Write a function to reverse a linked list

Iterative Method

Iterate trough the linked list. In loop, change next to prev, prev to current and current to next.

Implementation of Iterative Method

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

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

/* Function to reverse the linked list */
static void reverse(struct node** head_ref)
{
    struct node* prev   = NULL;
    struct node* current = *head_ref;
    struct node* next;
    while (current != NULL)
    {
        next  = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }
    *head_ref = prev;
}

/* Function to push a node */
void push(struct node** head_ref, int new_data)
```

Google™ Custom Search
GeeksforGeeks
53,528 people like GeeksforGeeks.
Interview Experiences
Advanced Data Structures
Dynamic Programming
Greedy Algorithms
Backtracking
Pattern Searching
Divide & Conquer
Mathematical Algorithms
Recursion
Geometric Algorithms

```
{
    /* 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;
}

/* Function to print linked list */
void printList(struct node *head)
{
    struct node *temp = head;
    while(temp != NULL)
    {
        printf("%d  ", temp->data);
        temp = temp->next;
    }
}

/* 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, 85);

     printList(head);
     reverse(&head);
     printf("\n Reversed Linked list \n");
     printList(head);
     getchar();
}
```

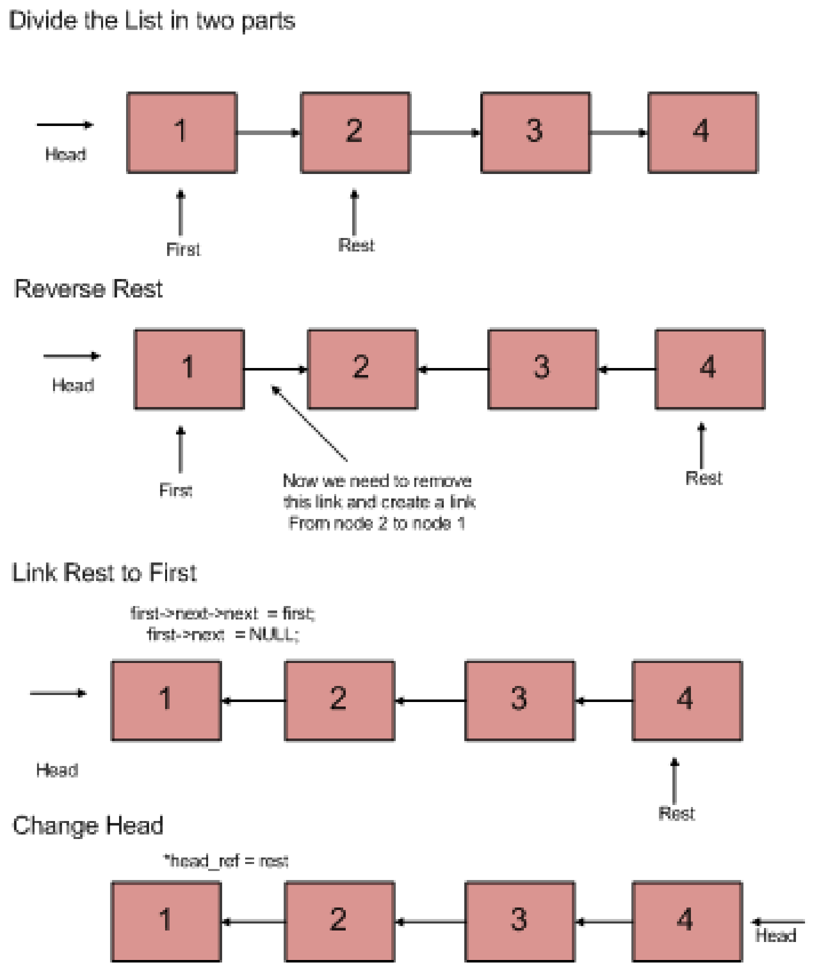
Time Complexity: O(n)
Space Complexity: O(1)

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

Recursive Method:

```
   1) Divide the list in two parts - first node and rest of the linked list.
   2) Call reverse for the rest of the linked list.
   3) Link rest to first.
   4) Fix head pointer
```



```
void recursiveReverse(struct node** head_ref)
{
    struct node* first;
    struct node* rest;

    /* empty list */
    if (*head_ref == NULL)
```

```
        return;

