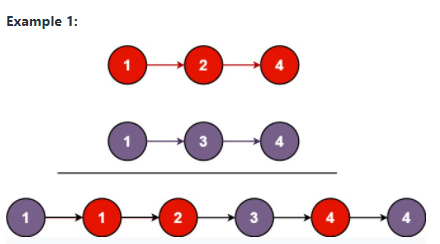
🡪问题1链表合并：合并两个单向链表，该单向链表的节点数值已经从小到大排序，要求合并后的链表节点的数值也从小到大



class Solution:

def mergeTwoLists(self, l1: ListNode, l2: ListNode) -> ListNode: # 输入参数为两个需合并链表的头节点，要求返回合并后的链表的头节点

if l1 and l2:

nodeList = []

while l1.next is not None: # 轮询链表1，将其中节点依次放入nodeList中

nodeList.append(l1)

l1 = l1.next

nodeList.append(l1)

while l2.next is not None: # 轮询链表2，将其中节点依次放入nodeList中

nodeList.append(l2)

l2 = l2.next

nodeList.append(l2)

nodeList = sorted(nodeList, key=lambda k: k.val) # 根据数值，对两链表的节点进行排序

for i in range(len(nodeList)-1): # 排序完成后，所需要的就是将节点的next指向连接起来。形成合并好的链表。

nodeList[i].next = nodeList[i+1]

return nodeList[0] # 返回合并后链表头节点

else:

if not l1 and not l2:

return None

elif not l2 and l1:

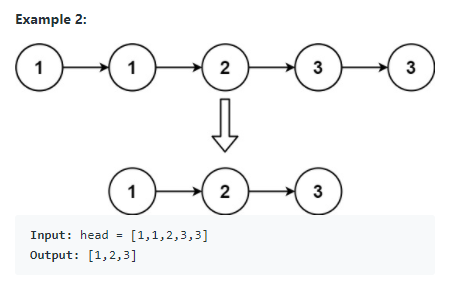
return l1

else:

return l2

🡪

问题2：在一个已经排序，从小到大的单向链表中，去除节点值重复的节点



class Solution:

def deleteDuplicates(self, head: ListNode) -> ListNode: # 输入参数为需要合并的链表头节点

result = head # 使用result变量记录头节点

if result:

checkingNode = head.next # 使用头节点作为指针1，指向当前节点。checkingNode作为另一个指针2，指向当前节点的下一个节点

while checkingNode:

if checkingNode.val != head.val: # 如果当前节点值不等于下一个节点值，则不改变节点中的引用。将指针1与指针2同时移动到下一个节点。

head = checkingNode

checkNode = checkingNode.next

else: # 如果当前节点值等于下一个节点值，则我们应该在下一个节点从链表中去除。所以将当前节点的next重新赋值，指向指针2所指节点的下一个节点。

head.next = checkingNode.next

checkingNode = checkingNode.next

return result # 最后返回去除重复后链表的头节点

🡪