107. Binary Tree Level Order Traversal II

107 Binary Tree Level Order Traversal II

- **Breadth-first Search** + Tree

Description

Given a binary tree, return the *bottom-up level order* traversal of its nodes' values. (ie, from left to right, level by level from leaf to root).

For example:
Given binary tree `[3,9,20,null,null,15,7]`,

```
    3
   / \
  9  20
    /  \
   15   7
```

return its bottom-up level order traversal as:

```
[
  [15,7],
  [9,20],
  [3]
]
```

1. Thought line

- as same as 102, 103

2. Breadth-first Search + Tree

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

/**
 * 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>> levelOrderBottom(TreeNode* root) {

        vector<vector<int>> result;
            queue<TreeNode*> que;
            if (root!=nullptr) que.emplace(root);

```

```
        while (!que.empty() || que.front()!=nullptr){
            queue<TreeNode*> tempQue;
            vector<int> tempVec;
            while (!que.empty()){
                tempVec.push_back(que.front()->val);
                if (que.front()->left!=nullptr ) tempQue.push(que.front()->left);
                if (que.front()->right!=nullptr) tempQue.push(que.front()->right);
                que.pop();
            }
            if(!tempVec.empty()) result.insert(result.begin(),tempVec);
            else break;
            if(!tempQue.empty()) que.swap(tempQue);
            else break;


        }
        return result;
    }
};
```