GeeksforGeeks
A computer science portal for geeks
Skip to content

IDE Q&A GeeksQuiz

- Home
- Algorithms
  - Analysis of Algorithms
  - Searching and Sorting
  - Greedy Algorithms
  - Dynamic Programming
  - Pattern Searching
  - Other String Algorithms
  - Backtracking
  - Divide and Conquer
  - Geometric Algorithms
  - Mathematical Algorithms
  - Bit Algorithms
  - Graph Algorithms
  - Randomized Algorithms
- Data Structures
  - Linked List
  - Stack
  - Queue
  - Binary Tree
  - Binary Search Tree
  - Heap
  - Hashing
  - Graph
  - Advanced Data Structure
  - Array
  - Matrix
  - Misc
- GATE
- Interview
- Q&A
- C
- C++
- Java
- Book
- Contribute
- Contests
- Jobs
- Ask a Q
- Array
- Bit Magic
- C/C++
- Article
- GFact
- Linked List
- MCQ
- Misc
- Output
- String
- Tree
- Graph
- IDE
- Android App

Menu ▼

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.

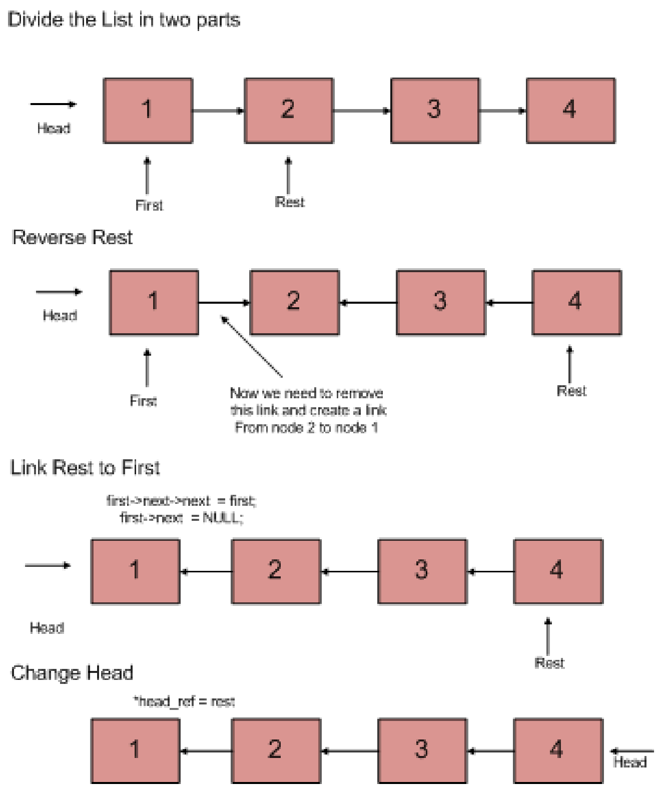
Implementation of Iterative Method

```
#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)
{
    /* 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)

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)

A Simpler and Tail Recursive Method

Below is C++ implementation of this method.

```
// A simple and tail recursive C++ program to reverse
// a linked list
#include<bits/stdc++.h>
using namespace std;

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

void reverseUtil(node *curr, node *prev, node **head);

// This function mainly calls reverseUtil()
// with prev as NULL
void reverse(node **head)
{
    if (!head)
        return;
    reverseUtil(*head, NULL, head);
}

// A simple and tail recursive function to reverse
// a linked list.  prev is passed as NULL initially.
void reverseUtil(node *curr, node *prev, node **head)
{
    /* If last node mark it head*/
    if (!curr->next)
    {
        *head = curr;

        /* Update next to prev node */
        curr->next = prev;
        return;
    }

    /* Save curr->next node for recursive call */
    node *next = curr->next;

    /* and update next ..*/
    curr->next = prev;

    reverseUtil(next, curr, head);
}

// A utility function to create a new node
node *newNode(int key)
{
    node *temp = new node;
    temp->data = key;
    temp->next = NULL;
    return temp;
}

