LeetCode：98. 验证二叉搜索树 题解：

给定一个二叉树，判断其是否是一个有效的二叉搜索树。

假设一个二叉搜索树具有如下特征：

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。
示例 1:

```
输入:
    2
   / \
  1   3
输出: true
```

示例 2:

```
输入:
    5
   / \
  1   4
     / \
    3   6
输出: false
解释: 输入为: [5,1,4,null,null,3,6]。
     根节点的值为 5 ，但是其右子节点值为 4 。
```

来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/validate-binary-search-tree

简单分析：

1. 判断是否是二叉搜索树，题目写的很明显，左子树的所有结点都小于根节点，右子树的所有结点都大于根节点
2. 根据上面分析得出：求左子树的最大上界，再求出右子树的最小下界，root的值要在这两个值之间

   并不是只要左孩子小于根节点，右孩子大于根节点就好了，这样会存在右子树有小于root的元素，左子树有大于root的元素
3. 中序历遍的结果就是排序树中所有元素排序后的结果。因此只要中序历遍一次，查看序列是否有序即可。
4. 由于满足有序这个特点，我们并不需要全部保存下来，只需要知道每两个元素之间是否是符合递增关系即可

show code：

```
/**
 * Definition for a binary tree node.
```

```
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
class Solution {
public:
    vector<int> ans;
    bool flag = false;
    void DFS(TreeNode* root)
    {
        if (root == nullptr||flag) return;
        DFS(root->left);
        ans.push_back(root->val);
        if (ans.size() == 2)
            if (ans[1] > ans[0]) ans[0] = ans[1], ans.pop_back();
            else flag = true;
        DFS(root->right);
    }
    bool isValidBST(TreeNode* root) {
        DFS(root);
        return !flag;
    }
};
```