```
/*
给定一个二叉树，返回其按层次遍历的节点值。 （即逐层地，从左到右访问所有节点）。

例如:
给定二叉树: [3,9,20,null,null,15,7],

    3
   / \
  9  20
    /  \
   15   7
返回其层次遍历结果：

[
  [3],
  [9,20],
  [15,7]
]
在真实的面试中遇到过这道题？

来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/binary-tree-level-order-traversal
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：

- 二叉树的层次遍历的常规方法一：借助辅助queue进行迭代
- 二叉树的层次遍历的常规方法二：前序递归遍历，根据层次计算进行数据保存。

方法一:C++,借助辅助queue进行迭代

```
/**
 * 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:
        vector<vector<int>> levelOrder(TreeNode* root)
        {
```

```
            vector<vector<int>>     ret_val             ;
            vector<int>             level_vector        ;
            queue<TreeNode*>        qn                  ;
            TreeNode*               temp        = NULL  ;
            int                     num_1       = 0     ;
            int                     num_2       = 0     ;
            int                     i           = 0     ;

            if( root == NULL )
            {
                return ret_val;
            }
            else
            {
                qn.push(root);
                num_1 = 0;
                num_2 = 1;
                while(1)
                {
                    num_1 = num_2           ;
                    num_2 = 0               ;
                    level_vector.clear()    ;
                    for(i = 0; i < num_1 ; i++)
                    {
                        temp = qn.front();
                        if(temp->left)
                        {
                            qn.push(temp->left);
                            num_2++;
                        }
                        if(temp->right)
                        {
                            qn.push(temp->right);
                            num_2++;
                        }
                        level_vector.push_back(temp->val);
                        qn.pop();
                    }
                    ret_val.push_back(level_vector);
                    if(num_2==0)
                    {
                        break;
                    }
                }
                return ret_val;
        }
    }
};

/*
执行结果：
通过
显示详情
```

```
执行用时 :12 ms, 在所有 C++ 提交中击败了67.53%的用户
内存消耗 :13.7 MB, 在所有 C++ 提交中击败了76.76%的用户
*/
```

方法二:C++,前序递归遍历

```
class Solution
{
    private:
        void __preOrder(TreeNode* node , int depth, vector<vector<int>>& ret_val)
        {
            if(node==NULL)
            {
                return ;
            }

            /* 根左右*/
            if(depth>=ret_val.size())
            {
                vector<int> temp;
                temp.push_back(node->val);
                ret_val.push_back(temp);
            }
            else
            {
                ret_val[depth].push_back(node->val);
            }

            __preOrder(node->left,depth+1,ret_val);
            __preOrder(node->right,depth+1,ret_val);
        }

    public:
        vector<vector<int>> levelOrder(TreeNode* root)
        {
            vector<vector<int>> ret_val;
            __preOrder(root,0,ret_val);
            return ret_val;
        }
};

/*
执行结果：
通过
显示详情
执行用时 :20 ms, 在所有 C++ 提交中击败了16.73%的用户
内存消耗 :15.7 MB, 在所有 C++ 提交中击败了5.05%的用户
*/
```

AlimyBreak
2019.07.30