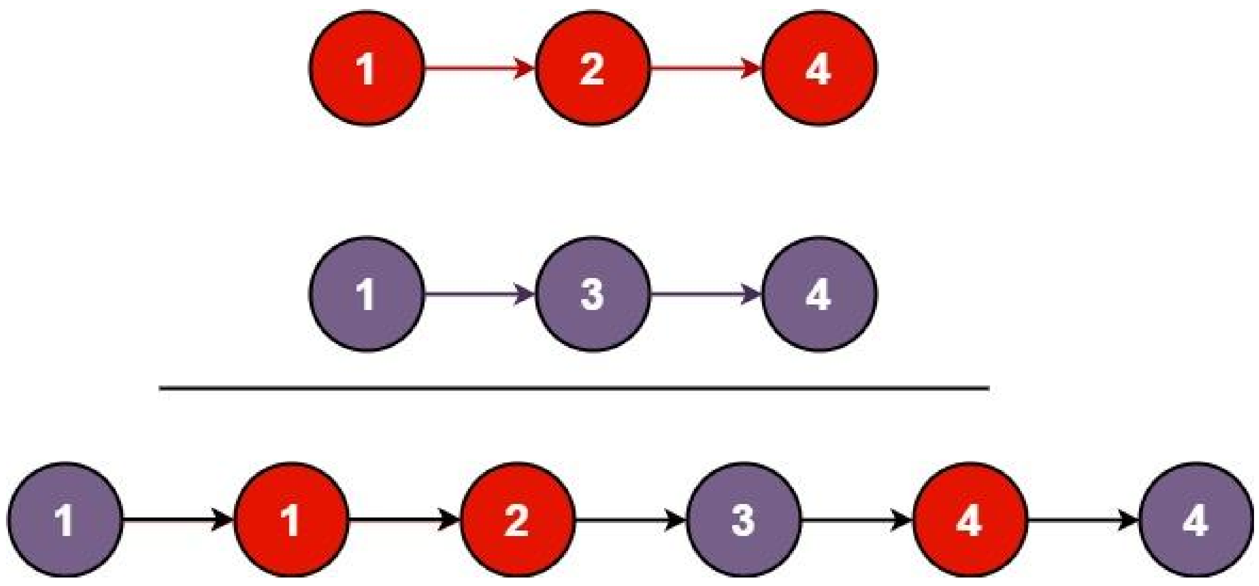
Problem Link:

https://leetcode.com/problems/merge-two-sorted-lists

Problem Description:

You are given the heads of two sorted linked lists `list1` and `list2`. Merge the two lists into 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*.



Problem Approach:

We use a simple step-by-step process with two pointers to merge the lists.

Solution:

We start by creating a dummy node called finalList with a value of 0. This node is just a placeholder to help us easily build the new list, and we'll use a current pointer to keep track of where we are as we merge the lists.

Next, we loop through both lists, comparing the values at the current nodes of list1 and list2. If list1's current value is smaller (or equal), we attach that node to the merged list and move list1 forward to the next node. If list2 has the smaller value, we attach its node instead and move list2 forward. After each comparison, we move the current pointer forward to keep building the list. Once we reach the end of one of the lists, we just attach the rest of the other list (since it's already sorted).
Finally, we return the merged list starting from finalList.next, skipping over the dummy node at the beginning.

Code (Python):

```python
def mergeTwoLists(self, list1: Optional[ListNode], list2: Optional[ListNode])
-> Optional[ListNode]:
       finalList = ListNode(0)
       current = finalList
       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
       if list1:
           current.next = list1
       if list2:
           current.next = list2
       return finalList.next
```