GeeksforGeeks

A computer science portal for geeks

GeeksQuiz

- Home
- Algorithms
- DS
- GATE
- Interview Corner
- Q&A
- C
- C++
- Java
- Books
- Contribute
- Ask a Q
- About

Array
Bit Magic
C/C++
Articles
GFacts
Linked List
MCQ
Misc
Output
String
Tree
Graph

Delete nodes which have a greater value on right side

Given a singly linked list, remove all the nodes which have a greater value on right side.

Examples:
a) The list 12->15->10->11->5->6->2->3->NULL should be changed to 15->11->6->3->NULL. Note that 12, 10, 5 and 2 have been deleted because there is a greater value on the right side.

When we examine 12, we see that after 12 there is one node with value greater than 12 (i.e. 15), so we delete 12.
When we examine 15, we find no node after 15 that has value greater than 15 so we keep this node.
When we go like this, we get 15->6->3

b) The list 10->20->30->40->50->60->NULL should be changed to 60->NULL. Note that 10, 20, 30, 40

and 50 have been deleted because they all have a greater value on the right side.

c) The list 60->50->40->30->20->10->NULL should not be changed.

Method 1 (Simple)
Use two loops. In the outer loop, pick nodes of the linked list one by one. In the inner loop, check if there exist a node whose value is greater than the picked node. If there exists a node whose value is greater, then delete the picked node.
Time Complexity: O(n^2)

Method 2 (Use Reverse)
Thanks to Paras for providing the below algorithm.
1. Reverse the list.
2. Traverse the reversed list. Keep max till now. If next node < max, then delete the next node, otherwise max = next node.
3. Reverse the list again to retain the original order.

Time Complexity: O(n)

Thanks to R.Srinivasan for providing below code.

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

/* structure of a linked list node */
struct node
{
     int data;
     struct node *next;
};

/* prototype for utility functions */
void reverseList(struct node **headref);
void _delLesserNodes(struct node *head);

/* Deletes nodes which have a node with greater value node
  on left side */
void delLesserNodes(struct node **head_ref)
{
    /* 1) Reverse the linked list */
    reverseList(head_ref);

    /* 2) In the reversed list, delete nodes which have a node
       with greater value node on left side. Note that head
       node is never deleted because it is the leftmost node.*/
    _delLesserNodes(*head_ref);

    /* 3) Reverse the linked list again to retain the
       original order */
    reverseList(head_ref);
}

/* Deletes nodes which have greater value node(s) on left side */
```

```
void _delLesserNodes(struct node *head)
{
     struct node *current = head;

     /* Initialize max */
     struct node *maxnode = head;
     struct node *temp;

     while (current != NULL && current->next != NULL)
     {
         /* If current is smaller than max, then delete current */
         if(current->next->data < maxnode->data)
         {
             temp = current->next;
             current->next = temp->next;
             free(temp);
         }

         /* If current is greater than max, then update max and
            move current */
         else
         {
             current = current->next;
             maxnode = current;
         }

     }
}

/* Utility function to insert a node at the begining */
void push(struct node **head_ref, int new_data)
{
     struct node *new_node =
               (struct node *)malloc(sizeof(struct node));
     new_node->data = new_data;
     new_node->next = *head_ref;
     *head_ref = new_node;
}

/* Utility function to reverse a linked list */
void reverseList(struct node **headref)
{
     struct node *current = *headref;
     struct node *prev = NULL;
     struct node *next;
     while(current != NULL)
     {
          next = current->next;
          current->next = prev;
          prev = current;
          current = next;
     }
     *headref = prev;
```

```
}

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

/* Driver program to test above functions */
int main()
{
    struct node *head = NULL;

    /* Create following linked list
      12->15->10->11->5->6->2->3 */
    push(&head,3);
    push(&head,2);
    push(&head,6);
    push(&head,5);
    push(&head,11);
    push(&head,10);
    push(&head,15);
    push(&head,12);

    printf("Given Linked List: ");
    printList(head);

    delLesserNodes(&head);

    printf("\nModified Linked List: ");
    printList(head);

    getchar();
    return 0;
}
```

Output:

