508. Most Frequent Subtree Sum

Add to List

Description SubmissionsSolutions

- Total Accepted: **7821**
- Total Submissions: **15062**
- Difficulty: **Medium**
- Contributors: **Cyber233**

Given the root of a tree, you are asked to find the most frequent subtree sum. The subtree sum of a node is defined as the sum of all the node values formed by the subtree rooted at that node (including the node itself). So what is the most frequent subtree sum value? If there is a tie, return all the values with the highest frequency in any order.

Examples 1

Input:

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

return [2, -3, 4], since all the values happen only once, return all of them in any order.

Examples 2

Input:

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

return [2], since 2 happens twice, however -5 only occur once.

Note: You may assume the sum of values in any subtree is in the range of 32-bit signed integer.

Subscribe to see which companies asked this question.

```
#include<iostream>
#include<stdio.h>
#include<algorithm>
#include<unordered_set>
#include<unordered_map>
#include<string>
#include<vector>
using namespace std;

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

typedef struct TreeNode TreeNode;

TreeNode* Insert(TreeNode *node, int key)
{
    if(node==NULL) return new TreeNode(key);
    if(key<node->val)
        node->left = Insert(node->left,key);
    else if (key>node->val)
        node->right = Insert(node->right,key);
    return node;
}

int countSubtreeSums(TreeNode *r, unordered_map<int,int> &counts, int &maxCount)
{
    if(r==NULL) return 0;
    int sum = r->val;
    sum+=countSubtreeSums(r->left,counts,maxCount);
    sum+=countSubtreeSums(r->right,counts,maxCount);
    counts[sum]++;
```

```
    maxCount = max(maxCount,counts[sum]);
    return sum;
}

vector<int> findFrequentTreeSum(TreeNode* root)
{
    unordered_map<int,int> counts;
    int maxCount = 0;
    countSubtreeSums(root,counts,maxCount);
    vector<int> maxSums;
    for(auto i:counts)
    {
        if(i.second==maxCount) maxSums.push_back(i.first);
    }
    return maxSums;
}

void destroyTree(TreeNode *root)
{
    if(root==NULL) return;
    destroyTree(root->left);
    destroyTree(root->right);
    TreeNode *p = root;
    delete p;
}
int main(int argc,char *argv[])
{

    /* 5 test case 1
     /  \
    2   -3
    */
    TreeNode *node = new TreeNode(5);
    node->left = new TreeNode(2);
    node->right = new TreeNode (-3);
    vector<int> res = findFrequentTreeSum(node);
    for(auto x:res) cout<<x<<endl;
    destroyTree(node);
    return 0;
}
```