Problem 652: Find Duplicate Subtrees

Problem Information

Difficulty: Medium
Acceptance Rate: 60.43%
Paid Only: No
Tags: Hash Table, Tree, Depth-First Search, Binary Tree

Problem Description

Given the `root` of a binary tree, return all **duplicate subtrees**.

For each kind of duplicate subtrees, you only need to return the root node of any **one** of them.

Two trees are **duplicate** if they have the **same structure** with the **same node values**.

Example 1:

![](https://assets.leetcode.com/uploads/2020/08/16/e1.jpg)

Input: root = [1,2,3,4,null,2,4,null,null,4] **Output:** [[2,4],[4]]

Example 2:

![](https://assets.leetcode.com/uploads/2020/08/16/e2.jpg)

Input: root = [2,1,1] **Output:** [[1]]

Example 3:

![](https://assets.leetcode.com/uploads/2020/08/16/e33.jpg)

Input: root = [2,2,2,3,null,3,null] **Output:** [[2,3],[3]]

Constraints:

* The number of the nodes in the tree will be in the range `[1, 5000]` * `-200 <= Node.val <= 200`

Code Snippets

C++:

```
/**
* Definition for a binary tree node.
* struct TreeNode {
* int val;
* TreeNode *left;
* TreeNode *right;
* TreeNode() : val(0), left(nullptr), right(nullptr) {}
* TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
* TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left),
right(right) {}
* };
*/
class Solution {
public:
vector<TreeNode*> findDuplicateSubtrees(TreeNode* root) {

}
};
```

Java:

```
/**
* Definition for a binary tree node.
* public class TreeNode {
* int val;
* TreeNode left;
* TreeNode right;
* TreeNode() {}
* TreeNode(int val) { this.val = val; }
* TreeNode(int val, TreeNode left, TreeNode right) {
* this.val = val;
* this.left = left;
* this.right = right;
* }
* }
*/
class Solution {
public List<TreeNode> findDuplicateSubtrees(TreeNode root) {

}
}
```

Python3:

```
# Definition for a binary tree node.
# class TreeNode:
# def __init__(self, val=0, left=None, right=None):
# self.val = val
# self.left = left
# self.right = right
class Solution:
def findDuplicateSubtrees(self, root: Optional[TreeNode]) ->
List[Optional[TreeNode]]:
```