```
Given Linked List: 12 15 10 11 5 6 2 3
Modified Linked List: 15 11 6 3
```

Source:
http://geeksforgeeks.org/forum/topic/amazon-interview-question-for-software-engineerdeveloper-about-linked-lists-6

Please write comments if you find the above code/algorithm incorrect, or find other ways to solve the same problem.

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.

DD • 15 days ago

```
void delLesserNodes(struct node *head)
{
struct node *curr = head;
struct node *temp = NULL;
while((curr) && (curr->next))
{
if(curr->data < curr->next->data)
{
curr->data = curr->next->data;
temp = curr->next;
curr->next = curr->next->next;
free(temp);
}
else
{
curr = curr->next;
}
}
}
```


Umesh Lohani • a month ago

Java code

```
public void deleteSmallRightNodes() {

Node current = head;

while (current != null) {

if (current.getValue() < current.getNext().getValue()) {

current.setValue(current.getNext().getValue());

current.setNext(current.getNext().getNext());

current = current.getNext();

} else {

current = current.getNext();

}

}

}
```

• Reply • Share ›

surbhijain93 • 2 months ago

My code

https://ideone.com/kl6QYo

• Reply • Share ›

SS • 2 months ago

Hello geeksforgeeks,
Do we require the variable "maxnode", current always points to max node so far, isn't it?
-Thanks

• Reply • Share ›

Antonov • 2 months ago

Wont recursion make for a prettier piece of code?

• Reply • Share ›

Tejwinder ➦ Antonov • 2 months ago

Yes using recursion and a current maximum there will not be any need to reverse the linked list.

• Reply • Share ›

Guest • 2 months ago

Many people seem to have the got the question wrong. Here is what is required 'traverse the list and remove an element from it if there exist any element greater in value to its right (not just immediate right) through the end'. Try with the following data input.

input : 14->7->9->18->11->15->10->12
output : 18->15->12

1 ^ | v • Reply • Share ›

Ibrahim Bayer • 3 months ago

I have solved that question by storing a stack.

//put bigger items in stack

Stack<node> stack = new Stack<>();

if (start == null){

return;

}

stack.push(start);

Node node = start.next;

// traverse all nodes

while(node != null){

//test all stack items

see more

^ | v • Reply • Share ›

lovey • 3 months ago

we can solve it without reversing it also.... by taking the reference from (http://www.geeksforgeeks.org/g...

here is the ideone link of the code: http://ideone.com/vfWa1o

^ | v • Reply • Share ›

Stack ➚ lovey • a month ago

the code is wrong it just checks with the next node not all the nodes in the list, say 12->15->10->11->16 then according to your logic 15 should not be deleted which is wrong.

1 ^ | v • Reply • Share ›

ic++ ➚ lovey • 2 months ago

very good!

• Reply • Share ›

Vishal Chandani • 3 months ago

```
#include <stdio.h>

#include <stdlib.h>

struct node

{

int data;

struct node *next;

};

struct node * DeleteRight(struct node *head)

{

struct node *current,*temp,*prev;

current = head;
```

see more

• Reply • Share ›

Naval • 3 months ago

Here id the code
http://ideone.com/bcAGLd

• Reply • Share ›

Naval • 3 months ago

solution without reversing list

```
#include<stdio.h>

#include<malloc.h>

struct node

{

int data;
```

```
struct node *next;

};

void append(struct node **,int );

void display(struct node *);

void delete(struct node **);
```

see more

• Reply • Share ›

Spm • 3 months ago

This can be done more simply.Please check this and reply back if there is any error or it can be made better and more efficient
"http://ideone.com/e.js/botOqa

• Reply • Share ›

Naval Spm • 3 months ago

this is wrong for 10->20-->30-->40-->50-->60-->70-->80--->NULL
according to your code list give us 20-->40-->60-->80-->NULL which is not correnct the list sholud be --80-->NULL

• Reply • Share ›

Abhishek • 4 months ago

Here it comes: THE POWER OF RECURSION :)

