Find Largest Value in Each Tree Row

You need to find the largest value in each row of a binary tree.

Example:

```
Input: 

          1
         / \
        3   2
       / \   \  
      5   3   9 

Output: [1, 3, 9]
```

Solution 1

```
def findValueMostElement(self, root):
    maxes = []
    row = [root]
    while any(row):
        maxes.append(max(node.val for node in row))
        row = [kid for node in row for kid in (node.left, node.right) if kid]
    return maxes
```

written by StefanPochmann original link here

Solution 2

Just a simple pre-order traverse idea. Use depth to expand result list size and put the max value in the appropriate position.

```java
public class Solution {
    public List<Integer> largestValues(TreeNode root) {
        List<Integer> res = new ArrayList<Integer>();
        helper(root, res, 0);
        return res;
    }
    private void helper(TreeNode root, List<Integer> res, int d){
        if(root == null){
            return;
        }
       //expand list size
        if(d == res.size()){
            res.add(root.val);
        }
        else{
        //or set value
            res.set(d, Math.max(res.get(d), root.val));
        }
        helper(root.left, res, d+1);
        helper(root.right, res, d+1);
    }
}
```

written by Ryan777 original link here

Solution 3

Alright, two binary tree level order traversal problems in one contest. This time, mission is to find the `max` of each level...

```
public class Solution {
    public int[] findValueMostElement(TreeNode root) {
        List<Integer> res = new ArrayList<>();
        if (root == null) return new int[0];

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

        while (!queue.isEmpty()) {
            int size = queue.size();
            int max = Integer.MIN_VALUE;
            for (int i = 0; i < size; i++) {
                TreeNode node = queue.poll();
                max = Math.max(max, node.val);
                if (node.left != null) queue.add(node.left);
                if (node.right != null) queue.add(node.right);
            }
            res.add(max);
        }

        int[] result = new int[res.size()];
        for (int i = 0; i < res.size(); i++) {
            result[i] = res.get(i);
        }

        return result;
    }
}
```

written by shawngao original link here