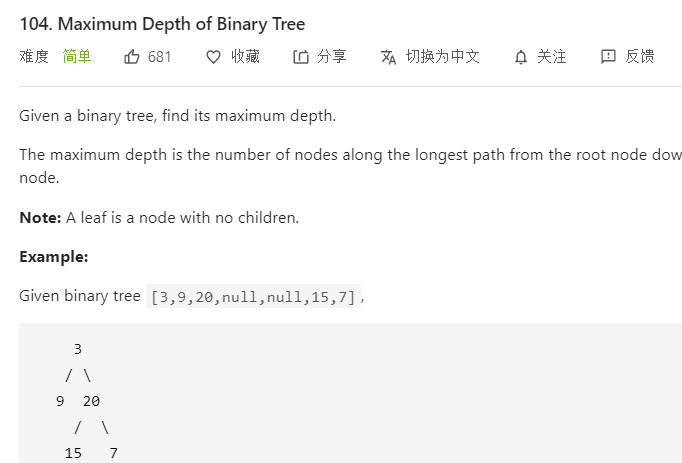
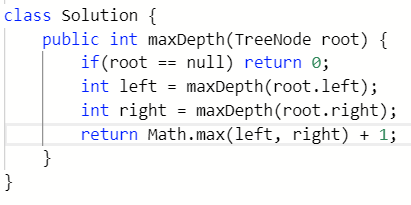
Tree

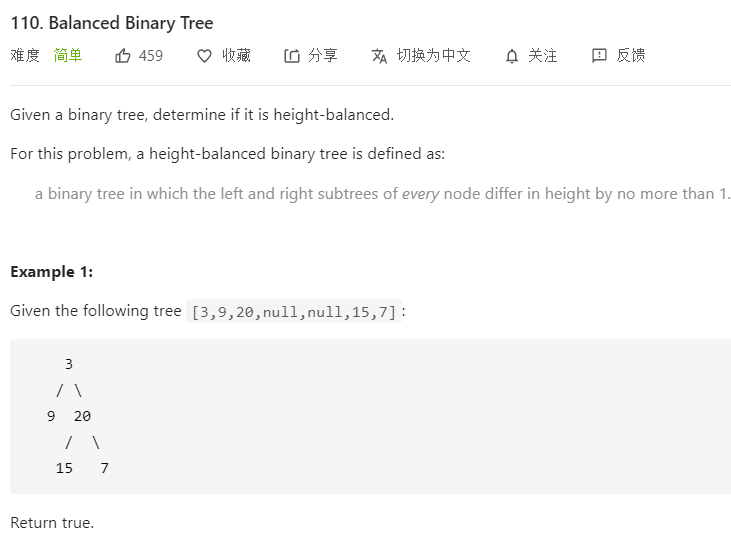
1.



