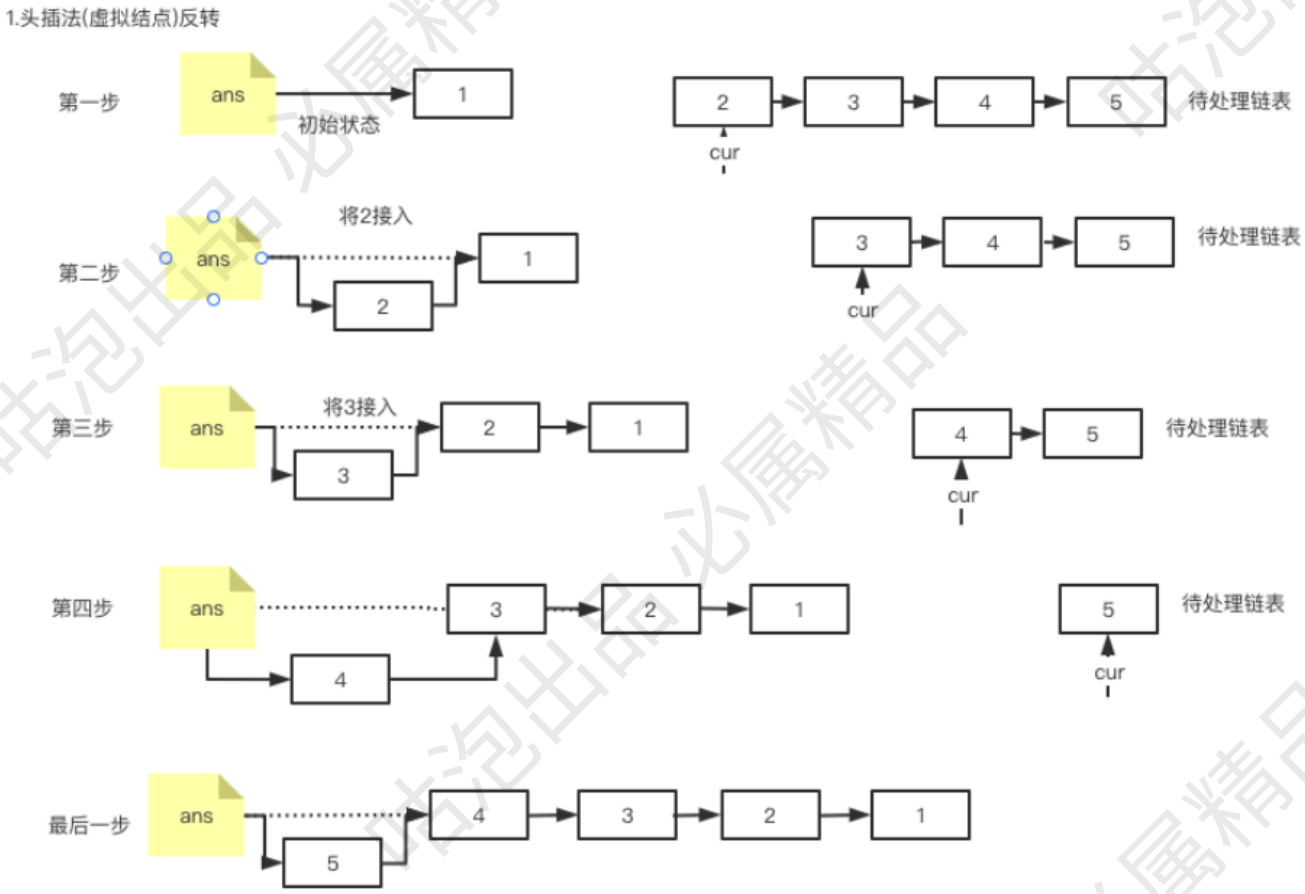
# 第二次直播

## 1.链表反转

重点学习：

1.链表整体反转

2.三个拓展：在链表上指定一个区间来反转、K个一组反转（最难的）、两两一组反转




```
    while (cur != null) {
        ListNode next = cur.next;
        cur.next = ans.next;
        ans.next = cur;
        cur = next;
    }
```

## 虚拟头结点：

1.思想：创建一个空的结点ans(-1)，然后让ans.next=head。

2.这么做的好处：可以不用考虑链表head的元素的问题（增加和删除来体会）

3.反转链表的过程：依次从旧的链表上拿到结点，此时可以逐个接到ans的后面，（插队的过程），全部完成之后，ans后面的元素就被反转了。

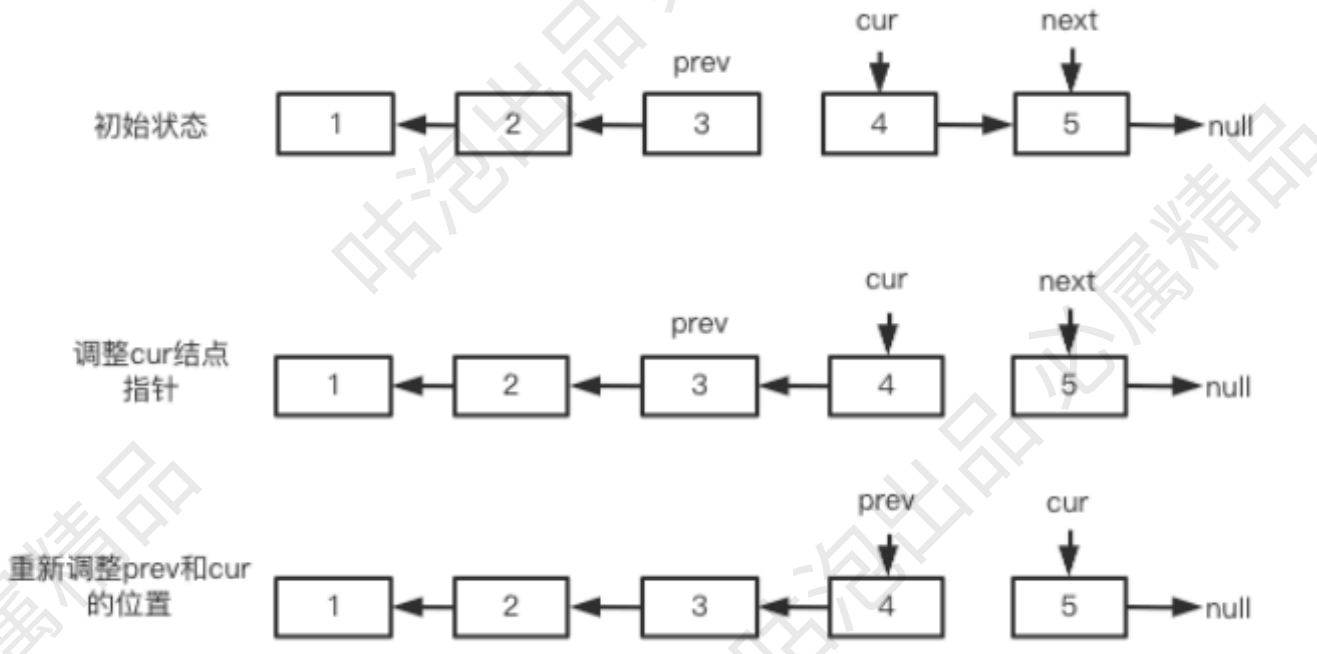
4.指定区间反转[left,right]：此时链表的第left-1个元素，就是我们上面的ans。

5.处理完之后，虚拟头结点不用删掉

ans.next就是head

```
f () {
ans
return ans.next;
}
```

# 不带头结点



```
while(){
next=cur.next
cur.next=prev
prev=cur
cur=next
}
```

**链表反转作业：**

1.增加和删除（帮助理解带和不带虚拟头的区别）

2.对照图理解反转的过程

核心点：我们改的是结点的next的指向，而不是结点的位置

对照讲义的图，注意理解起始状态和最终状态的变化。

留意中间过程是如何进行的

# 双向链表

本质上还是链表调整指针的指向

增删如何进行。

# 2.数组

双指针->滑动窗口

## 2.1 数组的存储特征

```
 int[] arr = new int[10]
【0 0 0 0 0 0 0 0 0 0 】

我们认为的初始化：
【1 2 3 4 0 0 0 0 0 0 】
其实计算机用新的数据将需要的位置(0)给覆盖了

【1 2 0 4 0 0 5 0 0 9 0 】


【1 2 4 5 9 9 null 0 0 null 0 】


【1 2 4 5 9  10 | 0 -1 10 9 5 】
size=5+1=6
```

```
array[4]=array[9]

插入10


有效元素有多少?
arr.length：表示的是数组空间的大小，数组不变，length就不变
增加一个变量来标记数组中的有效元素个数：size
```

## 2.2 作业

自己基于数组搞增删改查，保证在数组的首位置、末尾和中间操作元素都可以。

## 2.3 双指针和应用

### 快慢型双指针：

基本思想：定义两个变量，slow和fast，fast负责向后遍历元素，从0到slow的位置是我们最终想要的结果。

fast的移动条件：逐个遍历整个数组的元素。

slow移动的条件：当fast找到某个满足特定要求的元素才会移动

最终fast到达数组终点，此时从0到slow的位置是我们最终想要的结果。

## 对撞型

基本思想：定义两个变量，left和right，从两头向中间走。

left和right的移动条件基本是相反的，具体根据题目的要求进行。

移动的目标是将右侧的有效元素与左侧的无效元素交换，

最终的结果是从0到left，就是我们要的结果。right向后的都是无效元素。

拓展：快速排序：对撞型双指针+二叉树的前序遍历

# 拓展

拓展1：从双指针到滑动窗口

拓展2：队栈Hash，比较简单，自己看一下就行了。

# 学习的优先级

1.链表基础->反转（必须搞清楚），K个一组量力而行，尽量搞定。

2.数组的特征->数组的增删改查，数组讲义里的题目不必都做，到2.8就可以了。

3.双指针思想和经典题目

4.队栈Hash

5.滑动窗口