1. Copy List with Random Pointer

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

Return a [**deep copy**](https://en.wikipedia.org/wiki/Object_copying#Deep_copy) of the 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] where:

* val: an integer representing Node.val
* random\_index: the index of the node (range from 0 to n-1) where random pointer points to, or null if it does not point to any node.

**Example 1:**



img

Input: head = [[7,null],[13,0],[11,4],[10,2],[1,0]]  
Output: [[7,null],[13,0],[11,4],[10,2],[1,0]]

**Example 2:**



img

Input: head = [[1,1],[2,1]]  
Output: [[1,1],[2,1]]

**Example 3:**



Input: head = [[3,null],[3,0],[3,null]]  
Output: [[3,null],[3,0],[3,null]]

**Example 4:**

Input: head = []  
Output: []  
Explanation: Given linked list is empty (null pointer), so return null.

**Constraints:**

* -10000 <= Node.val <= 10000
* Node.random is null or pointing to a node in the linked list.
* Number of Nodes will not exceed 1000.

**解** 同133，用一个hash表记录当前的节点是否被复制。节点访问顺序进行就可以

/\*  
// 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) {  
 unordered\_map<Node\*, Node\*>hash;  
 Node\* p = head;  
 Node\* pre = new Node(0);  
   
 while(p){  
 if(hash[p] == NULL)hash[p] = new Node(p->val);  
 if(p->random && hash[p->random] == NULL){  
 hash[p->random] = new Node(p->random->val);  
 }  
 hash[p]->random = hash[p->random];  
 pre->next = hash[p];  
 pre = hash[p];  
 p = p->next;   
 }  
 return hash[head];  
 }  
};