

# 107. Binary Tree Level Order Traversal II

## 107 Binary Tree Level Order Traversal II

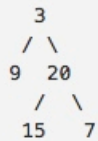
- Breadth-first Search + Tree

### 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]`,



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

```
1 /**
2  * Definition for a binary tree node.
3  * struct TreeNode {
4  *     int val;
5  *     TreeNode *left;
6  *     TreeNode *right;
7  *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
8  * };
9  */
10
11 /**
12  * Definition for a binary tree node.
13  * struct TreeNode {
14  *     int val;
15  *     TreeNode *left;
16  *     TreeNode *right;
17  *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
18  * };
19  */
20 class Solution {
21 public:
22     vector<vector<int>> levelOrderBottom(TreeNode* root) {
23
24         vector<vector<int>> result;
25         queue<TreeNode*> que;
26         if (root!=nullptr) que.emplace(root);
27     }
```

```

28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
};

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;
}
};

```