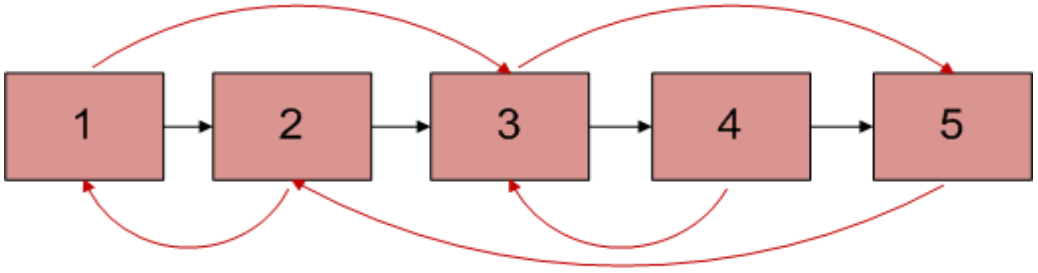
Copy a linked list with next and arbit pointer

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.

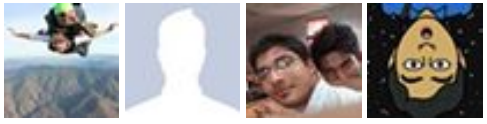


Arbitrary pointers are shown in red and next pointers in black

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.

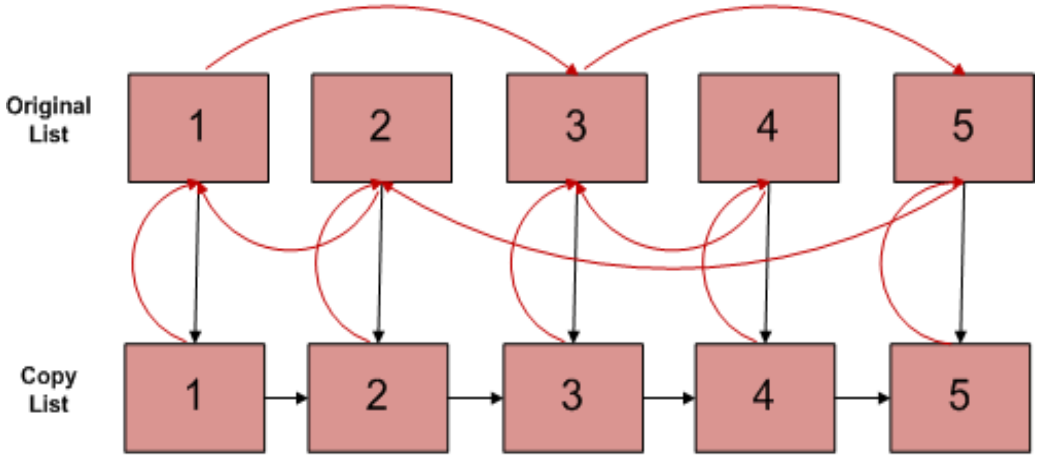


Figure 2

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

Geometric Algorithms

Popular Posts

All permutations of a given string
Memory Layout of C Programs
Understanding “extern” keyword in C
Median of two sorted arrays
Tree traversal without recursion and without stack!
Structure Member Alignment, Padding and Data Packing
Intersection point of two Linked Lists
Lowest Common Ancestor in a BST.
Check if a binary tree is BST or not
Sorted Linked List to Balanced BST

the copy of 2 and insert it between 2 & 3.. Continue in this fashion, add the copy of N afte 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)

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

705 Subscribe

Related Tpoics:

- Given a linked list, reverse alternate nodes and append at the end
- Pairwise swap elements of a given linked list by changing links
- Self Organizing List | Set 1 (Introduction)
- Merge a linked list into another linked list at alternate positions
- QuickSort on Singly Linked List
- Delete N nodes after M nodes of a linked list
- Design a stack with operations on middle element
- Swap Kth node from beginning with Kth node from end in a Linked List

15 Tweet 0 13

Writing code in comment? Please use **ideone.com** and share the link here.

