Question:

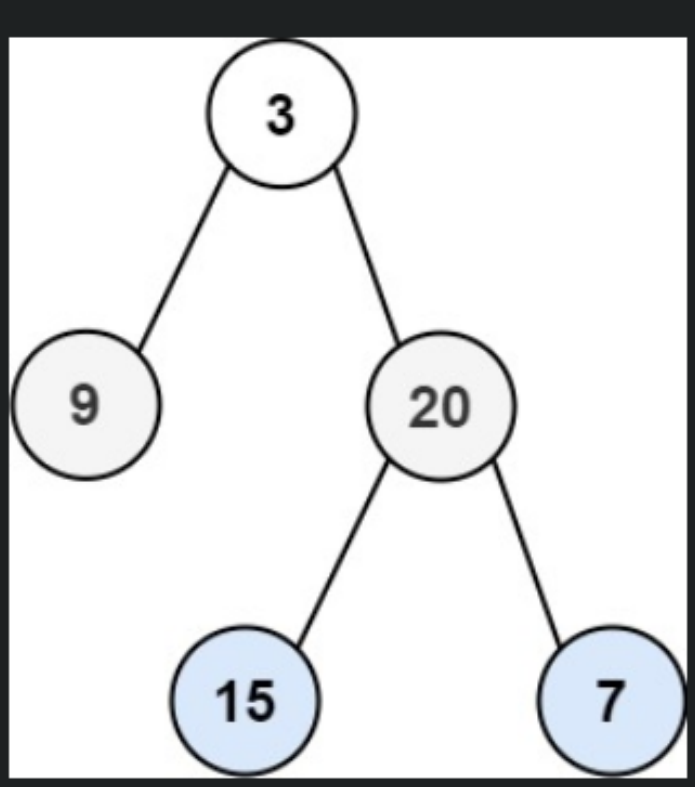
102. Binary Tree Level Order Traversal

Medium 14.6K 290

Companies

Given the `root` of a binary tree, return *the level order traversal of its nodes' values*. (i.e., from left to right, level by level).

Example 1:




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

Code:

```
public List<List<Integer>> levelOrder(TreeNode root) {
    List<List<Integer>> result = new ArrayList<>();

    if (root == null) {
      return result;
    }
    Queue<TreeNode> queue = new LinkedList<>();
    queue.offer(root);

    while(!queue.isEmpty()){
      int levelSize = queue.size();
      List<Integer> currentLevel = new ArrayList<>();

      for(int i = 0; i < levelSize; i++){
        TreeNode currentNode = queue.poll();
        currentLevel.add(currentNode.val);

        if(currentNode.left != null){
          queue.offer(currentNode.left);
        }
        if(currentNode.right != null){
          queue.offer(currentNode.right);
        }
      }
      result.add(currentLevel);
    }
    return result;
}
```

Question:

103. Binary Tree Zigzag Level Order Traversal

Medium 10.3K 269

Companies

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).

Since, We need to traverse level wise, we'll use BFS.

Here, We'll just maintain a boolean pointer to alternate the traversal (left to right then right to left)

Code:

```
public List<List<Integer>> zigzagLevelOrder(TreeNode root) {
  List<List<Integer>> result = new ArrayList();
  if(root == null){
    return result;
  }

  Queue<TreeNode> queue = new LinkedList();
  queue.offer(root);

  boolean isLeftToRight = true; //pointer

  while(!queue.isEmpty()){
    int levelSize = queue.size();
    List<Integer> currentLevel = new ArrayList<>();

    for(int i = 0; i < levelSize; i++){
      TreeNode currentNode = queue.poll();

      if(isLeftToRight){
        currentLevel.add(currentNode.val);
      }
      else{
        currentLevel.add(0, currentNode.val);
      }
      if (currentNode.left != null) {
        queue.offer(currentNode.left);
      }
      if (currentNode.right != null) {
        queue.offer(currentNode.right);
      }

    }
    result.add(currentLevel);
    isLeftToRight = !isLeftToRight;
  }

  return result;
}
```

Question:

107. Binary Tree Level Order Traversal II

Medium 4.7K 318

Companies

Given the `root` of a binary tree, return *the bottom-up level order traversal of its nodes' values*. (i.e., from left to right, level by level from leaf to root).

Same pattern as level order traversal.
Here, we need to return bottom-up traversal so we'll just add the currentLevel in our result at 0th index.
This is the only change we've to do.

```
result.add(0, currentLevel);
```

Question:

637. Average of Levels in Binary Tree

Easy 5K 310

Companies

Given the `root` of a binary tree, return *the average value of the nodes on each level in the form of an array*. Answers within 10^{-5} of the actual answer will be accepted.

