**100. the same tree**

主要思路：递归recursion

递归有两大部分：终点（递归出口，逻辑边界）和自身调用部分（递归表达式，规律等）

写好一个递归算法要把握三个方面：1.提取重复逻辑 2.合适的退出条件3.控制逻辑边界。

终点：（我一般先写终点，比较容易想到且好写）

1. 合适的退出条件：

找到不一样的节点，return 0

1. 控制逻辑边界

当两个二叉树都为空时，即递归到了叶子结点时，return 1。

自身调用部分：

1. 提取重复逻辑：

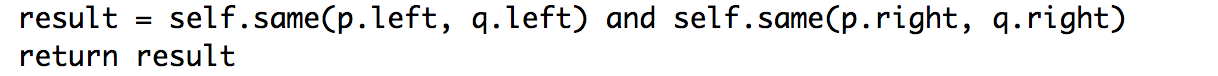
这题的重复逻辑很明显，就是判断每个节点的值是否相等。（p.val = q.val）

When何时用这部分:当然是没有到递归终点的时候。但是这一题不能浅显的只考虑除了递归终点的情况，因为这一题需要对结点的val做比较， 所以NoneType object（叶子结点）一定要被排除在外（if **p and q and** (p.val == q.val):），不然会报错**Line 21: AttributeError: 'NoneType' object has no attribute 'val'**

然后你可能会有疑问，这个刚刚不是在控制逻辑边界时考虑了吗？这里需要注意之前考虑的是两个二叉树都同时为空的情况，当一个为空另一个有值，就容易出bug了。比如：



if p and q and (p.val == q.val):这个判断语句也很巧妙，因为一定要注意判断的顺序，比如写成if (p.val == q.val) and p and q:就会报错。思考题：这是为什么？ 因为只有在先确定 p q不是NoneType的情况下，.val才不会报错

How这部分怎么计算:当两颗树的节点值一样时就可以对他们的字节点进行递归判断了，并且记录子节点的判断结果

What返回什么:

return 1: 当两个二叉树都为空时，即递归到了叶子结点时

return 0:节点不一样时

return result: 用and逻辑连接左右两支的递归返回结果。