58 Comments GeeksforGeeks

Sort by Newest

Join the discussion…

xgme • 5 days ago

When I first encountered this question I couldn't find the solution because I ass supposed to be in all software systems. I'm glad I failed to answer because of

• Reply • Share ›

pulkit mehra • 2 months ago

If the List has Read Only access then the above discussed methods will fail.

Recent Comments

Abhi You live US or India?

Google (Mountain View) interview · 49 minutes ago

Aman Hi, Why arent we checking for conditions...

Write a C program to Delete a Tree. · 1 hour ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 1 hour ago

Sanjay Agarwal bool tree::Root_to_leaf_path_given_sum(tree...

Root to leaf path sum equal to a given number · 1 hour ago

GOPI GOPINATH @admin Highlight this sentence "We can easily...

Count trailing zeroes in factorial of a number · 1 hour ago

newCoder3006 If the array contains negative numbers also. We...

Find subarray with given sum · 2 hours ago

key and a value associated with that key.

Steps: (The code syntax isn't coming properly in comments so check online)

1. Copy the original list to new list with next pointers intact, leave arbit pointers

2. While copying make a map

```
map <node*, node*=""> map_hash
```

In our case map will have the key as the original list node and value as copy lis

3 Map would be somewhat like this

Original --> Copy
Node1 --> Node1(copy)
Node2 --> Node2(copy)

see more

3 • Reply • Share ›

darkpassenger • 4 months ago

if we don't want to modify the original list the with the help of 2 hash maps u ca

2 • Reply • Share ›

neelabhsingh • 6 months ago

This problem can be done in two scan Linked List

1st Scan: In first scan change change next pointer of original LL to the corresp
change random pointer of new LL to the corresponding same node of the origi

current1 is pointing to the start node of the originalLL.

current2 is pointing to the start node of the newLL

current1=root1;// root node of the original node

current2=roott2// root node of the original node

while(current1!=NULL)

{

temp=current1->next;

current1->next=current2;

see more

5 ^ | v • Reply • Share ›

pavansrinivas • 6 months ago

What if the Original list is read only???

^ | v • Reply • Share ›

Prateek Rathore • 7 months ago

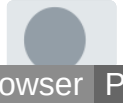
Saravanan Mani , nice trick !!!

10 ^ | v • Reply • Share ›

Neha Garg • 8 months ago

@GeeksforGeeks where do you need to insert copy element in original list ??
we can do it just first copy the whole list using next pointers ... then copy the a
list using using a loop till end and picking one element of each LL ??
this will help in reduing two step first insert in between and second to separate
correct me if i m wrong.....

^ | v • Reply • Share ›



Neha Garg • 8 months ago

@Saravanan Mani @GeeksforGeeks where do you need to insert copy eleme
we can do it just first copy the whole list using next pointers ... then copy the a
list using using a loop till end and picking one element of each LL ??
this will help in reduing two step first insert in between and second to separate
correct me if i m wrong.....

ankit • 8 months ago

Recursive method(using hashmap):
where hashmap contains (address of list 1 node,corresponding list 2 node's a

```
node* copy(node* n1)
{
if(n is null)
return null;

if(if n1 is in hasmap)

return n2;

else

create new node n2;
insert into hashMap(n1,n2);

n2->data=n1->data;
n2->next=copy(n1->next);
n2->arbitrary=copy(n1->arbitrary);

return n2;
}
```

Can anybody pls verify this method.

@meya ankit • 11 days ago

Cool solution for cases when input list is read only.

Here is the working solution for this approach.

http://ideone.com/oqJ429

• Reply • Share ›

Pavan • 10 months ago

```
 // Method 2 code ,check if there is any bug
#include <stdio.h>
#include <stdlib.h>
struct node {
    int data;
    struct node* arbit;
    struct node* next;
};
struct node *newNode(int data)
{
    struct node *new_node = (struct node *) malloc(sizeof(struct node
    new_node->data = data;
    new_node->next = NULL;
    new_node->arbit = NULL;
    return new_node;
}
void printList(struct node *node)
{
```

see more

2 • Reply • Share ›

Gaurav Garg Pavan • 10 months ago

No bug pandu ;)
code working fine !
keep posting.

```
/* Paste your code here (You may delete these lines if not wri
```