    /* suppose first = {1, 2, 3}, rest = {2, 3} */
    first = *head_ref;
    rest  = first->next;

    /* List has only one node */
    if (rest == NULL)
        return;

    /* reverse the rest list and put the first element at the end */
    recursiveReverse(&rest);
    first->next->next  = first;

    /* tricky step -- see the diagram */
    first->next  = NULL;

    /* fix the head pointer */
    *head_ref = rest;
}
```

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

References:
http://cslibrary.stanford.edu/105/LinkedListProblems.pdf

Recent Comments

Abhi You live US or India?

Google (Mountain View) interview · 47 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...

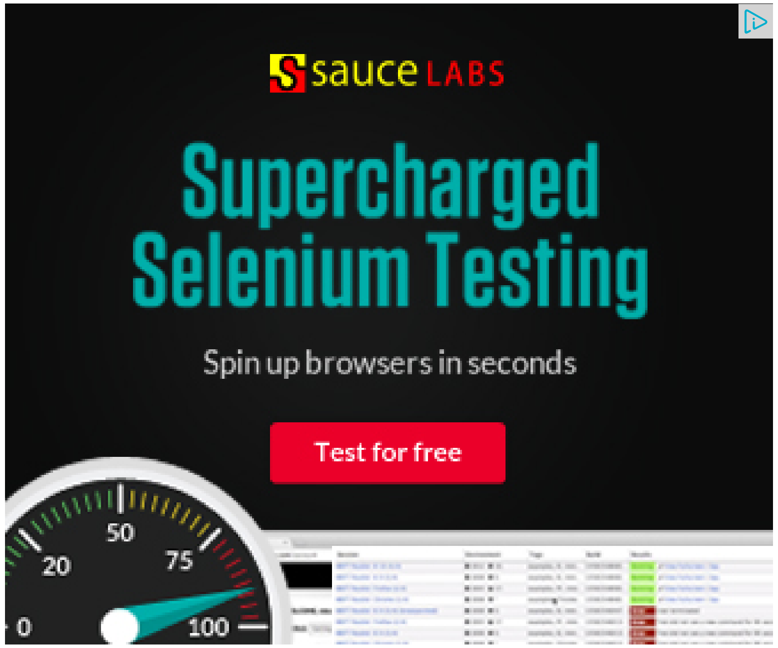
Backtracking | Set 2 (Rat in a Maze) · 1 hour ago

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



hour ago

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

Find subarray with given sum · 2 hours ago

AdChoices

- ▶ Linked List
- ▶ C++ Code
- ▶ Linked Data

AdChoices

- ▶ Programming C++
- ▶ C++ Reverse List
- ▶ Java Reverse

AdChoices

- ▶ Java Reverse
- ▶ Reverse Polarity
- ▶ Function Value

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

29 Tweet 0 3

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

79 Comments **GeeksforGeeks**

Sort by Newest

Join the discussion

VaraKalyan M • 23 days ago
I think
/* fix the head pointer */
*head_ref = rest;
this step should be above recursive call. Otherwise only two elements will be |
Reply • Share ›

Vesper VaraKalyan M • 15 days ago
https://www.google.co.in/url?s...
Reply • Share ›

AMIT JAMBOTKAR VaraKalyan M • 23 days ago
check ones..it will work................
first->next->next = first;

/* tricky step -- see the diagram */
first->next = NULL;

/* fix the head pointer */
*head_ref = rest;
just here think about function call stack....
Reply • Share ›

VaraKalyan M AMIT JAMBOTKAR • 23 days ago
As the statement is below the recursive call, it will be executed
executed. At the end, rest will be pointing to head->next and 2 e
missing any thing here.
Reply • Share ›

AMIT JAMBOTKAR • 25 days ago

For Java lovers

public class LinkedList<e> implements Cloneable{

Node<e> head = null;

public class Node<t> {

T value;

Node<t> nextReference;

private Node(T value) {

this.value = value;

this.nextReference = null;

}

public Node(T value, Node<t> ref) {

see more

Reply • Share ›

Saket Pandey • a month ago

@geeks: pl comment if its wrong

Link oldFirstNode = lnkList.head;
lnkList.reverseList(lnkList.head);
oldFirstNode.setNext(null);
System.out.println(lnkList);
.
.
.
public void reverseList(Link curr){

