138. Copy List with Random Pointer

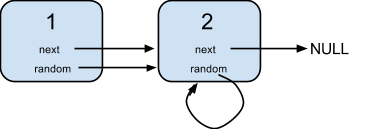
Medium

1308379FavoriteShare

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.

Example 1:



Input:  
{"$id":"1","next":{"$id":"2","next":null,"random":{"$ref":"2"},"val":2},"random":{"$ref":"2"},"val":1}  
  
Explanation:  
Node 1's value is 1, both of its next and random pointer points to Node 2.  
Node 2's value is 2, its next pointer points to null and its random pointer points to itself.

Note:

1. You must return the copy of the given head as a reference to the cloned list.

/\*

// Definition for a Node.

class Node {

public:

int val;

Node\* next;

Node\* random;

Node() {}

Node(int \_val, Node\* \_next, Node\* \_random) {

val = \_val;

next = \_next;

random = \_random;

}

};

\*/

class Solution {

public:

Node\* copyRandomList(Node\* head) {

if(head==NULL) return NULL;

//1. duplicate every node in the same list.

Node\* curr=head;

while(curr!=NULL){

Node\* next = curr->next;

Node\* fresh = new Node(curr->val,curr->next,curr->random);

curr->next = fresh;

//curr->next->next=fresh;

curr=curr->next->next;

}

//2. Modify random pointers to point to new duplicates created above.

curr=head->next;

while(curr!=NULL){

if(curr->random!=NULL) curr->random=curr->random->next;

if(curr->next==NULL) break;

curr=curr->next->next;

}

//3. separate into two linked lists

curr=head;

Node\* ret = head->next;

while(curr!=NULL){

Node\* next = curr->next;

curr->next=next->next;

if(next->next!=NULL) next->next=next->next->next;

curr=curr->next;

}

return ret;

}

};

Success

[Details](https://leetcode.com/submissions/detail/211321803/)

Runtime: 36 ms, faster than 100.00% of C++ online submissions for Copy List with Random Pointer.

Memory Usage: 21.9 MB, less than 5.21% of C++ online submissions forCopy List with Random Pointer.