0098. 验证二叉搜索树

ITCharge 大约 2 分钟

- 标签：树、深度优先搜索、二叉搜索树、二叉树
- 难度：中等

题目链接

- 0098. 验证二叉搜索树 - 力扣

题目大意

描述：给定一个二叉树的根节点 `root` 。

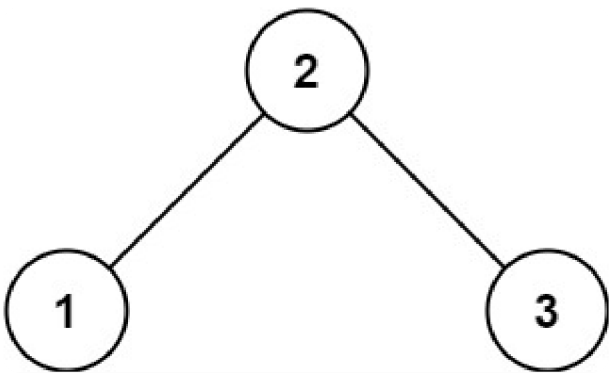
要求：判断其是否是一个有效的二叉搜索树。

说明：

- **二叉搜索树特征**：
  - 节点的左子树只包含小于当前节点的数。
  - 节点的右子树只包含大于当前节点的数。
  - 所有左子树和右子树自身必须也是二叉搜索树。
- 树中节点数目范围在$[1, 10^4]$ 内。
- $-2^{31} \leq Node.val \leq 2^{31} - 1$。

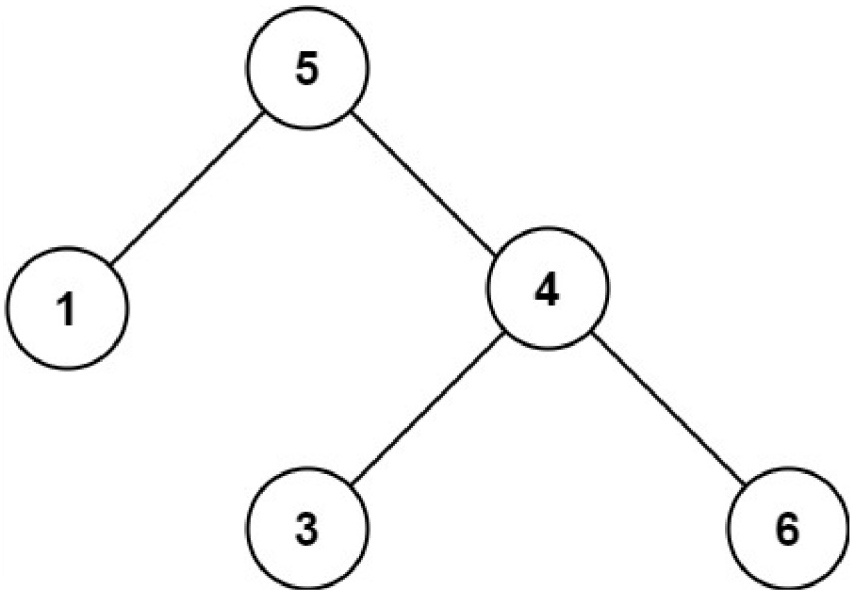
示例：

- 示例 1：



```py
输入：root = [2,1,3]
输出：true
```

- 示例 2：




```py
输入：root = [5,1,4,null,null,3,6]
输出：false
解释：根节点的值是 5 ，但是右子节点的值是 4 。
```

解题思路

思路 1：递归遍历

根据题意进行递归遍历即可。前序、中序、后序遍历都可以。

1. 以前序遍历为例，递归函数为： `preorderTraversal(root, min_v, max_v)` 。
2. 前序遍历时，先判断根节点的值是否在 `(min_v, max_v)` 之间。
    1. 如果不在则直接返回 `False` 。
    2. 如果在区间内，则继续递归检测左右子树是否满足，都满足才是一棵二叉搜索树。
3. 当递归遍历左子树的时候，要将上界 `max_v` 改为左子树的根节点值，因为左子树上所有节点的值均小于根节点的值。
4. 当递归遍历右子树的时候，要将下界 `min_v` 改为右子树的根节点值，因为右子树上所有节点的值均大于根节点。

思路 1：代码

```py
class Solution:
    def isValidBST(self, root: TreeNode) -> bool:
        def preorderTraversal(root, min_v, max_v):
            if root == None:
                return True
            if root.val >= max_v or root.val <= min_v:
                return False
            return preorderTraversal(root.left, min_v, root.val) and
preorderTraversal(root.right, root.val, max_v)

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

思路 1：复杂度分析

- **时间复杂度**：$O(n)$，其中 n 是二叉树的节点数目。
- **空间复杂度**：$O(n)$。递归函数需要 栈空间，栈空间取决于递归深度，最坏情况下递归深度为 n，所以空间复杂度为 $O(n)$。