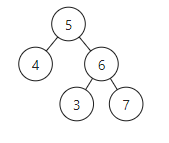
98.验证二叉搜索树

1.用先序遍历的递归法不能解决

按照之前做的，在二叉树上递归的方法，即单层逻辑为：判断左孩子的节点值是否小于当前根节点的节点值，右孩子的节点值是否大于当前根节点的节点值，是存在问题的。

因为二叉搜索树要求，左子树上的所有节点的值都小于根节点的值，右子树上的所有节点的值都大于根节点的值，上述方法将大小关系局限于子树内，遇到下面这样的二叉树就无法正确判断：



2.为了解决上述问题想到的递归法

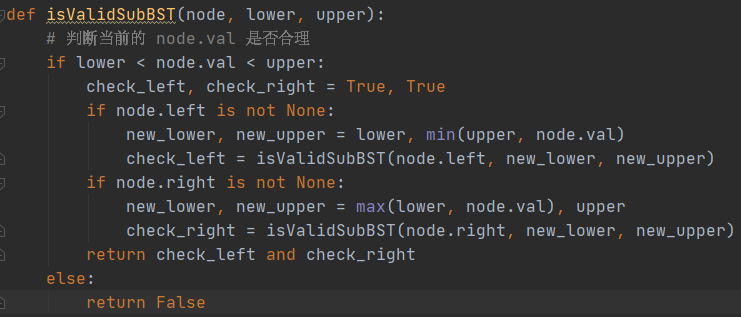
既然从根节点判断两个孩子的方法存在问题，可以将问题转换为判断当前节点的值是否满足条件。

