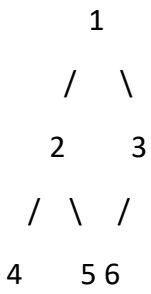
Check Completeness of Binary Tree LeetCode

Given the root of a binary tree, determine if it is a *complete binary tree*.

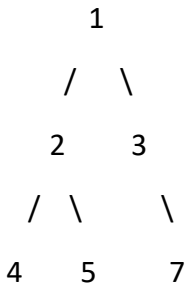
In a **complete binary tree** every level, except possibly the last, is completely filled, and all nodes in the last level are as far left as possible. It can have between 1 and 2^h nodes inclusive at the last level h.

Example:



Output: The Given Binary Tree is Complete Binary Tree

Example:



Output: The Given Binary Tree is not Complete Binary Tree

Approach 1: Recursive function to check if a binary tree is a complete binary tree

Function Purpose:

Check if a given binary tree is a complete binary tree using a recursive approach.

Explanation:

- **countNodes Function:**
    - Counts the total number of nodes in the binary tree.
- **isCBTHelper Function:**
    - Helper function for recursive checking of whether a binary tree is complete.
    - Checks if the current node is beyond the total number of nodes.

    - Recursively checks the left and right subtrees.
    - Returns true if both subtrees are complete.
- **isCompleteTreeRecursively Function:**
    - Initializes the recursive check by counting nodes and calling the helper function.

Time Complexity:

- **The time complexity is O(N), where N is the total number of nodes in the binary tree.**

Space Complexity:

- **The space complexity is O(H), where H is the height of the binary tree.**

Approach 2: Iterative function to check if a binary tree is a complete binary tree

Function Purpose:

Check if a given binary tree is a complete binary tree using an iterative approach.

Explanation:

- **isCompleteTreeIteratively Function:**
    - Uses an iterative level-order traversal with a queue to check for a complete binary tree.
    - Flags a non-complete structure when a null node is encountered.
    - Returns true if the loop completes without returning false.

Time Complexity:

- **The time complexity is O(N), where N is the total number of nodes in the binary tree.**

Space Complexity:

- **The space complexity is O(N), where N is the total number of nodes in the binary tree.**

Conclusion:

- Both recursive and iterative approaches are provided to check if a given binary tree is a complete binary tree.
- **Approach 1 is having better space complexity.**