[Click on the logo to find **Problem Statement**]

Intuition

This code merges two sorted linked lists, list1 and list2, into a single sorted linked list. It initializes a new linked list (head) and a pointer (current) to the head. The algorithm iterates through both input lists simultaneously. At each step, it compares the current nodes from list1 and list2, appends the smaller node to the result, and advances the corresponding list pointer. This process continues until one of the lists is exhausted. Finally, the algorithm links the remaining nodes from the non-empty list to the result. The merged list is returned. The approach leverages the sorted nature of the input lists to efficiently merge them in ascending order.

Approach

1. **Initialization:**
    - Create a dummy node (`head`) to simplify the handling of the merged list.
    - Initialize a pointer (`current`) to the dummy node.
2. **Iterative Merging:**
    - While both `list1` and `list2` are not empty:
        - Compare the values of the current nodes in `list1` and `list2` .
        - Append the node with the smaller value to the merged list (`current.next`).
        - Move the corresponding list pointer (`list1` or `list2`) to the next node.
        - Move the `current` pointer to the newly added node.
3. **Linking Remaining Nodes:**
    - Once one of the lists is exhausted, link the remaining nodes from the non-empty list to the merged list.
4. **Return Merged List:**
    - Return the merged list starting from the next of the dummy node (`head.next`).

Complexity

- Time complexity: O(n)
- Space complexity: O(1)

Code

```
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
class Solution:
    def mergeTwoLists(self, list1: Optional[ListNode], list2:
                      Optional[ListNode]) -> Optional[ListNode]:
        head = ListNode()
        current = head
        while list1 and list2:
            if list1.val < list2.val:
                current.next = list1
                list1 = list1.next
            else:
                current.next = list2
                list2 = list2.next
            current = current.next

        current.next = list1 or list2
        return head.next
```

If you want to see more solutions to coding problems, you can visit: