Sum of Longest Bloodline of Binary Tree GFG

Find the sum of all nodes on the longest path from root to leaf node.
If two or more paths compete for the longest path, then the path having maximum sum of nodes is being considered.

Example 1:

```
              5
            /   \
          3       7
        /   \       \
      11      1       6
    /   \           /   \
   9     12       13     15
```

Output: The Sum of Longest Bloodline: (5 → 7 → 6 → 15) → 33

Example 2:

```
              5
            /   \
          3       7
        /   \       \
      11      1       6
    /   \           /   \
   9     12       13     15
        /
      1
```

Output: The Sum of Longest Bloodline: (5 → 3 → 11 → 12 → 1) → 32

Approach 1: Function to calculate the sum of the longest bloodline in a binary tree using recursion

- The recursive approach uses a helper function **solve** to traverse the tree.

- It maintains two variables, **maxHeight** and **maxSum**, to track the maximum height and the corresponding maximum sum encountered during traversal.
- The helper function calculates the sum along a path from the root to a leaf and compares it with the maximum sum found so far.
- If the current path has a greater sum or the same sum but a greater height, it updates **maxSum** and **maxHeight**.
- The main function initializes **maxSum** and **maxHeight** with the root node's value and height 0 and then calls the **solve** function.
- Finally, it returns the maximum sum found.

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(H), where H is the height of the binary tree, due to the function call stack.**

Approach 2: Function to calculate the sum of the longest bloodline in a binary tree using iteration.

- The iterative approach uses a queue for level-order traversal.
- It maintains a pair **(height, sum)** for each node in the queue, where **height** represents the height of the node in the tree, and **sum** represents the sum of node values along the path from the root to that node.
- It starts with the root node and initializes **maxSum** and **maxHeight** accordingly.
- During traversal, it updates **maxSum** and **maxHeight** when a higher sum or the same sum with a greater height is encountered.
- The process continues until all nodes are processed.

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 level-order traversal.**

Space Complexity:

- **The space complexity is O(N), where N is the number of nodes in the tree, due to the queue.**

Conclusion:

Both the recursive and iterative approaches effectively calculate the sum of the longest bloodline in a binary tree. The recursive approach uses depth-first traversal, while the iterative approach uses level-order traversal. Both approaches have a time complexity of O(N) and provide the expected maximum sum.

The choice between the two approaches depends on the specific requirements and constraints of the problem.