```
if(curr.getNext() == null){
head = curr;
return;
}
reverseList(curr.getNext());
Link tmp = curr.getNext();
tmp.setNext(curr);
}
```

^ | ⌄ • Reply • Share ›

aishInch • a month ago

what is the need of declaring the reverse function static here??

^ | ⌄ • Reply • Share ›

Guest • 2 months ago

how in diagram 2 when rest reaches null ,next poiner of all node except first is

^ | ⌄ • Reply • Share ›

Var • 2 months ago

```
public class ReverseALinkedList{

public static void main(String[] args)

{

Node a=new Node(1);

Node b=new Node(2);

Node c=new Node(3);

Node d=new Node(4);
```

Node e=new Node(5);

Node f=new Node(6);

Node g=new Node(7);

Node h=new Node(8);

see more

1 · Reply · Share ›

armgeek · 2 months ago

Simple Solution would be this ::

Consider the LL as below ::

```
struct node
{
int d;
struct node *next;
}*q,*start;

void reverse()
{
struct node *p1;
struct node *p2;
p1=p2=NULL;

while(q!=NULL)
{
p1=p2;
```

see more

1 · Reply · Share ›

armgeek armgeek • 2 months ago

sorry i might have missed the braces. please excuse me.

• Reply • Share ›

padma • 2 months ago

*head_ref = rest;
what does it do???

• Reply • Share ›

Amit • 3 months ago

```
node* reverse(node* head, node* pre){

if(head->next == NULL){

head->next = pre;

return head;

}

//temp is always the head of the reversed linked list

node* temp = reverse(head->next, head);

head->next = pre;

// cout<<temp->key;

return temp;

}
```

• Reply • Share ›

Kunal Arora • 3 months ago

Can anyone explain me what this line is doing...Thanks
first->next->next=first ;

Reply • Share ›

Pankaj Kushwaha → Kunal Arora • 3 months ago

Its basically for reversing the links. Suppose that you have a link list wit
to second node, then after putting first->next->next=first, second node

Reply • Share ›

Ignite → Pankaj Kushwaha • a month ago

can it be like that..?
rest->next= first;

Reply • Share ›

Kunal Arora → Pankaj Kushwaha • 3 months ago

Thanks a lot dude.....

Reply • Share ›

Kunal Arora • 3 months ago

I have implemented the reverse function in yet another recursive way.....

@admin please comment if it is wrong or i left some case

```
void reverse(struct node *head)

{

struct node *p=head,*q=head;

if(p==NULL)

return;

reverse(p->next);
```

p->data=q->data;

printf("%d",q->data);

q=q->next;

}

• Reply • Share ›

Himanshu Dagar • 3 months ago

very good method for reversing in recursive way
(y)

1 • Reply • Share ›

Pankaj • 4 months ago

@admin, in recursion method, if we divide the list into two equal parts and pro
linked list, we would obtain the output a4a3a2a1 by first reversing first half a2a
and then complete, a4a3a2a1(second block followed by first block). is it corre

• Reply • Share ›

Vivek • 5 months ago

recursive implementation.
pls go through this sol.

```
struct node * reverse(struct node *head)
{
static struct node *first=NULL;
struct node *second;
if(!head)
return first;
second=head->next;
head->next=first;
first=head;
return reverse(second);
}
```

Reply • Share ›

n00b • 8 months ago

```
struct node *rreverse(struct node *current, struct node **prev)
{
        struct node *next;

        if (!current)
                return *prev;

        next = current->next;
        current->next = *prev;
        *prev = current;

        return rreverse(next, prev);

...

        struct node *prev = NULL;
        head = rreverse(head, &prev);
```

Short and sweet ;)

8 • Reply • Share ›

Shivendu Kumar • 9 months ago

This code may cause problem if the link list is empty.
Also, in this solution, your link list will be reversed but the start pointer (that you
function) will point to last node of reversed link list. Rest of your list will be lost.

ptr value and return that saved ptr value at the end of the function.

4 • Reply • Share ›

Shivendu Kumar • 9 months ago

