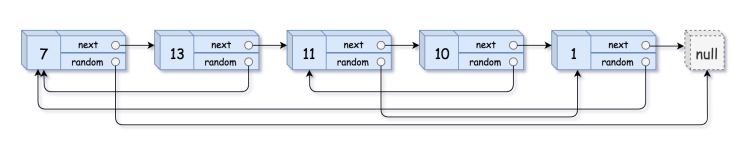
**难度: 简单**

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。

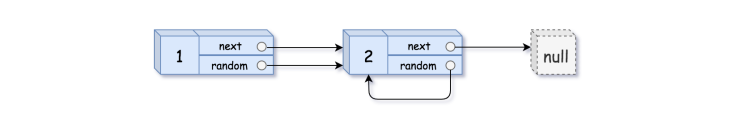
示例 1：



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]]

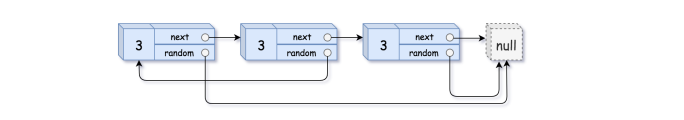
示例 2：



输入：head = [[1,1],[2,1]]

输出：[[1,1],[2,1]]

示例 3：



输入：head = [[3,null],[3,0],[3,null]]

输出：[[3,null],[3,0],[3,null]]

示例 4：

输入：head = []

输出：[]

解释：给定的链表为空（空指针），因此返回 null。

提示：

-10000 <= Node.val <= 10000

Node.random 为空（null）或指向链表中的节点。

节点数目不超过 1000 。

来源：力扣（LeetCode）

链接：https://leetcode-cn.com/problems/fu-za-lian-biao-de-fu-zhi-lcof

著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。

**思路1:**

**注:** 代码有注释

|  |
| --- |
| public Node copyRandomList(Node head) {  //判断是否为空  if (head == null) {  //空直接返回null  return null;  }  //头节点给到临时节点  Node temp = head;  //创建一个hashMap  HashMap<Node,Node> map = new HashMap<>();  //遍历  while(temp != null) {  //以temp作为key,创建val相等的Node  map.put(temp,new Node(temp.val));  temp = temp.next;  }  //重新把头节点给到临时节点  temp = head;  //遍历  while(temp != null) {  //节点指向下一个节点  map.get(temp).next = map.get(temp.next);  //给刚刚创建的Node节点的random 赋值节点  map.get(temp).random = map.get(temp.random);  temp = temp.next;  }  //返回  return map.get(head);  } |