二叉搜索树的属性求解

有效二叉搜索树：对于二叉树中任意一节点，它的左子树节点值均小于它，右子树节点均大于它

二叉搜索树的重要性质：它的中序遍历是单调递增的

二叉搜索树中的搜索

本题用一个while循环即可，当root不为空时：如果val大于root的值，就让root往右走；val小于root的值，就让root往左走；如果val等于root的值就返回，循环外返回null（说明没找到）

写法上的细节：循环先判断val是否等于root的值，然后判断val大于/小于root的值时要写成一行，如果分开写这两种情况写成两个if，会导致第一个if结束后root可能为空，影响第二个if的判断

题目：

700. 二叉搜索树中的搜索（简单）

验证二叉搜索树

基于二叉树的定义可知：二叉树的**中序遍历**得到的值是单调递增的

本题思路：基于二叉树的中序遍历程序框架，比较栈弹出的节点值和上一次栈弹出的节点值的大小关系，如果出现本次弹出的值小于等于上次弹出的值，即不是二叉搜索树；程序结束后都未出现这种情况，就是二叉搜索树

具体实现：设置temp节点储存上次弹出栈的值，初始化为空，循环内（else分支内）判断条件是当temp不为空且temp的值大于等于当前弹出栈的节点值（即cur）时，就返回false，然后把cur传给temp，再对cur进行中序遍历的相应操作

题目：

98. 验证二叉搜索树（中等）

二叉搜索树的最小绝对差

本题思路：采用**中序遍历**迭代，同“验证二叉搜索树”类似，不断最小化当前弹出栈的节点值和上次弹出栈的节点值的差，最后得到结果

题目：

530. 二叉搜索树的最小绝对差（中等）

二叉搜索树中的众数

数据结构：众数指的是出现次数最多的数，**涉及到统计次数/频率，因而想到哈希表**

具体方法：

（1）先对二叉搜索树进行中序遍历（此处采用递归），用map型哈希表按遍历顺序将二叉树的节点值（key）和对应次数（value）加入

（2）遍历哈希表的键值对（Map.entry的用法），找到出现次数的最大值

（3）再次遍历哈希表的键值对，将出现次数最大值对应的节点值（key）加入list数组中

（4）将list数组中的元素加入最终答案

涉及到的写法（回顾）：

（1）将键值对加入哈希表：map.put(node.val, map.getOrDefault(node.val, 0) + 1);

（2）访问哈希表中的键值对：for (Map.Entry<Integer, Integer> entry : map.entrySet())

（3）访问哈希表键值对中的key/value：entry.getValue() / entry.getKey()

（4）访问list数组中对应位置的值：result.get(i)

题目：

501. 二叉搜索树中的众数（简单）