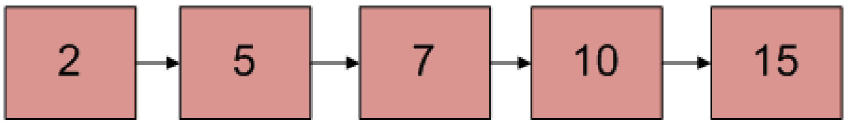
Given a linked list which is sorted, how will you insert in sorted way

Algorithm:

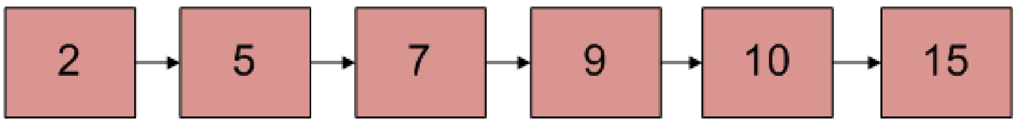
Let input linked list is sorted in increasing order.

```
1) If Linked list is empty then make the node as head and return it.
2) If value of the node to be inserted is smaller than value of head node
    then insert the node at start and make it head.
3) In a loop, find the appropriate node after which the input node (let 9) is
    to be inserted. To find the appropriate node start from head, keep moving
    until you reach a node GN (10 in the below diagram) who's value is
    greater than the input node. The node just before GN is the appropriate
    node (7).
4) Insert the node (9) after the appropriate node (7) found in step 3.
```

Initial Linked List



Linked List after insertion of 9



Implementation:

C/C++

```
/* Program to insert in a sorted list */
#include<stdio.h>
#include<stdlib.h>

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

/* function to insert a new_node in a list. Note that this
  function expects a pointer to head_ref as this can modify the
  head of the input linked list (similar to push())*/
void sortedInsert(struct node** head_ref, struct node* new_node)
{
    struct node* current;
    /* Special case for the head end */
    if (*head_ref == NULL || (*head_ref)->data >= new_node->data)
    {
        new_node->next = *head_ref;
        *head_ref = new_node;
    }
    else
    {
        /* Locate the node before the point of insertion */
        current = *head_ref;
        while (current->next!=NULL &&
               current->next->data < new_node->data)
        {
            current = current->next;
        }
        new_node->next = current->next;
        current->next = new_node;
    }
}

/* BELOW FUNCTIONS ARE JUST UTILITY TO TEST sortedInsert */

/* A utility function to create a new node */
struct node *newNode(int new_data)
{
    /* allocate node */
    struct node* new_node =
        (struct node*) malloc(sizeof(struct node));

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

    return 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 count function*/
int main()
{
    /* Start with the empty list */
    struct node* head = NULL;
    struct node *new_node = newNode(5);
    sortedInsert(&head, new_node);
    new_node = newNode(10);
    sortedInsert(&head, new_node);
    new_node = newNode(7);
    sortedInsert(&head, new_node);
    new_node = newNode(3);
    sortedInsert(&head, new_node);
    new_node = newNode(1);
    sortedInsert(&head, new_node);
    new_node = newNode(9);
    sortedInsert(&head, new_node);
    printf("\n Created Linked List\n");
    printList(head);

    return 0;
}
```

Run on IDE

Java

```
// Java Program to insert in a sorted list
class LinkedList
{
    Node head;  // head of list

    /* Linked list Node*/
    class Node
    {
        int data;
        Node next;
        Node(int d) {data = d; next = null; }
    }

    /* function to insert a new_node in a list. */
    void sortedInsert(Node new_node)
    {
         Node current;

         /* Special case for head node */
         if (head == null || head.data >= new_node.data)
         {
            new_node.next = head;
            head = new_node;
         }
         else {

            /* Locate the node before point of insertion. */
            current = head;

            while (current.next != null &&
                   current.next.data < new_node.data)
                  current = current.next;

            new_node.next = current.next;
            current.next = new_node;
         }
     }

                  /*Utility functions*/

    /* Function to create a node */
    Node newNode(int data)
    {
       Node x = new Node(data);
       return x;
    }

     /* Function to print linked list */
     void printList()
     {
         Node temp = head;
         while (temp != null)
         {
            System.out.print(temp.data+" ");
            temp = temp.next;
         }
     }

     /* Drier function to test above methods */
     public static void main(String args[])
     {
         LinkedList llist = new LinkedList();
         Node new_node;
         new_node = llist.newNode(5);
         llist.sortedInsert(new_node);
         new_node = llist.newNode(10);
         llist.sortedInsert(new_node);
         new_node = llist.newNode(7);
         llist.sortedInsert(new_node);
         new_node = llist.newNode(3);
         llist.sortedInsert(new_node);
         new_node = llist.newNode(1);
         llist.sortedInsert(new_node);
         new_node = llist.newNode(9);
         llist.sortedInsert(new_node);
         System.out.println("Created Linked List");
         llist.printList();
     }
}
/* This code is contributed by Rajat Mishra */
```

