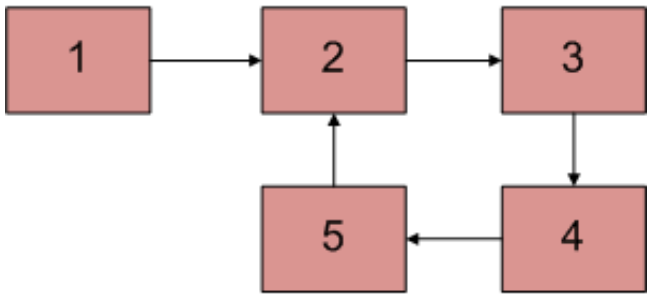
GeeksforGeeks

A computer science portal for geeks

Write a C function to detect loop in a linked list

Below diagram shows a linked list with a loop



Following are different ways of doing this

Use Hashing:

Traverse the list one by one and keep putting the node addresses in a Hash Table. At any point, if NULL is reached then return false and if next of current node points to any of the previously stored nodes in Hash then return true.

Mark Visited Nodes:

This solution requires modifications to basic linked list data structure. Have a visited flag with each node. Traverse the linked list and keep marking visited nodes. If you see a visited node again then there is a loop. This solution works in O(n) but requires additional information with each node.

A variation of this solution that doesn't require modification to basic data structure can be implemented using hash. Just store the addresses of visited nodes in a hash and if you see an address that already exists in hash then there is a loop.

Floyd's Cycle-Finding Algorithm:

This is the fastest method. Traverse linked list using two pointers. Move one pointer by one and other pointer by two. If these pointers meet at some node then there is a loop. If pointers do not meet then linked list doesn't have loop.

Implementation of Floyd's Cycle-Finding Algorithm:

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

/* Link list node */
struct node
{
  int data;
  struct node* next;
};

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;
}

int detectloop(struct node *list)
{
  struct node  *slow_p = list, *fast_p = list;

  while(slow_p && fast_p &&
          fast_p->next )
  {
    slow_p = slow_p->next;
    fast_p  = fast_p->next->next;
    if (slow_p == fast_p)
    {
       printf("Found Loop");
       return 1;
    }
  }
  return 0;
}

/* Drier program to test above function*/
int main()
{
  /* Start with the empty list */
  struct node* head = NULL;

  push(&head, 20);
  push(&head, 4);
  push(&head, 15);
  push(&head, 10);

  /* Create a loop for testing */
  head->next->next->next->next = head;
  detectloop(head);

  getchar();
}
```

Time Complexity: O(n)
Auxiliary Space: O(1)

References:
http://en.wikipedia.org/wiki/Cycle_detection
http://ostermiller.org/find_loop_singly_linked_list.html

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

Tags: Linked Lists, loop

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

wrestler • 14 days ago

in while condition, slow need not be there.. Only while(fast && fast->next) will solve the purpose.

AYUSH KUMAR • 3 months ago

c++ code

http://ideone.com/q3amq5

Arjun K • 5 months ago

Here is C program to check self loop in a graph > http://www.msccomputerscience....

codecrecker • 6 months ago

```
#include<stdio.h>

#include<stdlib.h>

typedef struct nd node;

struct nd{

int d;

node *n;
```

```
};

node *head,*ptr;

node *createNode(int d)

{

node *tmp;
```

see more

1 • Reply • Share ›

Mohit Chikara • 6 months ago

it will work and slowptr and fastptr can match at any node in the loop, not necessarily at the starting node of the loop.for ex if list is as 1->2->3->->4->2
than slowptr will move as 1->2->3->4->2->3->4->2
and fastptr will move as 1->3->1->3->1->3 and match will occur at node 3

1 • Reply • Share ›

saurav Mohit Chikara • 5 months ago

fastptr will be 1->3->2->4
slowptr will be 1->2->3->4

• Reply • Share ›

xoxo • 6 months ago

```
bool floyd()

{

node *ptr1,*ptr2;

ptr1 = root;

ptr2 = root;

while(ptr1->next!=NULL)

{

if((ptr2->next == NULL) || (ptr2->next->next == NULL))

return false;

else

{
```

see more

See more

• Reply • Share ›

Anil Kumar K K • 7 months ago

There is a problem with the above code : It detects loop at wrong location .

Example -

```
{

/* Start with the empty list */

struct node* head = NULL;

push(&head, 20);

push(&head, 4);

push(&head, 15);

push(&head, 10);

/* Create a loop for testing */

head->next->next->next->next = head->next;

detect_loop(head);

getchar();

