112. Path Sum

Total Accepted: 120350
Total Submissions: 374276
Difficulty: Easy
Given a binary tree and a sum, determine if the tree has a root-to-leaf path such that adding up all the values along the path equals the given sum.

For example:
Given the below binary tree and sum = 22,

```
              5
             / \
            4   8
           /   / \
          11  13  4
         /  \      \
        7    2      1
```

return true, as there exist a root-to-leaf path 5->4->11->2 which sum is 22.

113. Path Sum II

Total Accepted: 94312
Total Submissions: 315819
Difficulty: Medium
Given a binary tree and a sum, find all root-to-leaf paths where each path's sum equals the given sum.

For example:
Given the below binary tree and sum = 22,

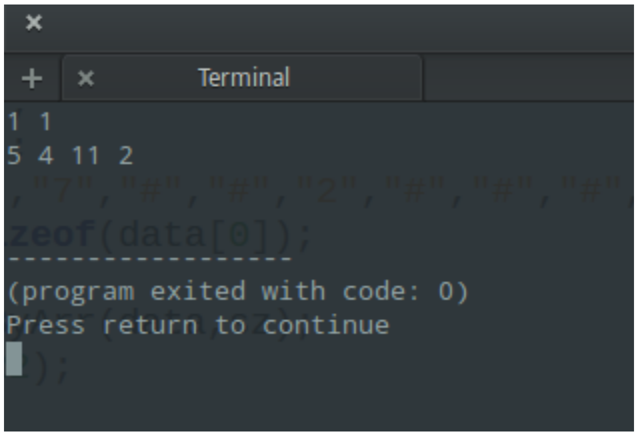
```
              5
             / \
            4   8
           /   / \
          11  13  4
         /  \    / \
        7    2  5   1
```

return
[
   [5,4,11,2],
   [5,8,4,5]
]

```
#include<iostream>
#include<sstream>
#include<vector>
#include<string>
```

```
#include<algorithm>
#include<limits>
#include<stdio.h>
//author:DemonMikalis
using namespace std;
int counter;
struct TreeNode
{
     int val;
     struct TreeNode *left;
     struct TreeNode *right;
     TreeNode(int x)
     {
          val=x;
          left=NULL;
          right=NULL;
     }
};

bool hasPathSum(TreeNode* root, int sum)
{
     if(root==NULL) return false;
     if(root->val==sum && root->right==NULL && root->left==NULL)
     return true;
     return hasPathSum(root->left,sum-root->val) || hasPathSum
(root->right,sum-root->val);
}

void findPath(TreeNode* root, int sum, vector<int> &path,
vector<vector<int> >&paths)
{
     if(root==NULL) return;
     path.push_back(root->val);
     if(root->val==sum && root->right==NULL && root->left==NULL)
     paths.push_back(path);
     findPath(root->left,sum-root->val,path,paths);
     findPath(root->right,sum-root->val,path,paths);
     path.pop_back();
}

vector<vector<int> > pathSum(TreeNode* root, int sum)
{
     vector<int> path;
     vector<vector<int> > paths;
     findPath(root,sum,path,paths);
     return paths;
}
```

```
/*
              5
             / \
            4   8
           /   / \
          11  13  4
         /  \      \
        7    2      1
*/
TreeNode *createTree()
{
     TreeNode *tree = new TreeNode(5);
     tree->left=new TreeNode(4);
     tree->right=new TreeNode(8);
     tree->left->left=new TreeNode(11);
     tree->left->left->left=new TreeNode(7);
     tree->left->left->right=new TreeNode(2);
     tree->right->left = new TreeNode(13);
     tree->right->right = new TreeNode(4);
     tree->right->right->right = new TreeNode(1);
     return tree;
}

void printTree(TreeNode *node)
{
     if(node==NULL) return;
     cout<<node->val;
     printTree(node->left);
     printTree(node->right);
}

TreeNode *createTreeByArr(string arr[],int arrsz)
{
     if(counter>=arrsz-1) return NULL;
     string arrstr = arr[counter];
     if(arrstr=="#") return NULL;
     int value=-1;
     stringstream ss;
     ss<<arrstr;
     ss>>value;
     TreeNode *node = new TreeNode(value);
     counter++;
     node->left = createTreeByArr(arr,arrsz);
     counter++;
     node->right= createTreeByArr(arr,arrsz);
     return node;
}
int main(int argc,char *argv[])
{
     TreeNode *tree = createTree();
     int ans = hasPathSum(tree,22);
     string data[] =
{"5","4","11","7","#","#","2","#","#","#","8","13","#","#","4","#
","1","#","#"};
     int sz = (int)sizeof(data)/sizeof(data[0]);
     counter=0;
     TreeNode *tree2 = createTreeByArr(data,sz);
     // test path sum 1
     int ans2 = hasPathSum(tree2,22);
     counter=0;// reset
     cout << ans << " "<< ans2 << endl;

     // test path sum 2
     vector<vector<int> > paths = pathSum(tree2,22);
     for(int i=0;i<(int)paths.size();i++)
     {
          for(int j=0;j<(int)paths[0].size();j++)
          {
               printf("%d ",paths[i][j]);
          }
          printf("\n");
     }
     return 0;
}
```



```
1 1
5 4 11 2
------------------
(program exited with code: 0)
Press return to continue
```