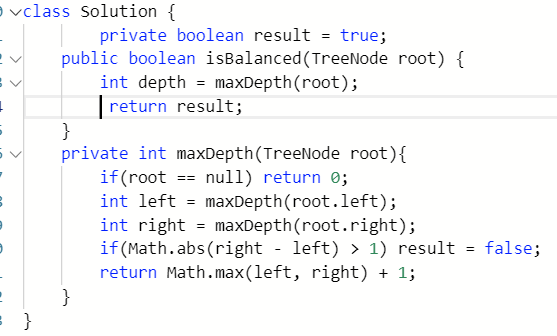
解法：用得到左右的最长深度最后+1



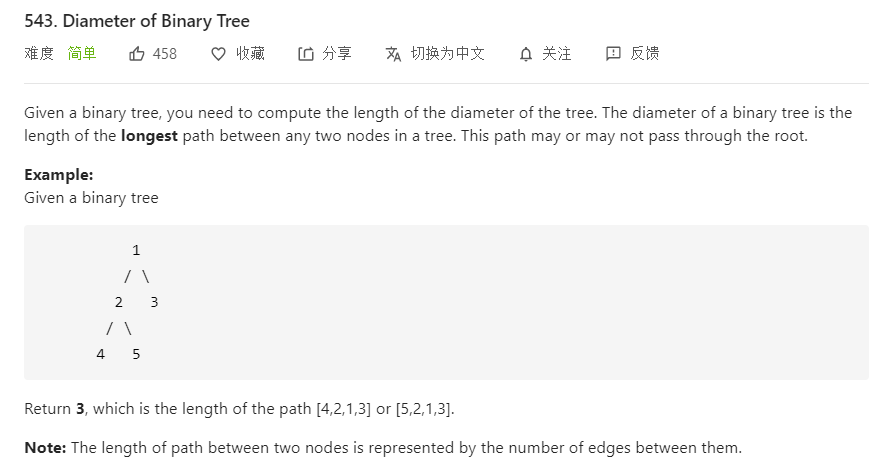
2.



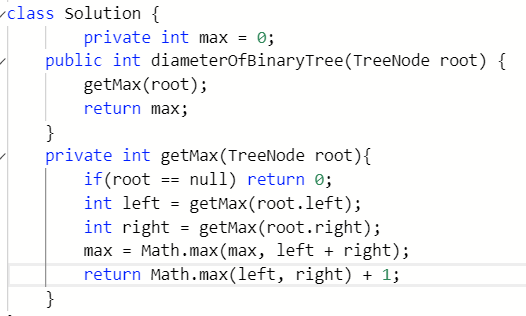
解法：和上一题一样只不过每次递归都看下左右相差，如果大于1 则变量设为false



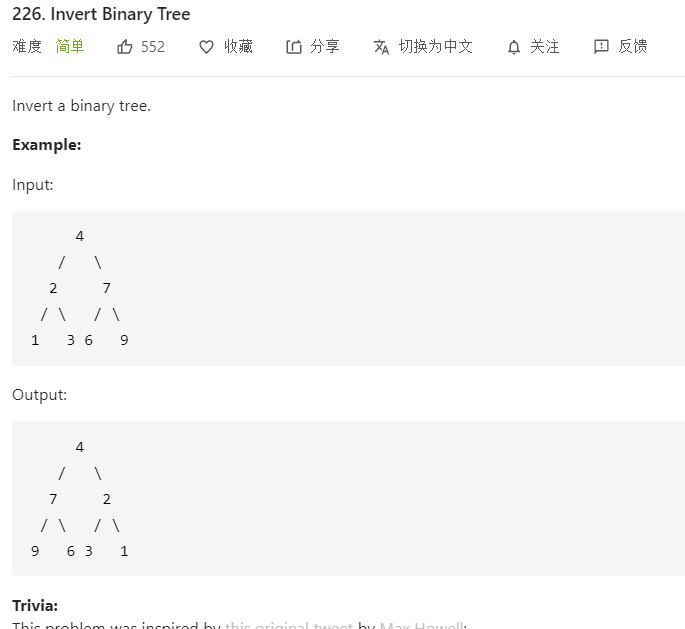
3.