}
```

./a.out

Loop detected... [20]

• Reply • Share ›

algo1 • 7 months ago

Is slow != NULL necessary ? Are there any test cases to justify this?

2 • Reply • Share ›

Guest ➚ algo1 • 7 months ago

i guess it is to support empty list..or am i wrong?

• Reply • Share ›

Guest ➚ Guest • 6 months ago

Fast_p satisfy that condition for empty list.... slow_p not necessary according to me...

1 • Reply • Share ›

rahul • 7 months ago

when the detect loop is returning 1 then what is accepting it in main() ??

• Reply • Share ›

jayasurya j rahul • 7 months ago

In this context nothing catches the value returned by the function..but it is obvious right?

• Reply • Share ›

Pranav Kumar Jha • 7 months ago

Instead of
while(slow_p && fast_p &&
fast_p->next),

while(fast_p &&
fast_p->next) will also do the work!

6 • Reply • Share ›

DS+Algo=Placement • 8 months ago

plz someone tell me the code using hashing

• Reply • Share ›

Saurabh • 8 months ago

A very good explanation is available here.

http://stackoverflow.com/quest...

It explains why the algorithm works!! Have a look guys!!

• Reply • Share ›

Vatsalya Chauhan Saurabh • 8 months ago

Thanks a lot brother. it is really awesome.....

But i have a doubt, about it's Time Complexity: O(n).
Is it correct ?

• Reply • Share ›

leotreim scofield Vatsalya Chauhan • 7 months ago

yes.. it is...
SInce, the algorithm ends when (slow_p == fast_p), the traversal will go through maximum number of nodes where the last node is pointing to first node instead of null. In that case, the number of nodes the slow_p will cover is ~ = (n+n) which is equivalent to O(n)

• Reply • Share ›

Shikha Baranwal • 8 months ago

can u guys please explain how will I find out d node where loop has been occurred?

• Reply • Share ›

Rohit Pujar ➚ Shikha Baranwal • 8 months ago

run two pointers from the beginning node. Increment 'fast' pointer by two nodes and 'slow' pointer by one. After each iteration check if fast==slow. If they're equal, loop has been detected

• Reply • Share ›

Shikha Baranwal ➚ Rohit Pujar • 3 months ago

thank u.

• Reply • Share ›

Abhishek Kannojia • 9 months ago

I think checking *slow_p* for NULL is redundant here and

```
while(slow_p && fast_p && fast_p->next )
```

can be rewritten as

```
while(fast_p && fast_p->next)
```

as *slow_p* will be behind *fast_p* always and thus will never be NULL. Additionally *slow_p* and *fast_p* both starts from same point, so in case of NULL list *fast_p* will terminate the loop.

8 • Reply • Share ›

GOPI GOPINATH ➚ Abhishek Kannojia • 9 months ago

Yes, it can be eliminated here.

• Reply • Share ›

sumit agarwal • 9 months ago

how is the Floyd's cycle-finding algorithm fastest??? according to me it's complexity is more than O(n) something like O(a*n) where a>1
please clear my doubt!!!

1 • Reply • Share ›

GOPI GOPINATH ➚ sumit agarwal • 9 months ago

its linear because we can detect the loop in O(length of the loop + position of meeting point of both the pointers) This is because each time they move, The difference between them is reduced by '1' , Hence even if they move in cycle, the complexity will

still be linear.

• Reply • Share ›

sangee • 9 months ago

Can someone explain how dis happens?..how do the tortoise and hare eventually meet up?

• Reply • Share ›

GOPI GOPINATH ➚ sangee • 9 months ago

Draw a list with loop and have 2 pointers starting from head with different speeds as mentioned(fast and slow ptrs) .you can find the answer. A similar example : Assume a 5km run in a ground and people initially start from the same position, after a while, each one may be covering different number of rounds but their physical distance in the ground may be equal, guess How ? I leave it for u

• Reply • Share ›

SANTOSH KUMAR MISHRA • 9 months ago

