```
/*
给你一个二叉树的根节点 root。设根节点位于二叉树的第 1 层，而根节点的子节点位于第 2 层，依此类推。

请你找出层内元素之和 最大 的那几层（可能只有一层）的层号，并返回其中 最小 的那个。



示例：
          1
       7      0
     7   -8


输入：[1,7,0,7,-8,null,null]
输出：2
解释：
第 1 层各元素之和为 1，
第 2 层各元素之和为 7 + 0 = 7，
第 3 层各元素之和为 7 + -8 = -1，
所以我们返回第 2 层的层号，它的层内元素之和最大。


提示：

树中的节点数介于 1 和 10^4 之间
-10^5 <= node.val <= 10^5

来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/maximum-level-sum-of-a-binary-tree
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:

- 常规方法:BFS+结果比较

方法一:C++_BFS

```
/**
 * 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
{
    public:
```

```
        int maxLevelSum(TreeNode* root)
        {
            queue<TreeNode*>    qtn                             ;
            int                 ret_val             =   0       ;
            int                 num_1               =   0       ;
            int                 num_2               =   0       ;
            int                 i                   =   0       ;
            vector<int>         vi                              ;
            TreeNode*           tnd                 =   NULL    ;
            int                 level_sum           =   0       ;
            if(root)
            {
                qtn.push(root);
                num_1 = 0;
                num_2 = 1;
                while(num_2)
                {
                    num_1       =   num_2   ;
                    num_2       =   0       ;
                    level_sum   =   0       ;
                    for(i = 0 ; i < num_1 ; i++)
                    {
                        tnd = qtn.front();
                        level_sum += tnd->val;
                        if(tnd->left)
                        {
                            num_2++;
                            qtn.push(tnd->left);
                        }
                        if(tnd->right)
                        {
                            num_2++;
                            qtn.push(tnd->right);
                        }
                        qtn.pop();
                    }
                    vi.push_back(level_sum);
                }
                ret_val = 0;
                for(i=1;i<vi.size();i++)
                {
                    if(vi[i] > vi[ret_val])
                    {
                        ret_val = i ;
                    }
                }
                ret_val += 1;
            }
            return ret_val;
        }
};
/*
执行结果:
通过
显示详情
执行用时 :272 ms, 在所有 C++ 提交中击败了85.06%的用户
内存消耗 :72.9 MB, 在所有 C++ 提交中击败了100.00%的用户
*/
```

AlimyBreak
2019.10.14