• Reply • Share ›

Karshit • 10 months ago

My Code.. with a built in function to test the copied and original Linked List.. Ho

```
#include <iostream>

using namespace std;

struct node {
    int data;
    node *next;
    node *arbit;
};

node *create(int n)
{
    if (n == 0)
        return NULL;

    node *head = new node();
```

see more

• Reply • Share ›

abhishek08aug • a year ago

Intelligent :D

• Reply • Share ›

Vijay Muvva • a year ago

Hello,

I don't see any need for the auxiliary space in the first method.

Why can't we restore the original list while adding arbitrary links to the new list
temp = newListNode.next.arbitrary;
newListNode.arbitrary = originalListNode.arbitrary.next;
originalListNode.next = temp;

Do this for every node.

Note: initially new list arbitrary links points to the original list's corresponding nc

Please correct if I'm wrong..

• Reply • Share ›

anshul.chauhan Vijay Muvva • 10 months ago

yes you are wrong...try it with an example...you can't point any node to
done because node can point to previous nodes as well and u've alrea

```
/* Paste your code here (You may delete these lines if not wri
```

1 • Reply • Share ›

neham • a year ago

there is more simpler version of method 1.
mahajanneha.blogspot.com/2013/...

this approach also takes O(n) auxiliary space.

• Reply • Share ›

whizkid08 • a year ago

Code for 1st method:

```
 #include <stdio.h>

struct Node
{
        int data;
        struct Node *arbit;
        struct Node *next;
};

typedef struct Node Node;
void push(Node **head, int val)
{
        Node* temp = (Node*) malloc(sizeof(Node));
        temp->data = val;

        if(*head == 0)
```

see more

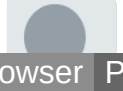
▲ | ▼ • Reply • Share ›

whizkid08 ➦ whizkid08 • a year ago

Oops. Didn't restore original list. Any idea on how to achieve:
(2)Store the node and its next pointer mappings of original linked list.

It can be easily done by having a 3rd list. Any efficient way in C?

▲ | ▼ • Reply • Share ›



Aishwarya • a year ago

```
#include<iostream>
#include<vector>
using namespace std;

struct node
{
   int data;
   node *next, *random;
   node(int _data):data(_data),next(NULL),random(NULL){}
};

node *addEnd(node **head, int data)
{
   while(*head)head=&(*head)->next;
   *head=new node(data);
   return *head;
}
```

see more

• Reply • Share ›

rahulcynosure ➦ Aishwarya • a year ago

Your code has a small bug. Check the output .
original list before copying
0 1 2 3 4 5
1 3 4 5 2 5

original list after copying
0 1 2 3 4 5
1 3 4 5 2 5

copied list

0 1 2 3 4 5
1 3 4 5 3 5

in the copied list random pointer , node with value 4 is pointing to node node with value 4 is pointing to node with value 2.

Here's the corrected code.

//this function returns the head of the copied list.

see more

• Reply • Share ›

sushdragon • a year ago

there is no need of extra space. Just change copy_node->arbitrary->next=cop

• Reply • Share ›

user123 • 2 years ago

how is the 1st solution done in O(n) and 2nd one in O(1)?i can't understand...s

```
/* Paste your code here (You may delete these lines if not writing c
```

• Reply • Share ›

user123 user123 • 2 years ago

with respect to space i'm asking..

• Reply • Share ›

GeeksforGeeks • 2 years ago

@WgpShashank:
1. if wen can't modify the original linked list e.g. its read only
If we can't modify the list then we can create a copy in more than O(n) time cc

2. if elements are duplicate or all the elements are same.
There should not be any problem when there are duplicates as both of the app
data.

@Ruonan Zhao, @Ishan and @Elijah:
Thanks for pointing this out. We have renmoved method 1 from the original po

WgpShashank • 2 years ago

@All Some Issue That I Can See with each approaches are .

