合并两个有序的单链表

【题目】

给定两个有序单链表的头节点 head1 和 head2，请合并两个有序链表，合并后的链表依然有序，并返回合并后链表的头节点。

例如：

0->2->3->7->null

1->3->5->7->9->null

合并后的链表为：0->1->2->3->3->5->7->7->9->null

【难度】

士 ★☆☆☆

【解答】

本题比较简单，假设两个链表的长度分别为 M 和 N，直接给出时间复杂度为 $O(M+N)$、额外空间复杂度为 $O(1)$的方法。具体过程如下：

1．如果两个链表中有一个为空，说明无须合并过程，返回另一个链表的头节点即可。

2．比较 head1 和 head2 的值，小的节点也是合并后链表的最小节点，这个节点无疑应该是合并链表的头节点，记为 head；在之后的步骤里，哪个链表的头节点的值更小，另一个链表的所有节点都会依次插入到这个链表中。

3．不妨设 head 节点所在的链表为链表 1，另一个链表为链表 2。链表 1 和链表 2 都从头部开始一起遍历，比较每次遍历到的两个节点的值，记为 cur1 和 cur2，然后根据大小关系做出不同的调整，同时用一个变量 pre 表示上次比较时值较小的节点。

例如，链表 1 为 1->5->6->null，链表 2 为 2->3->7->null。

cur1=1，cur2=2，pre=null。cur1 小于 cur2，不做调整，因为此时 cur1 较小，所以令 pre=cur1=1，然后继续遍历链表 1 的下一个节点，也就是节点 5。

cur1=5，cur2=2，pre=1。cur2 小于 cur1，让 pre 的 next 指针指向 cur2，cur2 的 next 指针指向 cur1，这样，cur2 便插入到链表 1 中。因为此时 cur2 较小，所以令 pre=cur2=2，然后继续遍历链表 2 的下一个节点，也就是节点 3。这一步完成后，链表 1 变为

1->2->5->6->null，链表 2 变为 3->7->null，cur1=5，cur2=3，pre=2。

cur1=5，cur2=3，pre=2。此时又是 cur2 较小，与上一步调整类似，这一步完成后，链表 1 变为 1->2->3->5->6->null，链表 2 为 7->null，cur1=5，cur2=7，pre=3。

cur1=5，cur2=7，pre=3。cur1 小于 cur2，不做调整，因为此时 cur1 较小，所以令 pre=cur1=5，然后继续遍历链表 1 的下一个节点，也就是节点 6。

cur1=6，cur2=7，pre=5。cur1 小于 cur2，不做调整，因为此时 cur1 较小，所以令 pre=cur1=6，此时已经走到链表 1 的最后一个节点，再往下就结束，如果链表 1 或链表 2 有任何一个走到了结束，就进入步骤 4。

4．如果链表 1 先走完，此时 cur1=null，pre 为链表 1 的最后一个节点，那么就把 pre 的 next 指针指向链表 2 当前的节点（即 cur2），表示把链表 2 没遍历到的有序部分直接拼接到最后，调整结束。如果链表 2 先走完，说明链表 2 的所有节点都已经插入到链表 1 中，调整结束。

5．返回合并后链表的头节点 head。

全部过程请参看如下代码中的 merge 方法。

```
public class Node {
        public int value;
        public Node next;

        public Node(int data) {
                this.value = data;
        }
}

public Node merge(Node head1, Node head2) {
        if (head1 == null || head2 == null) {
                return head1 != null ? head1 : head2;
        }
        Node head = head1.value < head2.value ? head1 : head2;
        Node cur1 = head == head1 ? head1 : head2;
        Node cur2 = head == head1 ? head2 : head1;
        Node pre = null;
        Node next = null;
        while (cur1 != null && cur2 != null) {
                if (cur1.value <= cur2.value) {
                        pre = cur1;
                        cur1 = cur1.next;
                } else {
                        next = cur2.next;
                        pre.next = cur2;
                        cur2.next = cur1;
                        pre = cur2;
                        cur2 = next;
```

```
                }
            }
            pre.next = cur1 == null ? cur2 : cur1;
            return head;
        }
```

按照左右半区的方式重新组合单链表

【题目】

给定一个单链表的头部节点 head，链表长度为 N，如果 N 为偶数，那么前 $N/2$ 个节点算作左半区，后 $N/2$ 个节点算作右半区；如果 N 为奇数，那么前 $N/2$ 个节点算作左半区，后 $N/2+1$ 个节点算作右半区。左半区从左到右依次记为 L1->L2->…，右半区从左到右依次记为 R1->R2->…，请将单链表调整成 L1->R1->L2->R2->…的形式。

例如：

1->null，调整为 1->null。

1->2->null，调整为 1->2->null。

1->2->3->null，调整为 1->2->3->null。

1->2->3->4->null，调整为 1->3->2->4->null。

1->2->3->4->5->null，调整为 1->3->2->4->5->null。

1->2->3->4->5->6->null，调整为 1->4->2->5->3->6->null。

【难度】

士 ★☆☆☆

【解答】

假设链表的长度为 N，直接给出时间复杂度为 $O(N)$、额外空间复杂度为 $O(1)$的方法。具体过程如下：

1．如果链表为空或长度为 1，不用调整，过程直接结束。

2．链表长度大于 1 时，遍历一遍找到左半区的最后一个节点，记为 mid。

例如：1->2，mid 为 1；1->2->3，mid 为 1；1->2->3->4，mid 为 2；1->2->3->4->5，mid 为 2；1->2->3->4->5->6，mid 为 3。也就是说，从长度为 2 开始，长度每增加 2，mid 就往后移动一个节点。