Remove duplicates from an unsorted linked list

Write a removeDuplicates() function which takes a list and deletes any duplicate nodes from the list. The list is not sorted.

For example if the linked list is 12->11->12->21->41->43->21 then removeDuplicates() should convert the list to 12->11->21->41->43.

METHOD 1 (Using two loops)
This is the simple way where two loops are used. Outer loop is used to pick the elements one by one and inner loop compares the picked element with rest of the elements.

Thanks to Gaurav Saxena for his help in writing this code.

```
/* Program to remove duplicates in an unsorted array */

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

/* A linked list node */
struct node
{
 int data;
 struct node *next;
};

/* Function to remove duplicates from a unsorted linked list */
void removeDuplicates(struct node *start)
{
  struct node *ptr1, *ptr2, *dup;
  ptr1 = start;

  /* Pick elements one by one */
  while(ptr1 != NULL && ptr1->next != NULL)
  {
     ptr2 = ptr1;

     /* Compare the picked element with rest of the elements */
     while(ptr2->next != NULL)
     {
       /* If duplicate then delete it */
       if(ptr1->data == ptr2->next->data)
       {
          /* sequence of steps is important here */
          dup = ptr2->next;
          ptr2->next = ptr2->next->next;
          free(dup);
       }
       else /* This is tricky */
       {
          ptr2 = ptr2->next;
       }
     }
     ptr1 = ptr1->next;
  }
}

/* UTILITY FUNCTIONS */
/* Function to push a node */
void push(struct node** head_ref, int new_data);

/* Function to print nodes in a given linked list */
void printList(struct node *node);

/* Druver program to test above function */
```

```
int main()
{
  struct node *start = NULL;

  /* The constructed linked list is:
   10->12->11->11->12->11->10*/
  push(&start, 10);
  push(&start, 11);
  push(&start, 12);
  push(&start, 11);
  push(&start, 11);
  push(&start, 12);
  push(&start, 10);

  printf("\n Linked list before removing duplicates ");
  printList(start);

  removeDuplicates(start);

  printf("\n Linked list after removing duplicates ");
  printList(start);

  getchar();
}

/* 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 nodes in a given linked list */
void printList(struct node *node)
{
  while(node != NULL)
  {
    printf("%d ", node->data);
    node = node->next;
  }
}
```

Time Complexity: O(n^2)

METHOD 2 (Use Sorting)
In general, Merge Sort is the best suited sorting algorithm for sorting linked lists efficiently.
1) Sort the elements using Merge Sort. We will soon be writing a post about sorting a linked list. O(nLogn)
2) Remove duplicates in linear time using the algorithm for removing duplicates in sorted Linked List. O(n)

Please note that this method doesn't preserve the original order of elements.

Time Complexity: O(nLogn)

METHOD 3 (Use Hashing)
We traverse the link list from head to end. For every newly encountered element, we check whether it is in the hash table: if yes, we remove it; otherwise we put it in the hash table.

Thanks to bearwang for suggesting this method.

Time Complexity: O(n) on average (assuming that hash table access time is O(1) on average).

Please write comments if you find any of the above explanations/algorithms incorrect, or a better 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

Like 9 Tweet 0 +1 1

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

78 Comments GeeksforGeeks Login

Recommend Share Sort by Newest

Join the discussion…

Ansuraj Khadanga • 14 days ago
My implementation to remove duplicates from unsorted linked list using Hash

http://ideone.com/iNsbTN

• Reply • Share ›

magicsign • 2 months ago

Method3, C#, enjoy :

static LinkedList<string> removeDuplicates(LinkedList<string> e) {

//Parse the whole Linked list, if the current element is found in the Hash table, remove it otherwise insert it in the Hash Table

System.Collections.Hashtable hashTable = new System.Collections.Hashtable();

LinkedListNode<string> current = e.First;

while(current != null){

if(hashTable.Contains(current.Value)){

e.Remove(current);

}

else{

see more

• Reply • Share ›

Hinata Hyuga • 2 months ago

method 3: hashing will also cost the space complexity of O(n).
where as other method do not use extra space.

