Find Bottom Left Tree Value

Given a binary tree, find the leftmost value in the last row of the tree.

Example 1:

```
Input:

    2
   / \
  1   3

Output:
1
```

Example 2:

```
Input:

        1
       / \
      2   3
     /   / \
    4   5   6
       /
      7

Output:
7
```

Note: You may assume the tree (i.e., the given root node) is not**NULL**.

Solution 1

Typical way to do binary tree level order traversal. Only additional step is to remember the `first` element of each level.

```
public class Solution {
    public int findLeftMostNode(TreeNode root) {
        if (root == null) return 0;

        int result = 0;
        Queue<TreeNode> queue = new LinkedList<>();
        queue.add(root);

        while (!queue.isEmpty()) {
            int size = queue.size();
            for (int i = 0; i < size; i++) {
                TreeNode node = queue.poll();
                if (i == 0) result = node.val;
                if (node.left != null) queue.add(node.left);
                if (node.right != null) queue.add(node.right);
            }
        }

        return result;
    }
}
```

written by shawngao original link here

Solution 2

```
public class Solution {
    int ans=0, h=0;
    public int findBottomLeftValue(TreeNode root) {
        findBottomLeftValue(root, 1);
        return ans;
    }
    public void findBottomLeftValue(TreeNode root, int depth) {
        if (h<depth) {ans=root.val;h=depth;}
        if (root.left!=null) findBottomLeftValue(root.left, depth+1);
        if (root.right!=null) findBottomLeftValue(root.right, depth+1);
    }
}
```

No global variables, 6ms (faster):

```
public class Solution {
    public int findBottomLeftValue(TreeNode root) {
        return findBottomLeftValue(root, 1, new int[]{0,0});
    }
    public int findBottomLeftValue(TreeNode root, int depth, int[] res) {
        if (res[1]<depth) {res[0]=root.val;res[1]=depth;}
        if (root.left!=null) findBottomLeftValue(root.left, depth+1, res);
        if (root.right!=null) findBottomLeftValue(root.right, depth+1, res);
        return res[0];
    }
}
```

written by ckcz123 original link here

Solution 3

Got the idea from @fallcreek. Doing BFS right to left means we can simply return the **last** node's value and don't have to keep track of the first node in the current tree row.

Python:

```
def findLeftMostNode(self, root):
    queue = [root]
    for node in queue:
        queue += filter(None, (node.right, node.left))
    return node.val
```

Java:

```
public int findLeftMostNode(TreeNode root) {
    Queue<TreeNode> queue = new LinkedList<>();
    queue.add(root);
    while (!queue.isEmpty()) {
        root = queue.poll();
        if (root.right != null)
            queue.add(root.right);
        if (root.left != null)
            queue.add(root.left);
    }
    return root.val;
}
```

written by StefanPochmann original link here