Zig Zag Traversal Of Binary Tree
Problem Statement: Given the root of a binary tree, return the zigzag level order traversal of Binary Tree. (i.e., from left to right, then right to left for the next level and alternate between).
Examples:

Example 1:

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

Output: [[3],[20,9],[15,7]]

Explaination: From the root, we follow this terminology, left to right -> right to left -> left to right and so on so forth.

Example 2:

Input: root = [[0]]

Output : [[0]]

Explanation: We just have a single node which acts as the root, so going from left to right, we get just one node that is the root node itself.

Intuition: Considering the fact that we need to print the nodes, level by level, our first guess would definitely be that it must be related to level order traversal. If we closely examine, for even levels we need to go from left to right while for odd levels we need to go from right to left.
Approach: The above idea, could be implemented with a queue. We initially keep an empty queue and push the root node. We also need to keep the left to right bool variable that keeps check of the current level we are in. As we traverse nodes in the queue, we need to push them in a temporary array. If left to right is false we need to reverse the array and push it in our data structure or else, simply push it in our data structure. In the end, when we have finished traversing the current level, we need to toggle our left to the right variable.
Code:

- C++ Code
- Java Code

```cpp
#include<bits/stdc++.h>

using namespace std;

class Node {
  public:
    int val;
  Node * left, * right;
};

vector < vector < int >> zigzagLevelOrder(Node * root) {
  vector < vector < int >> result;
  if (root == NULL) {
    return result;
  }

  queue < Node * > 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++) {
            Node * 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;
}
Node * newNode(int data) {
    Node * node = new Node;
    node -> val = data;
    node -> left = NULL;
    node -> right = NULL;
    return node;
}
int main() {
    int i, j;
    Node * root = newNode(3);
    root -> left = newNode(9);
    root -> right = newNode(20);
    root -> right -> left = newNode(15);
    root -> right -> right = newNode(7);
    vector < vector < int >> ans;
    ans = zigzagLevelOrder(root);
    cout << "Zig Zag Traversal of Binary Tree" << endl;
    for (i = 0; i < ans.size(); i++) {
        for (j = 0; j < ans[i].size(); j++) {
            cout << ans[i][j] << " ";
        }
        cout << endl;
    }
    return 0;
}
```

Output:

Zig Zag Traversal of Binary Tree

3

20 9

15 7

Time Complexity: O(N)

Space Complexity: O(N)