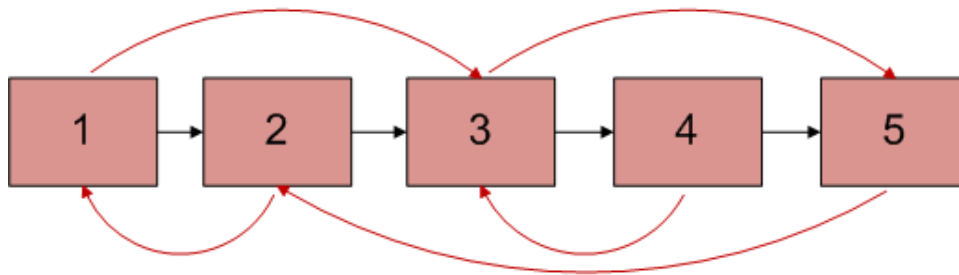


Clone a linked list with next and random pointer | Set 1

You are given a Double Link List with one pointer of each node pointing to the next node just like in a single link list. The second pointer however CAN point to any node in the list and not just the previous node. Now write a program in **$O(n)$ time** to duplicate this list. That is, write a program which will create a copy of this list.

Let us call the second pointer as arbit pointer as it can point to any arbitrary node in the linked list.



Arbitrary pointers are shown in red and next pointers in black

Figure 1

Method 1 (Uses $O(n)$ extra space)

This method stores the next and arbitrary mappings (of original list) in an array first, then modifies the original Linked List (to create copy), creates a copy. And finally restores the original list.

1) Create all nodes in copy linked list using next pointers.

3) Store the node and its next pointer mappings of original linked list.

3) Change next pointer of all nodes in original linked list to point to the corresponding node in copy linked list.

Following diagram shows status of both Linked Lists after above 3 steps. The red arrow shows arbit pointers and black arrow shows next pointers.

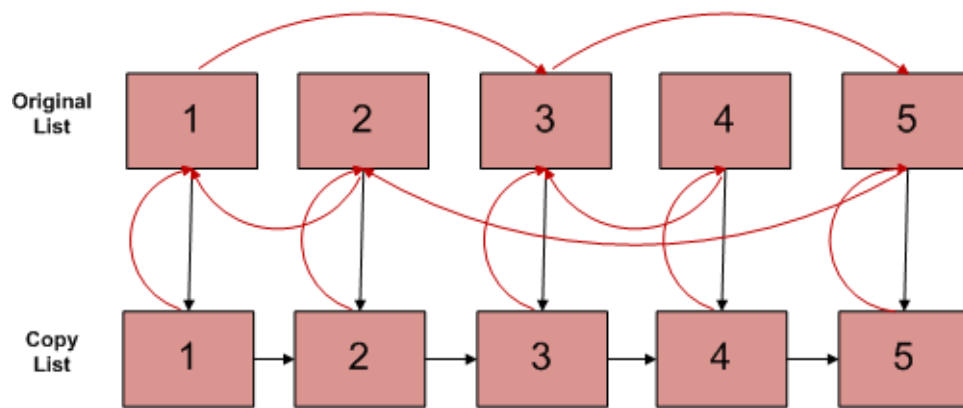


Figure 2

- 4) Change the arbit pointer of all nodes in copy linked list to point to corresponding node in original linked list.
- 5) Now construct the arbit pointer in copy linked list as below and restore the next pointer of nodes in the original linked list.

```
copy_list_node->arbit =
    copy_list_node->arbit->arbit->next;
copy_list_node = copy_list_node->next;
```

- 6) Restore the next pointers in original linked list from the stored mappings(in step 2).

Time Complexity: $O(n)$

Auxiliary Space: $O(n)$

Method 2 (Uses Constant Extra Space)

Thanks to Saravanan Mani for providing this solution. This solution works using constant space.

- 1) Create the copy of node 1 and insert it between node 1 & node 2 in original Linked List, create the copy of 2 and insert it between 2 & 3.. Continue in this fashion, add the copy of N after the Nth node
- 2) Now copy the arbitrary link in this fashion

```
original->next->arbitrary = original->arbitrary->next; /*TRAVERSE
TWO NODES*/
```

This works because `original->next` is nothing but copy of original and `Original->arbitrary->next` is nothing but copy of arbitrary.

- 3) Now restore the original and copy linked lists in this fashion in a single loop.

```
original->next = original->next->next;
copy->next = copy->next->next;
```

- 4) Make sure that last element of `original->next` is NULL.