• Reply • Share ›

jeyrs • 2 months ago

```
Node removeDuplicates(Node head){
for(Node i = head; i != null ; i = i.next){
for(Node j = i; j != null; j = j.next){
if(i.data == j.data) i.next = j.next;
}
}
return head;
}
```

With
class Node{

int data;
Node next;
}

• Reply • Share ›

Hinata Hyuga jeyrs • 2 months ago

1. you cannot use i.next = j.next, if first element is equal to 4th element the all the nodes from ith to jth node will be lost (node 2 and node 3).
2. also you need to delete the ptr to free the memory.

1 • Reply • Share ›

maniac87 • 4 months ago

/**
* Method 3 : Removing duplicates when we have a buffer(Using hash map)
* Complexity : putting elements in has map is O(1) containdKey : O(1) (assuming good hash function)
* traversing O(n) deleting O(n) Total : O(n)
*
* @param head
*/
public void removeDuplicatesWithBuffer(final LinkedListNode head) {
//if the linked list is empty
if(head == null) {
throw new RuntimeException("linked list is empty");
//can also return;
}
LinkedListNode current = head ;
LinkedListNode previous = null;
final Map<integer, integer=""> map = new HashMap<integer, integer="">();
while(current != null) {

see more

• Reply • Share ›

Guest maniac87 • 3 months ago

Please check on this linklist.

1->2->3->1->4->5->4->5->2->6->7

It will not remove repeated 5.

3 • Reply • Share ›

Jerry Goyal • 4 months ago

using count array in O(n) time

```
void duplicateusingarray(struct node* head){
int a[1000]={0};

struct node* tmp=head;

struct node* prev=NULL;

while(tmp!=NULL){

a[tmp->data]++;

if(a[tmp->data]>1){

prev->next=tmp->next;

tmp=tmp->next;

}
```

see more

• Reply • Share ›

Ekta Goel • 4 months ago

Create a count array and store the count of values of the node in linked list. Traverse the list from head to tail and for every node we encounter check its value in count array, if its 1 then move to the next node. If its greater than 1,assuming we are keeping track of the previous pointer, it now points to current->next and delete the current pointer.

• Reply • Share ›

Jerry Goyal Ekta Goel • 4 months ago

what if data value stored in linked list is huge...?we can't make array of that size.

• Reply • Share ›

Ekta Goel Jerry Goyal • 4 months ago

This was a possible method and obviously for large values , we have certain limitations. However, we can alternatively use maps in stl. Insert the (value,count) pair in map. If the value is not in stl, it'l be inserted and if it'll be there, the find() function returns the iterator to the existing value. Increment the count. PS: map is taken as a tree in stl, and operations on it are logarithmic (searching).

1 • Reply • Share ›

Sneha Ekta Goel • 4 months ago

Unordered map gives best performance. Its average time complexity is O(1)

• Reply • Share ›

Chanakya Nani • 5 months ago

Another solution using arrays where it removes not only duplicates but more than one occurrence of an element. It also preserves the order of the elements.
Time complexity: O(n)
Space complexity: max number you'll accommodate in the linked list
http://ideone.com/9G7a2D

```
void RemoveDupUnsorted(node*& list){
node* oldList = list;

int a[100003]={0};
node* temp = list->next;
a[list->data]++;

while(temp->next!=NULL){
a[temp->data]++;
if(a[temp->data]>1){
list->next = temp->next;
temp = list->next;
}
```

see more

1 ^ | v • Reply • Share ›

shark123 • 5 months ago

Will the hashing technique preserve the original order of elements?

^ | v • Reply • Share ›

Ravindar Dev ↪ shark123 • 5 months ago

Yes ,for sure !!
Cause you are not reordering the list.
It will run like this:-
for each element in the list [given order] ,we check if there is entry in hash table and if found u remove that element otherwise not. This way we can remove the duplicate while preserving the order..

