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

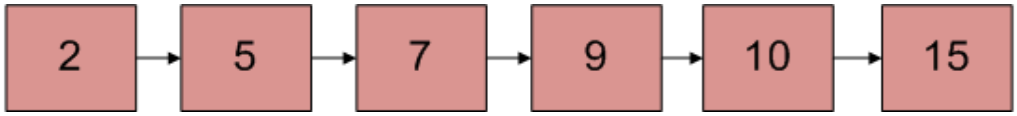
Algorithm:
Let input linked list is sorted in increasing order.

```
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:

```
/* 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);

    getchar();
    return 0;
}
```

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.

```
new_node->next = *current; /* Copies the value-at-address current to new_node

*current = new_node;  /* Fix next pointer of the node (using it's address) af
```

Time Complexity: O(n)

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

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

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

Praveen • 11 days ago

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

Prince Vijay Pratap • a month ago

https://ideone.com/SeUX2k

1 • Reply • Share ›

Pawan Dwivedi • 2 months ago

http://ideone.com/wMWWM5

1 • Reply • Share ›

Vaibhav Sharma • 3 months 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 • 21 days 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 • 4 months 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

• Reply • Share ›

shashsriv93 • 6 months 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 • 6 months 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 • 6 months 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 • 5 months 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 ›

mb1994 • 9 months 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 ›

Dipankar Jana • 9 months ago

This should work. But it fails if the value to be inserted is less than the value of the head node.

```
Node *InitList(int key)

{
```

```
Node *newNode = new Node();

newNode->data = key;

newNode->next = NULL;

return newNode;

}

Node *SortedInsert(Node *head, int key){

if(head == NULL) return InitList(key);

Node *temp = head;
```

see more

• Reply • Share ›

prasun_goyal • 9 months ago

a more simple approach

```
// SORTED LINKED LIST

// ELEMENTS GET INSERTD IN WAY THAT

// LIST REMAINS SORTED

#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node* next;

};
```

see more

1 • Reply • Share ›

Jynxta • 9 months ago

Sorted Insert done recursively. Remaining functionality remained the same in this example (although I also opted to just do the node creation and passing by ref at the same time (sortedInsert(&head, NewNode(n)); as I found it more readable.

(sortedInsert(&head, newNode(4)); as I found it more readable.

Note (and reason for edit): I tested this only against the main function provided so... feel free to fix it up if you find faults :)

```
//function to insert a new_node in list recursively
void sortedInsert(struct node **head_ref, struct node* new_node)
{
if ((*head_ref == NULL) || (new_node->data <= (*head_ref)->data))
{
new_node->next = *head_ref;
*head_ref = new_node;
return;
}

sortedInsert(&((*head_ref)->next), new_node);

}
```

• Reply • Share ›

ashish jaiswal ➚ Jynxta • 3 months ago

this is cool...!!

• Reply • Share ›

Khatri • 10 months ago

@geeksforgeeks
Can you please explain how Murat M Ozturk's solution is simplified ? Is it faster or In what sense it is better than solution provided above this?

6 • Reply • Share ›

RK ➚ Khatri • 7 months ago

The algorithm is basically the same. Its just that his solution is shorter.

• Reply • Share ›

ravi m • a year ago

```
#include<stdio.h>

#define s sizeof(int)

void main()

{

int i = -1;

// printf("%d", sizeof(short int));
```

if(i < s)

printf("t");

else

printf("f");

}

// answer is f how it is possible to get out put f, condition is if(-1 < sizeof(int)) it is true, but i got it is false..please give me solution...

• Reply • Share ›

mareen ravi m • a year ago

"f" gets printed because "sizeof()" returns unsigned value when it is compared to signed integer "-1" sign bit of "i" is high and of "s" is low ,so "i" is greater

1 • Reply • Share ›

mareen • a year ago

*current = new_node;
i do not get this line . how can it make the node's (before current) next point to new node ??

1 • Reply • Share ›

AMIT JAMBOTKAR • a year ago

IMPLEMENTED IN JAVA GENERIC WAY:

public class LinkedList<e extends="" number=""> implements Cloneable{

Node<e> head = null;

//Adding at the End

class Node<t extends="" number=""> {

T value;

Node<t> nextReference;

public Node(T value) {

this.value = value;

this.nextReference = null;

}

see more

• Reply • Share ›

Dark Protocol • a year ago

For Larger List size (n>10000000), Skip list is more appropriate

• Reply • Share ›

Himanshu Dagar • a year ago

can refer to below code

http://ideone.com/R5rl9g

1
• Reply • Share ›

Daniel YIn • a year ago

```
node * sortedInsert(node * n, int d){
if (n == NULL || n->data >d) return new node(d,n);
else if (n->data == d) return n;
else {
n->next = sortedInsert(n->next,d);
return n;
}
}
```


• Reply • Share ›

mahi2 • a year ago

This problem can be solved if we maintain 2 pointers...and move one pointer (tmp2) ahead of the other (tmp1) in the loop..and compare the value of the node to be inserted with the data value of tmp1 and tmp2. At one point data (node)> data(tmp1) and data(node) < data(tmp2).. insert the node at that point!

• Reply • Share ›

Xristos Mpalis • 2 years ago

I want this code in java, please.

• Reply • Share ›

AMIT JAMBOTKAR ➚ Xristos Mpalis • a year ago

```
public class LinkedList<e extends="" number=""> implements Cloneable{

    Node<e> head = null;

    //Adding at the End

    class Node<t extends="" number=""> {

    T value;

    Node<t> nextReference;
```

```
Node<t> nextReference;

public Node(T value) {

this.value = value;

this.nextReference = null;

}

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

see more

• Reply • Share ›

nitin • 2 years ago

```
#include
#include
struct node
{
int data;
struct node *link;
};
void insert1(struct node **p,int data)
{
struct node *temp,*t,*s;
temp=(struct node *)malloc(sizeof(struct node));
temp->data=data;
temp->link=NULL;
if((*p)==NULL)
{
*p=temp;
}
else
```

see more

• Reply • Share ›

Chuantao Zang • 2 years ago

This does not work if the node is the largest, you should add sereral lines more as follows.
/* Locate the node before the point of insertion */.

current = *head_ref;.

while (current->next!=NULL && current->next->data < new_node->data).

