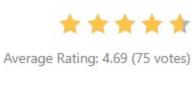
■ Articles > 83. Remove Duplicates from Sorted List

# 83. Remove Duplicates from Sorted List 💆

April 18, 2016 | 108.6K views



Example 1:

Given a sorted linked list, delete all duplicates such that each element appear only once.

Articles

Discuss

### Input: 1->1->2

```
Output: 1->2
Example 2:
  Input: 1->1->2->3->3
```

```
Output: 1->2->3
```

## Approach 1: Straight-Forward Approach **Algorithm**

Solution

#### is sorted, we can determine if a node is a duplicate by comparing its value to the node after it in the list. If it is a duplicate, we change the next pointer of the current node so that it skips the next node and points

• Space complexity : O(1). No additional space is used.

After another loop iteration, one of two things happen.

duplicates up to current.

duplicates.

# directly to the one after the next node.

**С**ору Java public ListNode deleteDuplicates(ListNode head) { 2 ListNode current = head; while (current != null && current.next != null) { 3 if (current.next.val == current.val) { 4

This is a simple problem that merely tests your ability to manipulate list node pointers. Because the input list

```
5
                  current.next = current.next.next;
              } else {
   6
   7
                  current = current.next;
  8
  9
  10
          return head;
  11
Complexity Analysis
   ullet Time complexity : O(n). Because each node in the list is checked exactly once to determine if it is a
      duplicate or not, the total run time is O(n), where n is the number of nodes in the list.
```

### We can prove the correctness of this code by defining a loop invariant. A loop invariant is condition that is

Correctness

true before and after every iteration of the loop. In this case, a loop invariant that helps us prove correctness is this:

And so it can not contain any duplicate elements. Now suppose current is now pointing to some node in the list (but not the last element), and the part of the list up to current contains no duplicate elements.

At the last iteration of the loop, current must point to the last element, because afterwards,

All nodes in the list up to the pointer current do not contain duplicate elements.

We can prove that this condition is indeed a loop invariant by induction. Before going into the loop,

current points to the head of the list. Therefore, the part of the list up to current contains only the head.

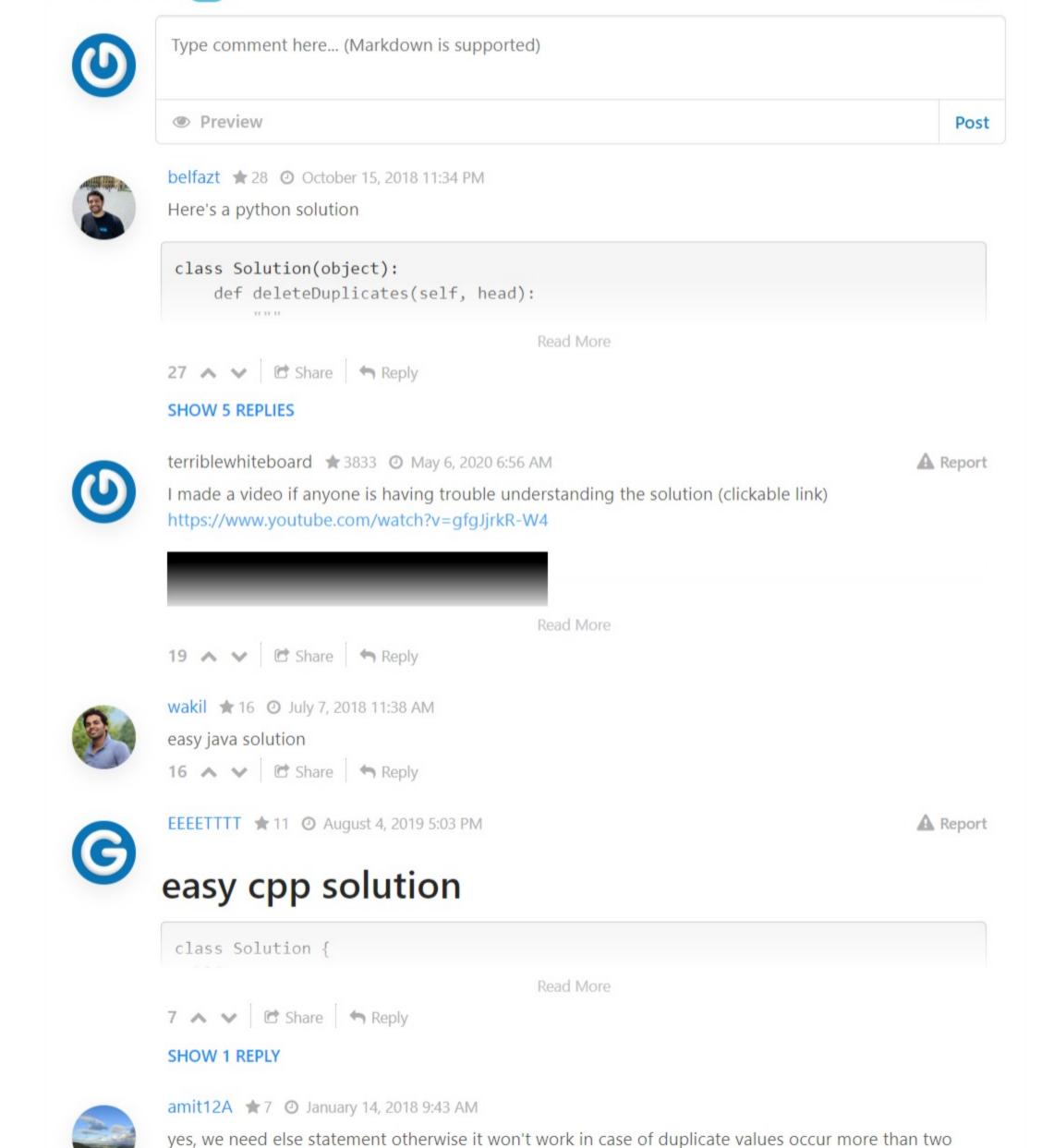
deleted, and current stays pointing to the same node as before. Therefore, the condition still holds; there are still no duplicates up to current. 2. current.next was not a duplicate of current (and, because the list is sorted, current.next is also not a duplicate of any other element appearing before current ). In this case, current moves forward one step to point to current.next. Therefore, the condition still holds; there are no

