112. Path Sum

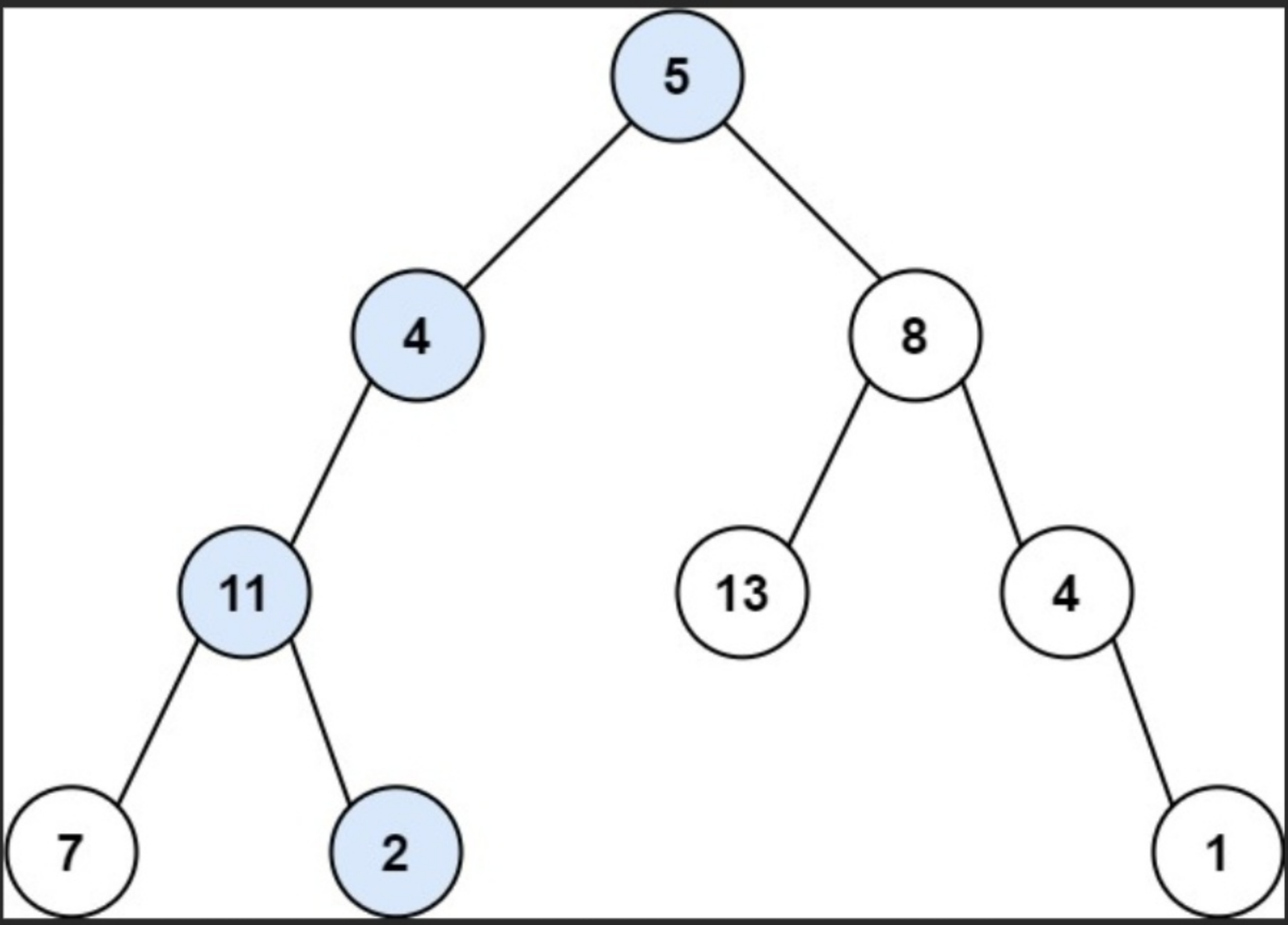
Easy 9K 1K

Companies

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`.

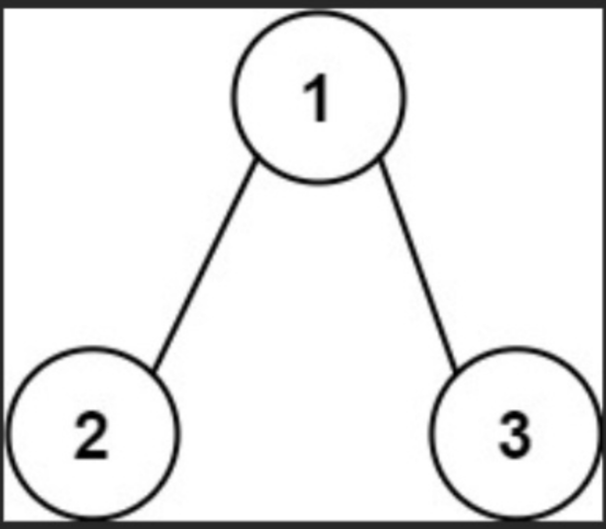
A **leaf** is a node with no children.

Example 1:



```
Input: root = [5,4,8,11,null,13,4,7,2,null,null,null,1], targetSum = 22
Output: true
Explanation: The root-to-leaf path with the target sum is shown.
```

Example 2:



```
Input: root = [1,2,3], targetSum = 5
Output: false
Explanation: There two root-to-leaf paths in the tree:
(1 --> 2): The sum is 3.
(1 --> 3): The sum is 4.
There is no root-to-leaf path with sum = 5.
```

Example 3:

```
Input: root = [], targetSum = 0
Output: false
Explanation: Since the tree is empty, there are no root-to-leaf paths.
```

Approach 1: Recursive implementation

```
bool check( root , target , Sum)
{
        if( root is null)
            return false

        Sum = Sum + root ->val

        if( root is leaf node && Sum == target)
            return true

        return  check(root->left , target , Sum) ||
                check (root -> right , target , Sum)
}
```

$T(n) : O(n)$

Approach 2: Iterative implementation

we can keep track of Current Sum along with node as a pair in queue

```
queue <pair<Treenode*, int >> q , Sum=0
q.push({ root , root->val })
```

```
while(q is not empty)
{
        auto node = q.front().first
          int  sum = q.front() .second

      if(node is leaf node && targetSum == Sum)
            return true

      if( node has non null left node)
          q.push({node->left,     sum +
                                  node -> left ->val})

      if ( node has non null right node)
          q.push({ node->right ,   sum +
                                   node->right->val})
}
```

T(n) : O(n)
S(n) : O(max nodes at
  a level)