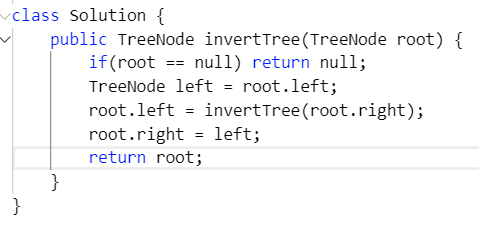
解法：用一个变量去记录每次递归左右的儿子数合的最大值



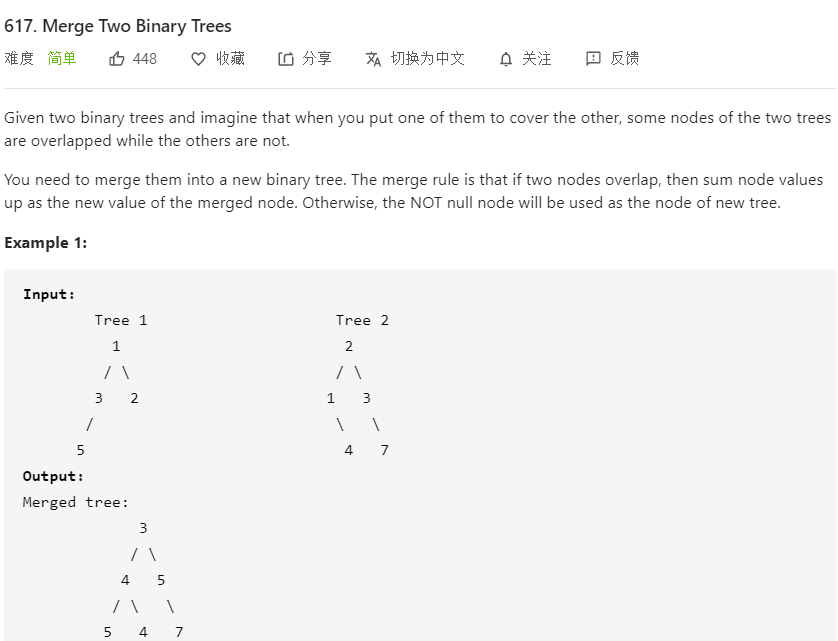
4.



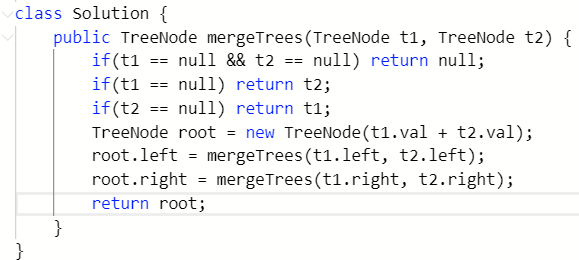
解法：用递归，先存下左儿子，再把左右儿子调换，最后返回root



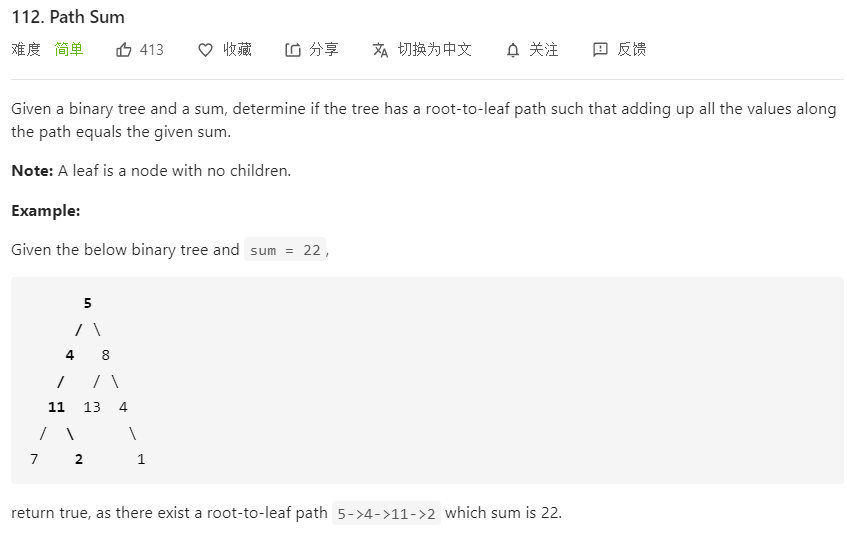
5.



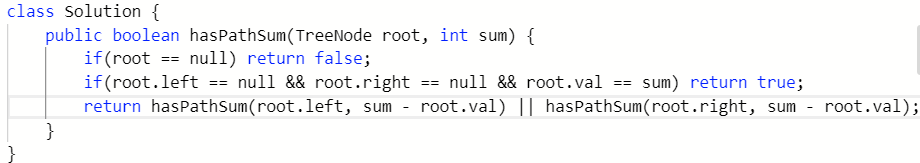
解法：先设置前置条件，如果树1是null则返回树2，反之亦然，然后创建新的节点root接左右儿子