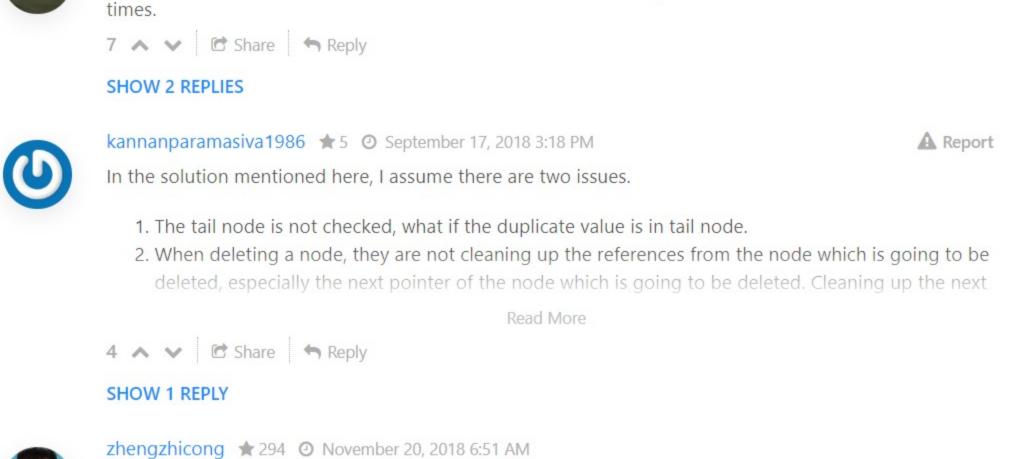
1. current.next was a duplicate of current. In this case, the duplicate node at current.next is

O Previous Next 🕑 Comments: 65 Sort By ▼

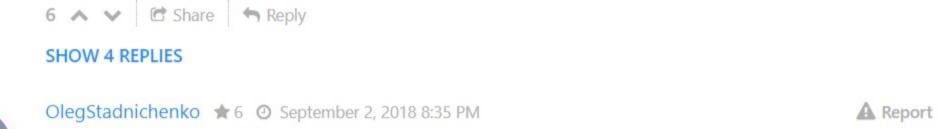
Rate this article: \* \* \* \* \*

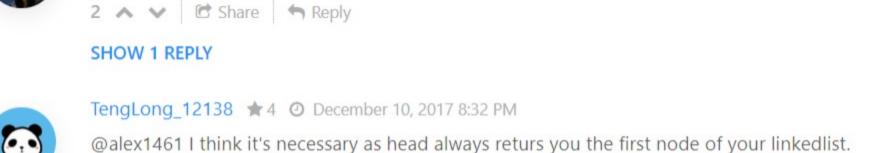
current.next = null. Therefore, after the loop ends, all elements up to the last element do not contain



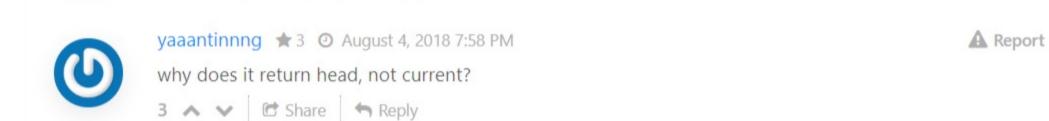








It's not clear should we change the input structure or create new. i mean in description.



(1) 2 3 4 5 6 7 >

2 A V C Share Reply

**SHOW 3 REPLIES** 

python3: