1. 题目
2. 题干

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 。

进阶：

你可以在 O(n log n) 时间复杂度和常数级空间复杂度下，对链表进行排序吗？

1. 示例

**输入：**head = [4,2,1,3]

**输出：**[1,2,3,4]

1. 题解
2. 思路

之前我们使用插入排序对链表进行了排序，时间复杂度为O(n2)，这里我们使用归并排序进行优化，将时间复杂度降低到O(nlogn)。归并排序需要找到链表的中间节点，中间节点通过快慢指针来获取，然后对两边的链表分别进行递归归并排序，将两个排好序的链表再整合起来，整合的过程就是简单的判断大小，将小的节点添加到新的链表中。

1. 代码

Java：

/\*\*

 \* 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) {

        // 归并排序

        // 尾部是null

        return sortList(head, null);

    }

    public ListNode sortList(ListNode head, ListNode tail){

        if(head == null){

            return head;

        }

        if(head.next == tail){

            head.next = null;

            return head;

        }

        // 找到中间节点

        ListNode low = head, fast = head;

        while(fast != tail){

            fast = fast.next;

            low = low.next;

            if(fast != tail){

                fast = fast.next;

            }

        }

        ListNode mid = low;

        ListNode s1 = sortList(head, mid);

        ListNode s2 = sortList(mid, tail);

        // 归并两个队列

        ListNode sorted = merge(s1, s2);

        return sorted;

    }

    // 正常的归并算法

    public ListNode merge(ListNode head1, ListNode head2){

        ListNode dumHead = new ListNode(0);

        ListNode temp = dumHead, temp1 = head1, temp2 = head2;

        while(temp1 != null && temp2 != null){

            if(temp1.val <= temp2.val){

                temp.next = temp1;

                temp1 = temp1.next;

            }else{

                temp.next = temp2;

                temp2 = temp2.next;

            }

            temp = temp.next;

        }

        if(temp1 != null){

            temp.next = temp1;

        }else if(temp2 != null){

            temp.next = temp2;

        }

        return dumHead.next;

    }

}