Time Complexity: $O(n)$

Auxiliary Space: $O(1)$

Refer Following Post for Hashing based Implementation.

[Clone a linked list with next and random pointer | Set 2](#)

Asked by Varun Bhatia. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.



[131 Comments](#) Category: [Linked Lists](#)

Related Posts:

- [Merge two sorted linked lists such that merged list is in reverse order](#)
- [Compare two strings represented as linked lists](#)
- [Rearrange a given linked list in-place.](#)
- [Sort a linked list that is sorted alternating ascending and descending orders?](#)
- [Select a Random Node from a Singly Linked List](#)
- [Merge Sort for Doubly Linked List](#)
- [Point to next higher value node in a linked list with an arbitrary pointer](#)
- [Swap nodes in a linked list without swapping data](#)

([Login](#) to Rate and Mark)

3.2 Average Difficulty : **3.2/5.0**
Based on **4** vote(s)

☐
☐

Add to TODO List

Mark as DONE

Like Share 27 people like this.

Writing code in comment? Please use code.geeksforgeeks.org, generate link and share the link here.

131 Comments**GeeksforGeeks****1 Login** ▾ **Recommend** 4 **Share****Sort by Newest** ▾

Join the discussion...

**Vijayakumar G** • a month ago

Java Implementation for method 2,

[http://code.geeksforgeeks.org/...](http://code.geeksforgeeks.org/)

^ | ▾ • Reply • Share ▸

**xudifsd** • 3 months agoillustration for second method see <http://fisherlei.blogspot.sg/2...>

^ | ▾ • Reply • Share ▸

**dseth2** • 3 months agoMethod-1 implementation <https://ideone.com/TYd5o8>

^ | ▾ • Reply • Share ▸

**xtreme** • 6 months ago

easy implementation for method 2-

<http://ideone.com/pFVCB2>

^ | ▾ • Reply • Share ▸

**Vijayakumar G** → xtreme • a month ago

You must do the step3 in single loop. Not in a separate loop.

^ | ▾ • Reply • Share ▸

**Swati S** • 6 months ago

in Method 2 we are still creating n copied nodes. So the auxiliary space is again $O(n)$. Can someone explain how it is constant space?

^ | ▾ • Reply • Share ▸

**Vijayakumar G** → Swati S • a month ago

Swati,

Auxiliary space is something which we use for temporary processing of data. But in method 2 we are not using any auxiliary space. Actual space allocated is returned as cloned list. (no single space is used for temporary processing. so no

auxiliary space)

^ | v • Reply • Share ›



huyue → Swati S • 6 months ago

In Method2, the space needed for output, instead of auxiliary space ...Method1's space complexity is $O(n)$ because in step2, it "store the node and its next pointer mappings of original linked list".

^ | v • Reply • Share ›



Dhruv Gosain → Swati S • 6 months ago

here additional space will be used to save the original link pointers. As you can see in the image that the next pointers of the original list have been modified to point to the corresponding item in the copied list. When the new list will be created then the original pointers will be restored to the original pointers

^ | v • Reply • Share ›



Gaurav Chauhan → Dhruv Gosain • 5 months ago

yes but we can change both the ptrs i.e. the changed next ptr of original list and the arbit ptr of new list at same tym. so no need of storing the next ptrs to revert them back. aux space will be $O(1)$ then

^ | v • Reply • Share ›



Bhavik Patel → Swati S • 6 months ago

I didnt understand the same thingmay be it is wrong....

^ | v • Reply • Share ›



Dman • 6 months ago

Implemented all the three methods in this solution

1) using hashing- stl

2) using queues

3) Inplace solution

<https://ideone.com/IPNQOJ>

1 ^ | v • Reply • Share ›



SANKHYA • 6 months ago

The code for the second method is here:

<http://code.geeksforgeeks.org/...>

Request for GeeksforGeeks team: Please upload the code on the main page.

3 ^ | v • Reply • Share ›



Mayank Gupta → SANKHYA · 6 months ago

Nice code !!

^ | v · Reply · Share ›



Wang Zhen · 6 months ago

I understand the method 2, but didn't get why auxiliary space is $O(1)$? In my opinion, method 2 uses the same spaces as the method 1. am i right?

^ | v · Reply · Share ›



sunny rawani → Wang Zhen · 6 months ago

when you will reset original list next pointer then you need $O(n)$ see step 1 and 6 of method 1. while there isn't a need in method 2.