Run on IDE

Python

```
# Python program to insert in sorted list

# Node class
class Node:

    # Constructor to initialize the node object
    def __init__(self, data):
        self.data = data
        self.next = None

class LinkedList:

    # Function to initialize head
    def __init__(self):
        self.head = None

    def sortedInsert(self, new_node):

        # Special case for the empty linked list
        if self.head is None:
            new_node.next = self.head
            self.head = new_node

        # Special case for head at end
        elif self.head.data >= new_node.data:
            new_node.next = self.head
            self.head = new_node

        else :

            # Locate the node before the point of insertion
            current = self.head
            while(current.next is not None and
                 current.next.data < new_node.data):
                current = current.next
            new_node.next = current.next
            current.next = new_node

    # Function to insert a new node at the beginning
    def push(self, new_data):
        new_node = Node(new_data)
        new_node.next = self.head
        self.head = new_node

    # Utility function to prit the linked LinkedList
    def printList(self):
        temp = self.head
        while(temp):
            print temp.data,
            temp = temp.next


# Driver program
llist = LinkedList()
new_node = Node(5)
llist.sortedInsert(new_node)
new_node = Node(10)
llist.sortedInsert(new_node)
new_node = Node(7)
llist.sortedInsert(new_node)
new_node = Node(3)
llist.sortedInsert(new_node)
new_node = Node(1)
llist.sortedInsert(new_node)
new_node = Node(9)
llist.sortedInsert(new_node)
print "Create Linked List"
llist.printList()

# This code is contributed by Nikhil Kumar Singh(nickzuck_007)
```

Run on IDE

Output:

```
Created Linked List
1 3 5 7 9 10
```

Shorter Implementation using double pointers

Thanks to Murat M Ozturk for providing this solution. Please see Murat M Ozturk's comment below for complete function. The code uses double pointer to keep track of the next pointer of the previous node (after which new node is being inserted).

Note that below line in code changes *current* to have address of next pointer in a node.

```
   current = &((*current)->next);
```

Also, note below comments.

```
    /* Copies the value-at-address current to
      new_node's next pointer*/
    new_node->next = *current;
    /* Fix next pointer of the node (using it's address)
       after which new_node is being inserted */
    *current = new_node;
```

Time Complexity: O(n)

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

FLIGHT CENTRE The Airfare Experts
LOWEST AIRFARE GUARANTEE
LOWEST AIRFARE GUARANTEE
We will beat available airfare quotes for international flights departing from New Zealand or you fly free!* Excludes domestic airfares. Conditions apply*
MORE INFO

Company Wise Coding Practice   Topic Wise Coding Practice

90 Comments Category: Linked Lists

Related Posts:

- Convert a Binary Tree to a Circular Doubly Link List
- Subtract Two Numbers represented as Linked Lists
- Rearrange a given list such that it consists of alternating minimum maximum elements
- Flatten a multi-level linked list | Set 2 (Depth wise)
- Decimal Equivalent of Binary Linked List
- Merge K sorted linked lists
- Check if a linked list is Circular Linked List
- Delete middle of linked list

(Login to Rate and Mark)

1.6 Average Difficulty : **1.6/5.0**
Based on **82** vote(s)

☐ Add to TODO List
☐ Mark as DONE

Writing code in comment? Please use code.geeksforgeeks.org, generate link and share the link here.

90 Comments **GeeksforGeeks** 1 **Login**

Recommend 10 **Share** **Sort by Newest**

Join the discussion…

zordan • 17 days ago

what is the difference b/w struct node**head_ref and struct node*head as in both the cases we are defining head node..........

Reply • Share ›

Shashank Rajput • 2 months ago

/* C++ code using constructors, destructor, new and delete.

-> Here i used "void InsertSorted(int);" function to add elements to empty linked list, link list with one element and link list with more than one element in the sorted order.

