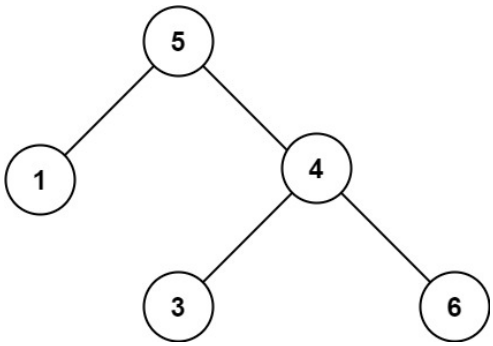
Example 1:



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

```
# Definition for a binary tree node.
# class TreeNode(object):
#     def __init__(self, val=0, left=None, right=None):
#         self.val = val
#         self.left = left
#         self.right = right
class Solution(object):
    def helper(self, root, prev, result):
        if root is None: return
        self.helper(root.left, prev, result)
        if prev[0] and root.val <= prev[0].val:
            result[0] = False
            return
        prev[0] = root
        self.helper(root.right, prev, result)


    def isValidBST(self, root):
        """
        :type root: TreeNode
        :rtype: bool
        """
        prev = [None]
        result = [True]
        self.helper(root, prev, result)
        return result[0]
```