Problem 83: Remove Duplicates from Sorted List

Problem Information

Difficulty: Easy
Acceptance Rate: 55.75%
Paid Only: No
Tags: Linked List

Problem Description

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.

Code Snippets

C++:

```
/**
* Definition for singly-linked list.
* struct ListNode {
* int val;
* ListNode *next;
* ListNode() : val(0), next(nullptr) {}
* ListNode(int x) : val(x), next(nullptr) {}
* ListNode(int x, ListNode *next) : val(x), next(next) {}
* };
*/
class Solution {
public:
ListNode* deleteDuplicates(ListNode* head) {

}
};
```

Java:

```
/**
* Definition for singly-linked list.
* public class ListNode {
* int val;
* ListNode next;
* ListNode() {}
* ListNode(int val) { this.val = val; }
* ListNode(int val, ListNode next) { this.val = val; this.next = next; }
* }
*/
class Solution {
public ListNode deleteDuplicates(ListNode head) {

}
}
```

Python3:

```
# 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: Optional[ListNode]) -> Optional[ListNode]:
```