^ | v • Reply • Share ›

Ethan Lim • 5 months ago

```
/*
* Author : ethanlim
* Description : A few questions should be asked is what type of data the linked list contain.
* It would greatly simply the hashing method with O(1) auxiliary memory.
*
* An example here is the linked list contained only ASCII characters from 'a' to 'z'.
* There are 26 alphabets hence we can use a 32 bit integer to represent this.
* We just check if the bit is set in the integer via AND operation or set it otherwise
* with a OR operation
*/

#include <iostream>
#include <cstddef>

class Node{
private:
unsigned int key_;
char data_;
```

see more

1 • Reply • Share ›

ritesh kumar • 6 months ago

Remove Duplicates using hashing

```
void removeDup(struct node** head_ref)

{

struct node *cur, *coming, *del;

map<int, int=""> occurences;

map<int, int=""> :: iterator it;

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

return;

cur = *head_ref;

coming = cur->link;
```

see more

• Reply • Share ›

swati • 7 months ago

while using hashing,it could be o(n^2) when there is so much collision.

• Reply • Share ›

Vãîbhåv Joshî swati • 7 months ago

no we'll put element when its not process before.... if its present then simply check in O(1) time and remove dup node....if its not present then put in hashTable....

^ | ⌄ • Reply • Share ›

setu • 7 months ago

Can some validate the time complexity of the Algorithm

http://stackoverflow.com/a/217...

Thanks in Advance

^ | ⌄ • Reply • Share ›

Daggerhunt • 8 months ago

Can someone post the implementation for Method 3 (Using Hashing)?

^ | ⌄ • Reply • Share ›

Guest ➦ Daggerhunt • 4 months ago

```
/**
* Method 3 : Removing duplicates when we have a buffer(Using hash map)
* Complexity : putting elements in has map is O(1) containdKey : O(1) (assuming good hash function)
* traversing O(n) deleting O(n) Total : O(n)
*
* @param head
*/
public void removeDuplicatesWithBuffer(final LinkedListNode head) {
//if the linked list is empty
if(head == null) {
throw new RuntimeException("linked list is empty");
//can also return;
}
LinkedListNode current = head ;
LinkedListNode previous = null;
final Map<integer, integer=""> map = new HashMap<integer, integer="">();
while(current != null) {
```

see more

^ | ⌄ • Reply • Share ›

Vijai • 8 months ago

If we have high range for value in linked list nodes hashing is not practical

2 ^ | ⌄ • Reply • Share ›

kumar praharsh Rakheja • 9 months ago

Not getting the output can someone explain ???

Not getting the output.....can someone explain.....???

```
#include"stdio.h"
#include"malloc.h"

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

struct node *head = NULL;

void insert(int num)
{
struct node *q = (struct node *)malloc(sizeof(struct node));
q->next = NULL;
q->data = num;
if(head==NULL)
```

see more

^ | ⌄ • Reply • Share ›

sonu431 • 9 months ago

In the method 3 what is key and value for the process to carried out? plz explain

^ | ⌄ • Reply • Share ›

Bharath G M ➦ sonu431 • 8 months ago

you can give it anything. It actually doesn't matter. Make the node's data as a key.

1 ^ | ⌄ • Reply • Share ›

pk28 ➦ sonu431 • 8 months ago

key will be your node's data and its value you can take as 1.

^ | ⌄ • Reply • Share ›

Vivek Garg • 10 months ago

Here is c++ implementation for this topic using hashTable..

```
void removeDuplicates(linkedlist list){

node *temp=list.head;

linkedlist hashlist[100];

node *prevNode=NULL;/*It is used because my pop function take the prevNode as input and remove the node after prevNode.*/
```

```
while(temp!=NULL)

{

if(isNodeInhashlist(temp,hashlist))

{

list.pop(prevNode);
```

see more

^ | ⌄ • Reply • Share ›

Jerry Goyal ➦ Vivek Garg • 4 months ago
can we use hashtable method in C?
^ | ⌄ • Reply • Share ›

