```
/*
给定一个二叉树，它的每个结点都存放着一个整数值。

找出路径和等于给定数值的路径总数。

路径不需要从根节点开始，也不需要在叶子节点结束，但是路径方向必须是向下的（只能从父节点到子节点）。

二叉树不超过1000个节点，且节点数值范围是 [-1000000,1000000] 的整数。

示例：

root = [10,5,-3,3,2,null,11,3,-2,null,1], sum = 8

      10
     /  \
    5   -3
   / \    \
  3   2   11
 / \   \
3  -2   1

返回 3。和等于 8 的路径有：

1.  5 -> 3
2.  5 -> 2 -> 1
3.  -3 -> 11

来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/path-sum-iii
著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。
*/
```

分析:

- 任务要求遍历所有的节点及所有节点的分支,采取两重递归的话,栈开销预计不小;
- 方法一:BFS+DFS
  - BFS:遍历所有节点
  - DFS:遍历从当前节点开始的所有可能分支
- 方法二:DFS+DFS

方法一:C++_BFS+DFS

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

```
 */
class Solution
{
    private:
        int ret_val     = 0;
        int target_sum  = 0;

        void __pathSum(TreeNode* node , int level_sum)
        {
            if(node==NULL)
            {
                return ;
            }

            int temp = level_sum + node->val;

            if(temp==target_sum)
            {
                ret_val++;
                //return;
            }
            __pathSum(node->left,temp);
            __pathSum(node->right,temp);
        }
    public:
        int pathSum(TreeNode* root, int sum)
        {
            ret_val     = 0;
            target_sum  = sum;
            queue<TreeNode*>    qtn     ;
            TreeNode*           temp    ;
            if(root!=NULL)
            {
                qtn.push(root);
                while(!qtn.empty())
                {
                    temp = qtn.front();
                    qtn.pop();
                    __pathSum(temp,0);
                    if(temp->left)
                    {
                      qtn.push(temp->left);
                    }
                    if(temp->right)
                    {
                        qtn.push(temp->right);
                    }
                }

            }
            return ret_val;
        }
};

/*
执行结果：
通过
显示详情
```

```
执行用时 :40 ms, 在所有 C++ 提交中击败了49.50% 的用户
内存消耗 :14.5 MB, 在所有 C++ 提交中击败了93.90%的用户
*/
```

方法二:C++_DFS+DFS

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
class Solution
{
    private:
        int ret_val     = 0;
        int target_sum  = 0;

        void __pathSum(TreeNode* node , int level_sum)
        {
            if(node==NULL)
            {
                return ;
            }

            int temp = level_sum + node->val;

            if(temp==target_sum)
            {
                ret_val++;
                //return;
            }
            __pathSum(node->left,temp);
            __pathSum(node->right,temp);
        }


        /* 根左右*/
        void dfs(TreeNode* node)
        {
            if(node==NULL)
            {
                return;
            }

            __pathSum(node,0);
            dfs(node->left);
            dfs(node->right);
        }

    public:
        int pathSum(TreeNode* root, int sum)
        {
```

```
            ret_val     = 0;
            target_sum  = sum;
            dfs(root);
            return ret_val;
        }
};

/*
执行结果：
通过
显示详情
执行用时 :44 ms, 在所有 C++ 提交中击败了41.65% 的用户
内存消耗 :14.6 MB, 在所有 C++ 提交中击败了84.96%的用户
*/
```

AlimyBreak
2019.09.15