```
public void deleteNodesWithLargeInRight() {

deleteNodesWithLargeInRight(head, null);

}

private Node<t> deleteNodesWithLargeInRight(Node<t> current, Node<t> prev) {

if(current == null) {

return null;

}

Node<t> maxInRight = deleteNodesWithLargeInRight(current.getNext(), current);

if(maxInRight != null && maxInRight.getData().compareTo(current.getData()) > 0) {

if(current == head) {
```

see more

• Reply • Share ›

karunakar vankayala • 4 months ago

efficient approach:

```
struct node *alt(struct node *head)

{

struct node *max=head,*temp1=max,*temp=head->next;

while(temp!=NULL)

{

if(max->data > temp->data)

{

max->next=temp->next;

free(temp);

temp=max->next;
```

see more

• Reply • Share ›

Amita • 4 months ago

```
current = head;
prev = null;
while(current != null){
if(current->next->data > current->data){
t = current;
if( prev == NULL)
{
head = current->next;
}
else

{
prev->next = current->next;
}

current = current->next;
free(t);
}
```

```
else{
prev = current;
current = current->next;
}

}
```

1 • Reply • Share ›

Naval ➚ Amita • 3 months ago

first of all you have to read question carefully carefully your code give 20--40--60--80--NULL which is not correct answer should be according to question 80--NULL

• Reply • Share ›

Rajat Panjwani • 4 months ago

It can also be done as:(using two pointers)

```
current = head;
prev = null;
while(current != null){
if(current->next->data > current->data){
t = current;
prev->next = current->next;
current = current->next;
free(t);
}
else{
prev = current;
current = current->next;
}

}
```

1 • Reply • Share ›

Amita ➚ Rajat Panjwani • 4 months ago

it will give error in the first turn when prev = NULL ;
and we want line 6 as - prev->next = current->next;

1 • Reply • Share ›

abhishek verma • 5 months ago

r_ptr == running pointer.

```
int removeNodeWithRightSideHighVal()
{
r_ptr=start;
node *d_tmp;
```

```
for (;r_ptr;)
{
if ( r_ptr==start || r_ptr == start->next )
{
if (r_ptr->next && r_ptr->data < ((node*)r_ptr->next)->data )
{
d_tmp=r_ptr;
if (r_ptr == start)
start=start->next;
else
start->next=r_ptr->next;
free(d_tmp);
```

see more

• Reply • Share ›

yss • 5 months ago

why do we need to reverse the list ? instead we can keep track of smallest element, if the next element is greater than smallest one remove the smallest and update the smallest to the compared node(as the nodes before smallest node would be greater in value and there wont be a smallest to right side as we will remove smallest and update the new smallest) ??

• Reply • Share ›

Swati • 6 months ago

the language of question suggests that we are looking for greater integer on the next node.

• Reply • Share ›

JP • 6 months ago

We can do it without reversing a list by using recursion..
1) recurse list till end.
2)In each recursive call if (max>val) then remove node.

Code is given below:

```
void leader(struct node **start, struct node *ptr, int *max){
if(ptr==NULL)
return;
leader(start,ptr->next,max);
if(ptr->data < *max)
free(ptr);
else{
*max=ptr->data;
ptr->next=*start;
*start=ptr;
```

```
}
}
```

see more

• Reply • Share ›

Mayank Vats • 6 months ago

How about this approach...
1) Sort the list in descending order
2) Then match node by node. If the node in the unsorted list is less then skip it. Also, keep a track of it for future reference. Else, continue to the next node in both.
3) Skip the node in the sorted list if it is present in the tracked nodes we've kept separately.

This is not the best approach but just checking out the logic.

• Reply • Share ›

RK- An Unproven Theorem ➚ Mayank Vats • 5 months ago

@Mayank Vats can you please explain the complexity of your code?

• Reply • Share ›

Mayank Vats ➚ RK- An Unproven Theorem • 5 months ago

O(nlogn).. I didn't say it's the best approach... I was just mentioning another approach...

• Reply • Share ›

RK- An Unproven Theorem ➚ Mayank Vats • 5 months ago

Yeah I know....but I'd like to practically implement your approach..

• Reply • Share ›

Guest • 6 months ago

this can be done using same logic as this question

http://www.geeksforgeeks.org/i...

here is the code