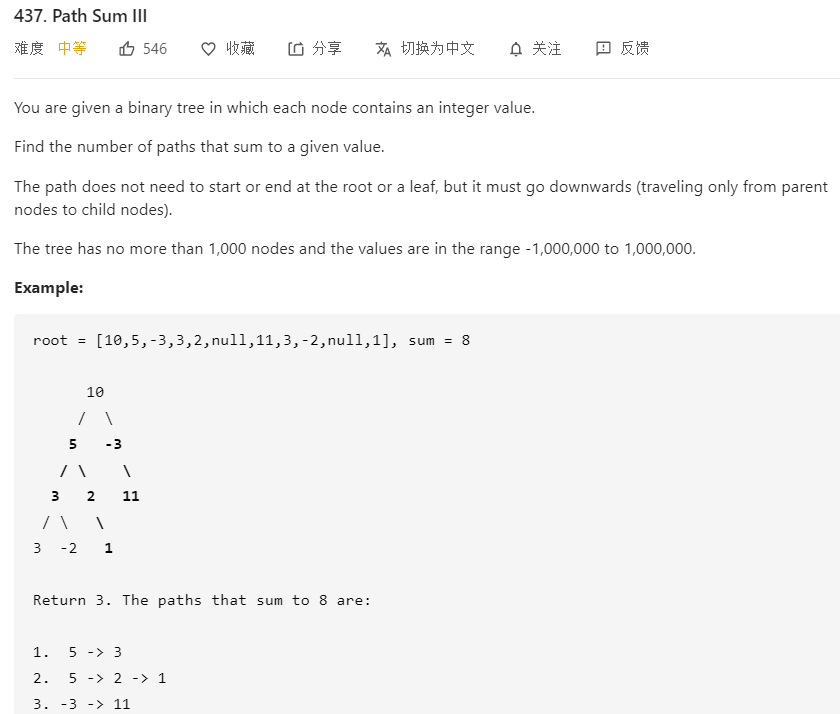
6.



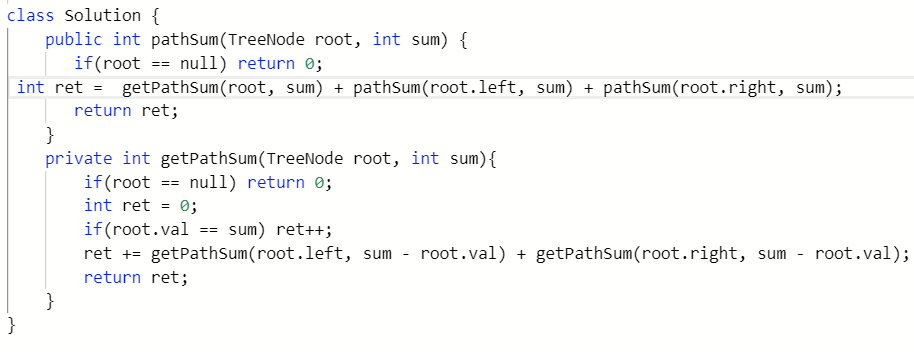
解法：如果root是null就说明没找到，如果当前节点的左右儿子都是null而且当前节点val = sum了就行，因为sum能减到与当前节点val一样。否则用当前节点的左儿子和sum-当前节点的val去找和右儿子一样去找，只要有一方找到就行



7.



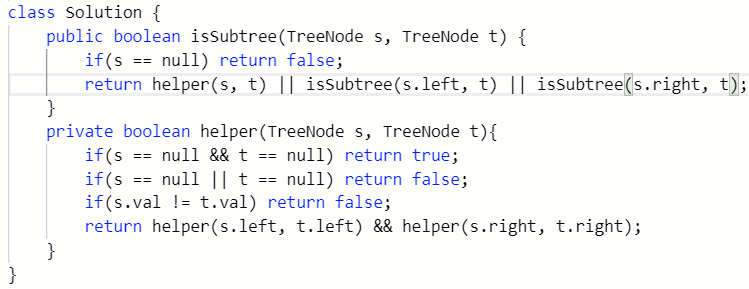
解法：把root放到一个方法里如果sum==root的val则ret值++接着去看左右儿子的ret的val，加起来。原root也要经历这个过程



8.



