Check if the given Binary Tree a Min Heap

Given a binary tree. The task is to check whether the given tree follows the **min heap** property or not.  
**Note:**Properties of a tree to be a min heap - Completeness and Value of node less than or equal to its child.

Example 1:

7

/ \

5 3

/ \ / \

4 3 2 1

Output: The Binary Tree is not a Min Heap

Example 2:

10

/ \

20 30

/ \

40 50

Output: The Binary Tree is a Min Heap

**Approach 1: Function to check if the binary tree is a heap using the recursive approach**

* **Function Purpose:** To check if the binary tree is a min heap using a recursive approach.
* **Explanation:**
  + This approach checks if the binary tree is a min heap by recursively examining each node.
  + It starts at the root and checks if the current node is less than its child nodes (left and right).
  + It also ensures that the tree is complete by checking that nodes at the same level are filled from left to right.
  + The recursion continues down the tree, checking each subtree.
  + The tree is a min heap if the value of each node is less than the values of its children, and the tree is complete.
* **Time Complexity:** **O(N), where N is the number of nodes in the tree.**
* **Space Complexity: O(H), where H is the height of the tree, as it uses the call stack for recursion.**

**Approach 2: Function to check if the binary tree is a heap using the iterative approach**

* **Function Purpose:** To check if the binary tree is a min heap using an iterative approach.
* **Explanation:**
  + This approach checks if the binary tree is a min heap by performing a level-order traversal of the tree.
  + It uses a queue to traverse the tree level by level, starting from the root.
  + At each level, it checks if the current node's value is less than its child nodes (left and right).
  + If any node violates the min heap property, the tree is not a min heap.
  + Additionally, it ensures that the tree is complete by verifying that nodes at the same level are filled from left to right.
* **Time Complexity: O(N), where N is the number of nodes in the tree.**
* **Space Complexity: O(N), as it stores nodes at each level in a queue.**

**Approach 3: Function to check if the binary tree is a heap using an optimized approach**

* **Function Purpose:** To check if the binary tree is a min heap using an optimized iterative approach.
* **Explanation:**
  + This approach checks if the binary tree is a min heap by performing a level-order traversal of the tree.
  + Similar to the iterative approach, it uses a queue to traverse the tree level by level.
  + At each level, it checks if the current node's value is less than its child nodes (left and right).
  + If any node violates the min heap property, the tree is not a min heap.
  + It also ensures that the tree is complete by verifying that nodes at the same level are filled from left to right.
* **Time Complexity:** **O(N), where N is the number of nodes in the tree.**
* **Space Complexity: O(N), as it stores nodes at each level in a queue.**

**Conclusion:**

**Among the three approaches, the recursive approach may be preferred for its simplicity and efficient space usage (O(H)),** while the iterative approaches also work but have a slightly higher space complexity (O(N)).