

链表的双指针技巧

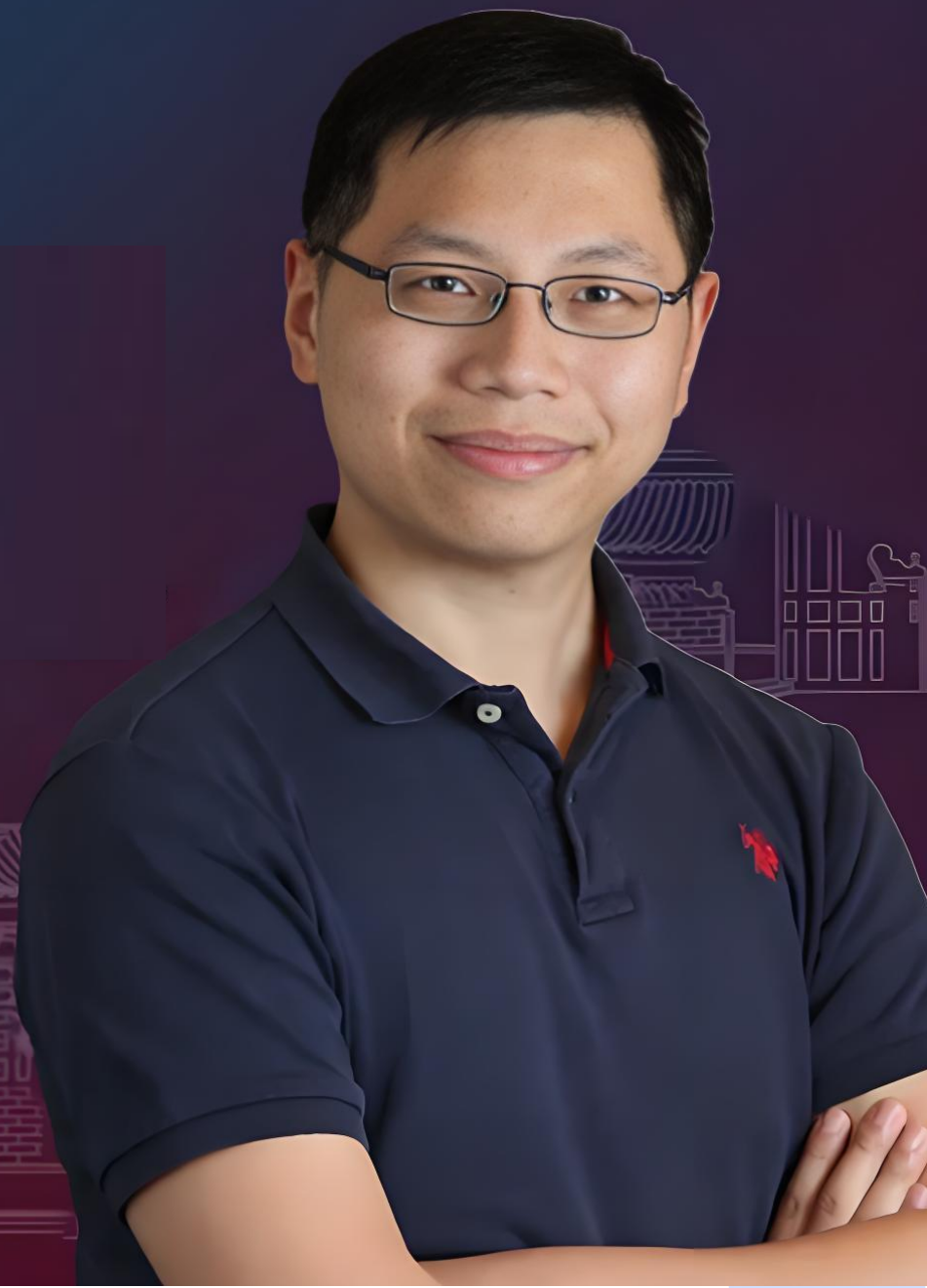
- 找单链表倒数第 K 个结点
- 找单链表中间结点
- 反转单链表
- 单链表求交点
- 单链表判环
- 其他

