复制带随机指针的链表

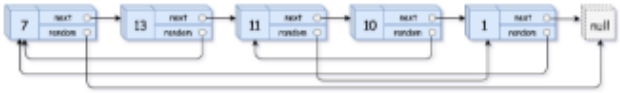
给定一个链表，每个节点包含一个额外增加的随机指针，该指针可以指向链表中的任何节点或空节点。

要求返回这个链表的 深拷贝。

我们用一个由 n 个节点组成的链表来表示输入/输出中的链表。每个节点用一个 [val, random_index] 表示：

- val：一个表示 Node.val 的整数。
- random_index：随机指针指向的节点索引（范围从 0 到 n-1）；如果不指向任何节点，则为 null 。

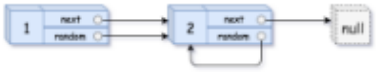
示例 1：



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。

```
}/**
 * Definition for a Node.
 * struct Node {
 *     int val;
 *     struct Node *next;
 *     struct Node *random;
 * };
 */

struct Node* copyRandomList(struct Node* head) {
    if(head==NULL)
        return NULL;
    //创建新链表进行拷贝
    //创建新链表的头
    struct Node* head1=(struct Node*)malloc(sizeof(struct Node));
    head1->val=head->val;
    head1->next=NULL;
    head1->random=NULL;

    //先拷贝主结构链表，忽略随机指针指向
    struct Node* h=head;
    struct Node* h1=head1;
    while(h->next)
    {
        h=h->next;
        struct Node* tmp=(struct Node*)malloc(sizeof(struct Node));
        tmp->val=h->val;
        tmp->next=NULL;
        tmp->random=NULL;
        h1->next=tmp;
        h1=h1->next;
    }

    //拷贝随机指针
    //头指针先归位，方便遍历
```

```
    h=head;
    h1=head1;
    while(h)
    {
        int index=0;
        struct Node * cur=head, * cur1=head1;
        if(h->random)
        {
            while(cur && cur!=h->random){
                index++;
                cur = cur->next;
            }
            while(index){
                index--;
                cur1 = cur1->next;
            }
            h1->random=cur1;
        }
        else
        {
            h1->random=NULL;
        }
        h=h->next;
        h1=h1->next;
    }
    return head1;
}
```