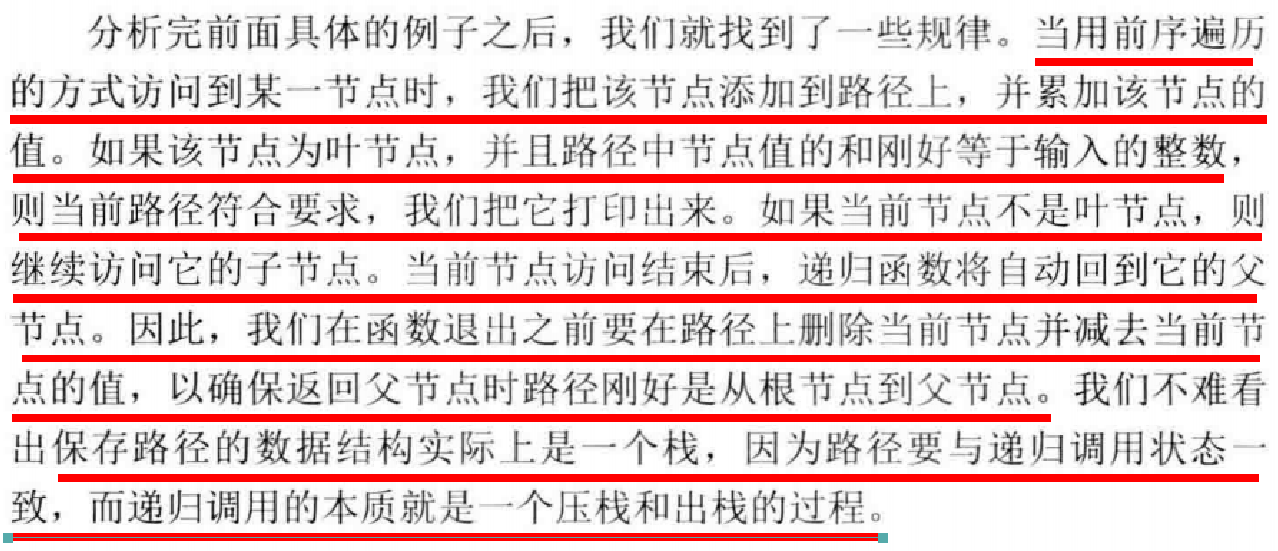
面试题34：二叉树中和为某一值的路径

题目描述

输入一颗二叉树的跟节点和一个整数，打印出二叉树中结点值的和为输入整数的所有路径。路径定义为从树的根结点开始往下一直到叶结点所经过的结点形成一条路径。(注意: 在返回值的list中，数组长度大的数组靠前)

解题思路



分析完前面具体的例子之后，我们就找到了一些规律。当用前序遍历的方式访问到某一节点时，我们把该节点添加到路径上，并累加该节点的值。如果该节点为叶节点，并且路径中节点值的和刚好等于输入的整数，则当前路径符合要求，我们把它打印出来。如果当前节点不是叶节点，则继续访问它的子节点。当前节点访问结束后，递归函数将自动回到它的父节点。因此，我们在函数退出之前要在路径上删除当前节点并减去当前节点的值，以确保返回父节点时路径刚好是从根节点到父节点。我们不难看出保存路径的数据结构实际上是一个栈，因为路径要与递归调用状态一致，而递归调用的本质就是一个压栈和出栈的过程。

C++

```
/*
struct TreeNode {
    int val;
    struct TreeNode *left;
    struct TreeNode *right;
    TreeNode(int x) :
            val(x), left(NULL), right(NULL) {
    }
};*/
class Solution {
public:
    vector<vector<int> > FindPath(TreeNode* root,int expectNumber) {
        //用带记忆的DFS来解决
        vector<vector<int>> ret;
        vector<int> trace;
        if(root)
            dfs(root,expectNumber,ret,trace);
        return ret;
    }
    void dfs(TreeNode* root,int s,vector<vector<int>> & ret,vector<int> & trace){
        trace.push_back(root -> val);
        if(!root -> left && !root -> right){
            if(s == root -> val)
                ret.push_back(trace);
        }
        if(root -> left)
```

```
            dfs(root -> left,s - root -> val,ret,trace);
        if(root -> right)
            dfs(root -> right,s - root -> val,ret,trace);
        trace.pop_back();
    }
};
```

Java

```
import java.util.ArrayList;
/**
public class TreeNode {
    int val = 0;
    TreeNode left = null;
    TreeNode right = null;

    public TreeNode(int val) {
        this.val = val;

    }

}
*/
public class Solution {
    public ArrayList<ArrayList<Integer>> FindPath(TreeNode root,int target) {
```

```java
        //递归
        ArrayList<ArrayList<Integer>> paths = new ArrayList<ArrayList<Integer>>();
        if(root == null)
            return paths;
        find(paths,new ArrayList<Integer>(),root,target);
        return paths;
    }
    public void find(ArrayList<ArrayList<Integer>> paths,ArrayList<Integer> path,TreeNode root,int target){
        path.add(root.val);
        if(root.left == null && root.right == null){
            if(target == root.val){
                paths.add(path);
            }
            return;
        }
        ArrayList<Integer> path2 = new ArrayList<>();
        path2.addAll(path);
        if(root.left != null)
            find(paths,path,root.left,target - root.val);
        if(root.right != null)
```

```
            find(paths,path2,root.right,target - root.val);
        }
}
```

Python 2.7.3

```
# -*- coding:utf-8 -*-
# class TreeNode:
#     def __init__(self, x):
#         self.val = x
#         self.left = None
#         self.right = None
class Solution:
    # 返回二维列表，内部每个列表表示找到的路径
    def FindPath(self, root, expectNumber):
        # write code here
        res = []
        treepath = self.dfs(root)
        for i in treepath:
            if sum(map(int,i.split('->'))) == expectNumber:
                res.append(list(map(int,i.split('->'))))
        return res

    def dfs(self,root):
        if not root:
            return[]
        if not root.left and not root.right:
            return [str(root.val)]
        treePath = [str(root.val) + "->" + path for path in self.dfs(root.left)]
        treePath += [str(root.val) + "->" + path for path in self.dfs(root.right)]
        return treePath
```