```
bool DetectLoop(node *head)
{
node *slowptr,*fastptr;
slowptr = fastptr = head;
if(head == NULL)
{
printf("\n\nList is empty.");
return FALSE;
}
while(slowptr && fastptr && fastptr->next)
{
fastptr = fastptr->next->next;
slowptr = slowptr->next;
if(slowptr == fastptr)
return TRUE;
}
return FALSE;
}
```

• Reply • Share ›

ALEX • 10 months ago

we moves fast pointer by 2.....

Here the reason.....

what if we want to print that node value, or if we want to find on which node we are having loop....?????

ANS:

step1: Firstly find the loop in the linked list using fast pointer and slow pointer.

step2: set the slow point on head.

step3: now move fast pointer and slow pointer by one skip.

step4: slow_pointer and fast_pointer will meet at that node on which loop is occur.....

you can take any example and check it...

Correct me if i'm wrong.....

6 • Reply • Share ›

The_Geek ALEX • 6 months ago

Can anyone explain d logic behind it, because its working fine fr all the lists which I tested.

• Reply • Share ›

Guest ALEX • 7 months ago

0,1,2,3,4,5 loops to 2
according to the steps fast and slow meets at 4
and step 3 to find the meeting node will make them meet again at 4

• Reply • Share ›

The_Geek Guest • 6 months ago

No dear, They will meet at 2 only. We have to skip by only one step for both fast and slow, after we found that there exists a loop.

• Reply • Share ›

Veer • 10 months ago

while(slow_p && fast_p &&
fast_p->next)

What does 'fast->next' adds to the condition ?
Its works fine with just 'slow_p && fast_p' .

• Reply • Share ›

nilayan_ahmed Veer • 10 months ago

fast_p->next->next will crash if fast_p->next is NULL and fast_p->next is NULL for the last node hence the check.

4 • Reply • Share ›

Suyash Soni • a year ago

What does that condition inside while loop while(slow_p && fast_p && fast_p->next) mean?

What does that condition inside while loop while(slow_p && fast_p && fast_p->next) mean?
Please tell me that when will this loop stop?

• Reply • Share ›

tharun Suyash Soni • 9 months ago

Divide the problems into 3 cases :

Even number of nodes[fast!=NULL]
Odd number nodes[fast->next!=NULL]
Only one node[slow!=NULL]

• Reply • Share ›

walter white Suyash Soni • 10 months ago

if there is no loop , we get at some point slow_p or fast_p = NULL, to move out of while loop

• Reply • Share ›

Kevin Alex Mathews • a year ago

I think we can use Depth First Search graph for the problem. In the DFS spanning tree, if there is any back edge, then that points to the presence of loop in the linked list.

Time complexity of O(n+e) is comparable to Floyd's cycle detection algorithm's O(n), where n is the number of nodes in the list and e is the number of pointers.

1 • Reply • Share ›

popeye Kevin Alex Mathews • 10 months ago

That's the same as marking the node as visited. (2nd method)

1 • Reply • Share ›

Himanshu Dagar • a year ago

I think that there is no need of checking slow_p condition in the while loop
(It will even work without this condition)

2 • Reply • Share ›

Himanshu Dagar • a year ago

Floyd's Cycle Finding algorithm
(we can not remove that loop from the linked list with the help of this algo.)

2 • Reply • Share ›

neelabhsingh • a year ago

suppose there is node which pointing to itself then is there is loop or not. Obviously this node is last node in the linked list. Instead point to NULL it is pointing to itself ? Please clear it..

• Reply • Share ›

swebdev neelabhsingh • a year ago

swebdev › › neelabhsingh • a year ago

Ofcourse there is loop.. The double pointer and single-pointer will point to same node at the beginning and also after one unit of movement, thus they will point to same node, returning true.

^ | v • Reply • Share ›

Sumit Nathany • 2 years ago

@Vibhu - How will yoou sort the linked list if it doesn&#039t end (there is no NULL). Also you don&#039t have the number of elements in it.

^ | v • Reply • Share ›

Abhilasha Mishra • 2 years ago

```
node * loop(node*head).
{
node *p,*q;
p=q=head; int flag=0;.
while(q!=NULL)
{
if(q->next==q)
flag=1;
else
{
while(p!=q)
{
{
if(p!=q->next)
p=p->next;
else
flag=1;
}
```

see more

^ | v • Reply • Share ›

ABHILASHA31 • 2 years ago

```
node * loop(node*head)
{
node *p,*q;
p=q=head; int flag=0;
while(q!=NULL)
{
if(q->next==q)
flag=1;
else
{
```

```
{
while(p!=q)
{
{
if(p!=q->next)
p=p->next;
else
flag=1;
}
```

see more

• Reply • Share ›

Akshay Jindal • 2 years ago

@Vibhu Tiwari:
nyc idea...(y)

• Reply • Share ›

Karshit • 2 years ago

My code for Cycle detection and Removal...
HOpe you guys find it useful.

```
[sourcecode language="C++"]
#include <iostream>

using namespace std;

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

void print(node *head)
{
while (head != NULL) {
cout << head -> data << " ";
head = head -> next;
}
```

see more

• Reply • Share ›

Gaurav Verma • 2 years ago

geek ban ja tu bhi ab

• Reply • Share ›

Vibhu Tiwari • 2 years ago

A loop in a linked list can just be determined by first sorting the linked list and then determining if any node&#039s value is less than previous.If yes then the linked list has a loop.

2 ^ | v • Reply • Share ›

Load more comments

Subscribe Add Disqus to your site Privacy

- GeeksforGeeks

  Like

  93,235 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

- lebron

  since the array size is 5, it takes constant...

  K'th Smallest/Largest Element in Unsorted Array | Set 3 (Worst Case Linear Time) · 3 hours ago

- lebron

  merge sort

  K'th Smallest/Largest Element in Unsorted Array | Set 3 (Worst Case Linear Time) · 3 hours ago

- Shubham Sharma

  You saved my time :)

  Searching for Patterns | Set 2 (KMP Algorithm) · 3 hours ago

- Prakhar

  Why so many LOCs, if I'm not wrong (please...

  Largest Sum Contiguous Subarray · 4 hours ago

- Aayush Gupta

  For R4 Q3, Another solution would be to use a...

  Amazon Interview Experience | Set 168 · 5 hours ago

- EigenHarsha

  For Power Of 2, We Simply Doing.. var1 =

  Practo Interview Experience | Set 2 (Off-Campus) · 5 hours ago