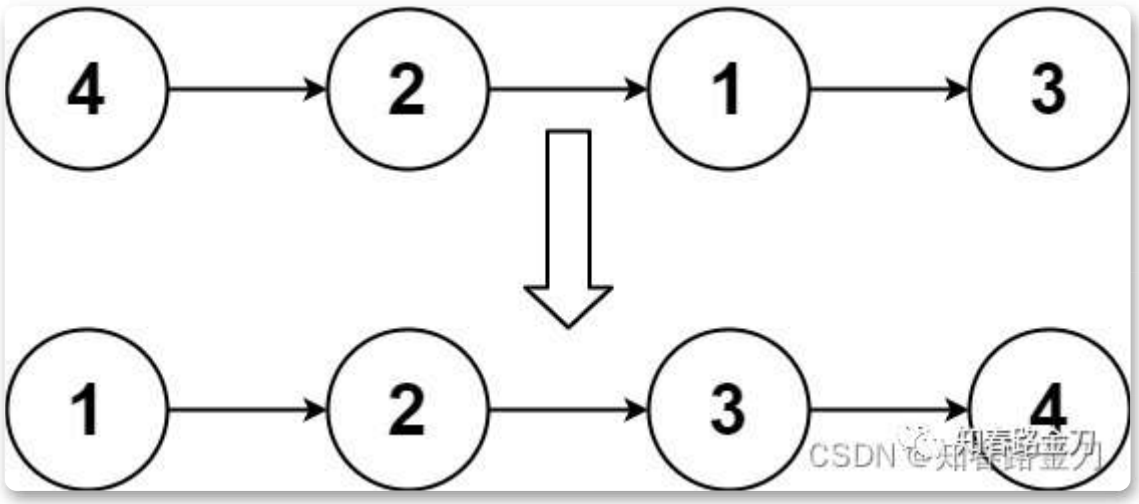
> PS：本道题也是在大厂互联网面试中出现频率非常高的题目，虽然字数不多，但做出最优解还是有一定难度的，请大家仔细阅读。

题目介绍[1]