1. if wen can't modify the original linked list e.g. its read only

2. if elements are duplicate or all the elements are same.

in these case we have to think some other algorithm ? isn't it
?

```
/* Paste your code here (You may delete these lines if not writing c
```


Sudha Malpeddi • 3 years ago

```
/* Paste your code here (You may delete these lines if not writing c
void copy(struct node *p, struct node **q)
{
        if(p!=NULL)
        {
                *q=(struct node*)malloc(sizeof(struct node));
                (*q)->num=p->num;
                (*q)->next=NULL;
                copy(p->next, &((*q)->next));
```

```
        }
    }
```

 • Reply • Share ›

levis ➚ Sudha Malpeddi • 3 years ago

This doesnt work ...In the last step of recursion you are trying to point t
should to point to the new list ...

 • Reply • Share ›

Ruonan Zhao • 3 years ago

Hi, I think the restriction assumption is wrong for the first solution. The restricti
by only one arbit pointer in a linked list. Even if we obey this restriction, there s
restore the original list.

For example?
In this case
original_Node1->arbit=original_Node3
original_Node2->arbit=original_Node1
original_Node3->arbit=original_Node2

the original_node2-next cannot be restore by original_node2->next = original_r
original_node2->next->next is copy_node1 and it's arbit has being change to c

So the restriction assumption should be:
The node's arbit will point to the node which is after that node.

 • Reply • Share ›

Ishan ➚ Ruonan Zhao • 3 years ago

Yes, I also came across the same thing, the 1st approach only works i
which is after the current node in the list.

points to its corresponding node in the duplicated list, thereby producin

Reply • Share ›

Elijah Ishan • 2 years ago

A very valid point. Even I figured out the same case.

```
/* Paste your code here (You may delete these lines if
```

Reply • Share ›

Raja • 3 years ago

First figure doesn't have the restriction mentioned. A node should be pointed b
3's arbit and 4's arbit are pointing to node 5.

Please correct me if i'm wrong.

--
Raja

Reply • Share ›

GeeksforGeeks Raja • 3 years ago

@Raja: Thanks for pointing this out. We shall upadate the post with m

Reply • Share ›

wgpshashank • 3 years ago

```
struct node
{
  struct node *next;
  struct node *random;
  int val;
```

```
};
void push(struct node **head_ref, int new_data)
{
     struct node *new_node =
              (struct node *)malloc(sizeof(struct node));
     new_node->val= new_data;
     new_node->next = *head_ref;
     *head_ref = new_node;
}
struct node* copy_list(struct node *root)
{
```

see more

• Reply • Share ›

disappearedng • 3 years ago

I am writing this to practice,

How can you write a function, deep_equal, that:
1) Traverse all the nodes at least once and check that the corresponding node value
2) Terminates?

My approach has been the following:
Given n1, n2:
1) Call on n1.next and n2.next (not n1 and n2) recursively
2) if you are at n1 and n2, return True

This does NOT guarantee to stop because you can simply construct a list whe pointer points at 2 and 3 respectively and then this recursion will never stop.

How do I go about it?

• Reply • Share ›

disappearedng disappearedng • 3 years ago

Note the method of doing the following:

I came up with a two pass algorithm which involves:
1st pass: check both l1 and l2 by traversing next
2nd pass: check both l1 and l2's arbit pointers pointing to nodes with th
the list via next (NOT via the arbit pointers to avoid cycles).

This does **not** work.

Look at the scenario below:

l1: 1->2->3->2
there is only 1 arbit pointer and it's from 3 to the first 2

l2: 1->2->3->2
there is only 1 arbit pointer and it's from 3 to the SECOND 2.

This algorithm will fail.

Anyone able to help?

^ | v • Reply • Share ›

wgpshashank • 3 years ago

```
 copy_linked_lists(struct node *q, struct node **s)
{
   if(q!=NULL) {
     *s=malloc(sizeof(struct node));
     (*s)->data=q->data;
     (*s)->link=NULL;
     copy_linked_list(q->link, &((*s)->link));
   }
}
```

Reply • Share ›

sri • 3 years ago

The first algo does not make sense. If you change all arbit pointers first and th
cannot.. as the arbit have been modified and vice versa.