-> To search for element in the linked list greater than the given element and to insert, for loop is used >> for(pred = head, tmp = head->next; tmp != 0 && !(tmp->data >= el); pred = pred->next, tmp = tmp->next);

-> pred and tmp pointers are used to search for element and keep track of previous node. */

http://code.geeksforgeeks.org/...

Reply • Share ›

fatih tekin • 3 months ago

Solution can be optmized to log(n) time complexity by using black red tree or I have chosen java's treemap(which uses red black tree) in my implementation.
http://code.geeksforgeeks.org/...

1 • Reply • Share ›

fatih tekin • 3 months ago

import java.util.Map.Entry;
import java.util.TreeMap;

public class LinkedListSortedWay {

TreeMap<integer,node> treeMap = new TreeMap<integer,node>();

public LinkedListSortedWay() {}

public static void main(String[] args) {

LinkedListSortedWay linkedListSortedWay = new LinkedListSortedWay();

linkedListSortedWay.insert(5);
linkedListSortedWay.insert(10);
linkedListSortedWay.insert(7);
linkedListSortedWay.insert(3);
linkedListSortedWay.insert(1);
linkedListSortedWay.insert(9);

see more

1 ^ | v • Reply • Share ›

Arvind Tiwari • 9 months ago

insertion code using single pointer
http://code.geeksforgeeks.org/...

^ | v • Reply • Share ›

Bhavesh Upadhyay • 9 months ago

list* insert_sorted(list* head, int a){

list* tmp=head;

list* x=(list*)malloc(sizeof(list));

x->data=a;

x->next=NULL;

if(head->data>=a){

x->next=head;

head=x;

return head;

}

while(tmp->next){

see more

^ | v • Reply • Share ›

saiteja • 10 months ago

#incl...
.....
..
void meth(struct node * head,int value)

```
{
if(head=NULL) return;
else if(value<head->data){
node *temp=getnewnode(value); // return a struct node with giveen value
temp->next=head;
head=temp;
}
else{
while(curr->next->data<value) {="" curr="curr-">next;
}
node * temp1=getnewnode(value);
temp1->next=curr->next;
curr->next=temp1;
}
}
```

Reply • Share ›

tushal khandelwal • 10 months ago

```
void insert(int p)
{
struct node*temp1=head;
struct node*temp2=head;
int c=0;
struct node* temp=(struct node*)malloc(sizeof(struct node));

temp->data=p;
temp->link=NULL;

if(head==NULL)
{
head=temp;
return;
}
else
{
while(temp1!=NULL)
```

see more

Reply • Share ›

vijay • a year ago

```
#include<stdio.h>

#include<malloc.h>

#include<stdlib.h>

struct s{
```

struct s{

int num;

struct s *link;};

int main(){

char cont='y';

int move,numb,m,temp;

struct s *start,*link,*ptr,*new,*ptr1,*ptr2;

start=(struct s *)0;

see more

Reply • Share ›

Faheem Shah • a year ago

#include<iostream>

using namespace std;

struct Node

{

int data;

Node* Next;

};

class LinkList

{

Node *First;

public:

see more

Reply • Share ›

Faheem Shah Faheem Shah • a year ago

It is the Best Code Written By Me

Reply • Share ›

Jatin • a year ago

public void addAssending(int i) {

Node node = new Node();

```
node.setData(i);

if (HEAD == null) { // First time add
HEAD = node;
} else if (i < HEAD.getData()) { // New item is less then HEAD
node.setNext(HEAD);
HEAD = node;
} else { // Other cases
Node tmpNode = HEAD;
while (tmpNode.getNext() != null) {
if (tmpNode.getNext().getData() > i) {
break;
}
tmpNode = tmpNode.getNext();
}
node.setNext(tmpNode.getNext());
tmpNode.setNext(node);
}

}
```

^ | v • Reply • Share ›

मनोज पाटीदार • a year ago

why it uses node** head_ref. why not just node* head_ref? is there any reason for using pointer to pointer in case of head..

^ | v • Reply • Share ›

atul_gavle ➦ मनोज पाटीदार • a year ago

whenever there is a possibility of "head pointer being changed" in another function,then we should pass address of the variable so that we can view the changes made in head pointer"in our main()".since here head itself is a pointer ,so
we should receive it in a pointer to pointer i.e. * *head_pointer.if head could simply be a variable then we would have received it in a simply *head. :-)

2 ^ | v • Reply • Share ›

Akilan Arasu ➦ मनोज पाटीदार • a year ago

head pointer is being modified in the function. That's why we use node** head_ref and not node* head_ref.
node** head_ref ----> passes the address of head pointer.
node* head_ref ----> passes the address stored in head pointer. This will be the address of a node in the linked list.

