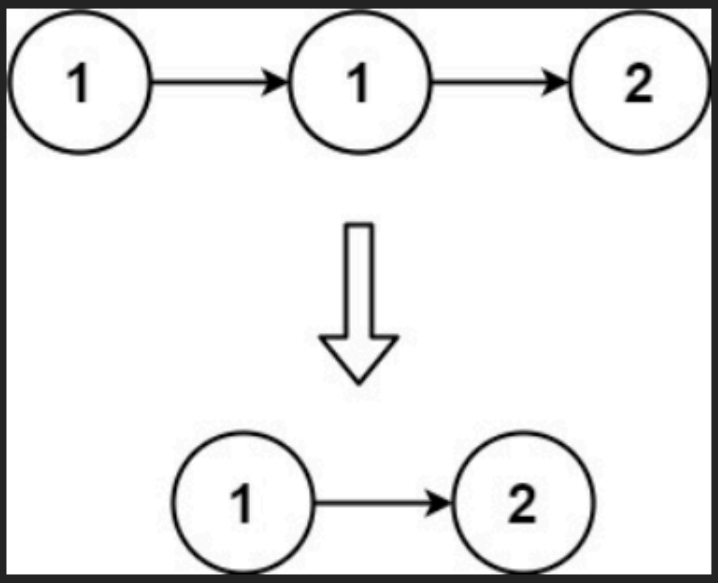
83. Remove Duplicates from Sorted List

Solved

Easy | Topics | Companies

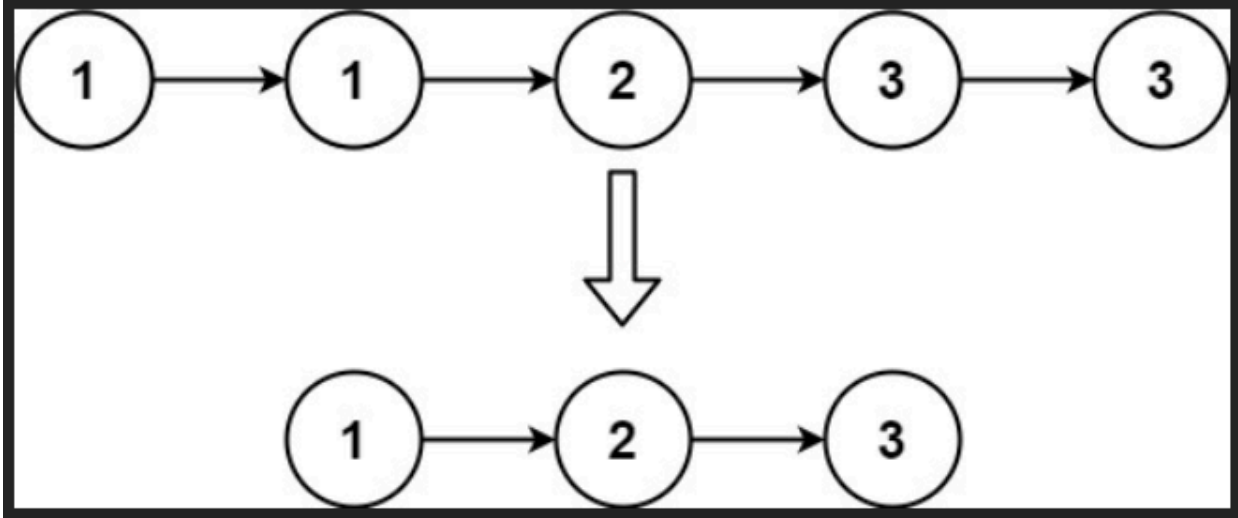
Given the `head` of a sorted linked list, *delete all duplicates such that each element appears only once*. Return *the linked list* ***sorted*** *as well*.

Example 1:




```
Input: head = [1,1,2]
Output: [1,2]
```

Example 2:




```
Input: head = [1,1,2,3,3]
Output: [1,2,3]
```

Constraints:

- The number of nodes in the list is in the range `[0, 300]`.
- `-100 <= Node.val <= 100`
- The list is guaranteed to be **sorted** in ascending order.

Python:

```
# Definition for singly-linked list.
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next

class Solution:
    def deleteDuplicates(self, head: ListNode) -> ListNode:
        current = head
        # Traverse the list
        while current and current.next:
            if current.val == current.next.val:
                # Skip the duplicate node
```

```
            current.next = current.next.next
        else:
            # Move to the next node if no duplicate
            current = current.next
    return head
```

JavaScript:

```
// Definition for singly-linked list.
function ListNode(val, next = null) {
  this.val = val;
  this.next = next;
}

var deleteDuplicates = function(head) {
  let current = head;

  while (current && current.next) {
    if (current.val === current.next.val) {
      // Skip duplicate node
      current.next = current.next.next;
    } else {
      // Move forward only if values are different
      current = current.next;
    }
  }

  return head;
};
```

Java:

```
// Works on judges that call ListNode.deserialize("[1,1,2,3,3]")
class ListNode {
    int val;
    ListNode next;

    ListNode() {}
    ListNode(int val) { this.val = val; }
    ListNode(int val, ListNode next) { this.val = val; this.next = next; }

    // Parse "[1,1,2]" -> linked list 1->1->2
    public static ListNode deserialize(String s) {
        if (s == null) return null;
        s = s.trim();
        if (s.isEmpty() || s.equals("[]")) return null;
        if (s.charAt(0) == '[' && s.charAt(s.length() - 1) == ']') {
```

```
            s = s.substring(1, s.length() - 1).trim();
        }
        if (s.isEmpty()) return null;

        String[] parts = s.split(",");
        ListNode dummy = new ListNode(0);
        ListNode cur = dummy;
        for (String p : parts) {
            p = p.trim();
            if (!p.isEmpty()) {
                cur.next = new ListNode(Integer.parseInt(p));
                cur = cur.next;
            }
        }
        return dummy.next;
    }

    // Turn list back into "[...]" string
    public static String serialize(ListNode head) {
        StringBuilder sb = new StringBuilder();
        sb.append("[");
        ListNode cur = head;
        while (cur != null) {
            sb.append(cur.val);
            if (cur.next != null) sb.append(",");
            cur = cur.next;
        }
        sb.append("]");
        return sb.toString();
    }

    @Override public String toString() {
        return serialize(this);
    }
}

class Solution {
    public ListNode deleteDuplicates(ListNode head) {
        ListNode cur = head;
        while (cur != null && cur.next != null) {
            if (cur.val == cur.next.val) {
                cur.next = cur.next.next; // skip duplicate
            } else {
                cur = cur.next;
```

```
            }
        }
        return head;
    }
}
```