Merge Two Sorted Lists

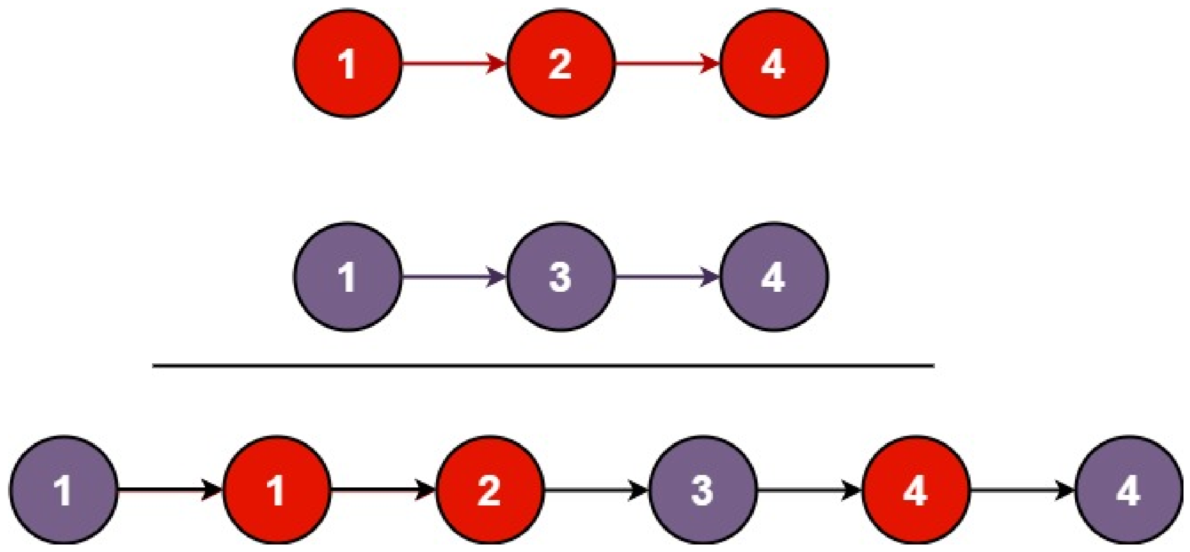
Difficulty	Easy
Category	LinkedList
Question	https://leetcode.com/problems/merge-two-sorted-lists/
Solution	https://youtu.be/XIdigk956u0
Status	Done

Question

> You are given the heads of two sorted linked lists `list1` and `list2`.
>
> Merge the two lists in a one **sorted** list. The list should be made by splicing together the nodes of the first two lists.
>
> Return *the head of the merged linked list*.

Example

Example 1:



```
Input: list1 = [1,2,4], list2 = [1,3,4]
Output: [1,1,2,3,4,4]
```

Example 2:

```
Input: list1 = [], list2 = []
Output: []
```

Example 3:

```
Input: list1 = [], list2 = [0]
Output: [0]
```

Idea

💡 Create a Dummy Node, Compare values from both LL and add the smaller ones to the Dummy and Return dummy.next (Head of the merged LL)

Solution

```
class Solution:
    def mergeTwoLists(self, list1: Optional[ListNode], list2: Optional[ListNode]) -> Optional[ListNode]:
        # Create a dummy node to simplify handling edge cases.
        dummy = ListNode()

        # Initialize the 'tail' pointer to the dummy node.
        tail = dummy

        # Iterate while both 'list1' and 'list2' are not empty.
        while list1 and list2:
            # Compare the values of the current nodes in 'list1' and 'list2'.
            if list1.val < list2.val:
                # If the value in 'list1' is smaller, append it to the result list.
                tail.next = list1
                list1 = list1.next
            else:
                # If the value in 'list2' is smaller or equal, append it to the result list.
                tail.next = list2
                list2 = list2.next

            # Move the 'tail' pointer to the last node in the result list.
            tail = tail.next

        # If there are remaining nodes in 'list1', append them to the result list.
        if list1:
            tail.next = list1
        # If there are remaining nodes in 'list2', append them to the result list.
        elif list2:
            tail.next = list2

        # Return the merged linked list, starting from the node after the dummy node.
        return dummy.next
```

Time and Space Complexity

📌 The Time complexity of this code is **O(m + n)**, where 'm' is the length of list1, and 'n' is the length of list2.

📌 The Space complexity of this code is **O(1)**, or constant space.

The code uses a constant amount of extra space for variables like `dummy`, `tail`, and other temporary variables. This space usage does not depend on the length of the input lists, so it's considered constant space