Step-wise Algorithm for Checking if a Binary Tree is a Sum Tree

This document explains the step-by-step algorithm to check whether a given binary tree is a **Sum Tree**. A Sum Tree is a binary tree where the value of each non-leaf node is equal to the sum of the values of its left and right subtrees.

Step-by-Step Algorithm:

1. Start with a recursive function `subtree(Node* root)` that returns the sum of a subtree if it satisfies the Sum Tree property, otherwise returns -1.
2. Base Case 1: If the current node (`root`) is NULL, return 0 (an empty tree contributes 0 to the sum).
3. Base Case 2: If the node is a leaf node (both left and right children are NULL), return the node's value.
4. Recursively call the function on the left child and store the result in `left`.
5. Recursively call the function on the right child and store the result in `right`.
6. If either `left` or `right` is -1, propagate -1 upwards because one of the subtrees is invalid.
7. Check if the current node's value equals `left + right`. If true, return the total sum of the subtree (`root->data + left + right`).
8. If the condition fails, return -1 because the Sum Tree property is violated.
9. In the main function `isSumTree(Node* root)`, call the `subtree` function.
10. If the result is not -1, return true (the tree is a Sum Tree). Otherwise, return false.

This algorithm ensures that every node is checked only once, making the solution efficient with a time complexity of **O(N)**, where N is the number of nodes in the tree.