Documentation Remove Duplicates from Sorted List II

Problem Description

Given the head of a sorted linked list, delete all nodes that have duplicate numbers, leaving only distinct numbers from the original list. Return the linked list sorted as well.

Example 1:

Input: head = [1,2,3,3,4,4,5]

Output: [1,2,5]

Example 2:

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

Output: [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.

Data Structure Definition

```
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next
```

Solution Implementation

The solution to remove duplicates from the sorted linked list involves iterating through the list and removing nodes that have duplicate values.

Explanation

1. **Initialization:**

- A dummy node is created with a value of 0 and its next pointer is set to the head of the input list. This dummy node helps to handle edge cases where the first few nodes might be duplicates and need removal.
- A pointer prev is initialized to point to the dummy node. This pointer is used to track the node before the sequence of duplicates.

2. **Iterating Through the List:**

- The while loop runs as long as head is not None.
- Within the loop, it checks if the current node (head) has a duplicate by comparing its value to the value of the next node (head.next).

3. **Handling Duplicates:**

- If duplicates are detected (head.val == head.next.val), another loop is used to skip all nodes with the same value by moving the head pointer to the next node until the end of the sequence of duplicates is reached.
- The prev node's next pointer is then updated to point to the node after the last duplicate, effectively removing the duplicates from the list.

4. **Moving the Pointers:**

- If no duplicates are detected, the prev pointer is moved to the head node.
- The head pointer is always moved to the next node in each iteration.

5. **Return the Result:**
    - Finally, the function returns the modified list starting from dummy.next.

This approach ensures that all nodes with duplicate values are removed, and the resulting list contains only distinct numbers from the original list. The use of a dummy node simplifies handling edge cases and ensures the function works correctly even if the initial nodes are duplicates.