链表

1. 翻转全部 ⭐ ⭐ ⭐

- 栈：O(n) ; O(n)
- 原地迁移：O(n); O(1)
  - 新建头节点 null
  - temp 节点指向cur.next
  - cur.next 指向新建头
  - 新建头更新至cur
  - cur 更新至temp
- 递归：O(n); O(n) 递归的空间复杂度需要考虑递归栈的栈空间
  - 返回条件：head==null 或 head.next == null
  - 用next节点记录head.next
  - 递归反转head.next
  - next.next 设为head
  - head.next 设为null

2. 翻转区间

- 先移动m
- 用1的方法翻转n次

3. 每k个翻转

- 递归：
  - 找到下一组的开头tail
  - 在cur 不等于tail之前，一路反转
  - 本轮头节点的next 设为ReverseKGroup(tail, k)

4. 合并两个已排序的链表 ⭐

- 归并

5. 合并k 个已排序的链表 ⭐ ⭐ ⭐

- 最小堆归并
- 堆内存n个待合并链表的头节点

6. 判断是否有环 ⭐ ⭐ ⭐

- 快慢双指针：
  - slow指针每次前进一个，fast指针每次前进两个
    - 若相遇，说明有环
    - 若没有相遇而是fast 为null，说明没有环

7. 环的入口节点 ⭐

- 6中，快慢指针相遇处一定在环内
- slow 从相遇节点出发

- fast 重新从头出发
- 依然按照快2慢1的步调同时出发，再次相遇时即为环入口处

8. 倒数最后k个节点 ⭐⭐⭐

- 双指针：O(n); O(1)
  - fast 先走k步
  - 随后双指针同步向后

9. 删除链表的倒数第n个节点

- 双指针：O(n), O(1)
  - 按照8 找到倒数第n个节点
  - pre.next = pre.next.next

10. 两个链表的第一个公共节点

- 双指针：令二者走相同的路径
  - 同步推进
  - 当其中一指针为null 时，将其指向另一链表的头

11. 链表相加 ⭐⭐

- 先反转，再从个位开始相加

12. 单链表排序 ⭐⭐⭐

- **辅助数组：T = O(NlogN); S = O(N)**
- **原地分治排序：T = O(NlogN); S = O(1)**
  - 快慢指针，slow，fast将链表分为两半
  - left = sortInList(head); right = sortInList(slow.next);
  - 将left 和 right **归并排序（11）**

13. 判断链表是否回文

- p1 正向遍历
- p2 遍历反转链表

14. 奇偶重排 ⭐⭐

- 双指针
  - odd = head, even = head.next
  - while (even != null && even.next != null)
    - odd.next = even.next
    - odd = odd.next
    - even.next = odd.next
    - even = even.next

15. 删除重复-I & 留一个

- while (cur.next != null)
  - if (cur.val == cur.next.val)
    - cur.next = cur.next.next

    - else
        - cur = cur.next

16. 删除重复-II & 一个不留

- dummy.next = head;
- cur = dummy
- while (cur.next != null && cur.next.next != null)
    - if (cur.next.val == cur.next.next.val)
        - while()
            - del
    - else
        - cur = cur.next