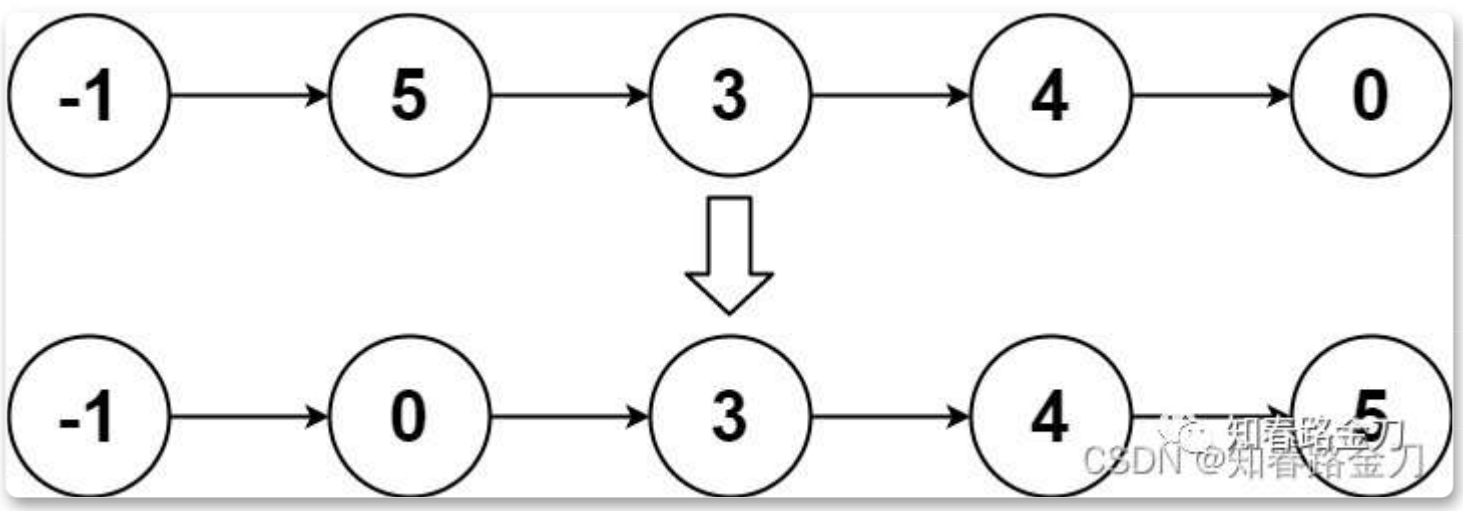
给你链表的头结点 `head` ，请将其按升序排列并返回排序后的链表。

示例 1



> **「输入」**：head = [4,2,1,3]
>
> **「输出」**：[1,2,3,4]

示例 2



> **「输入」**：head = [-1,5,3,4,0]
>
> **「输出」**：[-1,0,3,4,5]

题目解答

方法一：归并排序

思路和算法

本道题可以使用归并排序的思路来解答，归并排序的核心思想是分而治之，把一个大问题转换成多个**「子问题」**，通过对这些**「子问题」**求解，从而得出**「大问题」**的解。那么在这道题目中，**「大问题」**很容易理解，就是对整个链接进行排序，那么**「子问题」**是什么？我们该如果去理解呢？

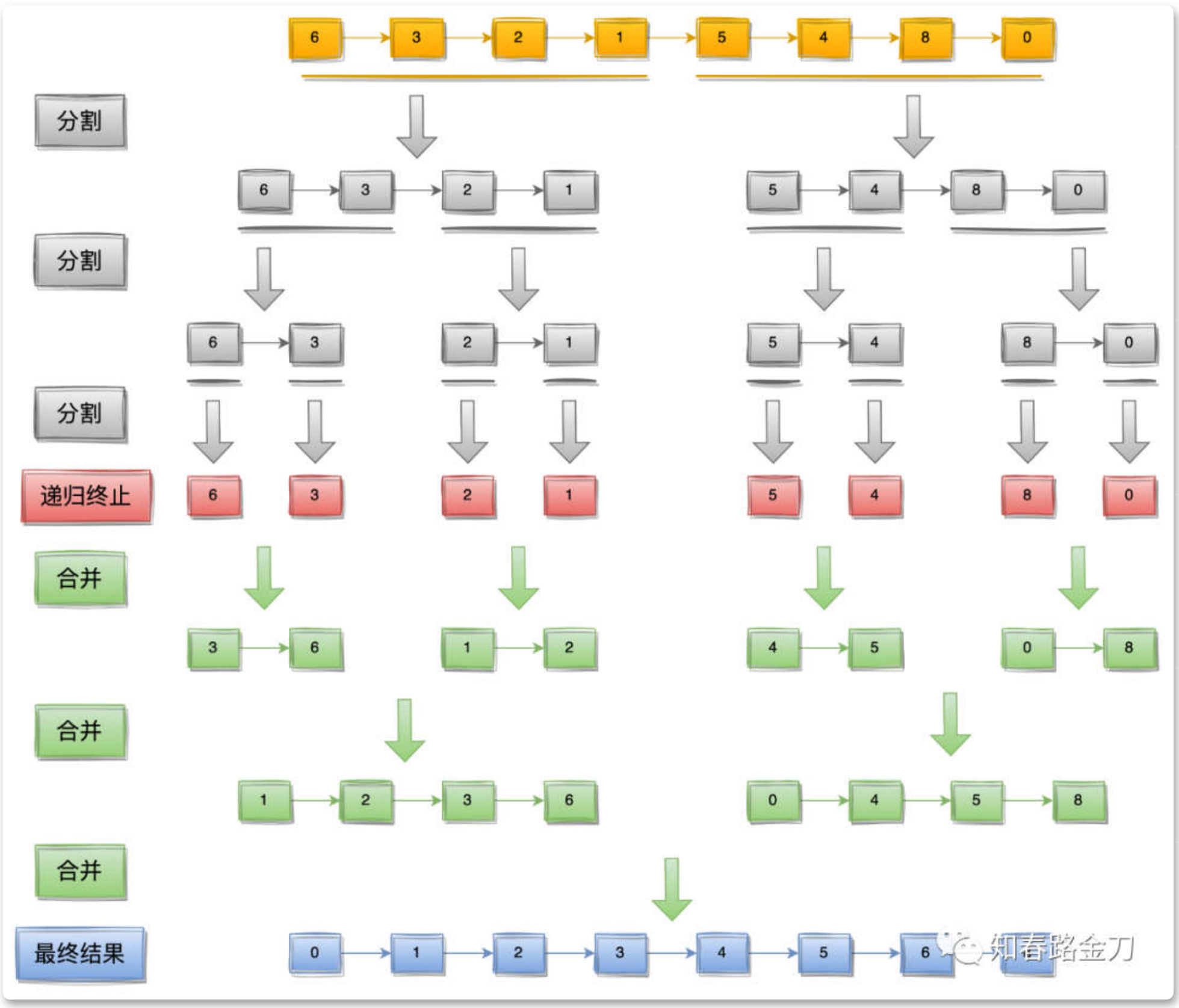
想一想看，如果给你两个已经排好序的链表，那么只需要对这两个已经排好序的链表按顺序进行合并就能得到一个有序的链表，所以‘子问题’意思是把对子链表进行排序。