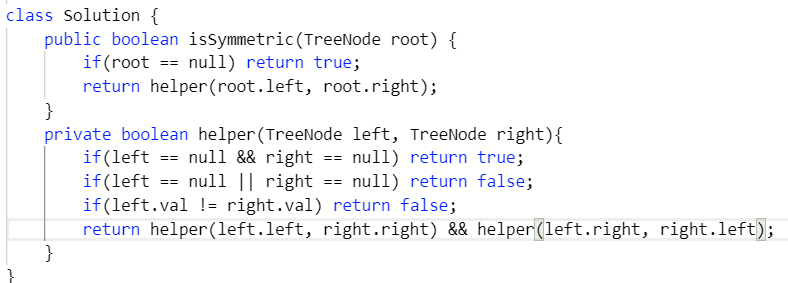
解法：对于两树root去看是不是他们同时为null，如果一个是null另一个不是则false，再比较val,不一样则也是false，再同时看root的左右，要都是true才行，同样的去看s的左儿子作为root和以及s的右儿子作为他的root去和t比较



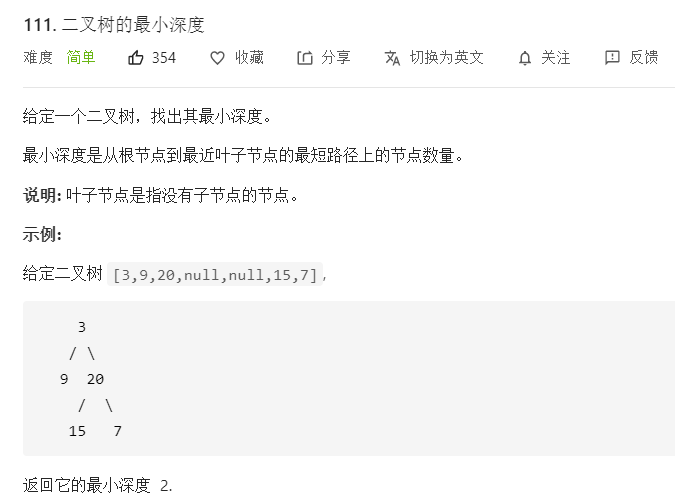
9.



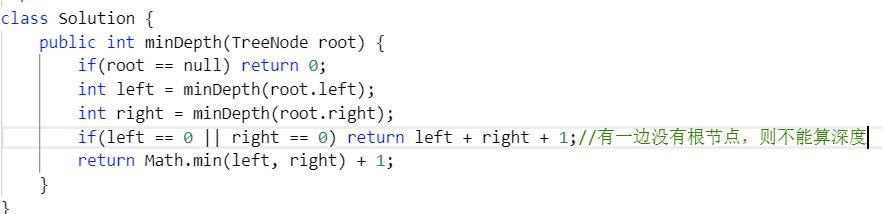
解法：去比较左右节点是不是同时为null，再去比较左节点的左儿子和右节点的右儿子，左节点的右儿子和右节点的左儿子



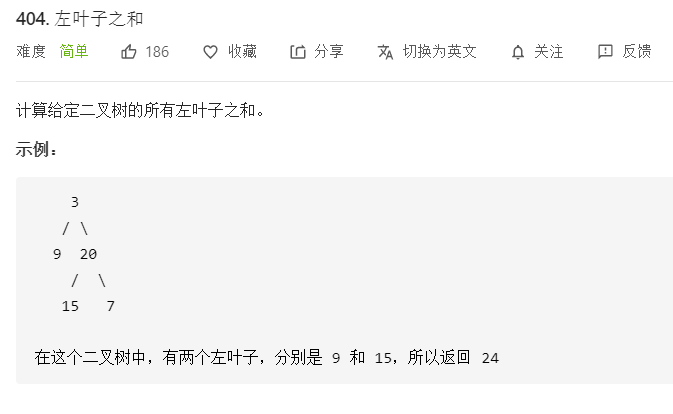
10.



