JavaScript Find Bottom Left Tree Value

Challenge

Given the `root` of a binary tree, return the leftmost value in the last row of the tree.

1st Example

```
Input: root = [2,1,3]
Output: 1
```

2nd Example

```
Input: root = [1,2,3,4,null,5,6,null,null,7]
Output: 7
```

Constraints

- `-2³¹ <= Node.val <= 2³¹ - 1`
- The number of nodes in the tree is in the range `[1, 10⁴]`.

Solution

```
const findBottomLeftValue = (root) => {
    if (root === null) return null;

    const queue = [root];
    let left = null;

    while (queue.length) {
        const node = queue.shift();

        if (node.left === null &&
            node.right === null) left = node.val;

        if (node.right) queue.push(node.right);

        if (node.left) queue.push(node.left);
    }

    return left;
};
```

Explanation

I've written a function called `findBottomLeftValue` that takes in a root node of a binary tree as an argument. The purpose of this function is to find and return the value of the leftmost node in the bottom level of the tree.

The function begins by checking if the root node is null. If it is, it means the tree is empty, so the function immediately returns null.

If the root node is not null, the function creates a queue array and adds the root node to it. This queue will be used to perform a breadth-first search traversal of the tree.

A variable named `left` is initialized as null. This variable will store the value of the leftmost node in the bottom level of the tree.

A while loop is used to iterate as long as the queue array has elements. This loop will continue until all nodes in the tree have been processed.

Inside the loop, the first element of the queue is removed using the `shift()` method and assigned to the variable `node`. This represents the current node being processed.

The function checks if the `node` has both left and right child nodes as null. If it does, it means the `node` is a leaf node and is the leftmost node in the bottom level. In this case, the value of the `node` is assigned to the `left` variable.

If the `node` has a right child, it is added to the end of the queue array using the `push()` method.

If the `node` has a left child, it is also added to the end of the queue array.

The loop continues until all nodes in the tree have been processed.

Finally, the `left` variable, which stores the value of the leftmost node in the bottom level, is returned as the result of the function.

In summary, this function performs a breadth-first search traversal

of a binary tree to find and return the value of the leftmost node in the bottom level of the tree.

Author: Trevor Morin