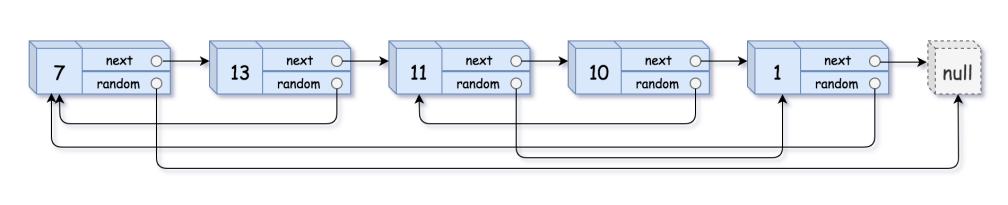
# 题目

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。

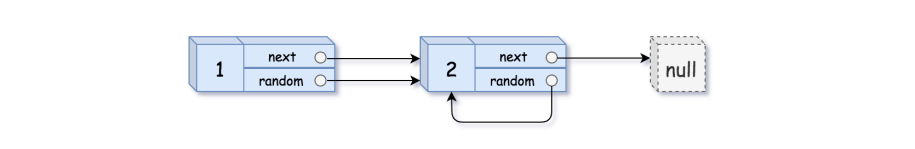
**示例 1：**



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]]

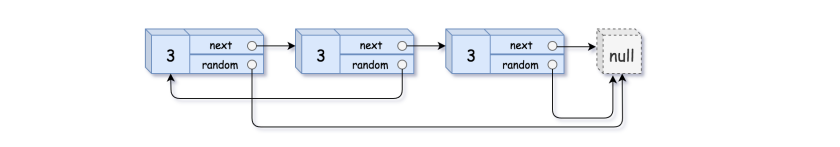
**示例 2：**



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 。

注意：本题与主站 138 题相同：

https://leetcode-cn.com/problems/copy-list-with-random-pointer/

# 分析

## 方法一：哈希表

**思路：**

复杂链表的复制涉及到深拷贝，因为除了next指针还有random指针。random指针寻找定位，将对应的关系复制不容易。

考虑到random指针把两个节点的关系联系了起来，所以思路可以开展到哈希表关联容器上，辅助实现深拷贝。

unordered\_map<node\*,node\*>mp,实现的是复制与深拷贝后一一对应的节点。

刚开始的深拷贝，将原先的节点深拷贝后的儿子放入map与其父亲一一对应

mp[t]=new Node(t->val);

next与random指针的复制对应关系！想清楚用unordered\_map的实现！

mp[t]->next=mp[t->next];

mp[t]->random=mp[t->random];

**代码：**

/\*

// Definition for a Node.

class Node {

public:

int val;

Node\* next;

Node\* random;

Node(int \_val) {

val = \_val;

next = NULL;

random = NULL;

}

};

\*/

class Solution {

public:

Node\* copyRandomList(Node\* head) {

if(head==NULL) return head;

unordered\_map<Node\*,Node\*>mp;

Node \*t=head;

while(t!=NULL){

mp[t]=new Node(t->val);//new节点存放到哈希表

t=t->next;

}

t=head;

while(t!=NULL){

if(t->next){

mp[t]->next=mp[t->next];

}

if(t->random){

mp[t]->random=mp[t->random];

}

t=t->next;

}

return mp[head];

}

};