138. Copy List with Random Pointer

Solved

Medium | Topics | Companies | Hint

A linked list of length `n` is given such that each node contains an additional random pointer, which could point to any node in the list, or `null`.

Construct a **deep copy** of the list. The deep copy should consist of exactly `n` **brand new** nodes, where each new node has its value set to the value of its corresponding original node. Both the `next` and `random` pointer of the new nodes should point to new nodes in the copied list such that the pointers in the original list and copied list represent the same list state. **None of the pointers in the new list should point to nodes in the original list**.

For example, if there are two nodes `X` and `Y` in the original list, where `X.random --> Y`, then for the corresponding two nodes `x` and `y` in the copied list, `x.random --> y`.

Return *the head of the copied linked list*.

The linked list is represented in the input/output as a list of `n` nodes. Each node is represented as a pair of `[val, random_index]`

Code

```
class Solution {
    func copyRandomList(_ head: Node?) -> Node? {
        guard let root = head else {
            return nil
        }

        var hash:[Node: Node] = [:]

        var curr = head

        while(curr != nil) {
            let new = Node(curr!.val)
            hash[curr!] = new
            curr = curr?.next
        }

        curr = head
        while(curr != nil) {
            let new = hash[curr!]
            if let next = curr!.next, let ne = hash[next] {
                new!.next = ne
            } else {
                new!.next = nil
            }
```

```
            if let random = curr!.random, let r = hash[random]
{
                new!.random = r
            } else {
                new!.random = nil
            }
            curr = curr?.next
        }

        return hash[head!]
    }
}
```