We've to find average of each level.
level wise traversal we'll use BFS
Here, we'll calculate the sum of each level & divide it by levelSize

Code:

```
public List<Double> averageOfLevels(TreeNode root) {
    List<Double> result = new ArrayList<>();
    if(root == null){
      return result;
    }
    Queue<TreeNode> queue = new LinkedList();
    queue.offer(root);

    while(!queue.isEmpty()){
      int levelSize = queue.size();
      double nodeSum = 0;

      for(int i = 0; i < levelSize; i++){
      TreeNode currentNode = queue.poll();
      nodeSum = nodeSum + currentNode.val;

      if(currentNode.left != null){
        queue.offer(currentNode.left);
      }

      if(currentNode.right != null){
        queue.offer(currentNode.right);
      }
    }
    result.add(nodeSum / levelSize);
  }
  return result;
}
```

Question:

515. Find Largest Value in Each Tree Row

Medium 3.5K 110

Companies

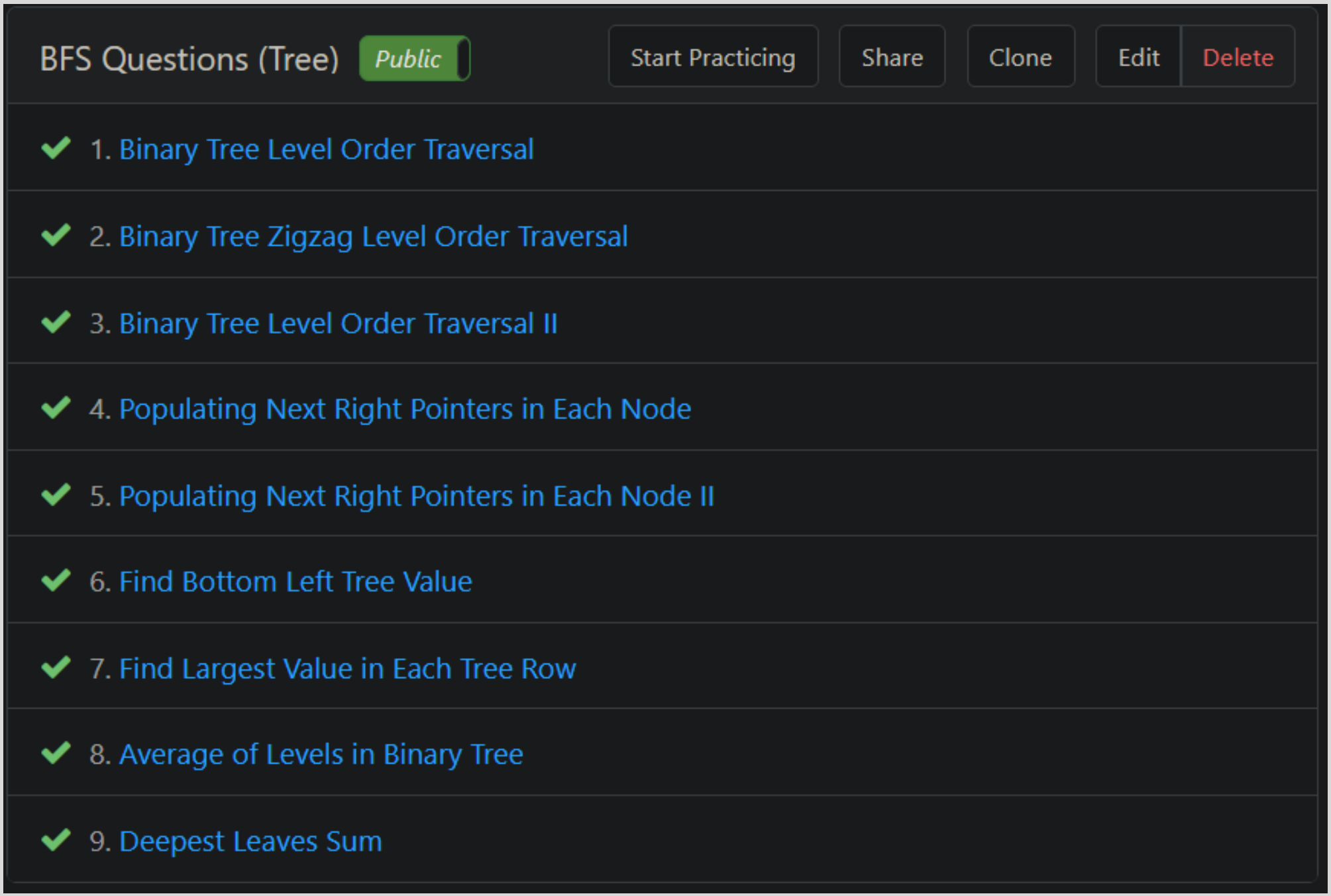
Given the `root` of a binary tree, return *an array of the largest value in each row* of the tree **(0-indexed)**.

Same Pattern again.
We've to find value in each row/level "BFS"
Here, we'll just maintain a "max" to compare the values & update it (level-wise)

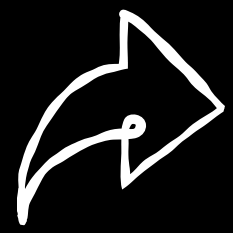
Code:

```
// Just the main logic

while(!queue.isEmpty()){
     int levelSize = queue.size();
// Taking MIN_VALUE here, since the question has -ve values
     int max = Integer.MIN_VALUE;

     for(int i = 0; i < levelSize; i++){
       TreeNode currentNode = queue.poll();

       if(currentNode.val > max){
// Compare & update the max
         max = currentNode.val;
       }
       if(currentNode.left != null){
         queue.offer(currentNode.left);
       }
       if(currentNode.right != null){
         queue.offer(currentNode.right);
       }
     }
     result.add(max);
   }
   return result;
```

Checkout the list I’ve created on Leetcode



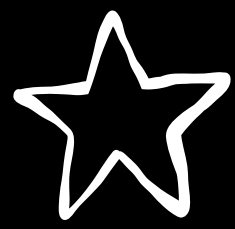
Like!



Share

DID YOU LIKE IT?

Comment



Save