Even if you modify arbit pointer of a node and then restore the next pointer of c
also the last node's back pointer shows problem. The next has been modified

Pls explain if i am missing something!!

Reply • Share ›

guest ➚ sri • 3 years ago

i don' tthink u read the algo carefully... it first modifies the arbit pointer c
pointer of corresponding node in orig list.
and for the last node how can the next point to 3??
for last node :
orig->next=orig->next->next->arbit;
but for last node : orig->next->next==NULL so we can set the next of la
this condition....

Reply • Share ›

kevalvora • 4 years ago

About the 1st algorithm, either I have not understood it or it is incomplete and c

In the example given, after 1st iteration, arbit of 1 in copy-list will point to 3 in c
to 2 in original-list. In 2nd iteration, arbit of 2 in original-list is pointing to 1 in orig
pointing back to 2 in original-list (due to the change in 1st iteration). So followin

> copy_list_node->arbit = copy_list_node->arbit->arbit->next

I might not have understood the algorithm, in that case, please explain where I

kevalvora kevalvora • 4 years ago

Now I have understood the algorithm and yes, it works properly.

In 1st loop we change the arbit nodes of copy-list and in a separate loo
original-list.

A wonderful algorithm :)

Stiju kevalvora • 3 years ago

1st method doubt :
1st loop we change the arbit nodes of copy-list
---
it will work fine

separate loop we change the next of nodes in original-list.

"orig_list_node->next =
orig_list_node->next->next->arbit"

since orig_list_node->next links to sibling node in copy list, now
copy_list_node->arbit which we changed in 1st loop.
wouldn't orig_list_node->next be replaced with arbit's of corresp
points to its actual arbit than its sibling on orig_list

Parthsarthi • 4 years ago

the first method is incorrect.I think we will have to store the next pointers in firs

After reading the question i tried solving it myself...
and came up with the method list below, then i looked into the three solution m
most.
it's really beautiful.

My solution is worst considering the space complexity,
but does not modifies the original list while on other hand all three solutions lis

Solution 4:

1) Traverse the list 1 and create list 2 (new copy) , just care about next link ptr
forget the previous link ptr completely.

While doing this create/update two hash tables

hash table 1

| list1ptr1 | list2ptr1|
| list1ptr2 | list2ptr |

see more

• Reply • Share ›

Sri1 Arif Ali Saiyed • a year ago

Algo:
This solution can be made much simpler with just one Hash Table. (Ke
(node_list1, node_list2). The hash table takes a list1 node as key and r

First iteration through original list1:
// Create new link list node
// After creating each node, add old and new nodes to HashMap
// If previous node is present, assign the next node for new list.

Second iteration through both list1 and list 2:
node_list2->arbit = HashMap(node->arbit);
//Move current nodes to node_list2->next; node_list1->next

Done! No need for array, index adjustment etc.

Sri1 ➚ Sri1 • a year ago

Complexity = O(n)
Storage is 2n storage of memory pointers, so still O(n)

Of course, if this is a very large data structure and it is a memo
Hashmap implementation, collisions, chaining etc. [if retrieval is
play.

Raja ➚ Arif Ali Saiyed • 3 years ago

do u have code for this?

Sambasiva • 4 years ago

```
#include <stdio.h>
#include <stdlib.h>

struct Node
{
        int data;
        struct Node *next;
        struct Node *arb;
};
```

```
typedef struct Node Node;
typedef Node* list;

Node* dupNode(Node *p)
{
        Node *temp = malloc(sizeof(Node));
```

see more

• Reply • Share ›

WgpShashank ➔ Sambasiva • 2 years ago

this method should be ?

Node* dupNode(Node *p)
{
Node *temp = malloc(sizeof(Node));

memcpy(temp, p, sizeof(Node));

temp->next=p->next;
p->next=temp;
return temp;

}

```
/* Paste your code here (You may delete these lines if not wri
```

• Reply • Share ›

WgpShashank ➔ WgpShashank • 2 years ago

@sambasiva

also it should be

p->next->arb = p->arb->next;

?