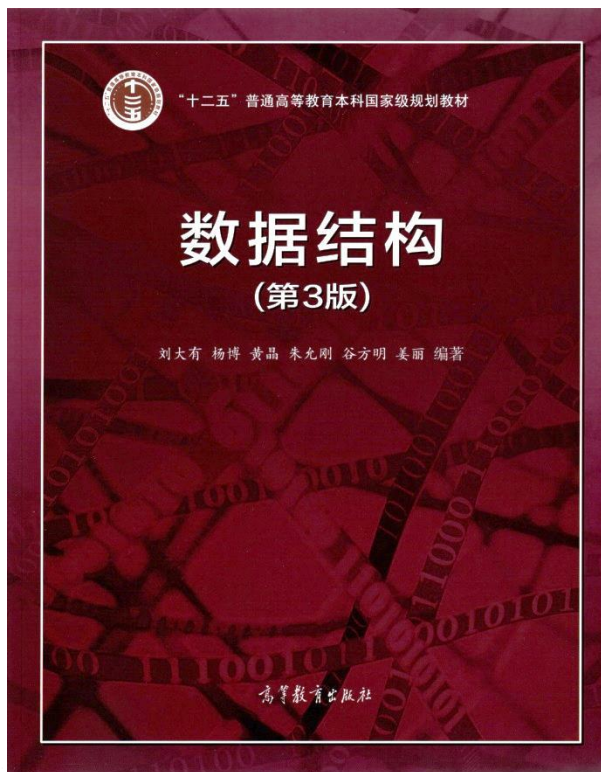
数据之法
结构之美
算法之道

数据结构与算法的高效学习方法
就是上手实践，从做小练习题到
实现大项目，不断提升对其掌握
程度。

——李沐

上海交通大学学士
卡内基梅隆大学博士
前亚马逊资深首席科学家





链表的双指针技巧

- 找单链表倒数第 K 个结点
- 找单链表中间结点
- 反转单链表
- 单链表求交点
- 单链表判环
- 其他

数据之法
结构之美
算法之道

找单链表倒数第 k 个结点

给定一个不含哨位结点的单链表，请设计一个尽可能高效的算法，**查找链表中倒数第 k 个结点**（ k 为正整数）。若查找成功，算法返回该结点的数据域之值；否则返回0。【**考研题全国卷，大厂面试题**LeetCode】

```
struct ListNode{
    int val;
    ListNode* next;
};
```

➤解法1:

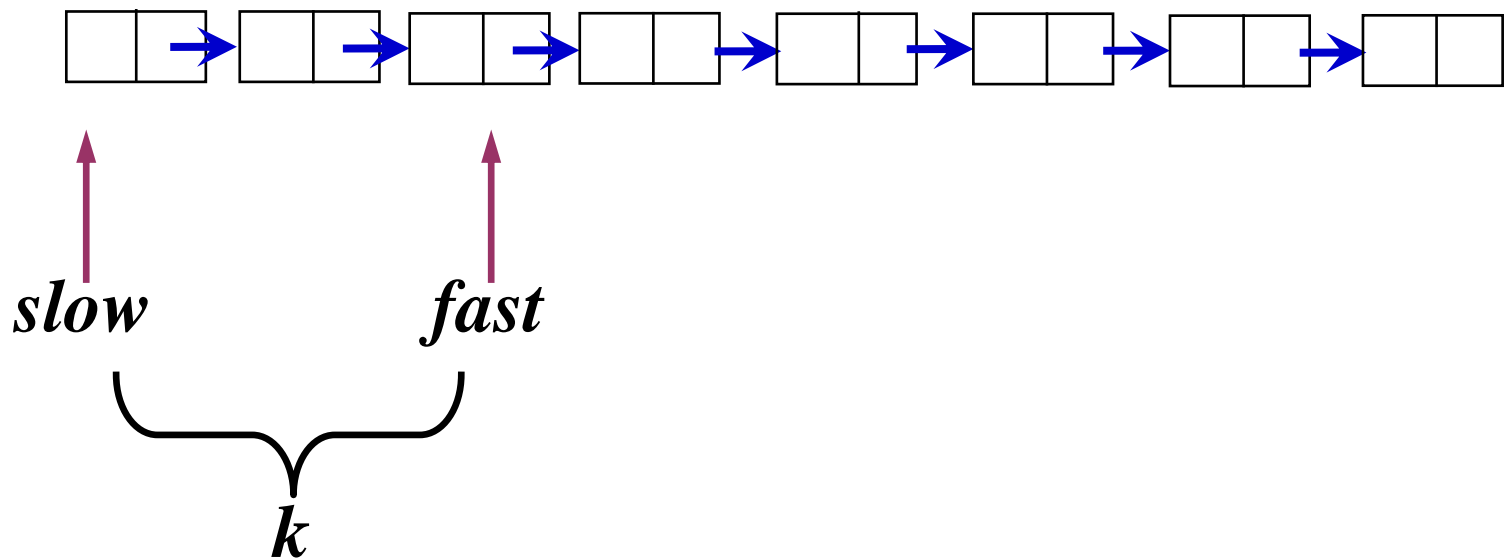
- ✓ 先找到最后一个结点，再往前找 $k-1$ 次前驱
- ✓ 找某个结点的前驱结点：时间 $O(n)$
- ✓ 整个算法时间复杂度 $O(n^2)$

