```swift
import UIKit
/// **剑指 Offer 34. 二叉树中和为某一值的路径**
/// 给你二叉树的根节点 root 和一个整数目标和 targetSum ，
/// 找出所有**从根节点到叶子节点**路径总和等于给定目标和的路径。
///
/// 叶子节点 是指没有子节点的节点。
/// 输入：root = `[5,4,8,11,null,13,4,7,2,null,null,5,1]`, targetSum = 22
/// 输出：`[[5,4,11,2],[5,8,4,5]] `
class Solution {
    func pathSum(_ root: TreeNode?, _ target: Int) -> [[Int]] {
        var track: [Int] = []
        var res: [[Int]] = []
        backTrack(root, &track, target, &res)
        return res
    }
    /// 本题实际上是一个标准的回溯算法
    /// - Parameters:
    ///   - root: 当前的树根节点
    ///   - track: 已经做出的选择，已经选的树的节点构成的列表
    ///   - target: 目标和
    ///   - res: 最终结果
    private func backTrack(_ root: TreeNode?,
                           _ track: inout [Int],
                           _ target: Int,
                           _ res: inout [[Int]]) {
        /// 如果当前节点为空直接返回
        if root == nil { return }
        /// 如果当前节点有值添加该值
        if let root = root { track.append(root.val) }
        /// 判断总和是否等于 target 同时判断当前的节点是否为最后一个点
        if ((track.reduce(0, +) == target) &&
            (root?.left == nil) &&
            (root?.right == nil) ) { res.append(track) }
        /// 回溯的标准步骤
        backTrack(root?.left, &track, target, &res)
        backTrack(root?.right, &track, target, &res)
        /// 撤销选择
        track.popLast()
    }
}

let solution = Solution()
let rootList1: [Int?] = [5,4,8,11,nil,13,4,7,2,nil,nil,5,1]
let root1 = buildTreeFromLevelOrder(rootList1)
print(solution.pathSum(root1, 22))
```