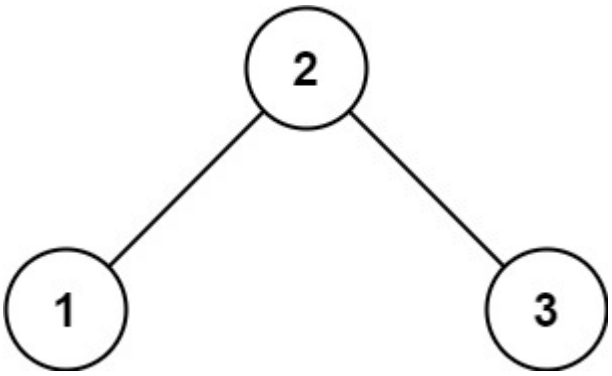
98. Validate Binary Search Tree

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

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

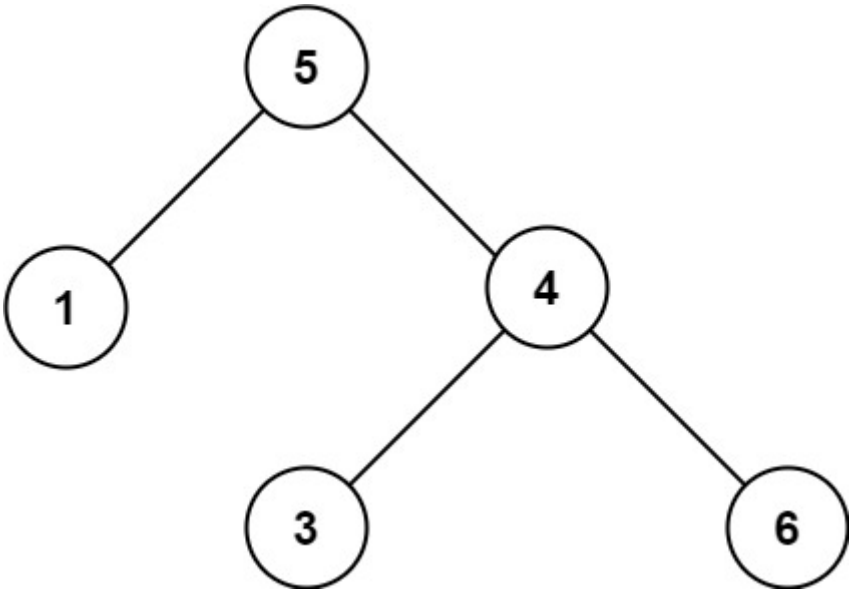
- The left subtree of a node contains only nodes with keys **less than** the node's key.
- The right subtree of a node contains only nodes with keys **greater than** the node's key.
- Both the left and right subtrees must also be binary search trees.

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<sup>4</sup>]`.
- `-2<sup>31</sup> <= Node.val <= 2<sup>31</sup> - 1`

```
import sys
class Solution:
    def isValidBST(self, root: Optional[TreeNode]) -> bool:
        low = -sys.maxsize
        high = sys.maxsize
        return self.isValidBSTHelper(root,low,high)

    def isValidBSTHelper(self,root,low,high):
        if root is None:
            return True
        if root.val<=low or root.val>=high:
            return False
        return self.isValidBSTHelper(root.left,low,root.val) and
self.isValidBSTHelper(root.right,root.val,high)
```

```
class Solution:
    def isValidBST(self, root: TreeNode) -> bool:
        nodes = []
        self.itemsOfBST(root,nodes)
        for i in range(1,len(nodes)):
            if nodes[i]-nodes[i-1]>0:
                continue
            else:
                return False
        return True









    def itemsOfBST(self,root,nodes):
        if root is None:
            return
        self.itemsOfBST(root.left,nodes)
        nodes.append(root.val)
        self.itemsOfBST(root.right,nodes)
```