➤解法2:

- ✓ 事实：倒数第 k 个，即正数第 $n-k+1$ 个， n 为链表长度
- ✓ 先遍历一遍链表，确定链表长度 n ，再遍历一遍链表找第 $n-k+1$ 个结点
- ✓ 遍历2次链表，时间复杂度 $O(n)$

➤ 解法3:

- ✓ 使用两个指针:*fast*和*slow*, 先把*fast*指向第*k*个结点, *slow*指向第1个结点。然后*fast*和*slow*同时并行向后移动, 当*fast*移动到最后一个结点时, *slow*正好指向倒数第*k*个结点。
- ✓ 只遍历1次链表, 时间复杂度 $O(n)$ 。



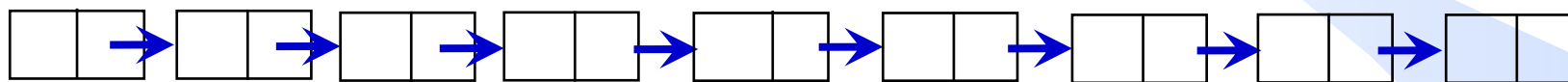
```
int kthToLast(ListNode* head, int k) {  
    ListNode *slow = head, *fast = head;  
    for(int i = 1; fast != NULL && i < k; i++)  
        fast = fast->next;    //让fast指向正数第k个结点  
    if(fast==NULL) return 0;    //链表长度不足k  
    while(fast->next != NULL) {  
        slow = slow->next;  
        fast = fast->next;  
    }  
    return slow->val;  
}
```

Talk is cheap,
show me the code



找单链表中间结点

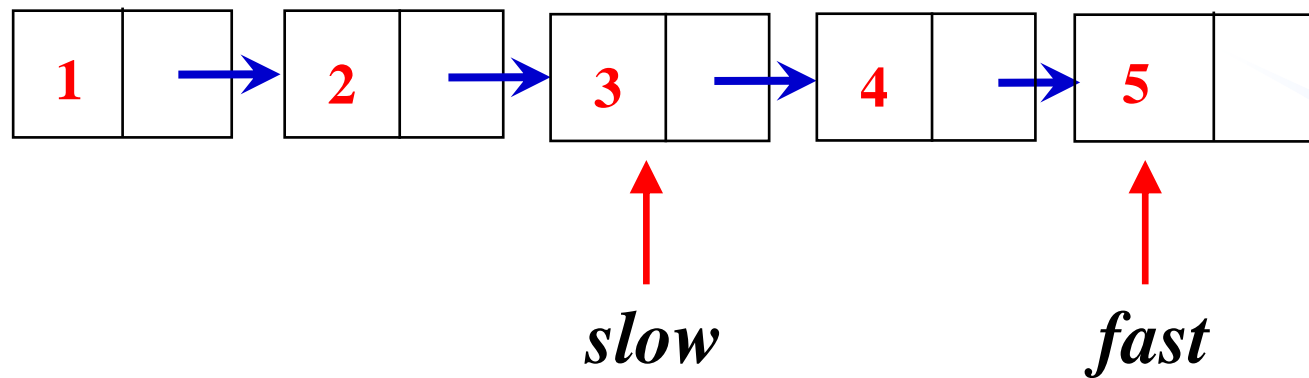
找单链表中间位置的结点，要求只遍历一次链表。若链表长度为偶数，返回两个中间结点中靠右的那个结点。【大厂面试题 [LeetCode 876](#)】



