# 437. Path Sum III (need to review)

link

## 自己写的暴力破解, 跟别人的比差一点.

```
public void helper(TreeNode node, int sum){
    if(node == null) return;
    helper(node.left, sum);
    helper(node.right, sum);
    dfs(node, sum);
}
public void dfs(TreeNode node, int sum){
    if(node == null) return;
    if(node.val == sum){
        res++;
    }
    dfs(node.left, sum - node.val);
    dfs(node.right, sum - node.val);
}
```

## 人家的.

```
public int pathSum(TreeNode root, int sum) {
    if (root == null) return 0;
    return pathSumFrom(root, sum) + pathSum(root.left, sum) +
pathSum(root.right, sum);
}

private int pathSumFrom(TreeNode node, int sum) {
    if (node == null) return 0;
    return (node.val == sum ? 1 : 0)
        + pathSumFrom(node.left, sum - node.val) + pathSumFrom(node.right, sum -
node.val);
}
```

## 真正的写法, 存currentSum, 然后通过之前的currentSum来判断是不是有路线.

## 比方我们要在1 2 1 1 1 找出和是3的, 可以建一个map, key是currentSum val是值, 第一次为1, 存在+1, 这样我们在2的时候可以用过contains(3 - 3)判断, 得加一个map.put(0,1)来包含跟节点, 这样4的时候contains(4-3)就找到了除去偷节点的两个值. 6的时候contains(6-3)除去了1和3两个节点

```
{
    {1:1}, {3,1}, {4,1}, {5,1}, {6,1}
}
```

## 偷的代码

```
    public int pathSum(TreeNode root, int sum) {
        HashMap<Integer, Integer> preSum = new HashMap();
        preSum.put(0,1);
        helper(root, 0, sum, preSum);
        return count;
    }
    int count = 0;
    public void helper(TreeNode root, int currSum, int target, HashMap<Integer,
    Integer> preSum) {
        if (root == null) {
            return;
        }

        currSum += root.val;

        if (preSum.containsKey(currSum - target)) {
            count += preSum.get(currSum - target);
        }

        if (!preSum.containsKey(currSum)) {
            preSum.put(currSum, 1);
        } else {
            preSum.put(currSum, preSum.get(currSum)+1);
        }

        helper(root.left, currSum, target, preSum);
        helper(root.right, currSum, target, preSum);
//分析这里节点算完了, 就得删了, 否则后面会重复计算?
        preSum.put(currSum, preSum.get(currSum) - 1);
    }
```