```swift
import Foundation
/// **剑指 Offer 35. 复杂链表的复制**
/// 请实现 copyRandomList 函数，复制一个复杂链表。
/// 在复杂链表中，每个节点除了有一个 next 指针指向下一个节点，
/// 还有一个 random 指针指向链表中的任意节点或者 null。
///
/// 输入：head = `[[7,null],[13,0],[11,4],[10,2],[1,0]]`
/// 输出：`[[7,null],[13,0],[11,4],[10,2],[1,0]]`
///
/// 输入：head = `[[1,1],[2,1]]`
/// 输出：`[[1,1],[2,1]]`
public class Node {
    public var val: Int
    public var next: Node?
    public var random: Node?
    public init(_ val: Int) {
        self.val = val
        self.next = nil
        self.random = nil
    }
}

class Solution {
    func copyRandomList(_ head: Node?) -> Node? {
        guard let head = head else { return nil }
        // 使用字典来建立原节点和新节点的映射关系
        var dic = [Node: Node]()
        var cur = head
        // 1. 若链表为空，直接返回 nil，表示空链表
        // 2. 创建一个空的字典 dic，用于存储原节点和新节点的映射关系
        // 3. 第一次遍历链表 head，复制链表中的每个节点，并建立 "原节点 -> 新节点"
         的映射关系
        while cur != nil {
            // 复制当前节点，并添加到 dic 中
            dic[cur] = Node(cur.val)
            cur = cur.next
        }
        cur = head
        // 4. 第二次遍历链表 head，构建新节点的 next 和 random 指向
        while cur != nil {
            // 通过 dic 获取新节点，设置新节点的 next 和 random 指向
            dic[cur]?.next = dic[cur?.next]
            dic[cur]?.random = dic[cur?.random]
            cur = cur?.next
        }
        // 5. 返回新链表的头节点
        return dic[head]
    }
}
```