^ | v · Reply · Share ›



gdb · 6 months ago

Why method 1 is storing nodes in map. `copy_list_node->arbit` can be used to restore original list on the fly. Please see the following modified pseudo code.

```
copy_list_node->arbit =  
copy_list_node->arbit->arbit->next;  
copy_list_node = copy_list_node->next;  
orig_list_node->next = copy_list_node->arbit;  
orig_list_node = orig_list_node->next ;
```

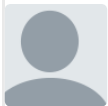
^ | v · Reply · Share ›



pravir ahuja · 7 months ago

can anyone tell how in method1 the node & its next pointer mapping of original linked list are stored ?

^ | v · Reply · Share ›



shubham garg · 7 months ago

I am not able to get why we need to store the mapping of original linked list and why we are changing the next pointer of original linked list to point to copied linked list. Why can't we do it like this...

using the next pointer of original linked list we create our copied list but all elements of copied list their arbitrary pointer will point to original list initially and now we will traverse the copied list and using its arbitrary pointer we can get value of its original arbitrary pointer as follows :

```
copy_node->arbitrary = copy_node->arbitrary->arbitrary;
```

So there is no use of extra space and plus this is done in $O(n)$ time also.

^ | v · Reply · Share ›



K V D Pratap → shubham garg · 7 months ago



copy_node->arbitrary= copy_node->arbitrary->arbitrary

This line gives the node in the original list. But we need to make arbit pointer in copy list to point a node in copy list, not the node in original list. Hope, the need of having copy_node->arbitrary= copy_node->arbitrary->arbitrary->next is clear.

^ | v • Reply • Share ›



Bewkoof_coder • 7 months ago

code to clone a linked list with time complexity O(n).

```
struct node *clone(struct node *start){
    struct node *t;
    if(start==NULL)
        return NULL;
    else{
        t=clone(start->next);
        t=insert(t,start->data);
        return t;
    }
}
```

^ | v • Reply • Share ›



Mr. Lazy → Bewkoof_coder • 7 months ago

I think you read the question with one eye closed! ;-[
what about the arbitrary pointer? ..

5 ^ | v • Reply • Share ›



numid • 7 months ago

In the first method we can do it in constant space by manipulating the original array in the following way in step 5 :

while making changes to copy list we can restore the original list as

```
if(cur->next->next)
    cur->next=cur->next->next->arbit;
else
    cur->next= NULL;
```

^ | v • Reply • Share ›



LondonMontmorency • 7 months ago



Unless statement of the problem explicitly allows modifying the original list, any solution that requires such modification is considered incorrect. The second approach is flawed in that regard. For example, the solution is not reentrant, which is a serious problem.

2 ^ | v • Reply • Share ›



acodebreaker → LondonMontmorency • 7 months ago

agree

^ | v • Reply • Share ›



coder123 • 8 months ago

the program is the implementation of method 2 but i cant understand teh mistake in it. It is not executing:/ Help!

```
#include<iostream>
```

```
using namespace std;
```

```
class list
```

```
{
```

```
public:
```

```
int data;
```

```
list *next;
```

```
list *start;
```

```
list *random;
```

see more

^ | v • Reply • Share ›



codelearner • 8 months ago

code for method 2 -> <https://ideone.com/zCOLDi>

and plzz can anyone tell me how to reconstruct original list in method 1 ???

^ | v • Reply • Share ›



Chauhan • 8 months ago

Working code for Method 1 - <http://ideone.com/XGFFtn>

^ | v • Reply • Share ›



Mr. Lazy • 8 months ago

Simple and clean implementation of method 2.

<http://ideone.com/m7DPje>

^ | v • Reply • Share ›



Abhilash Pareek • 9 months ago

What is wrong in this code.

```
copy_list_node->arbit =originals_list_node->arbit;
```

```
copy_list_node = copy_list_node->next;
```

```
original_list_node=original_list_node->next;
```

^ | v • Reply • Share ›



Chauhan → Abhilash Pareek • 8 months ago

You have not defined the relation between copy_list_node and original_list_node correctly. What you are trying to do is making the arbitrary pointer in the copy list point to the arbitrary node of the original list, while what you actually have to do is make the nodes of the copy list point to the arbitrary nodes in its own list. I hope you get the point now.

A working code - <http://ideone.com/XGFFtn>

1 ^ | v • Reply • Share ›



Ashish Maheshwari • 10 months ago

Here is the code for the second method in c++: <https://github.com/maheshwari-...>

^ | v • Reply • Share ›



Kshitij Bhutani • a year ago

How are we saying that second method uses constant space while first method is implemented using $O(n)$ extra space. Please explain

^ | v • Reply • Share ›



Ashish Maheshwari → Kshitij Bhutani • 10 months ago

What constant extra space means is that our function uses a constant amount of extra space at any given time which is independent of the input 'n' (this n maybe the no of elements in a list or the size of an array or anything else).

