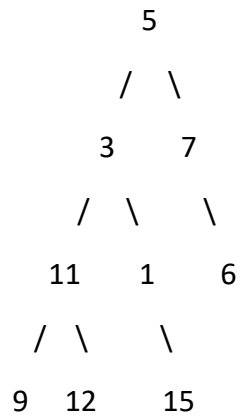
Finding Bottom Left Value in Binary Tree LeetCode

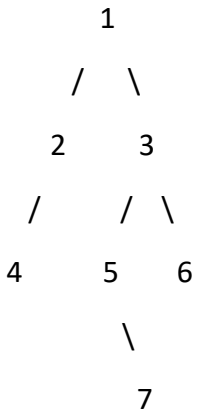
Given the root of a binary tree, return the leftmost value in the last row of the tree.

Example 1:



Output: The left-most value of Binary Tree: 9

Example 2:



Output: The left-most value of Binary Tree: 7

Approach 1: Function to find the bottom-left value of a binary tree using a recursive approach

- The recursive approach explores the tree by recursively traversing both the left and right subtrees.
- At each level, it checks if the current node's height is greater than the maximum height seen so far.
- If so, it updates the answer with the current node's value and the maximum height.
- The recursion continues until all nodes have been visited.
- The final answer represents the bottom-left value.

Time Complexity:

- **The time complexity is O(N), where N is the number of nodes in the tree, as each node is visited once.**

Space Complexity:

- **The space complexity is O(H), where H is the height of the tree, due to the recursive call stack.**

Approach 2: Function to find the bottom-left value of a binary tree using an iterative approach

- The iterative approach uses a level-order traversal with a queue.
- It starts at the root and explores nodes level by level.
- While processing each level, it updates the answer with the value of the leftmost node.
- The process continues until all levels have been traversed.
- The final answer represents the bottom-left value.

Time Complexity:

- **The time complexity is O(N), where N is the number of nodes in the tree, as each node is visited once during traversal.**

Space Complexity:

- **The space complexity is O(N) in the worst case due to the queue storing all nodes at a level.**

Conclusion:

Both the recursive and iterative approaches effectively find the bottom-left value in a binary tree. They share the same time complexity of O(N), ensuring efficient traversal of all nodes. However, their space complexities differ.

- The recursive approach has a space complexity of O(H).
- The iterative approach has a space complexity of O(N) due to the queue.

In terms of memory efficiency, the recursive approach is preferable when memory is a concern, as it has a lower space complexity. However, the iterative approach provides a straightforward solution and is suitable for relatively small trees.