// A utility function to print a linked list
void printlist(node *head)
{
    while(head != NULL)
    {
        cout << head->data << " ";
        head = head->next;
    }
    cout << endl;
}

// Driver program to test above functions
int main()
```

```
{
    node *head1 = newNode(1);
    head1->next = newNode(2);
    head1->next->next = newNode(2);
    head1->next->next->next = newNode(4);
    head1->next->next->next->next = newNode(5);
    head1->next->next->next->next->next = newNode(6);
    head1->next->next->next->next->next->next = newNode(7);
    head1->next->next->next->next->next->next->next = newNode(8);
    cout << "Given linked list\n";
    printlist(head1);
    reverse(&head1);
    cout << "\nReversed linked list\n";
    printlist(head1);
    return 0;
}
```

Output:

```
Given linked list
1 2 2 4 5 6 7 8

Reversed linked list
8 7 6 5 4 2 2 1
```

Thanks to Gaurav Ahirwar for suggesting this solution.

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

298 Comments Category: Linked Lists

Related Questions:

- Sort a linked list that is sorted alternating ascending and descending orders?
- Select a Random Node from a Singly Linked List
- Why Quick Sort preferred for Arrays and Merge Sort for Linked Lists?
- 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
- Generic Linked List in C
- Clone a linked list with next and random pointer | Set 2

Post navigation
← Write an Efficient C Program to Reverse Bits of a Number Write a C function to detect loop in a linked list →

Recommend 8 Share Sort by Newest

Join the discussion…

Monkey D. Luffy • 6 days ago
non recursive simple iterative O(n) time complexity and O(1) space complexity solution

https://ideone.com/Dlh707
• Reply • Share ›

M.Talha • 14 days ago
*head_ref = rest;

I agree that we must fix the head Node but. I cannot understand that(*head_ref = rest).It is in the recursive method. Is there anyone who illuminate me. Thanks :)
1 • Reply • Share ›

Belly M.Talha • 7 days ago
me too need explanation
• Reply • Share ›

Chaipau M.Talha • 10 days ago
There are plenty of explanations given in the below discussions...basically you have to understand each time in the recursive call we pass the "address" of "rest"...so the rest part is updated at the same address each time the recursion is called...
After: rest = first->next:
Add this statement : printf("The address of rest is at %u\n",&rest);
to understand better.
So coming to the *head_ref = rest statement...you may now probably understand...we assign where the head_ref points to (which is "head") to "rest"...which now contains the last element only.
As we move above the recursive stacks each time the "rest" was at the same address having the only last element in it.
(Correct me if I made a mistake.Thanks)
• Reply • Share ›

Manohar • 15 days ago
java version http://ideone.com/cMuTsW
1 • Reply • Share ›

Popeye Manohar • 14 days ago
nice
• Reply • Share ›

Anil Kumar Sahu • 17 days ago
void Reverse(struct List * CurrentNode)

{ if(CurrentNode==NULL)

return;

if(CurrentNode->next==NULL)

{

head=CurrentNode;

return ;

}

Reverse(CurrentNode->next);

CurrentNode->next->next=p;

CurrentNode->next=NULL;

}

• Reply • Share ›

Prateek Rathore • 18 days ago

//Refer : http://codingrecipies.blogspot...
public Node reverse(Node head){
Node revHead ; // to store the new Head pointer

if(head== null || head.next==null)
return head;

revHead=reverse(head.next) ;

head.next.next = head; // points to itself to create loop
head.next = null; // breaks the loop (old link)

return revHead ;
}

• Reply • Share ›

Vishwanath Rana • 18 days ago

Recursive method reverse link list ......

void reverse(struct node *cur, struct node *next_n, struct node **head)
{
if((cur == NULL) || (next_n == NULL)) { return; }

cur->next = next_n->next;
next_n->next = *head;
*head = next_n;

reverse(cur, cur->next, head);
}

• Reply • Share ›

jayasurya j • 21 days ago

simple recursive solution

