从上到下打印二叉树 II

从上到下按层打印二叉树，同一层的节点按从左到右的顺序打印，每一层打印到一行。

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]
]
```

解题思路：
此题与前一题基本思路一致 ，利用队列维护 只不过此处需要借助额外一个level来维护当前层所遍历的元素
输出结果将原有的一维数组，上升至二维数组

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 vector<int> levelOrder(TreeNode* root) {
        vector<int> ret;
        if(root==nullptr)
        {
            return ret;
        }
        queue<TreeNode*> q;
```

```
        q.push(root);//通过队列存储遍历树的结果
        while(!q.empty())
        {
            TreeNode* node=q.front();//将队头元素指向 node
            q.pop();
            ret.push_back(node->val);
            if(node->left) q.push(node->left);//左不为空 存储在队里
            if(node->right) q.push(node->right);//右不为空 存储在队里
        }
        return ret;
    }
 */
class Solution {
public:
    vector<vector<int>> levelOrder(TreeNode* root) {
        vector<vector<int>> ret;
        if(root==nullptr)
        {
            return ret;
        }
        //vector<int> level;
        queue<TreeNode*> q;
        q.push(root);//通过队列存储遍历树的结果
        while(!q.empty())
        {
            vector<int> level;//注意 level 的位置 只能为局部变量存储，如果全局会
重复记录上一层结果在 level 中    此处 level 只维护当前层遍历的元素
            int size=q.size();//记录当前队列大小
            for(int i=0;i<size;i++)
            {
                TreeNode* node=q.front();//将队头元素指向 node
                q.pop();
                level.push_back(node->val);
                if(node->left) q.push(node->left);//左不为空 存储在队里
                if(node->right) q.push(node->right);//右不为空 存储在队里
            }
            if(level.size()!=NULL)
            {
                ret.push_back(level);
            }
        }
        return ret;
    }
};
```