（1.1）LC107.二叉树的层序遍历II

107. 二叉树的层序遍历 II - 力扣（LeetCode）

方法一：reverse版

```
class Solution {
public:
    vector<vector<int>> levelOrderBottom(TreeNode* root) {
        vector<vector<int>> res;
        queue<TreeNode*> que;
        if(root != NULL) que.push(root);

        while(!que.empty()){
            //先出再进
            vector<int> vec;  //储存每一层的遍历
            int size = que.size();  //固定一下队列的大小；
            for(int i = 0;i < size;i++){
                TreeNode* tmp = que.front();

                vec.push_back(tmp->val);
                que.pop();
                if(tmp->left != NULL) que.push(tmp->left);
                if(tmp->right != NULL) que.push(tmp->right);
            }
            res.push_back(vec);
        }
        reverse(res.begin(),res.end());  //和102唯一的不同之处
        return res;
    }
};
```

方法二：insert头插版，不过时间开销大

```
class Solution {
public:
    vector<vector<int>> levelOrderBottom(TreeNode* root) {
        vector<vector<int>> res;
        queue<TreeNode*> que;
        if(root != NULL) que.push(root);

        while(!que.empty()){
            //先出再进
            vector<int> vec;  //储存每一层的遍历
            int size = que.size();  //固定一下队列的大小；
            for(int i = 0;i < size;i++){
```

```
                TreeNode* tmp = que.front();

                vec.push_back(tmp->val);
                que.pop();
                if(tmp->left != NULL) que.push(tmp->left);
                if(tmp->right != NULL) que.push(tmp->right);
            }
            res.insert(res.begin(),vec);  //不同之处

        }
        return res;
    }
};
```