```
/*
Another solution of Reversing the Link List using loop.
*/
void recRevLL(struct node **head)
{
struct node *f,*s;

if(*head==null)
return;

f=*head;
s=f->next;

if(s==null)
return;

f->next=null;

while(s!=null)
{
*head=s->next;
s->next=f;
f=s;
s=*head;
}

*head=f;
}
```

3 • Reply • Share ›

Chandu • 9 months ago

static struct node *ptr; // global variable

struct node *RecReverse(struct node *head)
{
if(head == NULL)
return NULL;
else if(head->next == NULL) {
ptr = head;
return head;
}

struct node *temp;
temp = RecReverse(head->next);

temp->next = head;
temp = temp->next;
temp->next = NULL;

return temp;
} // Print ptr after this step...

Reply • Share ›

hemadrigon • 9 months ago

for recursive reverse algo.
I am trying to understand how the fixing head pointer works.
I ran the code in gdb env and the rest ptr always correctly points to the addres
recursiveReverse are successful and the code below recurse func is being ex
understanding the rest ptr should point to address of value 2 in the end.

```
/* fix the head pointer */
   *head_ref = rest;
```

can anyone please put more light on this

• Reply • Share ›

Vijay Daultani hemadrigon • 9 months ago

Yes the code is correct and its working just fine..

Because...

If you read the code properly you would note that in the recursive call it
but actually it is &rest.

I am tracking down the series of call which will result the understanding

Assume linked list is 1 -> 2 -> 3-> 4

main()

{

reverse(&head) // head_ref -> head -> 1 {It means head_ref is pointing

// head and head is pointing to 1
// or I can say *head_ref = head and *head = 1

see more

7 • Reply • Share ›

hemadrigon • 9 months ago

I am not able to understand how fixing the head pointer works ..
/* fix the head pointer */
*head_ref = rest;

I ran the code in gdb environment to print addresses and the rest ptr always p
correct. Can anyone explain .. I thought the rest ptr would get updated and poi
thanks for help..

1 ^ | ˅ • Reply • Share ›

Ashok Ramnath • 10 months ago

simple recursion logic.

```
struct node* rec(struct node *ptr, struct node * prev).
{
struct node * temp;.
temp=ptr->next;
ptr->next=prev;
if(! temp)
return ptr;
rec(temp, ptr);

}
```

^ | ˅ • Reply • Share ›

Sunil Mourya • 10 months ago

Debarnob, Run this below recursion function.. i just tried to print pointer values
understand...

```
void recursiveReverse(struct node** head_ref)
{
struct node* first;
struct node* rest;
/* empty list */
if (*head_ref == NULL)
return;

/* suppose first = {1, 2, 3}, rest = {2, 3} */
first = *head_ref;
```

```
rest = first->next;

printf("[Push on Stack] %pt%pt%pn",first,rest,(*head_ref));

/* List has only one node */
if (rest == NULL)
```

see more

2 • Reply • Share ›

Rahul Sawhney • 10 months ago

Reverse of Link List can be done easily by taking 3 pointers.

```
void reverse(node *head).
{
node *p1,*p2,*p3;

p1=head;
p2=p1->next;
p3=p2->next;

p1->next=NULL;
p2->next=p1;

while(p3!=NULL)
{
p1=p2;
p2=p3;
p3=p3->next;
p2->next=p1;

}
head=p2;
```

}

8 ^ | ⌄ • Reply • Share ›

ram ➦ Rahul Sawhney • 3 months ago

good work

^ | ⌄ • Reply • Share ›

pranjalgupta • 10 months ago

We can also reverse the linked list without taking head's reference which lead
*headref=rest. Below is the function to do that:

list* recrev(list *head)
{
if(head==NULL)
return NULL;
if(head->next==NULL)
return head;
list* second = head->next;
head->next = NULL;
list* newhead = recrev(second);
second->next = head;
return newhead;
}
/* the value of newhead is calculated once and returned to every impending re

1 ^ | ⌄ • Reply • Share ›

shivi • 11 months ago

