Identical Linked Lists

Two Linked Lists are identical when they have same data and arrangement of data is also same. For example Linked lists a (1->2->3) and b(1->2->3) are identical. . Write a function to check if the given two linked lists are identical.

Method 1 (Iterative)
To identify if two lists are identical, we need to traverse both lists simultaneously, and while traversing we need to compare data.

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

```
/* Structure for a linked list node */
struct node
{
  int data;
  struct node *next;
};

/* returns 1 if linked lists a and b are identical, otherwise 0 */
bool areIdentical(struct node *a, struct node *b)
{
  while(1)
  {
    /* base case */
    if(a == NULL && b == NULL)
    {  return 1; }
    if(a == NULL && b != NULL)
    {  return 0; }
    if(a != NULL && b == NULL)
    {  return 0; }
    if(a->data != b->data)
    {  return 0; }

    /* If we reach here, then a and b are not NULL and their
       data is same, so move to next nodes in both lists */
    a = a->next;
    b = b->next;
  }
}

/* UTILITY FUNCTIONS TO TEST fun1() and fun2() */
/* Given a reference (pointer to pointer) to the head
  of a list and an int, push a new node on the front
  of the list. */
void push(struct node** head_ref, int new_data)
{
  /* allocate node */
  struct node* new_node =
          (struct node*) malloc(sizeof(struct node));

  /* put in the data  */
  new_node->data  = new_data;

  /* link the old list off the new node */
  new_node->next = (*head_ref);

  /* move the head to point to the new node */
  (*head_ref)    = new_node;
}

/* Druver program to test above function */
int main()
{
  struct node *a = NULL;
  struct node *b = NULL;

  /* The constructed linked lists are :
   a: 3->2->1
   b: 3->2->1 */
  push(&a, 1);
  push(&a, 2);
  push(&a, 3);

  push(&b, 1);
  push(&b, 2);
  push(&b, 3);

  if(areIdentical(a, b) == 1)
    printf(" Linked Lists are identical ");
  else
    printf(" Linked Lists are not identical ");

  getchar();
  return 0;
}
```

Method 2 (Recursive)
Recursive solution code is much cleaner than the iterative code. You probably wouldn't want to use the recursive version for production code however, because it will use stack space which is proportional to the length of the lists

```
bool areIdentical(struct node *a, struct node *b)
{
  if (a == NULL && b == NULL)
  {  return 1;  }
  if (a == NULL && b != NULL)
  {  return 0;  }
  if (a != NULL && b == NULL)
  {  return 0;  }
  if (a->data != b->data)
  {  return 0;  }

  /* If we reach here, then a and b are not NULL and their
       data is same, so move to next nodes in both lists */
  return areIdentical(a->next, b->next);
}
```

Time Complexity: O(n) for both iterative and recursive versions. n is the length of the smaller list among a and b.

Please write comments if you find the above codes/algorithms incorrect, or find better ways to solve the same problem.

Related Topics:

- Clone a linked list with next and random pointer | Set 2
- Given a linked list of line segments, remove middle points
- Construct a Maximum Sum Linked List out of two Sorted Linked Lists having some Common nodes
- 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

Like 8 Tweet 0 +1 0

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

28 Comments GeeksforGeeks Login

Recommend Share Sort by Newest

Join the discussion…

Prabhuraj Kanche • 3 days ago

```
bool areIdentical(struct node *a, struct node *b)

{

if (a == NULL && b == NULL)

return true;

if (a == NULL || b == NULL)

return false;

if (a->data != b->data)

return false;

return (a->next, b->next);

}
```

• Reply • Share ›

Ajitesh Mandal • 5 days ago

correct me if i am wrong

http://ideone.com/Uyj6PF

• Reply • Share ›

Ajitesh Mandal • 5 days ago

comment if i am wrong

http://ideone.com/uc3ybX

• Reply • Share ›

Guest • 8 months ago

Single line recursive solution:

```
bool identical(struct node *a,struct node *b)
{
return ((a==NULL && b==NULL)||((a->data==b->data)&&identical(a->next,b->next)));
}
```

15 • Reply • Share ›

Saurabh → Guest • 5 months ago

Great!!

• Reply • Share ›

Ravi → Guest • 6 months ago

Great Solution :)
I just want to ask that is this code can consider as a tail recursion or not ?

• Reply • Share ›

Sumit Kesarwani → Guest • 6 months ago

greate Coding Style...

• Reply • Share ›

Ravi → Sumit Kesarwani • 6 months ago

Great Solution
I just want to ask that is this code can consider as a tail recursion or not ?

• Reply • Share ›

Sumit Kesarwani → Ravi • 6 months ago

yes u can do...

• Reply • Share ›

Kim Jong-il → Guest • 7 months ago

Ohhh great one, It has given me a good lesson. Thanks.:)

2 • Reply • Share ›

Vinayak • 9 months ago

HI. I have created this simple method, and it works for all the cases.
can it be optimized, and can the use of exit be avoided?
thanks
Here's the ideone link-http://ideone.com/EJwNa6
temp1 and temp2 point to start of the respective lists

```
void identical (struct node*temp1, struct node*temp2){

while(temp1!=NULL || temp2!=NULL)
{
if(temp1->info!=temp2->info)
{
cout<<"The linked lists are not identical. "<<endl<<"mismatch found;="" cout<<endl;=""
exit(1);="" }="" temp1="temp1-">link;
temp2=temp2->link;
if(temp2==NULL ||temp1==NULL)break;
}
if(temp1==NULL &&temp2==NULL)//traversed both lists completely
cout<<" Lists are equal"<<endl; else="" cout<<"the="" linked="" lists="" are="" not=""
identical.="" "<<endl<<"difference="" in="" lists="" length"<<endl;="" }="">
```

