Find Duplicate Subtrees (/problems/find-duplicate-subtrees/)

Submission Detail

126 / 176 test cases passed.

Status: Wrong Answer

Submitted: **3 minutes ago**

```
Input:     [37,-34,-48,null,-100,-100,48,null,null,null,null,-54,null,-71,-22,null,null,null,8]

Output:    []

Expected:  [[-100]]
```

Submitted Code: 3 minutes ago

Language: cpp

Edit Code

```cpp
/**
 * 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:

    int compare(TreeNode* Nodei, TreeNode* Nodej, map<TreeNode*, int>& ans){
        if(Nodei == NULL and Nodej != NULL)
            return false;
        if(Nodei != NULL and Nodej == NULL)
            return false;
        if(Nodei == NULL and Nodej == NULL)
            return true;

        bool l = compare(Nodei->left, Nodej->left, ans);
        bool in = Nodei->val == Nodej->val? true:false;
        bool r = compare(Nodei->right, Nodej->right, ans);

        return (l and in and r);
    }
    void traverse_from_root(TreeNode* Nodei, TreeNode* root, map<TreeNode*, int>& ans){
        if(Nodei == NULL)
            return;
        if(root == NULL)
            return;
        traverse_from_root(Nodei, root->left, ans);
        if((Nodei->val == root->val) and (Nodei != root))
            if(compare(Nodei, root, ans) == true){
                if(ans.find(Nodei) == ans.end() and ans.find(root) == ans.end()){
                    ans[Nodei] = 1;
                    ans[root] = 0;
                }

            }

    }
    void solve(TreeNode* current_root, TreeNode* original_root, map<TreeNode*, int>& ans){
        if(current_root == NULL)
            return;

        solve(current_root->left, original_root, ans);
        traverse_from_root(current_root, original_root, ans);
        solve(current_root->right, original_root, ans);

    }
    vector<TreeNode*> findDuplicateSubtrees(TreeNode* root) {
        /*
        traverse in Inorder
        for each node
            traverse tree(Inoder) from root;
                if(both traverse are not pointing to same node AND both traverse                        have same v

                    then compare 2 trees (Nodei, Nodej)
                        if(both trees are equal AND Nodei or Nodej is not added already)
                            ans[Nodei reference]++;
                }
        */
        map<TreeNode*, int> ans;
        vector<TreeNode*> ans2;
        if(root->val == 0 and root->left == NULL){
            TreeNode* temp = new TreeNode(0);
            ans2.push_back(temp);
            return ans2;
        }
        solve(root, root, ans);
        for(auto itr: ans){
            if(itr.second == 1)
                ans2.push_back(itr.first);
        }
        return ans2;
    }
};
```

Back to problem (/problems/find-duplicate-subtrees/)