```
 Node* Reverse(Node *head,Node *prev)
{


       if(head==NULL)
               return prev;
```

```
        else
        {
                Node *temp=head->next;
                head->next=prev;
                prev=head;
                return Reverse(temp,prev);
        }
}
```

this seems much better and simpler?!!!

^ | ⌄ • Reply • Share ›

shivi ➦ shivi • 11 months ago

call this function with (head,NULL) will return new head!

^ | ⌄ • Reply • Share ›

Rakesh Rk • 11 months ago

It will take O(n^2) ryt?

^ | ⌄ • Reply • Share ›

Arnab Bhattacharjee • 11 months ago

You will understand once you see what&#039s happening. This is an exquisite
for knowing pointer tricks but its better to write the iterative version in general.

^ | ⌄ • Reply • Share ›

Debarnob Sarkar • 11 months ago

Cant understand hoe the "HEAD POINTER" IS BEING FIXED! :(
Can sumbody please explain?

^ | ⌄ • Reply • Share ›

Ankit Gupta • 11 months ago

node* reverse(node** start){
//start will be pointer to head
node *trav = *start;
while(trav->next!=NULL){
node *temp = trav->next->next;
trav->next->next = *start;
*start = trav->next;
trav->next=temp;
}return *start;
}

Reply • Share ›

Ankit Gupta • 11 months ago

Piyush Gandhi nope u can do that without a stack !!!!....

Reply • Share ›

Piyush Gandhi • 11 months ago

i thought it as well.....but this can be done using stack only ....and many corpo
stacks....dont know the reason.

Reply • Share ›

Ankit Gupta • 11 months ago

Alternate solution.....traverse through the linked list and keep inserting each no

Reply • Share ›

abhishek08aug • a year ago

Intelligent :D

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

Reply • Share ›

sd • a year ago

```
 NODE reverse_list_recursive(NODE head)
{
        NODE curr_node;
        curr_node = head;
        if (curr_node->next == NULL) {
                return curr_node;
        }
        else {
                reverse_list_recursive(curr_node->next)->next =     cı
                curr_node->next = NULL;
                return curr_node;
        }
}
```

^ | ⌄ • Reply • Share ›

Parikksit Bhisay • a year ago

There seems to be something wrong with the last line of the code. I think heac
because it gets reset in every recursion.

This line is incorrect in my opinion:
[sourcecode language="C"]
/* fix the head pointer */.
*head_ref = rest;.

However, I wouldn&#039t say I&#039m 100% sure because I tried a java impl
Here&#039s my java code if anyone bumps into this thread with the same pro
[sourcecode language="Java"]
public static void recursiveReverse(SLL head){.
//First we declare.

SLL first, rest;.

/*
* Assigning first and rest nodes as shown below:
* [2]->[8]->[5]->[9]->null.
* then, first node is [2] and the rest are [8],[5],[9].

see more

Reply • Share ›

Nishant Kumar • a year ago

Two more recursive method.
First method directly change the first node reference while second method ret
reversed linkList.

```
 struct link{
    int data;
    struct link* next;
};
typedef struct link node;

node* reverse1(node* local,node** start){
        if(local->next==NULL){
                *start=local;
                return local;
        }
        node* top=*start;
        node* tmp=reverse1(local->next,start);
        tmp->next=local;
```

see more

Reply • Share ›

Nishant Kumar ➦ Nishant Kumar • a year ago

modified method 1

```
node* reverse1(node* local,node** start){
        if(local->next==NULL){
                *start=local;
                return local;
        }
        node* tmp=reverse1(local->next,start);
        local->next=NULL;
        tmp->next=local;
        return local;
}
```

Reply • Share ›

Soumya Sengupta • a year ago

@geeksforgeeks......
awesum recursive code...........
seemed so easy..
:)

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

Reply • Share ›

rahulcynosure • a year ago

struct Node * recrev(struct Node * curr,struct Node * prev)
{
if(!curr){

return prev;
}

}
struct Node * newH = recrev(curr->next,curr);
curr->next=prev;
return newH;

}

from main call this function as :
struct Node * newH = recrev(head,NULL);

Reply • Share ›

Ratikanta Pal • a year ago

we need to track the head pointer.
it should be fix.

static int track=0;.

reverse(head);

public static void reverse(NodeLinkList node) {.
//boolean status=true;
if (node == null).
return;
NodeLinkList first = node, rest = node.next;.
if (rest == null).
return;
else{
reverse(rest);
//System.out.println("in rw : "+first.data+rest.data+start.data);.
first.next.next = first;.
first.next = null;.
if(track++==0)
start=rest;

```
    ;

    }
```

^ | v • Reply • Share ›

Load more comments

Subscribe   Add Disqus to your site