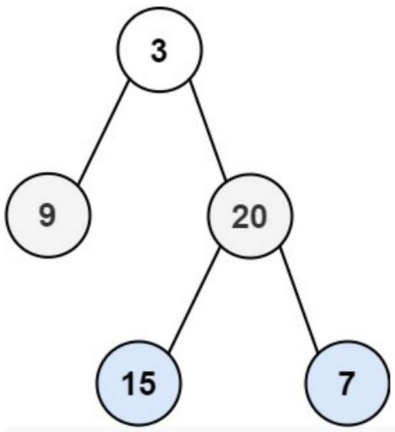
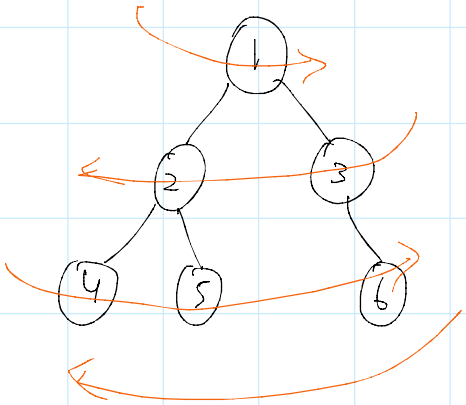
103. Binary Tree Zigzag Level Order Traversal

20 February 2022 10:14 AM

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




```
Input: root = [3,9,20,null,null,15,7]
Output: [[3],[20,9],[15,7]]
```



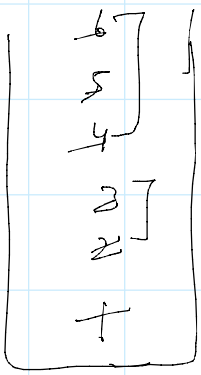
O/P ⇒

1
3 2
4 5 6

flag = 0/1

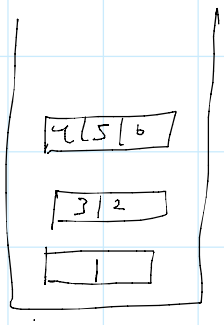
0 → (L → R)

1 → (R → L)



Queue

1
2 3



ds

+ After pushing 1 into the ds change the flag to 1 as one iteration or one level is over.

* Now as flag is 1 insert (2,3) in R to L order [3|2] and change flag to 0.

* Now flag is 0 (4, 5, 6) in L to R order [4|5|6]

Now your queue is empty and return the ds.

True L→R, false R→L

```java
class Solution {
    public List<List<Integer>> zigzagLevelOrder(TreeNode root) {
        Queue<TreeNode> queue = new LinkedList<TreeNode>();
        List<List<Integer>> wrapList = new LinkedList<List<Integer>>();

        if(root == null) return wrapList;

        queue.offer(root);
        boolean flag = true;
        while(!queue.isEmpty()){
            int levelNum = queue.size();
            List<Integer> subList = new ArrayList<Integer>(levelNum);
            for(int i=0; i<levelNum; i++) {
                int index = i;
                if(queue.peek().left != null) queue.offer(queue.peek().left);
                if(queue.peek().right != null) queue.offer(queue.peek().right);
                if(flag == true) subList.add(queue.poll().val);
                else subList.add(0, queue.poll().val);
            }
            flag = !flag;
            wrapList.add(subList);
        }
        return wrapList;
    }
}
```

Java

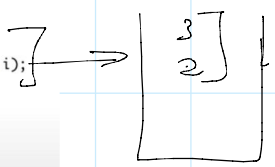
```cpp
*/
class Solution {
public:
    vector<vector<int>> zigzagLevelOrder(TreeNode* root) {
        vector<vector<int>> result;
        if (root == NULL) {
            return result;
        }

        queue<TreeNode*> nodesQueue;
        nodesQueue.push(root);
        bool leftToRight = true;

        while ( !nodesQueue.empty()) {
            int size = nodesQueue.size();
            vector<int> row(size);
            for (int i = 0; i < size; i++) {
                TreeNode* node = nodesQueue.front();
                nodesQueue.pop();

                // find position to fill node's value
                int index = (leftToRight) ? i : (size - 1 - i);

                row[index] = node->val;
                if (node->left) {
                    nodesQueue.push(node->left);
                }
                if (node->right) {
                    nodesQueue.push(node->right);
                }
            }
            // after this level
            leftToRight = !leftToRight;
            result.push_back(row);
        }
        return result;
    }
};
```



[2|3] else [3|2]

GFG

```cpp
class Solution{
    public:
    //Function to store the zig zag order traversal of tree in a list.
    vector <int> zigZagTraversal(Node* root)
    {
        vector<int> result;
        if(root == NULL) return result;
        queue<Node*> q;
        q.push(root);

        bool leftToRight = true;
        while(!q.empty()){

            int size = q.size();
            vector<int> ans(size);
            //Level process
            for(int i=0; i<size; i++){
                Node* frontNode = q.front();
                q.pop();
                //normal insert or reverse insert
                int index = leftToRight ? i : size - i - 1;
                ans[index] = frontNode -> data;
                if(frontNode -> left){
                    q.push(frontNode->left);
                }
                if(frontNode -> right){
                    q.push(frontNode->right);
                }
            }
            // direction change
            leftToRight = !leftToRight;
            for(auto i: ans){
                result.push_back(i);
            }
        }
        return result;
    }
};
```

Press Esc to exit full screen