• Reply • Share ›

Guest • 9 months ago

Hi. I have created this very simple method. This works for all of the cases, but is there a way in this code can be optimised.Also can the exit() function be replaced somehow here.Thanks

```
void identical (struct node*temp1, struct node*temp2){

while(temp1!=NULL || temp2!=NULL){
if(temp1->info!=temp2->info)
{
cout<<"The linked lists are not identical. "<<endl<<"mismatch found="" at="" "<<temp1-
="">info<<" and "<<temp2->info<<endl; exit(1);="" }="" temp1="temp1-">link;
temp2=temp2->link;
if(temp2==NULL ||temp1==NULL)break;
}
if(temp1==NULL &&temp2==NULL)//traversed both lists completely
cout<<"Vola! Lists are equal"<<endl; else="" cout<<"the="" linked="" lists="" are="" not=""
identical.="" "<<endl<<"difference="" in="" lists="" length"<<endl;="" }="">
```

• Reply • Share ›

Gaurav Nara • 9 months ago

We don't need 4 if conditions..

```
if(a==NULL&&b==NULL)
return 1;
if(a==NULL||b==NULL)
```

```
return 0;
if(a->data==b->data)
{
a=a->next;
b=b->next;
}
```

2 • Reply • Share ›

Himanshu Dagar • a year ago

can refer to below code for recursion method

http://ideone.com/Bx7UEb

• Reply • Share ›

neelabhsingh • a year ago

what is problem in this method

```
bool areIdentical(struct node *)
{
while(1)
{
if((a==NULL)&&(b==NULL))
return 1;
if(a->data==b->data)
{
a=a->next;
b=b->next;

}
else
return 0;
}
}
```

• Reply • Share ›

mahesh neelabhsingh • a year ago

@neelabhsingh consider two list viz. L1=1->2->NULL and L2=1->2->3->NULL. In your code for input L1 and L2 will give segmentation fault. Due to this condition (a->data==b->data) for a=NULL or b=NULL will raise error. Your code works for equal list only.

1 • Reply • Share ›

neelabhsingh mahesh • a year ago

thanks for explanation.

• Reply • Share ›

Sumit Gaur • 2 years ago

bool idendical (node *x, node *y).
{

if(x==NULL&&y==NULL)
return true;
return ((x->data==y->data)&&identical(x->next, y->next));
}

• Reply • Share ›

Saurav Sahu Sumit Gaur • a year ago

That will cause Segmentation fault if both lists are not of equal length.

2 • Reply • Share ›

Deepak Singh • 2 years ago

thanks for such a beautiful explanation. now concept of linked list isn&#039t tough anymore.

• Reply • Share ›

cyberlynxs • 3 years ago

In method 2(recursive soln), the function is tail-recursive. If the compilers implements tail-recursion(most of the latest compilers support it), a single stack-frame will be used. So, I think recursive soln can also be used safely. Can you please confirm it?

• Reply • Share ›

Ashish • 5 years ago

check if a and b are pointing to same node, then the lists would be same by default

the case can also be a Y shaped two LL. so the moment address matches, feel safe to declare them as same

• Reply • Share ›

Sambasiva • 5 years ago

```
 int areIdentical(list a, list b)
{
    for(; a && b && a->data == b->data; a = a->next, b = b->next);
    return !(a || b);
}
```

5 • Reply • Share ›

pradeep kumar Sambasiva • 9 months ago

could you please give me a brief about how its working.....
?

.

• Reply • Share ›

abhikumar18 Sambasiva • 2 years ago

awesome yar...

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

• Reply • Share ›

piyush Sambasiva • 3 years ago

GREAT..........

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

• Reply • Share ›

kapil • 5 years ago

How about comparing two linked list having same set of elements and same number of elements. The difference here is that elements can be in any order.

• Reply • Share ›

kartik kapil • 5 years ago

There can be two ways to solve this:
1) Sort both lists in O(mLogm + nLogn). After sorting, use the areIdentical() to compare the lists.

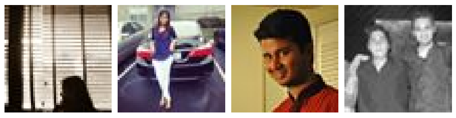
2) Use Hashing

• Reply • Share ›

-
- GeeksforGeeks

  Like

  93,347 people like GeeksforGeeks.



  Facebook social plugin
-
-
  - Interview Experiences
  - Advanced Data Structures
  - Dynamic Programming
  - Greedy Algorithms
  - Backtracking
  - Pattern Searching
  - Divide & Conquer
  - Mathematical Algorithms
  - Recursion
  - 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
- Follow @GeeksforGeeks Subscribe
- ## Recent Comments

  - Baba

    Traverse the tree in inorder fashion and insert...

Populate Inorder Successor for all nodes · 54 minutes ago

- creeping_death

  Ruby solution, slightly different, easier to...

  Longest Even Length Substring such that Sum of First and Second Half is same · 1 hour ago

- darkprotocol

  Anyone round-1 4th question?

  Amazon Interview | Set 121 (On-Campus for SDE-1) · 1 hour ago

- S.Nkm

  Would a simple answer involving the Unix epoch...

  Find number of days between two given dates · 1 hour ago

- Shomina

  Found Real Job INterview Questions Here...

  Flipkart Interview Experience | Set 20 (For SDE-II) · 2 hours ago

- Shimona

  Found Real Job INterview Questions here...

  Myntra Interview Experience | Set 4 (For Senior Software Engineer) · 2 hours ago

•