条件是从根节点上递归传递下来的，有一个上界和一个下界。

对于左孩子来说，需要修改上界；对于右孩子来说，需要修改下界。

与其保存所有可能的上下界值，不如只保存最大的下界和最小的上界，只要满足这两个条件，剩下的一定也能满足。

注意这种方法，对于根节点，上下界应该是节点值取值范围的上下界。

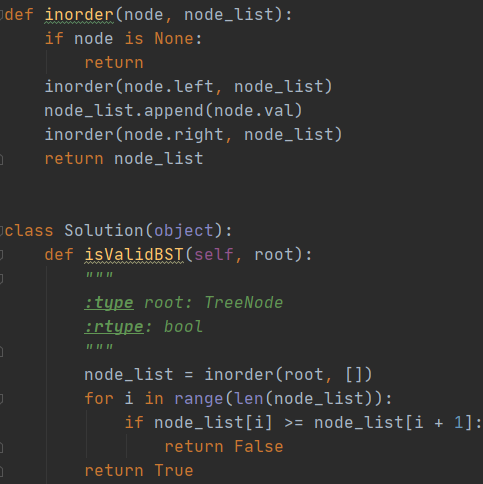


还有一个小问题是，为了防止左右子树判断时，上下界互相干扰，不要直接修改上下界的值，而是创建一个新的变量，用于保存新的上界或下界。

3. 真正解决问题的方法

二叉搜索树的性质：中序遍历得到的序列是有序的。

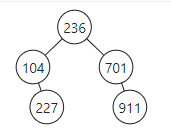
只要先对二叉搜索树做中序遍历，再对得到的序列判断是否有序即可：



530. 二叉搜索树的最小绝对差

1. 最小绝对差不一定发生在父子节点之间

因为二叉搜索树的性质，最小绝对差一定发生在中序遍历的相邻元素之间，但是中序遍历的相邻元素不一定是父子关系：



最小绝对差：236-227 = 9，但是227和236不是父子关系。

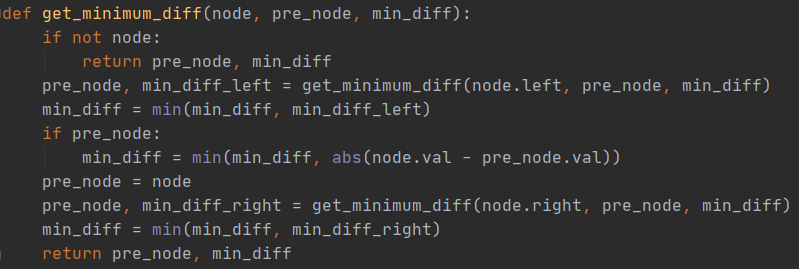
和98验证二叉树的问题一样，不能把递归局限在一棵父子关系的子树中。

2. 简易做法

先中序遍历二叉搜索树，再在有序序列中直接查找最小绝对差。

3. 对2的优化

可以在中序遍历中直接查找最小绝对差，需要记录上一个节点。



注意：在这种情况下，需要把pre\_node也作为返回值返回，否则在中序遍历中，无法更新pre\_node的值，会导致1那样的错误。

501. 二叉搜索树中的众数

目标是通过一次树上的中序遍历，就能够得到众数序列，并且不消耗多余的存储空间。

1. pre\_node初始条件判断

在这里，用于存储上一个节点值的pre\_node不是一个树节点，所以不能用if not pre\_node的方法判断初始条件，否则当pre\_node=0的时候会产生错误判断，对pre\_node进行错误的重新赋值。

从另一个角度来说，也可以把pre\_node设置为一个树节点或到树节点的指针，就像530题一样。

（Python的自动类型转换造成的问题）

1. 什么时候修改结果数组？

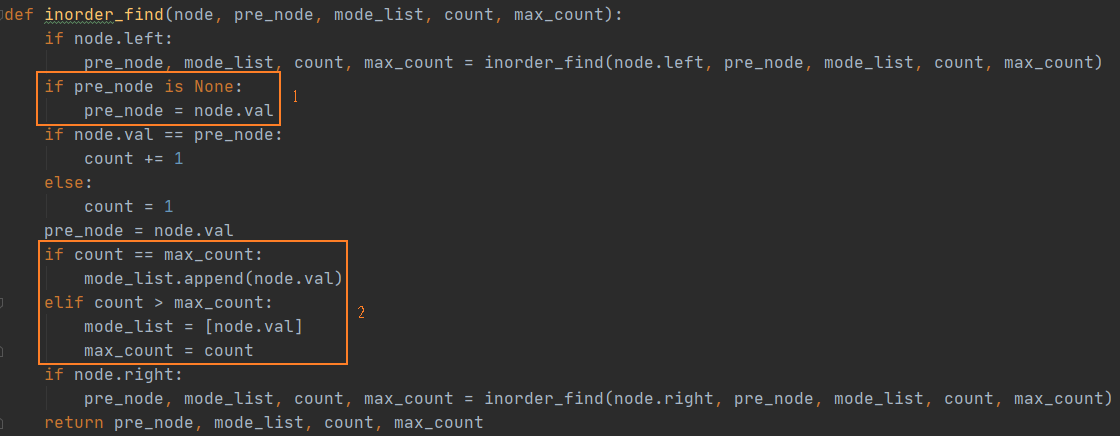
我一开始的想法是，当遍历到和pre\_node值不同的节点时，修改结果数组。

这样会产生两个问题：

① 需要添加到结果数组中的值是pre\_node的值，而不是node的值（因为此时count和max\_count的计数是pre\_node的）；

② 遍历结束之后，需要对最后遍历到的pre\_node值做一次额外的判断，否则如果最后遍历到的值也是众数的话，不能正常的加入结果数组（因为不会再有不同的节点了）。

与其考虑的这么麻烦，不如遍历到每个节点的时候，就直接判断一次是否能够加入结果数组或修改结果数组。



1. 类的成员变量

如上图所示，每次递归都需要把所有更新后的参数都返回，不仅很麻烦，而且很容易遗忘导致错误。

不如直接把这几个随着递归不断更新的变量作为类的成员变量，把递归函数也变成类的成员函数，这样就不需要每次回传这几个变量也能够实现随着递归更新了。