Root Equals Sum of Children in Binary Tree [LeetCode](https://leetcode.com/problems/root-equals-sum-of-children/description/)

Return true *if the value of the root is equal to the****sum****of the values of its two children, or*false*otherwise*.

Example:

26

/ \

10 3

/ \ \

4 6 3

Output: The Binary tree is not a Sum tree

**Approach 1: Recursive approach to check if a binary tree is a sum tree**

* In the **sumTreeRecursive** function, you check if a binary tree is a sum tree using a recursive approach.
* A sum tree is a binary tree in which the value of each node is equal to the sum of the values of its left and right subtrees.
* The base cases check if the node is null or if it's a leaf node (no left and right children), in which case it's considered a sum tree.
* Recursively, you check the left and right subtrees and verify if the current node satisfies the sum tree property.
* If all conditions are met, you return **true**; otherwise, you return **false**.

**Time Complexity:** **O(N), where N is the number of nodes in the binary tree. You visit each node once.**

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

**Approach 2: Iterative approach using a queue to check if a binary tree is a sum tree**

* In the **sumTreeIterative** function, you check if a binary tree is a sum tree using an iterative approach.
* You perform level-order traversal using a queue.
* At each node, you check if it satisfies the sum tree property (value equals the sum of its left and right children).
* If all nodes satisfy the property, the entire tree is considered a sum tree.

**Time Complexity:** **O(N), where N is the number of nodes in the binary tree. You visit each node once.**

**Space Complexity: O(H), where H is the height of the binary tree due.**

**Approach 3: Optimized recursive approach to check if a binary tree is a sum tree**

* In the **sumTreeOptimized** function, you check if a binary tree is a sum tree using an optimized recursive approach.
* A pair of values **(bool, int)** is returned from the helper function.
* The **bool** value indicates whether the subtree rooted at the current node is a sum tree.
* The **int** value represents the sum of all nodes in the subtree.
* The base cases are similar to Approach 1, but the optimized approach combines the checks for left and right subtrees and the current node's sum.

**Time Complexity:** **O(N), where N is the number of nodes in the binary tree. You visit each node once.**

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