```swift
import Foundation
/// **剑指 Offer 07. 重建二叉树**
/// 输入某二叉树的前序遍历和中序遍历的结果，请构建该二叉树并返回其根节点。
/// **前序遍历性质： 节点按照 [ 根节点 | 左子树 | 右子树 ] 排序。**
/// **中序遍历性质： 节点按照 [ 左子树 | 根节点 | 右子树 ] 排序。**
///
/// 假设输入的前序遍历和中序遍历的结果中都不含重复的数字。

/// **前序遍历的数字中第一个数字必为树的 root 节点**
/// **在中序遍历中找到该 root 节点并以该节点为中心将中序遍历数组划分为左右两部分**
/// **该左右两部分即为 root 的左右两子树**
/// 前序遍历: `[3, | 9, | 20, 15, 7]`
/// 中序遍历: `[9, | 3, | 15, 20, 7]`
/// 上述例子中 3 为 root 节点，[9]为左子树的所有节点，[15, 20, 7]为右子树的所有节点
/// 由于左子树 [9] 就一个数字且前序遍历中 9 为开头因此左子树只有一个 9 节点
/// 右子树的判断可以将两种遍历剩下的节点提取出来重复上述步骤即可得到
class Solution {
    /// 使用哈希表存储中序遍历的值与索引的映射
    private var dict: [Int: Int] = [:]
    private var preorder: [Int] = []

    func buildTree(_ preorder: [Int], _ inorder: [Int]) -> TreeNode? {
        /// 存放前序遍历用于后续 recur 使用
        self.preorder = preorder
        /// 获取中序遍历中每一个元素的索引位置
        inorder.enumerated().forEach { index, value in
            self.dict[value] = index
        }
        return self.recur(0, 0, inorder.count-1)
    }

    private func recur(_ root: Int, _ left: Int, _ right: Int) -> TreeNode? {
        if left > right { return nil }
        /// 获取根节点
        let node = TreeNode(self.preorder[root])
        /// 获取根节点所在的中序遍历内的位置
        let rootIndex = self.dict[self.preorder[root]] ?? 0
        /// 左子树的根的索引为先序中的根节点+1
        /// 递归左子树的左边界不变
        /// 递归左子树的右边界为中序中的根节点索引-1
        node.left = recur(root + 1, left, rootIndex - 1)
        /// 右子树的根的索引为前序中的 当前根位置 + 当前根中左子树的数量 + 1
        /// 递归右子树的左边界为中序中当前根节点+1
        /// 递归右子树的右边界为中序中原来右子树的边界
        node.right = recur(root + (rootIndex - left) + 1, rootIndex + 1, right)
        return node
    }
}
```