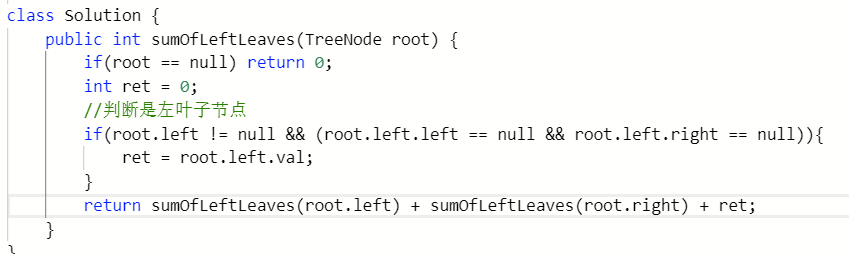
解法：用递归，然后用Math.min，但如果一侧没有儿子则不能去算深度



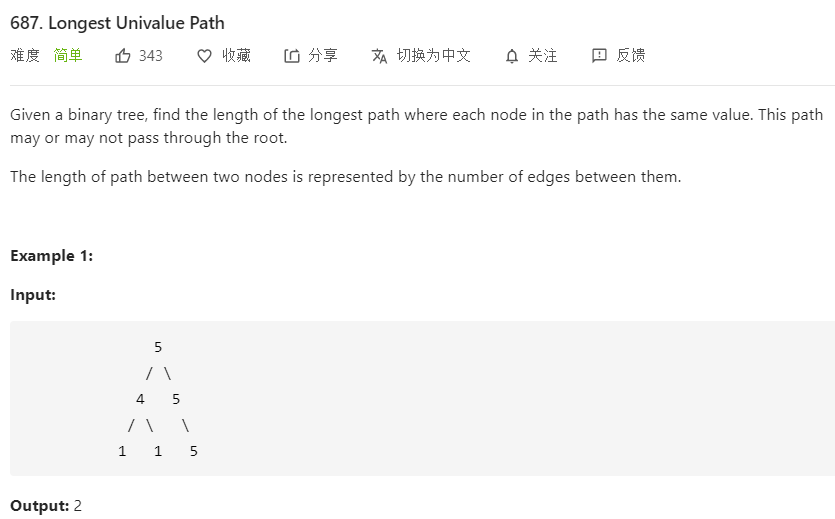
11.



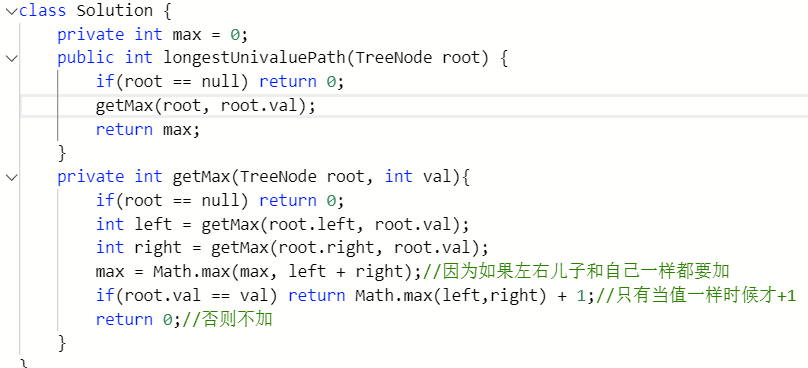
解法：要去找叶子节点里属于左儿子的节点，所以要去看左儿子和右儿子的左叶子节点并把结果返回