^ | v • Reply • Share ›

मनोज पाटीदार ➦ Akilan Arasu • a year ago

I got it thanks :)

• Reply • Share ›

Nikhil Agarwal • a year ago

here is a recursive solution:

```
#include <stdio.h>

#include <stdlib.h>

struct node

{

int data;

struct node *next;

};

struct node* sortedInsert(struct node *head, int key)

{

struct node *temp = newnode(key);
```

see more

• Reply • Share ›

Rajneesh Kumar • a year ago

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

struct node
{
int data;
struct node *next;
};
void sortedInsert(struct node **head,int key)
{
struct node *NewNode,*q,*p=*head;
NewNode=(struct node*)malloc(sizeof(struct node));
NewNode->data=key;

if(*head==NULL||key<p->data)
{
NewNode->next=*head;
*head=NewNode;
```

see more

• Reply • Share ›

shivangi goel • a year ago

why the push function doesnt work here.its giving the wrong output.

• Reply • Share ›

Vikram singh shivangi goel • a year ago

push function is work here .see this

```
#include <stdio.h>
#include<stdlib.h>
struct node
{
int data;
struct node *next;
};
void push(struct node**head,int new_data)
{
struct node* new_node=(struct node*)malloc(sizeof(struct node));
new_node -> data = new_data;
new_node ->next= *head;
*head=new_node;
}
// this function insert a new node in sorted order
void inserts(struct node**head,int x)
```

see more

• Reply • Share ›

ravi • a year ago

```
//what's wrong with this
//simply using the concept of previous node
void sortedinsert(struct node ** head_ref, node * new_node)

{

struct node * current;

struct node * prev;

if(*head_ref==NULL or new_node->data <= (*head_ref)->data)

{

new_node->link=*head_ref;

*head_ref=new_node;

}
```

see more

^ | ˅ • Reply • Share ›

Rohit Saluja ➦ ravi • a year ago

In your code after finding the position where the node is to be inserted you are linking the node incorrectly

instead of
new_node->link=prev->link;
prev->link=new_node;

the linking of the new_node should first be done to the current and then to the previous node
The correct way to do it is
new_node->link=current;
prev->link=new_node;

Hope this helps

^ | ˅ • Reply • Share ›

coolk • a year ago

Java approach
https://coderpad.io/DRGEAEWP

^ | ˅ • Reply • Share ›

Gautham Kumaran • a year ago

Java code

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

^ | ˅ • Reply • Share ›

AkankshaT94 • a year ago

A RECURSIVE APPROACH: Plz let me know abt its time and space complexity relative to the given answers.

```
void rec_ins(struct node *start,struct node *newptr)
{
if(start->next==NULL || (start->next->data)>=(newptr->data))
{ newptr->next=start->next;
start->next=newptr;
return;
}
rec_ins(start->next,newptr);
}

