**543. Diameter of Binary Tree**

Easy

Given a binary tree, you need to compute the length of the diameter of the tree. The diameter of a binary tree is the length of the **longest** path between any two nodes in a tree. **This path may or may not pass through the root. (没看到这句话，被testcase坑惨)**

**Example:**  
Given a binary tree

1

/ \

2 3

/ \

4 5

Return **3**, which is the length of the path [4,2,1,3] or [5,2,1,3].

**Note:** The length of path between two nodes is represented by the number of edges between them.

简单来讲这道题就是找二叉树的直径，然后对直径的定义要注意，1.是the length of the **longest** path between any two nodes in a tree，也就是说不是找树的高度。2. **This path may or may not pass through the root.我之前没注意这句话，以为最长路径一定会通过root，然后还再想为什么一直有testcase不通过。**

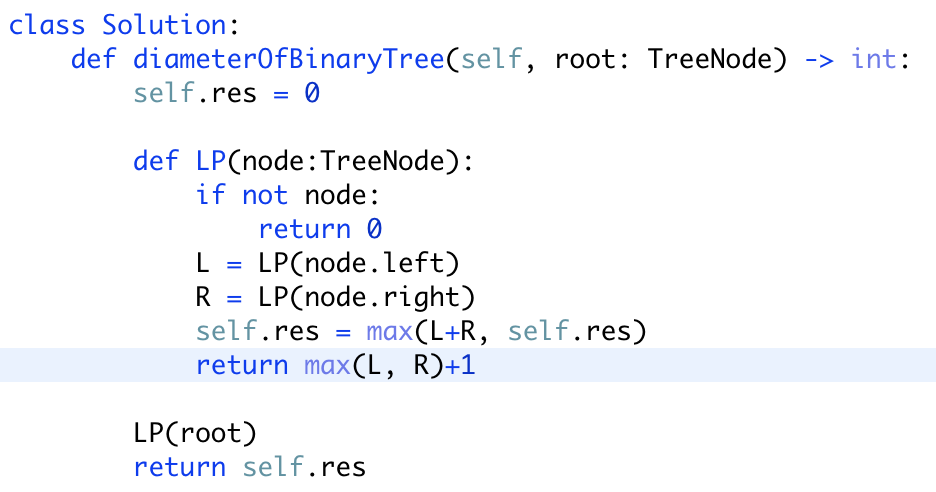
思路也比较好想，就是用DFS找到最深路径，也就是递归

1. 递归

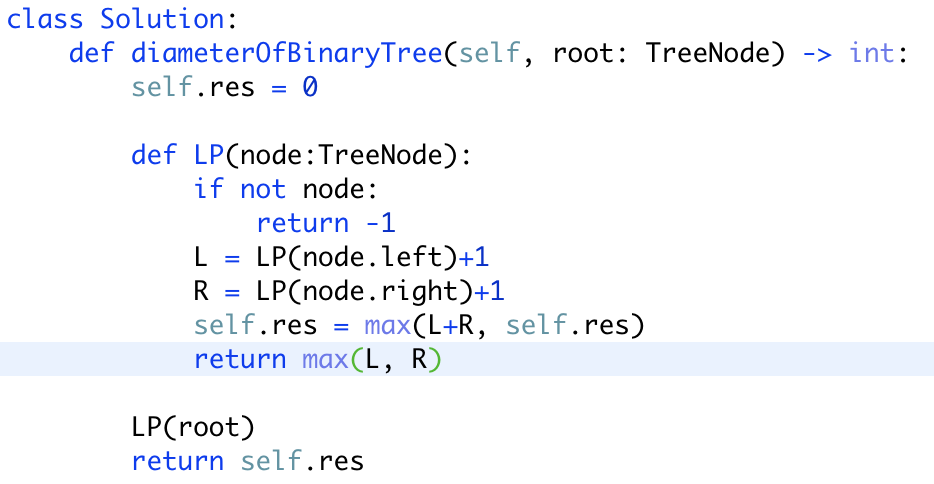
因为2那句话暗示了每一个node都可能成为支路接点（同时使用左右两支路的长度），所以我们需要一个变量res来存储目前最长直径，然后再在每一个节点和左右支路长度相加值做比较，记录最长值。Max(L+R, self.res)。

此外递归边界，即node = null时，按通常的逻辑应该返回0，也可以返回-1，这取决于你之前函数+1是怎么写的（例子放在后面）。

然后其他部分就是很惯常的操作了，没有特别的地方。分别计算左右支路的长度（调用自身）L=LP(node.left), R=LP(node.right)然后return max(L, R)这里加一与否取决于你前面有没有加一。

****

Return-1例子：



Tips：大家可以注意到，我这次调用函数的方式和以前不一样，这次新函数写在了原来函数的里面，为的是可以直接处理res变量，这样就不用将参数通过递归函数穿来穿去。然后需要注意，这里self的运用，因为定义在函数里面，所以定义和调用都不用加self（不用表示class solution的实例），而在调用res的时候就都有加上self（因为self是class里的属性变量，归solution这个类所有）

参考<https://blog.csdn.net/CLHugh/article/details/75000104>