‘归并排序’中的**「并」**字其实就代表对两个子链表进行合并，那么**「归」**字是什么意思呢？这道题可以使用递归的方式对子链表进行排序，所以**「归」**暗示着使用递归的方式对子问题进行求解，也就是使用递归来对子链表进行排序。

整体思路大概如下

1. 把无序链表head分成两个子链表，分别为headA和headB。
2. 使用递归的方式对headA和headB这两个子链表进行排序。
3. 按顺序对headA和headB这两个子链表进行合并。

先上图便于大家理解



代码实现

```
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode() {}
 *     ListNode(int val) { this.val = val; }
 *     ListNode(int val, ListNode next) { this.val = val; this.next = next; }
 * }
 */
class Solution {
    public ListNode sortList(ListNode head) {
        if (head == null) {
            return null;
        }
        ListNode mid = getMid(head);
        if (mid.next == null) {
            return mid;
        }
        ListNode headB = mid.next;
        mid.next = null;
        return mergeTwoList(sortList(head), sortList(headB));
    }
    public ListNode getMid(ListNode head) {
        head = new ListNode(0, head);
        ListNode slow = head;
        ListNode fast = head;
        while (fast != null && fast.next != null) {
            slow = slow.next;
            fast = fast.next.next;
        }
        return slow;
    }
    public ListNode mergeTwoList(ListNode l1, ListNode l2) {
        ListNode dump = new ListNode();
        ListNode last = dump;
        while (l1 != null && l2 != null) {
            if (l1.val <= l2.val) {
                last.next = l1;
                l1 = l1.next;
            } else {
                last.next = l2;
                l2 = l2.next;
            }
            last = last.next;
        }
        if (l1 != null) {
            last.next = l1;
        }
        if (l2 != null) {
            last.next = l2;
        }
        return dump.next;
    }
}
```

复杂度分析

N为链表的长度

- 时间复杂度为O(NlogN)
- 空间复杂度为O(logN)

面试tips

这道题考察的点很多，有链表归并，有获取链表的中间节点等等，每一个考点都可以作为一道题来进行考察。

参考资料

[1] **原题链接:**
https://leetcode-cn.com/problems/sort-list/solution/