While $O(n)$ extra space means the extra space increases/decreases with input 'n' (again, n can be in any unit).

^ | v • Reply • Share ›




acodebreaker → Ashish Maheshwari • 7 months ago

in method2 , space will increase as size of linked list will increase both are $O(n)$ in space but differs in auxillary space

^ | v • Reply • Share ›



me → Kshitij Bhutani • 10 months ago

 Space used in programs can roughly be classified as: Input, Auxiliary, Output. The Space complexity is usually the Auxiliary space used. Method 1 has Input (LinkedList), Auxiliary (HashMap) and Output (Cloned LinkedList), hence $O(N)$. Method 2 has Input (LinkedList) and Output (LinkedList). It doesn't use Auxiliary space, so it is $O(1)$.

^ | v • Reply • Share ›



neer1304 • a year ago

Implementation Using 2nd Method

<http://ideone.com/3oPn0T>

^ | v • Reply • Share ›



nitin • a year ago

Method 2 Implementation -

<http://ideone.com/rubFSV>

^ | v • Reply • Share ›



Kartik Bhatia • a year ago

Implementation of method 2 in c++:

<http://ideone.com/TaQ4of>

^ | v • Reply • Share ›



Kerem Sahin • a year ago

Method 2 Implementation

<http://ideone.com/uPifkM>

^ | v • Reply • Share ›



Ishan • a year ago

My solution to second method:

```
/**
 * Definition for singly-linked list with a random pointer.
 * struct RandomListNode {
 *   int label;
 *   RandomListNode *next, *random;
 *   RandomListNode(int x) : label(x), next(NULL), random(NULL) {}
 * };
 */
```

```
RandomListNode *copyRandomList(RandomListNode *head) {
    if(!head) {
```

```

return NULL;
}

/* Insert new list nodes in between the original linked list */
RandomListNode *temp = head;
while(temp!=NULL) {

```

[see more](#)

^ | v • Reply • Share ›



codecrecker • a year ago

```

#include<stdio.h>
#include<stdlib.h>
typedef struct nd node;
struct nd{
int d;
node *n;
node *r;
};
node *head,*ptr;
node *createNode(int d)
{
node *tmp;
tmp=(node *)malloc(sizeof(node));
tmp->n=NULL;
tmp->r=NULL;
if(!tmp)
{
printf("Memory error");

```

[see more](#)

^ | v • Reply • Share ›



Kartik Bhat • a year ago

can we use hash table for this?

1 ^ | v • Reply • Share ›



Aditya Goel → Kartik Bhat • a year ago

I also solved it using Hash Table first. I stored the old address (Original linked list) and the new address(copied linked list) as key-value pair for each node in HashTable. Then I traversed the original Linked List and for each node n

```

valueof(n)->arbit=valueof(n->arbit);

```

where valueof(key) will return the corresponding value in HashTable

that will set the arbit pointers in new linked list.

^ | v • Reply • Share ›



codecrecker → Kartik Bhat • a year ago

Sure u can use hashtable for this problem, just store new nodes in hash table:-
Loop1: copy to new linked list keep random pointer as NULL and store in hash table

Loop2: Now set random pointer according to hash

Done

^ | v • Reply • Share ›



Rohit Raj • a year ago

In Method 1.

Instead of doing step 2: Store the node and its next pointer mappings of original linked list. wouldn't this work.

While assigning arbit pointers to the copied list. simultaneously correct the original list using this:

originalList->next = originalList->next->next->arbit

?

1 ^ | v • Reply • Share ›



Ethan → Rohit Raj • a year ago

It's not gonna work if one original node points back to another as random pointer.

^ | v • Reply • Share ›



Rohit Raj → Ethan • a year ago

Check in the given example for node 4.

Can you give an example?

^ | v • Reply • Share ›



pol • a year ago

plz help me..

method 2, how space complexity is $O(1)$?? we are creating duplicate of each node after that node. so for n node n duplicate nodes. so its $O(n)$.

4 ^ | v • Reply • Share ›



spiderweb → pol • a year ago

yes even i doubt the same thing . In both the approaches n number of nodes have to be created.

1 ^ | v • Reply • Share ›

Load more comments

@geeksforgeeks, Some rights reserved

Contact Us!

About Us!

Advertise with us!