

## Zigzag Traversal (medium)

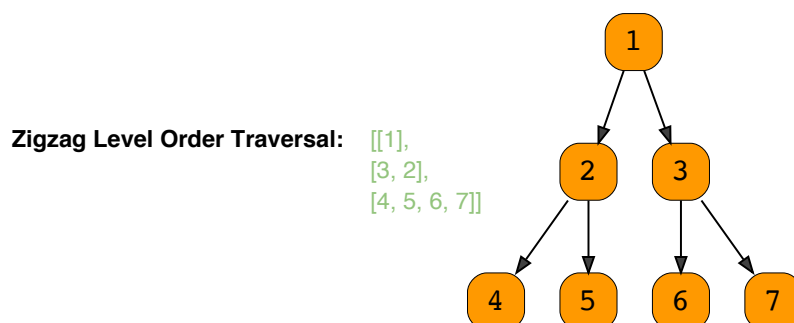
We'll cover the following ^

- Problem Statement
- Try it yourself
- Solution
- Code
  - Time complexity
  - Space complexity

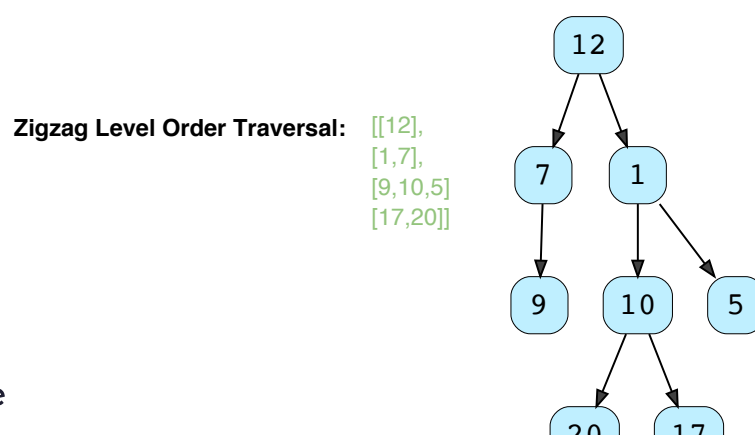
### Problem Statement #

Given a binary tree, populate an array to represent its zigzag level order traversal. You should populate the values of all **nodes of the first level from left to right**, then **right to left for the next level** and keep alternating in the same manner for the following levels.

#### Example 1:







#### Example 2:



## Try it yourself #





Try solving this question here:

 Java
  Python3
  JS
  C++

```

1 class TreeNode:
2     def __init__(self, val):
3         self.val = val
4         self.left, self.right = None, None
5
6 def traverse(root):
7     result = []
8     # TODO: Write your code here
9     return result
10
11
12 def main():
13     root = TreeNode(12)
14     root.left = TreeNode(7)
15     root.right = TreeNode(1)
16     root.left.left = TreeNode(9)
17     root.right.left = TreeNode(10)
18     root.right.right = TreeNode(5)
19     root.right.left.left = TreeNode(20)
20     root.right.left.right = TreeNode(17)
21     print("Zigzag traversal: " + str(traverse(root)))
22
23
24 main()
25

```

## Solution #

This problem follows the Binary Tree Level Order Traversal

(<https://www.educative.io/collection/page/5668639101419520/5671464854355968/5726607939469312/>) pattern. We can follow the same **BFS** approach. The only additional step we have to keep in mind is to alternate the level order traversal, which means that for every other level, we will traverse similar to Reverse Level Order Traversal

(<https://www.educative.io/collection/page/5668639101419520/5671464854355968/5765606242516992/>).

## Code #

Here is what our algorithm will look like, only the highlighted lines have changed:

 Java
  Python3
  C++
  JS

```

1 from collections import deque
2
3
4 class TreeNode:
5     def __init__(self, val):
6         self.val = val
7         self.left, self.right = None, None
8
9 def traverse(root):
10     result = []
11     # TODO: Write your code here
12     return result
13
14
15 def main():
16     root = TreeNode(12)
17     root.left = TreeNode(7)
18     root.right = TreeNode(1)
19     root.left.left = TreeNode(9)
20     root.right.left = TreeNode(10)
21     root.right.right = TreeNode(5)
22     root.right.left.left = TreeNode(20)
23     root.right.left.right = TreeNode(17)
24     print("Zigzag traversal: " + str(traverse(root)))
25
26
27 main()
28

```




```

0     self.val = val
7     self.left, self.right = None, None
8
9
10    def traverse(root):
11        result = []
12        if root is None:
13            return result
14
15        queue = deque()
16        queue.append(root)
17        leftToRight = True
18        while queue:
19            levelSize = len(queue)
20            currentLevel = deque()
21            for _ in range(levelSize):
22                currentNode = queue.popleft()
23
24                # add the node to the current level based on the traverse direction
25                if leftToRight:
26                    currentLevel.append(currentNode.val)
27                else:
28                    currentLevel.appendleft(currentNode.val)

```



#### Time complexity #

The time complexity of the above algorithm is  $O(N)$ , where 'N' is the total number of nodes in the tree. This is due to the fact that we traverse each node once.

#### Space complexity #

The space complexity of the above algorithm will be  $O(N)$  as we need to return a list containing the level order traversal. We will also need  $O(N)$  space for the queue. Since we can have a maximum of  $N/2$  nodes at any level (this could happen only at the lowest level), therefore we will need  $O(N)$  space to store them in the queue.

[← Back](#)

[Next →](#)

[Reverse Level Order Traversal \(easy\)](#)

[Level Averages in a Binary Tree \(easy\)](#)

☒ **Mark as Completed**

Report an Issue

Ask a Question

([https://discuss.educative.io/tag/zigzag-traversal-medium\\_\\_pattern-tree-breadth-first-search\\_\\_grokking-the-coding-interview-patterns-for-coding-questions](https://discuss.educative.io/tag/zigzag-traversal-medium__pattern-tree-breadth-first-search__grokking-the-coding-interview-patterns-for-coding-questions))