

剑指34 二叉树中和为某一值的路径

点评：本问题是典型的二叉树方案搜索问题，使用回溯法解决，其包含 先序遍历 + 路径记录 两部分。

Tips：将target改为不断减去节点，最终用target == 0表示达到目标状态，这样能少用一个成员变量。

TODO：做一次用栈的非递归遍历试试~~~

```
1 // 第一次尝试，性能非常低：
2 // 原因大概是vector<int>作为函数参数不断传递。
3 class Solution {
4 public:
5     vector<vector<int>> output;
6     int mtarget;
7     void DFS(TreeNode* node, int sum, vector<int> line) {
8         if (!node) return;
9         sum += node->val;
10        line.push_back(node->val);
11        if (node->left == NULL && node->right == NULL && sum == mtarget) {
12            output.push_back(line);
13            return;
14        }
15        DFS(node->left, sum, line);
16        DFS(node->right, sum, line);
17    }
18
19    vector<vector<int>> pathSum(TreeNode* root, int target) {
20        mtarget = target;
21        DFS(root, 0, {});
22        return output;
23    }
24 };
25
26
27 // 自我优化：
28 class Solution {
29 public:
30     vector<vector<int>> output;
31     vector<int> line;
32     int mtarget;
33     void DFS(TreeNode* node, int sum) {
34         if (!node) return;
35         sum += node->val;
36         line.push_back(node->val);
37         if (node->left == NULL && node->right == NULL && sum == mtarget) {
38             output.push_back(line);
39         }
40         DFS(node->left, sum);
41         DFS(node->right, sum);
```

```

42         line.pop_back();
43     }
44     vector<vector<int>> pathSum(TreeNode* root, int target) {
45         mtarget = target;
46         DFS(root, 0);
47         return output;
48     }
49 };
50
51 // 最终改进: 删除mtarget
52 class Solution {
53 public:
54     vector<vector<int>> output;
55     vector<int> line;
56     void DFS(TreeNode* node, int sum) {
57         if (!node) return;
58         sum -= node->val;
59         line.push_back(node->val);
60         if (node->left == NULL && node->right == NULL && sum == 0) {
61             output.push_back(line);
62         }
63         DFS(node->left, sum);
64         DFS(node->right, sum);
65         line.pop_back();
66     }
67     vector<vector<int>> pathSum(TreeNode* root, int target) {
68         DFS(root, target);
69         return output;
70     }
71 };

```