Documentation for "Remove Duplicates from Sorted List"

Problem Statement

Given the head of a sorted linked list, delete all duplicates such that each element appears only once. Return the modified linked list, which remains sorted.

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 leq text{Node.val} leq 100)
- The list is guaranteed to be sorted in ascending order.

Solution

The task is to remove duplicate elements from a sorted linked list. Since the list is sorted, duplicate elements will be consecutive. We can achieve the solution by iterating through the list and skipping nodes with duplicate values.

Approach

1. Initialize a pointer current at the head of the linked list.
2. Traverse the list while the current node and its next node are not None.

- If the value of the current node is the same as the value of the next node, skip the next node by updating the next pointer of the current node.
- Otherwise, move the current pointer to the next node.

3. Return the modified list starting from the head node.

Explanation

1. **Class Definitions:**

- *ListNode:* Defines a node in the linked list with a value (val) and a pointer to the next node (next).
- *Solution:* Contains the method deleteDuplicates to remove duplicates from the list.

2. **Method deleteDuplicates:**

- *Input:* head The head node of the sorted linked list.
- *Output:* The head node of the modified list with duplicates removed.
- *Algorithm:*
  - ✓ Initialize current to the head of the list.
  - ✓ Loop through the list using the condition while current and current.next.
  - ✓ If the value of the current node is equal to the value of the next node (current.val == current.next.val), update the next pointer of the current node to skip the next node (current.next = current.next.next).
  - ✓ If the values are not equal, move the current pointer to the next node (current = current.next).
  - ✓ Return the head of the modified list after removing duplicates.

This solution efficiently removes duplicates from a sorted linked list in a single pass with a time complexity of (O(n)), where (n) is the number of nodes in the list.