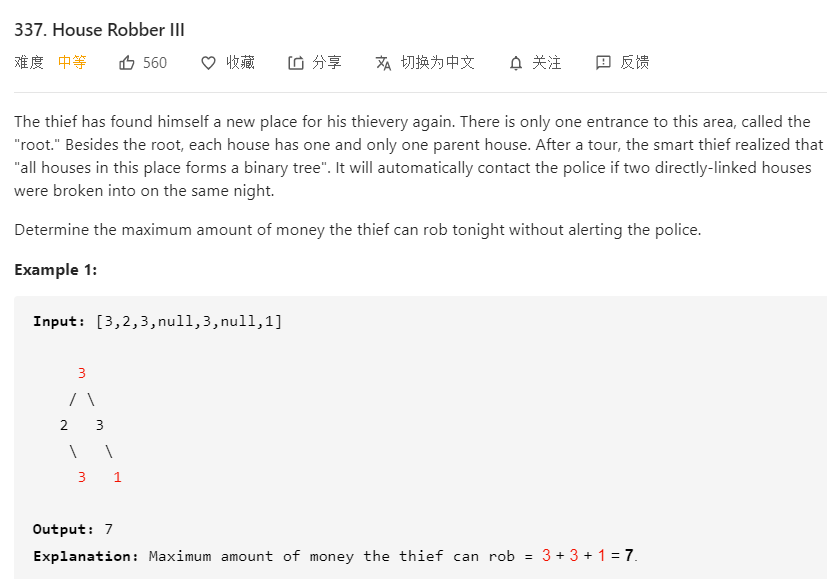
12.



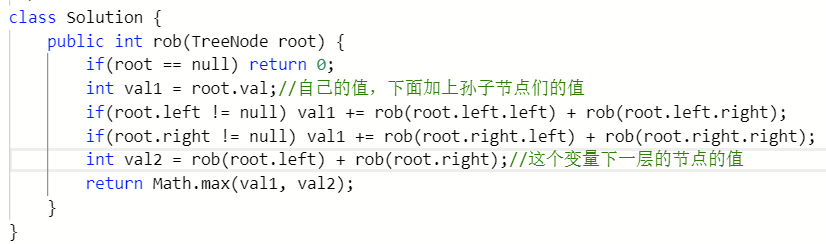
解法：



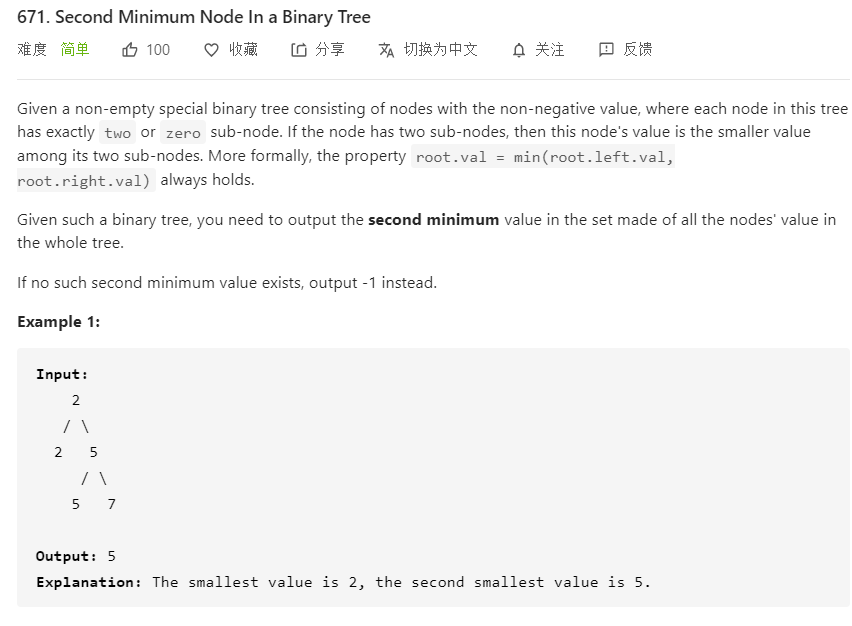
13.



解法：递归用两个值，一个记录自己和（如果下一层不是null）自己的两个孙子节点的和们，另一个记录自己的下一层的连个节点的和。最后返回两个值中大的那个



14.



题目：根节点是最小的，每个节点有0/2个儿子 - min heap 找第二小的节点

解法：如果当前节点不等于根节点，说明就是当前节点的值返回，



层序遍历BFS