```
void delete_greater(struct node** head){

struct node* temp, *current = *head;

if(*head == NULL){

printf("Empty list \n");

return;
```

```
}

while(current->next != NULL){

temp = current->next;
```

see more

2 • Reply • Share ›

devakar verma Guest • 5 months ago

We can do just passing *head, as we are not changing it.

• Reply • Share ›

Sandeep Reddy • 6 months ago

in Method 2:
Two times you are reversing, so there 2*n + traversing one.
the complexity will be O(3*n) not O(n).

• Reply • Share ›

Shashikant • 6 months ago

http://ideone.com/6h3JIT

done recursively using two pointers.
Time Complexity O(n)

• Reply • Share ›

guest • 6 months ago

@GeeksforGeeks: If it is only for deleting if right is max, then a simple solution is as follows, It will give correct output for the inputs you have given in problem, From what i understood about the problem is, delete the present node if its immediate next node is bigger, for that I've coded in a simple way, Please let me know if you have comments

```
void deleteNextRightMax()
{
Node *curr=head->next;
Node *prev=head;
Node *temp;
while (curr!=NULL) {
if ( curr->value > prev->value) {
temp=curr;
prev->value=curr->value;
prev->next=curr->next;
curr=curr->next;
delete temp;
} else {
```

```
prev=prev->next;
curr=curr->next;
}
}
```

1 • Reply • Share ›

abhishek verma guest • 5 months ago

Your solution is wrong !!!!!

Problem is to delete the current node if (curr->value < next->value)

and you are deleting current when (curr->value > prev-value) .

That is kind of reverse what is mentioned in the problem..... Let me know if you think otherwise.....

• Reply • Share ›

sonu431 • 6 months ago

can anyone make me understand that why in this function in _delLesserNodes(*head_ref) we are taking single pointer to the head_ref;

```
void delLesserNodes(struct node **head_ref)
{
reverseList(head_ref);

_delLesserNodes(*head_ref);
reverseList(head_ref);

}
```

• Reply • Share ›

guest sonu431 • 6 months ago

since head is declared locally and we wanted to alter the linked list we have used double pointer, since we now the head we just passed to, so it is not necessary to use a double pointer there, If something is wrong please correct me

• Reply • Share ›

Sumit Kesarwani • 7 months ago

/*Can Any One Please Help Me Where I Am Doing Wrong...*/

```
private LinkListNode deleteNextNodeOfMaxNode_usingDoubleLoop(LinkListNode listNode)

{

try

{
```

{

LinkListNode tempNode = listNode;

LinkListNode tempNode1 = null;

LinkListNode tempNode11 = tempNode;

LinkListNode tempNode_1 = tempNode11;

while(tempNode11!=null&&tempNode11.getNextLink()!=null)

{

if(tempNode11.getNodeData()<tempnode11.getnextlink().getnodedata()) {="" linklistnode="" temp="tempNode11;" system.out.println(temp.getnodedata());="" tempnode11="tempNode11.getNextLink();" temp="null;" }="" else="" {="" tempnode11="tempNode11.getNextLink();" }="" }="" return="" tempnode_1;="" }="" catch(exception="" ex)="" {="" ex.printstacktrace();="" }="" return="" null;="" }="">

• Reply • Share ›

nk • 7 months ago

