Zigzag Traversal (medium)

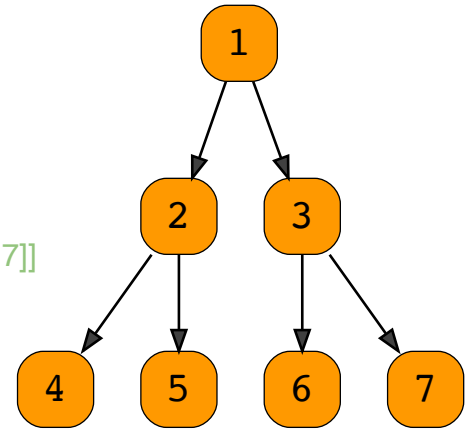
We'll cover the following

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.

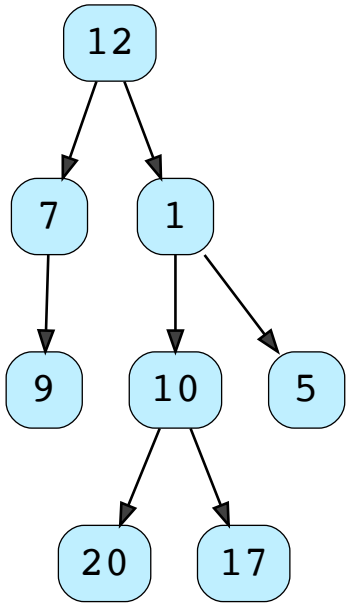
Example 1:

Zigzag Level Order Traversal: [[1],
[3, 2],
[4, 5, 6, 7]]



Example 2:

Zigzag Level Order Traversal: [[12],
[1,7],
[9,10,5]
[17,20]]



Try it yourself

Try solving this question here:

Java | Python3 | JS | C++

```
1  class TreeNode:
```

```
  def __init__(self, val):
    self.val = val
    self.left, self.right = None, None

def traverse(root):
  result = []
  # TODO: Write your code here
  return result


def main():
  root = TreeNode(12)
  root.left = TreeNode(7)
  root.right = TreeNode(1)
  root.left.left = TreeNode(9)
  root.right.left = TreeNode(10)
  root.right.right = TreeNode(5)
  root.right.left.left = TreeNode(20)
  root.right.left.right = TreeNode(17)
  print("Zigzag traversal: " + str(traverse(root
  

main()

```

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

```
def traverse(root):
  result = []
  if root is None:
    return result

  queue = deque()
  queue.append(root)
  leftToRight = True
  while queue:
    levelSize = len(queue)
    currentLevel = deque()
    for _ in range(levelSize):
      currentNode = queue.popleft()

      # add the node to the current level based
      if leftToRight:
        currentLevel.append(currentNode.val)
      else:
        currentLevel.appendleft(currentNode.val)

      # insert the children of current node in t
      if currentNode.left:
        queue.append(currentNode.left)
      if currentNode.right:
        queue.append(currentNode.right)

    result.append(list(currentLevel))
    # reverse the traversal direction
    leftToRight = not leftToRight
```

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

Reverse Level Order Traversal (easy)

Next

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)