337. House Robber III

♡ Add to List ▾

Description (?tab=Description)   Submission (?tab=Submission)   Solutions (?tab=Solutions)

Total Accepted: **36287**   Total Submissions: **86426**   Difficulty: **Medium**   Contributors: **Admin**

Notes

The thief has found himself a new place for his thievery again. There is only one entrance to this area, called the "root." Besides the root, each house has one and only one parent house. After a tour, the smart thief realized that "all houses in this place forms a binary tree". It will automatically contact the police if two directly-linked houses were broken into on the same night.

Determine the maximum amount of money the thief can rob tonight without alerting the police.

Example 1:

```
     3
    / \
   2   3
    \   \
     3   1
```

Maximum amount of money the thief can rob = 3 + 3 + 1 = **7**.

Example 2:

```
     3
    / \
   4   5
  / \   \
 1   3   1
```

Maximum amount of money the thief can rob = 4 + 5 = **9**.

Credits:

Special thanks to @dietpepsi (https://leetcode.com/discuss/user/dietpepsi) for adding this problem and creating all test cases.

Show Company Tags

Show Tags

Show Similar Problems

Have you met this question in a real interview? Yes No

Discuss (https://discuss.leetcode.com/category/421)   Pick One (/problems/random-one-question/)   Editorial Solution

C++ ▾

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
class Solution {
public:

    // check this: https://discuss.leetcode.com/topic/39659/easy-understanding-solution-with-dfs
    // similar idea but simpler expression

    map<pair<TreeNode*, bool>, int> dp;
    int helper(TreeNode* root, bool canInclude){
        if(root == nullptr) return 0;
        auto p = make_pair(root, canInclude);
        if(dp.find(p)!=dp.end()) return dp[p];
        if (canInclude) {
```

✉ Send Feedback (mailto:admin@leetcode.com?subject=Feedback)

```
            return dp[p] = max(root->val + helper(root->left, false) + helper(root->right, false),
                      helper(root->left, true) + helper(root->right, true));
        } else {
            return dp[p] = helper(root->left, true) + helper(root->right, true);
        }
    }

    int rob(TreeNode* root) {
        if(root == nullptr) return 0;
        int res = max(helper(root->left, true) + helper(root->right, true), root->val + helper(root->left, false) + helper
            ->right, false));
        return res;
    }
};
```

Notes

Custom Testcase □

Contribute Testcase ?

Run Code   Submit Solution

Frequently Asked Questions (/faq/) | Terms of Service (/tos/)

Privacv

✉ Send Feedback (mailto:admin@leetcode.com?subject=Feedback)