```
struct node* dlt_right_grt(struct node** head)
{ if(*head==NULL)
return NULL;

if((*head)->next!=NULL)
{
if((*head)->data < (*head)->next->data)
{
*head=(*head)->next;
}
(*head)->next= dlt_right_grt(&((*head)->next));
}
return (*head);
}
```

• Reply • Share ›

sonu431 ➚ nk • 6 months ago

ur code is working fine in the sense of printing the node.....but
u are not deleting the node as the question says.. so, though, it gives answers correct but it may cause danger to the memory....

• Reply • Share ›

sandeep nagendra • 7 months ago

void delete_less_nodes(struct Node** H)

```
{

struct Node* Temp = *H;
struct Node* N = NULL;
struct Node* prev = NULL;

while(Temp!=NULL && Temp->next!=NULL)
{
if(Temp->data < Temp->next->data)
{
N = Temp;
if(prev!=NULL)
prev->next = Temp->next;
if(*H == Temp)
{
*H = Temp->next;
prev = *H;
```

see more

• Reply • Share ›

Batman_ • 7 months ago

isn't this simple ?

```
deleteNode(node **root)

{

node *prevNode=NULL;

node *currNode=*root;

node *nextNode=(*root)->next;

while(nextNode!=NULL)

{

if(currNode->data<nextnode->data)

{

if(prevNode==NULL)
```

see more

• Reply • Share ›

Animesh • 7 months ago

no need for 2 loops..

 • Reply • Share ›

Ajay kumar • 8 months ago

My C++ implementation

```
#include<iostream>
#include<list>

using namespace std;
int main(){
    list <int> a;
    list <int>::iterator b, c;
    int input;
    cout << "Enter";
    while(cin >> input){
        a.push_back(input);
    }

    b = a.begin();
    c = a.begin();
```

see more

 • Reply • Share ›

DHANRAJ • 8 months ago

PASS THE ADDRESS OF FIRST ELEMENT OF THE LIST AND GET THE REQUIRED LIST

```
void delete_g(node **start)
{
if(*start==NULL || (*start)->next==NULL)
return ;
if((*start)->next->data > (*start)->data)
{
node *p=*start;
*start =(*start)->next ;
free(p);
delete_g(&(*start));
}
else
delete_g(&(*start)->next);
//printf("%d\n",start->data);
}
```

• Reply • Share ›

ajayv • 8 months ago

This is running perfectly without doing any reversal.. plzzz comment..

```
void arrangeNodes(struct node *head)
{
struct node *node = head;
struct node *temp;

while (node != NULL && node->next != NULL)
{
/* If node is smaller than its next, then delete current */
if(node->data < node->next->data)
{
temp = node->next;
node->data = temp->data;
node->next = temp->next;
free(temp);
}
else
node = node->next;
}
}
```

• Reply • Share ›

ajayv ajayv • 8 months ago

running code is here.... http://www.compileonline.com/c...

plz correct me if i left any test cases..

• Reply • Share ›

Dharmendra Verma ajayv • 7 months ago

ur code goes wrong for case 10->5->11

• Reply • Share ›

VINOD PAL • 8 months ago

//O(n) Using Recursive

```
#include<stdio.h>

#include<malloc.h>

typedef struct node{

int data ;
```

```
int data ;
struct node *next;
}N;
void push(N **new_node,int key){
N *temp=(N*) malloc(sizeof(N));
temp->data=key;
temp->next=(*new_node);
```

see more

• Reply • Share ›

Load more comments

Subscribe Add Disqus to your site Privacy

- 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

Recent Comments

- Rajeev

  Did you consider the scenario, when root itself...

  Given a binary tree, how do you remove all the half nodes? · 1 minute ago

- Anonymous

  Will it work for multi-dimensional array? I...

An Uncommon representation of array elements · 3 minutes ago

- Brahma Reddy Chilakala

  I think in first function there is no use. But...

  Count the number of occurrences in a sorted array · 8 minutes ago

- Rajeev

  Use a hash map to store the visited...

  Myntra Interview Experience | Set 5 · 16 minutes ago

- bhavi

  +1

  Majority Element · 16 minutes ago

- Nirav patel

  recursion of main #include<stdio.h>...

  How will you print numbers from 1 to 100 without using loop? · 44 minutes ago