연결 리스트를 O(nlogn)에 정렬하라.

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

```
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
```

1.리스트로 변환

```
class Solution:
    def sortList(self, head: ListNode) -> ListNode:
        def toList(head):
            result = []
            while head:
                result.append(head.val)
                head = head.next
            return result


        def toNode(list):
            result = temp = ListNode()
            for l in list:
                temp.next = ListNode(l)
                temp = temp.next
            return result.next


        result = toList(head)
        result.sort()
        result = toNode(result)
        return result
```

```
class Solution:
    def sortList(self, head: ListNode) -> ListNode:
        p = head
        lst: List = []
        while p:
            lst.append(p.val)
            p = p.next
        lst.sort()
        p = head
        for i in range(len(lst)):
            p.val = lst[i]
            p = p.next
        return head
```

2.merge sort

```
class Solution:
    def sortList(self, head: ListNode) -> ListNode:
        def mergeTwoLists(l1: ListNode, l2: ListNode) -> ListNode:
            if l1 and l2:
                if l1.val > l2.val:
                    l1, l2 = l2, l1
                l1.next = mergeTwoLists(l1.next, l2)
            return l1 or l2


        if not (head and head.next):
            return head
        half, slow, fast = None, head, head
        while fast and fast.next:
            half, slow, fast = slow, slow.next, fast.next.next
        half.next = None
        l1 = self.sortList(head)
        l2 = self.sortList(slow)
        return mergeTwoLists(l1, l2)
```