337. House Robber III

link

这个是我最开始写的.

```
    public int rob(TreeNode root) {
        return Math.max(dfs(root, true), dfs(root, false));
    }
    public int dfs(TreeNode root, boolean flag){
        if(root == null) return 0;
//最开始写在里面, 但是把它拿出来了, 因为有公共部分
        int leftFalse = dfs(root.left, false);
        int rightFalse = dfs(root.right, false);
        if(flag){
            int max =  leftFalse + rightFalse;
            return root.val > 0 ? root.val + max : max;
        }
        int left = Math.max(dfs(root.left, true), leftFalse);
        int right = Math.max(dfs(root.right, true), rightFalse);
        return left + right;
    }
```

分析链接, if(flag)和else都用到了相同的值进行计算, 考虑缓存或者动态规划了.

最开始我也想缓存, 但是是在函数里面缓存没有用, 考虑优化一个是放进map里面(全局变量)缓存, 还有一个是通过return两个值来缓存.

```
    public int rob(TreeNode root) {
        // Map<TreeNode, Integer> map = new HashMap();
        int[]res = dfs(root);
        return Math.max(res[0], res[1]);
    }
    int choose = 0;
    int notChoose = 0;
    public int dfs(TreeNode root, Map<TreeNode, Integer> map){
        if(root == null) return 0;
        int val = 0;
        if(root.left != null){
            if(!map.containsKey(root.left.left))
                map.put(root.left.left, dfs(root.left.left, map));
            if(!map.containsKey(root.left.right))
                map.put(root.left.right, dfs(root.left.right,map));
            val += map.get(root.left.left) + map.get(root.left.right);
        }
        if(root.right != null){
            if(!map.containsKey(root.right.left))
                map.put(root.right.left, dfs(root.right.left, map));
            if(!map.containsKey(root.right.right))
                map.put(root.right.right, dfs(root.right.right, map));
            val += map.get(root.right.left) + map.get(root.right.right);
        }
        return Math.max(root.val + val, dfs(root.right, map) + dfs(root.left, map));
    }
```

最简单的, 每次返回两个值, 这个节点被抢res0, 或者这个节点没有被抢res1.

```
 public int rob(TreeNode root) {
    int[] res = robSub(root);
    return Math.max(res[0], res[1]);
}

private int[] robSub(TreeNode root) {
    if (root == null) return new int[2];

    int[] left = robSub(root.left);
    int[] right = robSub(root.right);
    int[] res = new int[2];

    res[0] = Math.max(left[0], left[1]) + Math.max(right[0], right[1]);
    res[1] = root.val + left[0] + right[0];

    return res;
}
```

活着这样, 提前比较

```
public class Solution {
    public int rob(TreeNode root) {
        return maxMoney(root)[1];
    }

    // return int[2]: maxMoney[0] = max Money avoiding root itself, maxMoney[1] =
    max Money allowing root to be stolen
    private int[] maxMoney(TreeNode root) {
        if (root == null) return new int[2];
        int[] ans = new int[2],
                l = maxMoney(root.left),
                r = maxMoney(root.right);
        ans[0] = l[1] + r[1];
        ans[1] = Math.max(root.val + l[0] + r[0], ans[0]);
        return ans;
    }
}
```