Arun Dixit • 10 months ago
Following is java implementation using hashSet:

```
public void removeDuplicatesUnsorted(Node head){

HashSet<integer> elem = new HashSet<integer>();

Node curr=head.next;

Node prev=head;

elem.add(head.data);

while(curr!=null){

if(elem.contains(curr.data)){

prev.next=curr.next;

curr=null;

curr=prev.next;
```

see more

1 ^ | ⌄ • Reply • Share ›

Himanshu Dagar • a year ago
Hashing is Perfect method for this
We can use map<int,int> for this (one is key and another one is mapped value)
^ | ⌄ • Reply • Share ›

shark123 ➦ Himanshu Dagar • 5 months ago

Can you please explain your method?

• Reply • Share ›

Sunil • a year ago

I have a better solution for this. For hashtable we need to again use extra memory. space constraint. If the interviewer asks you to give a solution with linear time O(n) with no extra space. please try using bit vectors. It will solve the problem.

private void removeDuplicates(Node head) {

Node iter = head;

int checker = 0;

while(iter != null) {

if((checker & (1 << iter.data)) != 0) {

// Found duplicate

if(iter.next != null) {

iter.data = iter.next.data;

see more

• Reply • Share ›

ryan ➦ Sunil • 8 months ago

this solution is only vaild if the list contain number from 1 to 32.

• Reply • Share ›

Niks ➦ Sunil • a year ago

The above solution is basically trying to store the occurrence of a number in a 32 bit integer. What if I have two numbers like 5 and 37. When 5 is encountered the code sets bit5 to 1 and when 37 is encountered the 32 bit number wraps and again bit 5 is set. Does this not cause 5 and 32 to be treated as duplicates??

• Reply • Share ›

Akash Panda • a year ago

Full source code for solution using the Hashing method.

void RemoveDuplicatesInUnsorted(struct node **head_ref)

{

struct node *current=*head_ref;

int a[100]={0};

struct node *prev=current;

a[current->data]=1;

current=current->next;

while(current!=NULL)

{

if(a[current->data]==1)

see more

• Reply • Share ›

Rich • a year ago

Don't understand the following statement:

/* Pick elements one by one */

while (ptr1 != NULL && ptr1->next != NULL)
{
. . .
}

I think we could write like this:

/* Pick elements one by one */

while (ptr1 != NULL)

{
. . .
}

• Reply • Share ›

Codecrawler • a year ago

How can we make hash table for it? Please provide the code

• Reply • Share ›

danny ➦ Codecrawler • a year ago

1.) We will map every value of linked list to hash table.

2.) While mapping we can compare if the two values hash to same location in hash table then we get the address of the hashed value in linked list and delete it table,then we get the address of the hashed value in linked list and delete it.

3.)Arrange pointers of the linked list .

Reply • Share ›

Tandoori • a year ago

```
node * partition(node *start)

{

node *l1=start;

node *temp1=NULL;

node *temp2=NULL;

if(start->next==NULL)

return start;

node * l2=f_b_split(start);

if(l1->next!=NULL)

temp1=partition(l1);

if(l2->next!=NULL)
```

see more

Reply • Share ›

Prashant Rathi • a year ago

here is the code in c

```
void removeduplicates(N **r)
{
N *temp1=*r;
N *temp2=NULL;
N *temp3=NULL;
while(temp1->next!=NULL)
{
temp2=temp1;
while(temp2!=NULL)
{
temp3=temp2;
temp2=temp2->next;
if(temp2==NULL)
{
break;
```