void sortedInsert(struct node **href,struct node *newptr)
```

```
{
if((*href)==NULL || (*href)->data>=newptr->data)
{ newptr->next=(*href);
(*href)=newptr;
return;
}
rec_ins(*href,newptr);
}
```

Reply • Share ›

coderzz AkankshaT94 • a year ago

Well, I just read this article :: http://www.geeksforgeeks.org/g...
And, it seems that the space complexity is O(n) anyway, for all the algorithms, and the auxiliary space used by your code is again O(n).

1 • Reply • Share ›

coderzz AkankshaT94 • a year ago

Time complexity seems O(n) to me. And considering the fact that it is a recursive solution, so you are actually using the stack space, so Space complexity might be O(n) as well, given we consider stack space while computing space complexity.

Reply • Share ›

Hitesh Saini • a year ago

we can implement this in O(n/2) by moving our pointer by 2 position and if we found a bigger node then we compare our previous node's(just before our current node) value as well if it is also greater then we will store it after the previous node otherwise we will store it just before our current node

Reply • Share ›

Amit Hitesh Saini • a year ago

O(n/2) is actually O(n)

Reply • Share ›

Hitesh Saini Amit • a year ago

everyone knows that O(n/2) is O(n).

But the approach i used is definitely a better approach.

Reply • Share ›

lucy • a year ago

i think it will take O(n^2) if insert elements come in increasing order

Reply • Share ›

Rajat__Gupta • 2 years ago

can smbody plz help me to find the error

https://ideone.com/KcrkUe

Reply • Share ›

yokila arora Rajat__Gupta • a year ago

There are 2 errors that I found. Line 23, should be if(*start==NULL ||...)
Line 70, ptr = newnode(num);
Line 71, insert(&head, ptr);

Reply • Share ›

Praveen • 2 years ago

can anyone plz tell me how to insert a new element in singly linked list (which is not sorted) in ascending order

Reply • Share ›

Rohit Saluja Praveen • a year ago

First clear the basic of the linked list. It seems you don't know the alphabets and but you want to write a passage :|

Reply • Share ›

c0der_32 Praveen • a year ago

why are you even on g4g?

Reply • Share ›

V_CODER Praveen • a year ago

Its like saying i want to meet the richest man on globe who dont have money !!!!!

Reply • Share ›

jaysurya_j Praveen • a year ago

where do you want to insert?

Reply • Share ›

Prince Vijay Pratap • 2 years ago

https://ideone.com/SeUX2k

2 Reply • Share ›

Pawan Dwivedi • 2 years ago

http://ideone.com/wMWWM5

1 Reply • Share ›

Vaibhav Sharma • 2 years ago

cn anyone plz tell me whats the problem with the following function to insert the newnode .Even though it looks similar to the one given in the method 1 but its not

running in DevC++.
void insert(struct node **headr, int num)

{

struct node *newn;

newn=(struct node *)malloc(sizeof(struct node *));

newn->data=num;

struct node *p;

if(*headr == NULL || (*headr)->data >= newn->data)

{

newn->next = *headr;

see more

Reply • Share ›

Guest Vaibhav Sharma • 2 years ago
newn=(struct node *)malloc(sizeof(struct node *));
The above statement is buggy, it must be:
newn=(struct node *)malloc(sizeof(struct node));
Reply • Share ›

Rahul Magdum • 2 years ago
Go on traversing till you get larger number.. That is 10..
Current will point to 10.
You are sure that prev node was less than 9 and current node is greater than 9.
Then create new node, insert it ahead of 10, copy 10 into it.
and change current nodes value to 9
1 • Reply • Share ›

shashsriv93 • 2 years ago
public void insertSorted(int data){

SListNode current=head;

SListNode newnode=new SListNode(data);

if(current==null || newnode.item<=current.item){

newnode.next=current;

current=newnode;

return;

}

else{

current=head;

while(current.next!=null && current.next.item<newnode.item) current="current.next;" }="" newnode.next="current.next;" current.next="newnode;" }="">

^ | ⌄ • Reply • Share ›

tintin • 2 years ago

Works fine. The code is for java

public void insert(int data) {

Node newNode = new Node(data);

Node prev = null;

if (head == null || newNode.data < head.data)

{

newNode.next = head;

head = newNode;

return;

}

Node current = head.next;

see more

1 ^ | ⌄ • Reply • Share ›

Aman jain • 2 years ago

insertion can be done in log(n) time in sorted list:

you can see code here http://pastebin.com/SxsFfqGa

^ | ⌄ • Reply • Share ›

d07RiV ➦ Aman jain • 2 years ago

Your algorithm is still O(n), because it needs O(length) time to find the middle of a list. n+n/2+n/4+... = 2n. The only improvement is that it performs O(log n) comparisons, but if those are the bottle neck, perhaps you should consider using a different data structure.

1 ^ | ⌄ • Reply • Share ›

Manraj Singh ➦ d07RiV • a year ago

Manraj Singh ↗ d07RiV • a year ago

Skip list can be used.

^ | ⌄ • Reply • Share ›

d07RiV ↗ Manraj Singh • a year ago

He linked specific code, which uses plain single linked lists. If you actually need to insert in log(n) time, you can simply use std::set or similar.

^ | ⌄ • Reply • Share ›

mb1994 • 2 years ago

This should be a full working code:

```
node* insertSorted(node* head, int data)
{
if(head==NULL||head->data>data)
{
node* newnode=(node*)malloc(sizeof(node));
newnode->data=data;
newnode->link=head;

return head;
}

node* current=head,next=head->link;
while(next && next->data<data) {="" current="next;" next="next-">link;
}
newnode->link=current->link;
current->link=newnode;

return head;
}
```

^ | ⌄ • Reply • Share ›

Load more comments