hello

I am calling you on hangout

binary tree, left right

leaf value arr

1

2 3 [2, 3] leaf similar

1

2 3

4 5 [4,5,3]

1

4 2

5 3 [4,5,3]

find leaf of tree, True or False

1. input: tree root with API, leaf has order? 4 - 5 - 3, other tree 3-5-4 ? same order of leaf? left to right, value of node repeated? Y node order? N node value range? -10 000- 10 000
2. output: T/F
3. corner case: if one tree root is None? False; if both root is None？ T
4. brute force

Method:

1. traversal all node, save leaf in a list . time complexity O(N), space O(N)
   1. pre-order traversal
   2. level order traversal
2. compare two trees list tO(1) sO(1) # ??

time complexity O(N) space O(N)

class mySolution:

def judge(self, r1, r2): # output: T/F

if not r1 and not r2: # corner case

return True

if not r1 or not r2:

return False

l1 = self.findleaf(r1) # find leaf

l2 = self.findleaf(r2)

if l1 == l2:

return True

else:

return False

def findleaf(self, root): # ouput: list[leaf] left to right

if not root: # corner case

return []

res = []

if not root.left and not root.right: # root is leaf

res.append(root.val)

res += self.findleaf(root.left)

res += self.findleaf(root.right)

return res

**test code:**

corner case:

if one tree root is None? False

if both root is None？ True

case 1

1

2 3

4 5 [4,5,3]

case2

1

4 2

5 3 [4,5,3]

findleaf()

case1

root 1 res =[]

root.left = 2 = root

root.left = 4 = root self.findleaf(4)

root.left = None = root##

root 4 res [4]

root 5 res [4] + [5] = [4, 5]

root 3 root.left and root.right is None

[4,5,3]

case2

1

4 2

5 3 [4,5,3]

root 1

res []

root.l = 4

4.left and right == None

res [4]

root.r = 2 = root

root.l = 5, 5 left and right = None

res [4, 5]

root.r = 3, 3.left and right = None

res[4, 5, 3]

return