```
ListNode* reverseList(ListNode* head) {
if(NULL==head||NULL==head->next) return head;
struct ListNode *newhead=reverseList(head->next);
head->next->next = head;
head->next = NULL;
return newhead;
}
```

• Reply • Share ›

snapDragoon jayasurya j • 21 days ago

what is wrong with this piece of code ?? please help

node* reverseList(node* l){
if(l==NULL || l->next == NULL){return l;}
else{return (reverseList(l->next)->next = l);}
}

• Reply • Share ›

jayasurya j snapDragoon • 19 days ago

ur code after else statement is wrong, you are adding node after the head node, while it should have been added after the tail node

• Reply • Share ›

Ankit Singh • 22 days ago

Why in the function reverse of the 1st method static keyword is used?

• Reply • Share ›

Narendra • 25 days ago

Can be written like http://ideone.com/Ht46TJ

• Reply • Share ›

Narendra • 25 days ago

SListNode *reverseUtil(SListNode *t, SListNode **h)

{

if(!t) return NULL;

SListNode *x=reverseUtil(t->next, h);

if(!x)

*h=t;

else

x->next=t;

t->next=NULL;

return t;

}

void reverse(SListNode **h)
{
reverseUtil(*h, h);

}

• Reply • Share ›

lucy • 25 days ago

```
void reverseUtil(node *curr, node *prev, node **head)

{

/* If last node mark it head*/

if (!curr)

{

*head = prev;

/* Update next to prev node */

//curr->next = prev;

return;

}

/* Save curr->next node for recursive call */

node *next = curr->next;

/* and update next ..*/

curr->next = prev;

reverseUtil(next, curr, head);

}
```

• Reply • Share ›

priyanka • 25 days ago

```
listnode* reverse(listnode *A,listnode **h)

{

listnode *t;

if(A->next)

{

t = reverse(A->next,h);

t->next = A;

A->next = NULL;

}else

*h = A;

return A;

}
```

• Reply • Share ›

abhi divekar • a month ago

The recursive methods are NOT O(1) space, they are O(n) stack space for n pointers, which is not inconsequential.

• Reply • Share ›

SmitParsania • a month ago

```
struct node* reverse_list(node *local_head){
if(local_head==NULL)
return local_head;
if(local_head->next==NULL)
return local_head;

node * head = reverse_list(local_head->next);

local_head->next->next = local_head;

local_head->next = NULL;

return head;
}
```

1 • Reply • Share ›

Dhrtzzz • a month ago

```
void reverse(struct node* head)
{ struct node* temp=head;
if(temp==NULL)
{
head=temp;
return;
}
reverse(temp->next);
struct node *temp1=temp->next;
temp1->next=temp;
temp->next=NULL;
}
```

• Reply • Share ›

Harshit Gupta ➦ Dhrtzzz • 22 days ago

What is the Complexity of this one? O(n)?

• Reply • Share ›

Dhrtzzz ➦ Harshit Gupta • 19 days ago

Yes !!!

• Reply • Share ›

vishnu • a month ago

http://javaworldwide.blogspot....

• Reply • Share ›

Gnomy7 • a month ago

What is wrong with this code? :(

