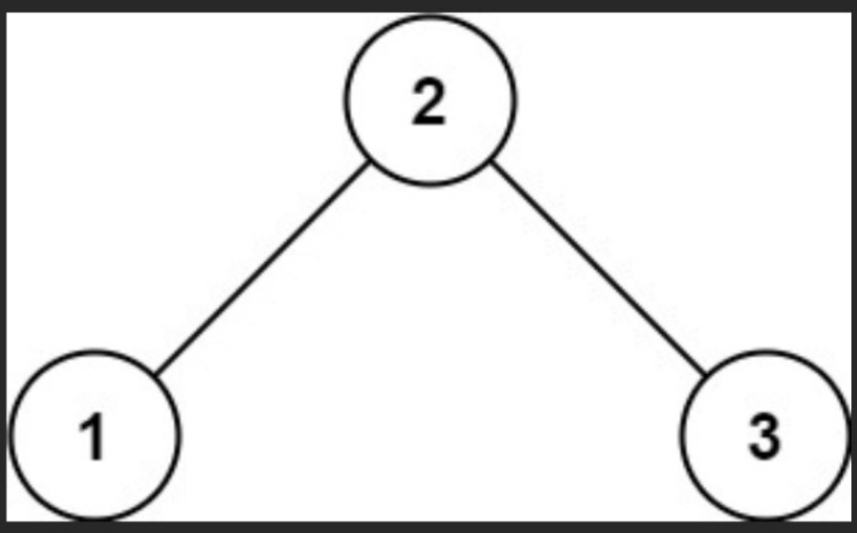
513. Find Bottom Left Tree Value

Medium 3.1K 252

Companies

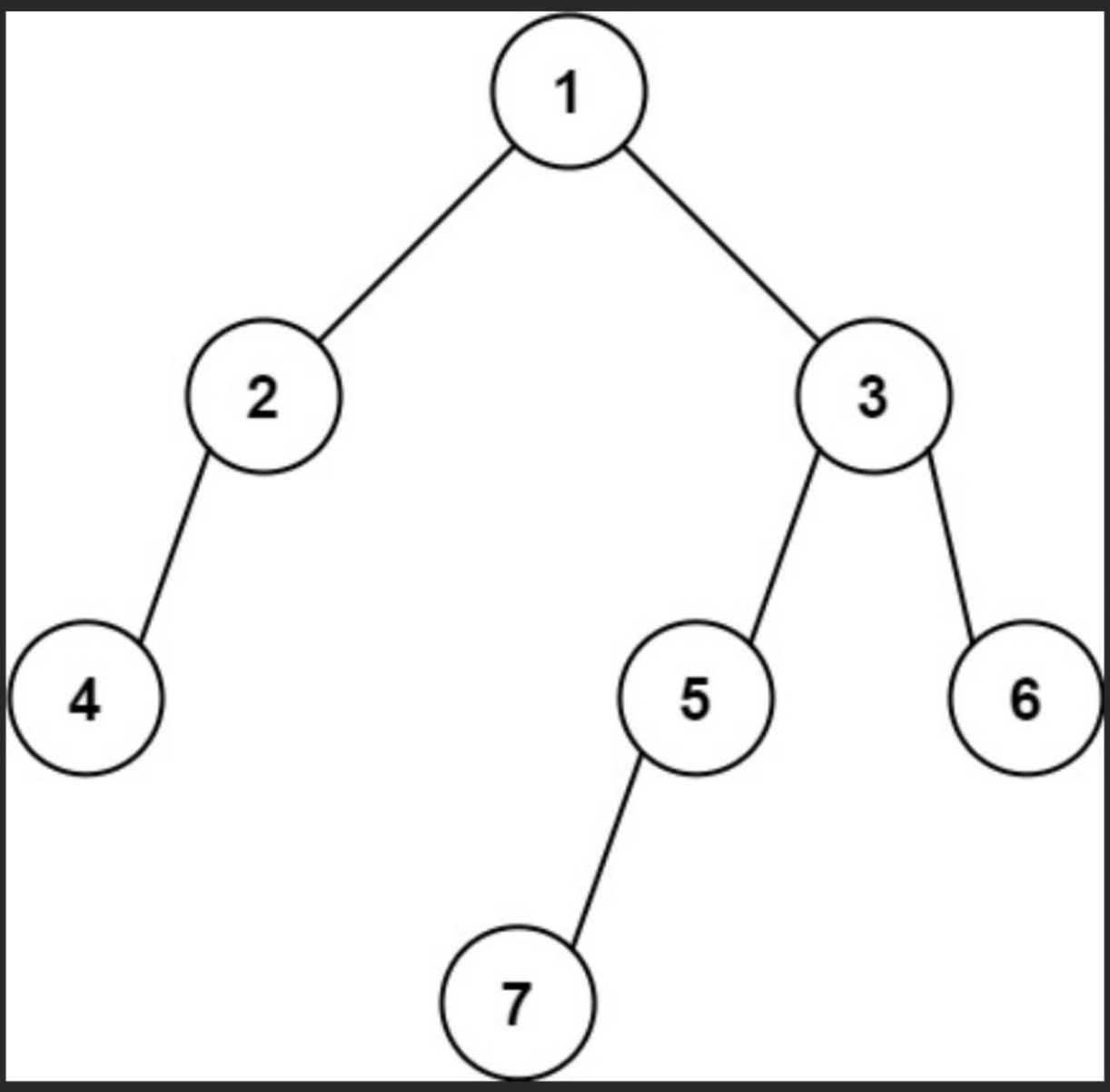
Given the `root` of a binary tree, return the leftmost value in the last row of the tree.

