Sum Tree

Given a Binary Tree. Check whether it is a Sum Tree or not.

A Binary Tree is a Sum Tree in which value of each node x is equal to sum of nodes present in its left subtree and right subtree . An empty tree is also a Sum Tree as sum of an empty tree can be considered to be 0. A leaf node is also considered as a Sum Tree.

Input:
3
/ \
1 2

Output: 1
Explanation: The given tree is a sum
tree so return a boolean true.

Input:

10
/
20 30
/ \
10 10

Output: 0
Explanation: The given tree is not a sum
tree. For the root node, sum of elements
in left subtree is 40 and sum of elements
in right subtree is 30. Root element = 10
which is not equal to 30+40.

```
def isSumTree(self,root):
        # Code here
        self.check = True
        self.helper(root)
        return self.check

    def helper(self,root):
        if root is None:
            return 0
        if root.left is root.right:
            return root.data
        lt = self.helper(root.left)
        rt = self.helper(root.right)
        if root.data!=(lt+rt):
            self.check = False
        return root.data+lt+rt
```

Binary Search Site same question : https://binarysearch.com/problems/Sum-Tree

Given a binary tree `root`, return whether for every node in the tree other than the leaves, its value is equal to the sum of its left child's value and its right child's value.

Constraints

- `n ≤ 100,000` where `n` is the number of nodes in `root`

root = [9, [1, null, null], [8, [6, [6, null, null], null], [2, null, null]]]

Output

True

```
def solve(self, root):
        self.check = True
        self.helper(root)
        return self.check

    def helper(self,root):
        if root is None:
            return 0
        if root.left is root.right:
            return root.val
        lt = self.helper(root.left)
        rt = self.helper(root.right)
        if root.val!=(lt+rt):
            self.check = False
        return root.val+lt+rt
```