```
#include <iostream>

using namespace std;
```

struct node{

int val;

node* next;

node(int v = 0){

val = v;

next = NULL;

}

};

see more

Reply • Share ›

Naive Coder • a month ago

Can somebody tell me if this is correct. I am basically a Python programmer. Just tried in C. Please correct it.

void recursiveReverse(struct node** head_ref)

{

struct node* first;

struct node* rest;

first = *head_ref;

rest = first->next;

if (first->next == NULL)

return first;

new_head = recursiveReverse(&rest);

new_head->next = first;
first->next = NULL

return first

}

Reply • Share ›

Dman • a month ago

Both implementations iterative and recursive
recursive is different

https://ideone.com/3dGSnX

Reply • Share ›

Goku • a month ago

@GeeksforGeeks check this simple recursive solution : http://ideone.com/oB7dsx

Reply • Share ›

Shashi Jey • a month ago
https://ideone.com/6rIihM
• Reply • Share ›

Unanymous • a month ago
can some one explain me about why *head_ref = rest; in the second method !!
I think it should update when the recursive call has been made but it is not updating...
• Reply • Share ›

Akshendra Unanymous • a month ago
We are passing rest using reference, so when on reaching the end, the value of rest is actually the last node, and not the one we set using first. The point is that its call be reference.
• Reply • Share ›

M.Talha Akshendra • 14 days ago
I also have a problem with that statement(*head_ref = rest;)!!
we have to fix the head Node however;
if we have (k) Node in the linkedlist
this statement is being executed (k) times. Why?
• Reply • Share ›

blank space • a month ago
https://ideone.com/WvtlAf
• Reply • Share ›

Deepak Sharma • a month ago
Tail recursion method would give segmentation fault if link list is empty i.e if current==NULL.
so, we need to add another base condition like
if(current==NULL)
return;
• Reply • Share ›

Klaus • 2 months ago
http://ideone.com/ZlNMs1 Smallest Recursive Implementation.
• Reply • Share ›

Pranjal Gupta • 2 months ago
can please tell me the mistake in this code for reversing the list ?
#include<iostream>

using namespace std;

struct node{

int data;

node *next;

};

void push(node **n,int m)

{

node *temp;

temp = new node;

see more

• Reply • Share ›

Shubham Hudda • 2 months ago

This will work right?

To call:
reverse(head,&head);

```
void reverse(struct node *p, struct node **head){
if(p->next==NULL){
*head=p;
return;
}

reverse(p->next,&head);
p->next->next=p;
p->next=NULL;
}
```

• Reply • Share ›

Harish Bisht • 2 months ago

```
private void reverse(node currentNode,node previousNode)

{

if(currentNode==null)

{

head=previousNode;

}

else

{

reverse(currentNode.next,currentNode);

currentNode.next=previousNode;

}

}
```

• Reply • Share ›

Dude • 2 months ago

here is an animation for the same:
http://animatedarena.com/Jex/f...

• Reply • Share ›

Chetan Pachpande • 2 months ago

I was working on C# implementation of Recursive versions, following are two recursive implementation in c# if anybody interested (above code is modified little bit for c# and another version is implement with help of this video: https://www.youtube.com/watch?...

```
/// <summary>
/// Reverses the linked list recursively.
/// </summary>
/// <param name="head">The head.</param>
public void ReverseLinkedListRecursive1(ref Node head)
{
Node first;
Node rest;
/* empty list */

if (head == null)
return;
/* suppose first = {1, 2, 3}, rest = {2, 3} */

first = head;
```

see more

^ | ⌄ • Reply • Share ›

Gaurav Verma • 2 months ago

simpler code:-

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

struct node {
struct node *next;
int data;
};
struct node * reverse(struct node *temp1,struct node *temp2);

void printll(struct node *p);
struct node * newnode(int num);

int main()
{

struct node *p;

p=newnode(1);
```

see more

^ | ⌄ • Reply • Share ›

Siva Praveen • 2 months ago

This might be smaller code for reverseutil

```
void reverseuntil(node * prev, node * current, node ** head_ref)

{

if(current == NULL)

return;

node * next = current ->next;

current -> next = prev;
```

current -> next = prev;

*head_ref = current;

reverseuntil(current,next,head_ref);

}

• Reply • Share ›

Return_0 • 2 months ago

Who all are finding it difficult to understand the recursive method..check this..its pretty simple..same logic..but used two parameters..to remove that confusing *head_ref = rest; and first->next->next

http://ideone.com/fBn1MT..

