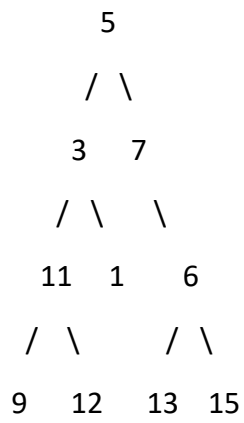
Path Sum of Binary Tree LeetCode

Given the root of a binary tree and an integer targetSum, return true if the tree has a **root-to-leaf** path such that adding up all the values along the path equals targetSum.

Example:



TargetSum: 31

Output: The Target Sum: 31 has Root-To-Leaf Path (5 – 3 – 11 – 12)

TargetSum: 15

Output: The Target Sum: 15 does not have Root-To-Leaf Path

Approach 1: Recursive approach to Check whether there's a path with a given sum

- In the **hasPathSumHelper** function, you recursively check whether there's a root-to-leaf path with a given sum.
- For each node, you check if it's a leaf node (no left or right children) and if the sum matches the targetSum.
- You recursively check both the left and right subtrees.
- The final result is **true** if any of the paths from root to leaf nodes adds up to the targetSum.

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: Optimized Recursive approach version to check whether there's a path with a given sum

- In the **hasPathSumRecursiveOptimized** function, you recursively check whether there's a root-to-leaf path with a given sum.
- For each node, you subtract its value from the targetSum.
- If the current node is a leaf node (no left or right children) and the remaining targetSum is 0, you return **true**.
- You recursively check both the left and right subtrees.
- The final result is **true** if any of the paths from root to leaf nodes adds up to 0.

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 3: Iterative approach: Check whether there's a path with a given sum

- In the **hasPathSum** function, you check whether there's a root-to-leaf path with a given sum using an iterative approach with a stack.
- You start from the root and push nodes onto the stack along with the current sum.
- If you encounter a leaf node with a sum equal to the targetSum, you return **true**.
- You continue to push left and right children onto the stack for further processing.
- If no path is found, you return **false** at the end.

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

Space Complexity: O(W), where W is the maximum width of the binary tree at any level.