o oundergroundsystem/) 8

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

76

77 78

79 80

};

```
Language: cpp
                                                                                                             Edit Code
 1
     * Definition for a binary tree node.
     * struct TreeNode {
            int val;
 4
 5
            TreeNode *left;
            TreeNode *right;
           TreeNode() : val(0), left(nullptr), right(nullptr) {}
 7
           TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
           TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 9
     * };
10
11
     */
12
    class Solution {
13
    public:
14
15
         int compare(TreeNode* Nodei, TreeNode* Nodej, map<TreeNode*, int>& ans){
16
             if(Nodei == NULL and Nodej != NULL)
17
                 return false;
18
             if(Nodei != NULL and Nodej == NULL)
19
                return false;
20
             if(Nodei == NULL and Nodej == NULL)
21
                 return true;
22
23
             bool 1 = compare(Nodei->left, Nodej->left, ans);
24
             bool in = Nodei->val == Nodej->val? true:false;
25
             bool r = compare(Nodei->right, Nodej->right, ans);
26
27
             return (1 and in and r);
28
29
         void traverse_from_root(TreeNode* Nodei, TreeNode* root, map<TreeNode*, int>& ans){
30
             if(Nodei == NULL)
31
                return;
32
             if(root == NULL)
33
                 return;
34
             traverse_from_root(Nodei, root->left, ans);
35
             if((Nodei->val == root->val) and (Nodei != root))
36
                 if(compare(Nodei, root, ans) == true){
                    if(ans.find(Nodei) == ans.end() and ans.find(root) == ans.end()){
37
38
                         ans[Nodei] = 1;
39
                         ans[root] = 0;
40
41
42
43
44
45
         void solve(TreeNode* current_root, TreeNode* original_root, map<TreeNode*, int>& ans){
46
             if(current_root == NULL)
47
                 return;
48
49
             solve(current_root->left, original_root, ans);
50
             traverse_from_root(current_root, original_root, ans);
51
             solve(current_root->right, original_root, ans);
52
53
         vector<TreeNode*> findDuplicateSubtrees(TreeNode* root) {
54
55
56
             traverse in Inorder
57
             for each node
                 traverse tree(Inoder) from root;
58
59
                    if(both traverse are not pointing to same node AND both traverse
                                                                                                             have same v
60
61
                         then compare 2 trees (Nodei, Nodej)
62
                             if(both trees are equal AND Nodei or Nodej is not added already)
63
                                 ans[Nodei reference]++;
64
65
             map<TreeNode*, int> ans;
66
67
             vector<TreeNode*> ans2;
68
             if(root->val == 0 and root->left == NULL){
69
                 TreeNode* temp = new TreeNode(0);
70
                 ans2.push_back(temp);
71
                 return ans2;
72
73
             solve(root, root, ans);
74
             for(auto itr: ans){
75
                 if(itr.second == 1)
```

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

ans2.push_back(itr.first);

return ans2;

https://leetcode.com/submissions/detail/686504698/

Copyright © 2022 LeetCode Help Center (/support) | Jobs (/jobs) | Bug Bounty (/bugbounty) | Online Interview (/interview/) | Students (/student) | Terms (/terms) | Privacy Policy (/privacy)

United States (/region)

https://leetcode.com/submissions/detail/686504698/