{.

current = current->next;.

}.

if(current->next!=NULL).

current0>next=new_node; //add to tail.
else.

{.

new_node->next = current->next;.

current->next = new_node;.

}.

• Reply • Share ›

Amit Kumar • 2 years ago

thats is what we all do...
if you are asking for insertion before a node then For that you can keep track of previous node or use linked_list_pointer->next->value to compare with..

• Reply • Share ›

Pallavee Gogoi • 2 years ago

insert into linked list after a given node.

• Reply • Share ›

Hina Jain • 2 years ago

@Murat M- I think your solutionn wont work when the node to be inserted turns out to be the largest...I have added a few checks for this condition....comments would be welcomed...

void sortedInsert(struct node** head_ref, struct node* new_node).
{
if (head_ref == NULL).
{.

return;.
}.

/* Locate the node before the point of insertion or if last node is reached we stop at last node */.

struct node** current = head_ref;.

while ((*current)->next!=NULL && (*current)->data < data).

{.

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

see more

• Reply • Share ›

ff • 5 years ago

hi ... please i want sorted with with only int

this funnction: sortedinsert(int)

• Reply • Share ›

Shekhu ff • 5 years ago

can you please explain your requirement with an example?

• Reply • Share ›

GeeksforGeeks • 5 years ago

@Murat M Ozturk: Thanks for the short and nice solution. We have added the solution to the original post.

5 • Reply • Share ›

rikitic GeeksforGeeks • 2 years ago

it can be done in less time by using binary search on linked list....correct me if i am wrong

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

• Reply • Share ›

GeeksforGeeks rikitic • 2 years ago

Binary Search can not be applied on Linked Lists. That is why we have skip lists (http://www.geeksforgeeks.org/s...

1 • Reply • Share ›

rikitic GeeksforGeeks • 2 years ago

its almost binary search

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

• Reply • Share ›

Murat M Ozturk • 5 years ago

Here is a simplified version of the sortedInsert() method:

```
 void sortedInsert(struct node** head_ref, struct node* new_node)
{
  if (head_ref == NULL)
  {
          return;
  }

        /* Locate the node before the point of insertion */
        struct node** current = head_ref;
        while (*current !=NULL && (*current)->data < data)
        {
          current = &((*current)->next);
        }

        new_node->next = *current;
        *current = new_node;
}
```

11 • Reply • Share ›

vipinkaushal ➚ Murat M Ozturk • 9 months ago

i think this function should return the header node

because if data is inserted at front then header will be changed
please correct if i'm wrong

2 • Reply • Share ›

hina ➚ Murat M Ozturk • 2 years ago

I think this wont work when the node to be inserted turns out to be the largest...I have added a few checks for this condition....Correct me if I am wrong...

• Reply • Share ›

hina ➚ hina • 2 years ago

Code with all the checks:
Correct me if i am wrong

void sortedInsert(struct node** head_ref, struct node* new_node).
{
/*if LL is empty */
if (head_ref == NULL)
{
*head_ref = new_node;
}

/* Locate the node before the point of insertion or if last node is reached we stop

/* Locate the node before the point of insertion or if last node is reached we stop at last node */.
struct node** current = head_ref;

//if new node is to be inserted at first position
if((*current)->data > new_node ->data)
{
new_node -> next = *current;

see more

• Reply • Share ›

olra ➚ Murat M Ozturk • 3 years ago

```
 /*
checking: if (head_ref == NULL) is included in while loop
so the code is :
*/
void sortedInsert(struct node** head_ref, struct node* new_node)
{
    /* Locate the node before the point of insertion */
    struct node** current = head_ref;
    while (*current !=NULL && (*current)->data < data)
    {
      current = &((*current)->next);
    }

    new_node->next = *current;
    *current = new_node;
}
```

1 • Reply • Share ›

Viky ➚ olra • 2 years ago

The second method of double pointer doesn't work for all cases ..

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

• Reply • Share ›

GeeksforGeeks ➚ Viky • 2 years ago

Could you please let us know the case for which it doesn't work?

• Reply • Share ›

Viky ➚ GeeksforGeeks • 2 years ago

If the list is empty, we should make head as the new node. But in this

case it returns NULL.

Also, Adding element to the end of the list doesn't work

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

• Reply • Share ›

Bunty Viky • a year ago

#include<stdio.h>

#include<conio.h>

struct node

{

int data;

struct node *next;

};

void printList(struct node *n)

{

while(n!=NULL)

{

see more

2 • Reply • Share ›

bunty Bunty • a year ago

neglect the </conio.h></stdio.h> at the end

• Reply • Share ›

-
  - 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

- Goku

  They are considering 0 based indexing instead...

  Write a function to get Nth node in a Linked List · 8 minutes ago

- lebron

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

  K’th Smallest/Largest Element in Unsorted Array | Set 3 (Worst Case Linear Time) · 4 hours ago

- lebron

  merge sort

  K’th Smallest/Largest Element in Unsorted Array | Set 3 (Worst Case Linear Time) · 4 hours ago

- Shubham Sharma

  You saved my time :)

  Searching for Patterns | Set 2 (KMP Algorithm) · 4 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 · 6 hours ago