Finding Paths with Target Sum in Binary Tree [LeetCode](https://leetcode.com/problems/path-sum-ii/description/)

Given the root of a binary tree and an integer targetSum, return *all****root-to-leaf****paths where the sum of the node values in the path equals*targetSum*. Each path should be returned as a list of the node****values****, not node references*.

A **root-to-leaf** path is a path starting from the root and ending at any leaf node. A **leaf** is a node with no children.

Example 1:

5

/ \

3 7

/ \ \

11 1 16

/ \ \

9 12 15

Target Sum = 28

Output: The Paths with Total Sum: 28 = [[5,3,11,9], [5,7,16]]

**Approach 1: Function to find all paths in a binary tree that sum up to the target sum using a recursive approach**

* The recursive approach explores the tree by recursively traversing both the left and right subtrees.
* At each node, it subtracts the current node's value from the target sum and appends the node's value to a temporary path.
* If it reaches a leaf node and the target sum is 0, it adds the path to the answer.
* The recursion continues until all nodes have been visited.
* The final answer represents all paths with the target sum.

**Time Complexity:**

* **The time complexity is O(N), where N is the number of nodes in the tree, as each node is visited once.**

**Space Complexity:**

* **The space complexity is O(H), where H is the height of the tree, due to the recursive call stack.**

**Approach 2: Function to find all paths in a binary tree that sum up to the target sum using an iterative approach**

* The iterative approach uses a level-order traversal with a queue.
* It starts at the root and explores nodes level by level.
* While processing each node, it maintains the current sum.
* If it reaches a leaf node and the current sum is equal to the target sum, it adds the path to the answer.
* The process continues until all levels have been traversed.
* The final answer represents all paths with the target sum.

**Time Complexity:**

* **The time complexity is O(N), where N is the number of nodes in the tree, as each node is visited once during traversal.**

**Space Complexity:**

* **The space complexity is O(N) in the worst case due to the queue storing all nodes at a level.**

**Conclusion:**

Both the recursive and iterative approaches effectively find all paths in a binary tree that sum up to the target sum. They share the same time complexity of O(N), ensuring efficient traversal of all nodes. However, their space complexities differ.

* The recursive approach has a space complexity of O(H).
* The iterative approach has a space complexity of O(N) due to the queue.

In terms of memory efficiency, the recursive approach is preferable when memory is a concern, as it has a lower space complexity. However, the iterative approach provides a straightforward solution and is suitable for relatively small trees.