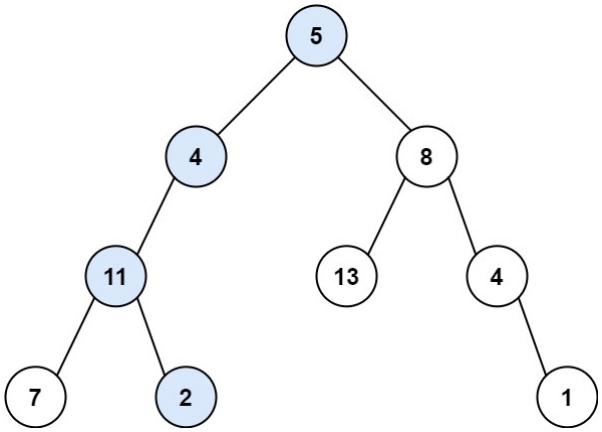
112 路径总和

Label：二叉树

给你二叉树的根节点 root 和一个表示目标和的整数 targetSum ，判断该树中是否存在 根节点到叶子节点 的路径，这条路径上所有节点值相加等于目标和 targetSum 。

叶子节点 是指没有子节点的节点。



- 递归

```
class Solution {
    public boolean hasPathSum(TreeNode root, int sum) {
        if (root == null) {  // 不能提前剪枝，因为可能会存在负数
            return false;
        }
        if (root.left == null && root.right == null) {
            return sum == root.val;
        }

        return hasPathSum(root.left, sum - root.val) || hasPathSum(root.right,
sum - root.val);
    }
}
```

- mine(不知道为什么，用例[1,2]1要要求返回false)

```
class Solution {
    public boolean hasPathSum(TreeNode root, int sum) {
        if (root == null) {  // 不能提前剪枝，因为可能会存在负数
            return false;
        }
        if (sum == root.val) {
            return true;
        }
        return hasPathSum(root.left, sum - root.val) || hasPathSum(root.right,
sum - root.val);
    }
}
```

- 广度优先(栈)、迭代

```
class Solution {
    public boolean hasPathSum(TreeNode root, int sum) {
        if (root == null) return false;
        Queue<TreeNode> queNode = new LinkedList<TreeNode>();
        Queue<Integer> queVal = new LinkedList<Integer>();  // 保存目前节点路径上加
和
        queNode.offer(root);
        queVal.offer(root.val);

        while (!queNode.isEmpty()) {
            TreeNode currNode = queNode.poll();
            int currValue = queVal.poll();

            if (currNode.left == null && currNode.right == null) {
                if (currValue == sum) {
                    return true;
                }
                continue;
            }

            // 入栈，因为是二叉树，所以入栈两次即可
            if (currNode.left != null) {
                queNode.offer(currNode.left);
                queVal.offer(currNode.left.val + currValue);
            }
            if (currNode.right != null) {
                queNode.offer(currNode.right);
                queVal.offer(currNode.right.val + currValue);
            }
        }
        return false;
    }
}
```