给出二叉树的根，找出出现次数最多的子树元素和。一个结点的子树元素和定义为以该结点为根的二叉树上所有结点的元素之和（包括结点本身）。然后求出出现次数最多的子树元素和。如果有多个元素出现的次数相同，返回所有出现次数最多的元素（不限顺序）。

示例 1

输入:

```
  5
 /  \
2    -3
```

返回 [2, -3, 4]，所有的值均只出现一次，以任意顺序返回所有值。

示例 2

输入:

```
  5
 /  \
2    -5
```

返回 [2]，只有 2 出现两次，-5 只出现 1 次。

提示： 假设任意子树元素和均可以用 32 位有符号整数表示。

分析，子元素和说白了，就是左右子树的和加上自己当前节点的值，然后又可以返回给上层节点作为递归条件。该遍历为后续遍历，动态的统计每一个节点值的情况。需注意，后续统计个数时，需要把重复的元素剔除，这时将该元素的个数统计置-1就ok

代码

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
int dfs(TreeNode* root,map<int,int> &bj,int &num){
    if(root->left!=NULL){
```

```
        root->val+=dfs(root->left,bj,num);
    }
    if(root->right!=NULL){
        root->val+=dfs(root->right,bj,num);
    }
    bj[root->val]++;
    if(bj[root->val]>num){
        num=bj[root->val];
    }
    return root->val;
}
void adddfs(vector<int> &result,TreeNode* root,map<int,int> &bj,int &num){
    if(bj[root->val]==num){
        result.push_back(root->val);
        bj[root->val]=-1;
    }
    if(root->left!=NULL){
        adddfs(result,root->left,bj,num);
    }
    if(root->right!=NULL){
        adddfs(result,root->right,bj,num);
    }
}
class Solution {
public:
    vector<int> findFrequentTreeSum(TreeNode* root) {
        vector<int> result;
        if(root==NULL){
            return result;
        }
        map<int,int> bj;
        int num=0;
        dfs(root,bj,num);
        adddfs(result,root,bj,num);
        return result;
```

```
    }
};
```