Example 1:




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

Example 2:




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

Constraints:

- The number of nodes in the tree is in the range `[1, 10^4]`.
- `-2^31 <= Node.val <= 2^31 - 1`

Approach 1: Recursive implementation

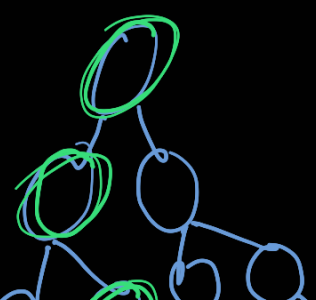
at a node we make a decision on which direction to move for bottom most left tree value:

if leftheight $\geq$ right height
    move left
else if leftheight < right height
    move right

```
find (root)
{
    if (root is a leaf node)
        return root->val

    leftheight = maxheight(root->left)
    rightheight = maxheight(root->right)

    if (leftheight ≥ rightheight)
        find(root->left)
    else
        find(root->right)
}
```

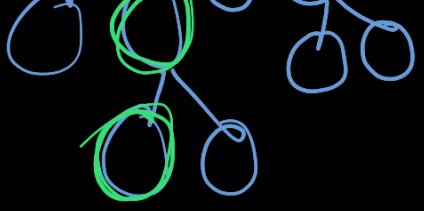


T(n): O(h x (h+h))

leftheight

right

height



note: The maxheight function gets called only on marked nodes two times. That's why T(n) is h * (h+th).

for skewed Trees
$T(n) = O(n^2)$

Approach 2: Recursive implementation

By keeping track of maxdepth and current depth throught all function calls we can determine bottom left tree value.

```
ans = -1;
get (root, int &maxdepth, depth)
{
    if(root is null)
        return

    if(depth > maxdepth)
    { ans = root->val
      maxdepth = depth
    }

    get(root->left, maxdepth, depth+1)
    get(root->right, maxdepth, depth+1)
}
```

}

T(n): O(n)

Approach 3: Iterative implementation

Just keep changing the ans to the first node of queue until all levels are traversed.

```
int ans;
q.push(root)

while( q is not empty)
{
    size = q.size()
    ans = q.front() ->val
    while (size--)
    {
        auto node = q.front()
        q.pop()
        if(node has non null left node)
            push left node to queue
        if(node has non null right node)
            push right node to queue
    }
}
```

note: we can also do using single while loop

T(n): O(n)
S(n): O(max nodes at