


Documentation for Find Bottom Left Tree Value


1. Problem Statement

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, 10⁴].

$-2^{31} \leq \text{Node.val} \leq 2^{31} - 1$

2. Intuition

In a level-order traversal (BFS), we process the tree level by level. The last level's leftmost node is simply the first node we encounter in that level. So, if we process the tree level-by-level and capture the first node at each level, the last one encountered will be our result.

3. Key Observations

- BFS processes nodes level by level, ideal for this problem.
- In BFS, if we push right before left, the last node processed will be the leftmost of the last level.
- This avoids the need to track each level separately.

4. Approach

We use Breadth-First Search (BFS):

- Initialize a queue and insert the root.

- While the queue is not empty:
 - Pop the front node.
 - Push right child first, then left child.
- After the loop ends, the last popped node is the leftmost node in the last level.

5. Edge Cases

- Tree has only one node → Return the value of that node.
- Tree is skewed (e.g., all left or all right) → Still works as it scans every level.
- Unbalanced tree → Still covered by BFS.

6. Complexity Analysis

□ Time Complexity

- $O(n)$: Every node is visited exactly once.

📦 Space Complexity

- $O(n)$: At most, the queue stores one level of the tree (in the worst case).

7. Alternative Approaches

🔄 DFS (Depth-First Search) with Depth Tracking:

- Track the maximum depth and update the leftmost value.
- More code and recursion stack usage.
- Still $O(n)$ time and space.

8. Final Thoughts

- BFS is ideal here as it naturally traverses the tree level-by-level.
- Reversing the child insertion order ensures we end with the leftmost node.
- This is a great example of how small changes in BFS logic can help solve specific positional problems efficiently.