



103. Binary Tree Zigzag Level Order Traversal

103 Binary Tree Zigzag Level Order Traversal

- Breadth-first Search + Queue + Tree
- Breadth-first Search + Stack + Tree

Description

 Discuss

 Pick One

Given a binary tree, return the *zigzag level order* traversal of its nodes' values. (ie, from left to right, then right to left for the next level and alternate between).

For example:

Given binary tree [3,9,20,null,null,15,7] ,

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

return its zigzag level order traversal as:

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

1. Thought line

2. Breadth-first Search + Queue + 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 class Solution {
11 public:
12     vector<vector<int>> zigzagLevelOrder(TreeNode* root) {
13         vector<vector<int>> result;
14         queue<TreeNode*> que;
15         if (root!=nullptr) que.emplace(root);
16         bool flag = true;
17         while (!que.empty() || que.front()!=nullptr){
```

```

18 queue<TreeNode*> tempQue;
19 vector<int> tempVec;
20 while (!que.empty()){
21     if (flag)
22         tempVec.push_back(que.front()->val);
23     else
24         tempVec.insert(tempVec.begin(), que.front()->val);
25     if (que.front()->left!=nullptr ) tempQue.push(que.front()->left);
26     if (que.front()->right!=nullptr) tempQue.push(que.front()->right);
27     que.pop();
28 }
29 if(!tempVec.empty()) result.push_back(tempVec);
30 else break;
31 if(!tempQue.empty()) que.swap(tempQue);
32 else break;
33 flag = !flag;
34
35 }
36 return result;
37 }
38 };

```

3. Breadth-first Search + Stack + Tree