}

see more

• Reply • Share ›

ubiquitous • 2 years ago

[sourcecode language="JAVA"]
void removeDuplicates()
{
if(head == null)return;
if(head.next==null)return;
Set<K> s = new HashSet<K>();
s.add(head.k);
Node<K> n = head;
while(n!=null && n.next!=null)
{
Node<K> temp = n.next;
while(true)
{
if(temp==null)break;
if(s.contains(temp.k))
{
n.next = temp.next;
temp=temp.next;

see more

1 • Reply • Share ›

Karshit • 2 years ago

My Code using Unordered_set.. hope you find it useful.. :)

```
 #include <iostream>
#include <unordered_set>

using namespace std;

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

node *create(int n)
{
    if (n == 0)
```

```
        return NULL;

    node *head = new node();
```

see more

1 ^ | ⌄ • Reply • Share ›

12rad • 2 years ago

For Java: Using hashMaps

```
public Linked_List removeOnlyDuplicates(Linked_List list){
               HashMap<Integer, Integer> map = new HashMap<Integer, Integer>();
               Linked_List.LNode node = list.head;
               Linked_List.LNode prev = list.head;
               while(node!=null){
                       if(map.containsKey(node.data)){
                               prev.nextPtr = node.nextPtr;

                       }else{
                               map.put(node.data, 0);
                               prev = node;
                       }
                       node = node.nextPtr;
               }
               return list;
       }/
```

^ | ⌄ • Reply • Share ›

Priyanka • 2 years ago

```
 void removeDuplicates(struct node *start)
{
  struct node *current=start, *save_current=NULL, *next=NULL;
  int hash_table[INT_MAX]={0};
  while(current)
 {
   if(has_table[current->data])
   {
     next=current->next;
     free(current);
     current=next;
     save_current->next=current;
   }
   else
```

```
    {
      hash_table[current->data]=1;
      save_current=current;
      current=current->next;


  }


 }
```

Reply • Share ›

hary Priyanka • 2 years ago

@priyanka, correct me if I am wrong here . Just modified your code a bit and also you have not made use of the hashfunc anywhere so you should use that as well. Yes do not forget to initialize your hashtbl with 0 before the start of everything.

```
while(current)
{
  if(has_table[current->data])
  {
    save_current->next=current->next;
    free(current);
    current=save_current->next;
  }
  else
  {
    hash_table[current->data]=1;
    save_current=current;
    current=current->next;
  }


}
```

Reply • Share ›

aravind Priyanka • 2 years ago

is save_current node the previous node of the node to be deleted?

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

Reply • Share ›

Kshitij Nagpal • 2 years ago

Awesome!

^ | ⌄ • Reply • Share ›

abcd • 2 years ago

Can someone please share the code for Method 3 listed here.

^ | ⌄ • Reply • Share ›

gr81 ➦ abcd • 2 years ago

```
 /* Paste your code here (You may delete these lines if not writing code) */
void remove_duplicate(struct node *head)
{
        map<int, int> freq;
        struct node *cur = head;
        struct node *prev = NULL;
        while(cur != NULL)
        {
                if(freq[cur->data])
                {
                        struct node *tmp = cur;
                        prev->next = cur->next;
                        cur = cur->next;
                        delete tmp;
                }
                else
                {
                        freq[cur->data]++;
                        prev = cur;
                        cur = cur->next;
                }
        }
}
```

^ | ⌄ • Reply • Share ›

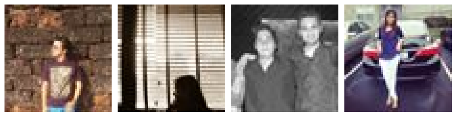
Load more comments

Subscribe   Add Disqus to your site   Privacy

-
- GeeksforGeeks

  Like

  93,248 people like GeeksforGeeks.

  Facebook social plugin
-
-
    - 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

    That would have higher time complexity.Method 1...

    Write a function to get the intersection point of two Linked Lists. · 7 minutes ago

  - Guest

    calling the 2nd solution as dp is stretching...

    Longest Even Length Substring such that Sum of First and Second Half is same · 21 minutes ago

  - Goku

    They are considering 0 based indexing instead...

    Write a function to get Nth node in a Linked List · 1 hour 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) · 5 hours ago

  - lebron

    merge sort

    K'th Smallest/Largest Element in Unsorted Array | Set 3 (Worst Case Linear Time) · 5 hours ago

  - Shubham Sharma

    You saved my time :)

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

-