1 • Reply • Share ›

Amit • 2 months ago

first->next->next = first;
This is confusing. Most of use here are new to these concepts.

first->next->next = first;

==

struct node *temp = first->next;
temp->next = first;

• Reply • Share ›

Gautham Kumaran • 2 months ago

java code for the iterative method

https://github.com/gautham20/g...

1 • Reply • Share ›

Sai Vamshi • 2 months ago

Some One please explain Me how the space complexity in recursive type reverse function is o(n) ??

• Reply • Share ›

V_CODER → Sai Vamshi • 2 months ago

Its O(1) only !!

1 • Reply • Share ›

Sai Vamshi → V_CODER • 2 months ago

Thanks..

• Reply • Share ›

Holden → Sai Vamshi • 2 months ago

If you consider recursion stack space, it is not O(1)

• Reply • Share ›

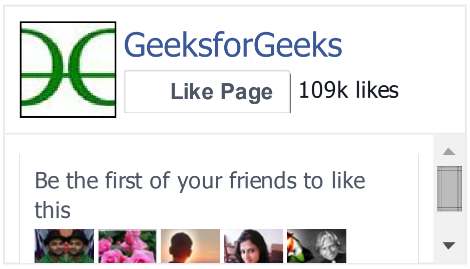
samsammy → V_CODER • 2 months ago

It's not O(1) , If you consider recursion stack space also.

2 • Reply • Share ›

Mr. Lazy • 2 months ago

Very simple logic to reverse linked list recursively. Check this code below, Also it is tail recursive and hence more efficient and compiler optimized than other non tail recursive codes .. :)



Popular Posts

- All permutations of a given string
- Memory Layout of C Programs
- Understanding “extern” keyword in C
- Median of two sorted arrays
- Topological Sorting
- 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

- Interview Experiences
- Advanced Data Structures
- Dynamic Programming
- Greedy Algorithms
- Backtracking
- Pattern Searching
- Divide & Conquer
- Mathematical Algorithms
- Recursion
- Geometric Algorithms
- Common Interview Puzzles

ADVERTISE WITH US

Recent Comments

- Anusha Hegde

  this works only if the list is sorted

  Remove duplicates from an unsorted linked list · 1 minute ago

- Ravi Aggarwal

  Another method using Morris Traversal...

  Two nodes of a BST are swapped, correct the BST · 11 minutes ago

- code down

  i see.

  Check if each internal node of a BST has exactly one child · 17 minutes ago

- code down

  @geeksforgeeks In approach 2, multiplication of...

  Check if each internal node of a BST has exactly one child · 19 minutes ago

- KochwaSchool

  there cannot be more than two consecutive...

  Find the element that odd number of times in O(Log n) time · 22 minutes ago

- Mysterious Mind

  Profile, CTC and how did you apply?

  Fiberlink (maas360) Interview Experience | Set 4 (Off-Campus) · 34 minutes ago

Follow @GeeksforGeeks

Tags

Adobe Advance Data Structures Advanced Data Structures Amazon array Backtracking Bit Magic C++ Cisco CN c puzzle D-E-Shaw DBMS Divide and Conquer Dynamic Programming Flipkart GATE GATE-CS-2006 GATE-CS-2008 GATE-CS-2011 GATE-CS-2012 GATE-CS-C-Language GATE-CS-DS-&-Algo GATE-CS-Older GFacts Goldman Sachs Google Graph Greedy Algorithm Hashing Interview Experience Java MathematicalAlgo Microsoft Morgan Stanley Operating systems Oracle Output of CPP Program Pattern Searching puzzle Recursion SAP Labs SnapDeal stack Tree Traveral

- **GeeksQuiz**
- **GeeksforGeeksIDE**
- **Data Structures**
- **Algorithms**
- **C Programming**
- **C++ Programming**
- **Java Programming**
- **Books**
- **Interview Experiences**
- **GATE CS**
- **GATE CS Forum**
- **Android App**