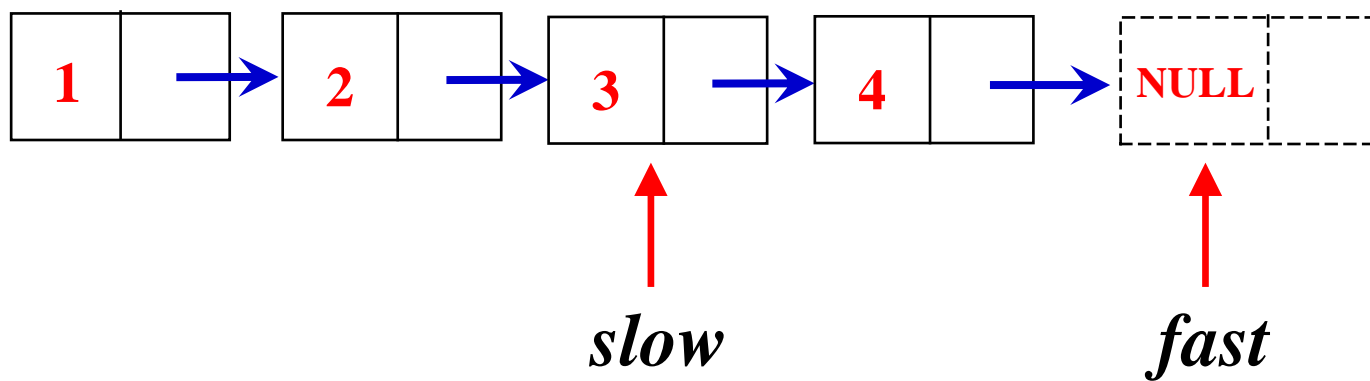
- ✓ 维护两个指针fast和slow
- ✓ slow每次移动1步，fast每次移动2步

```
slow = slow->next;  
fast = fast->next->next;
```


扫描结束的条件



`fast->next==NULL`



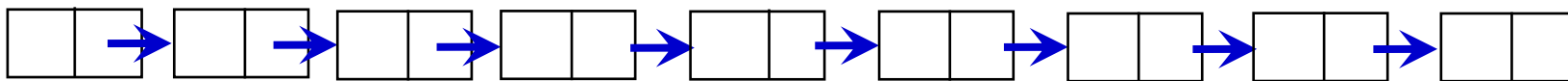
`fast==NULL`

Talk is cheap, show me the code

B

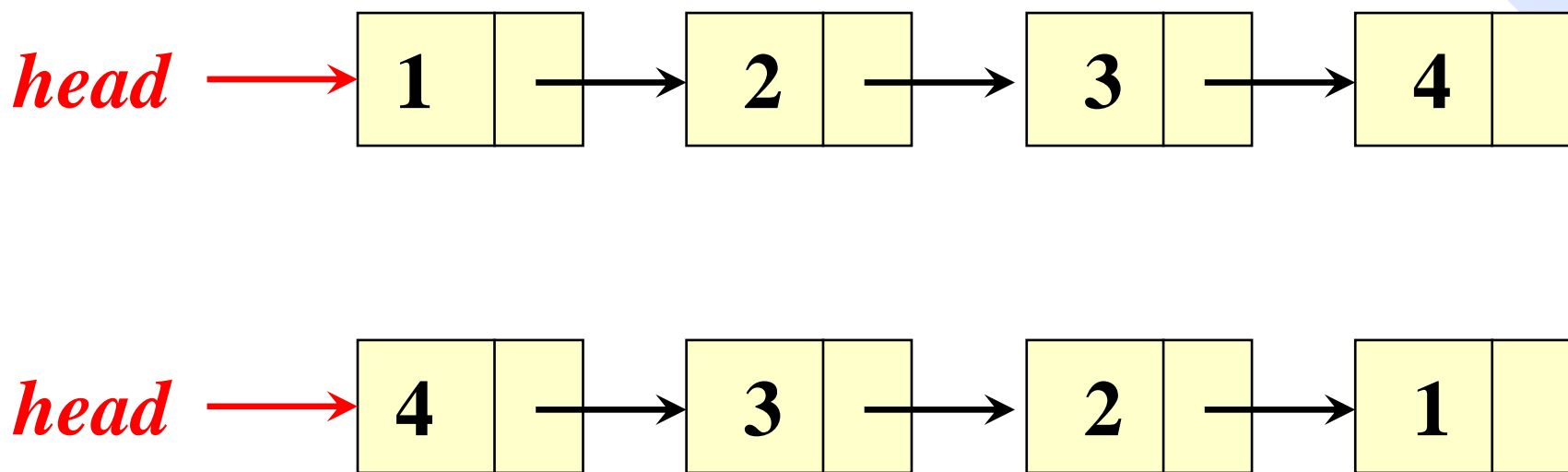
```
ListNode* middleNode(ListNode* head) {  
    ListNode *fast = head, *slow = head;  
    while (fast != NULL && fast->next != NULL) {  
        slow = slow->next;  
        fast = fast->next->next;  
    }  
    return slow;  
}
```

课下思考：对于长度为偶数的链表，若要返回第1个中间结点（即两个中间结点中靠左的那个结点），该如何修改上述代码？

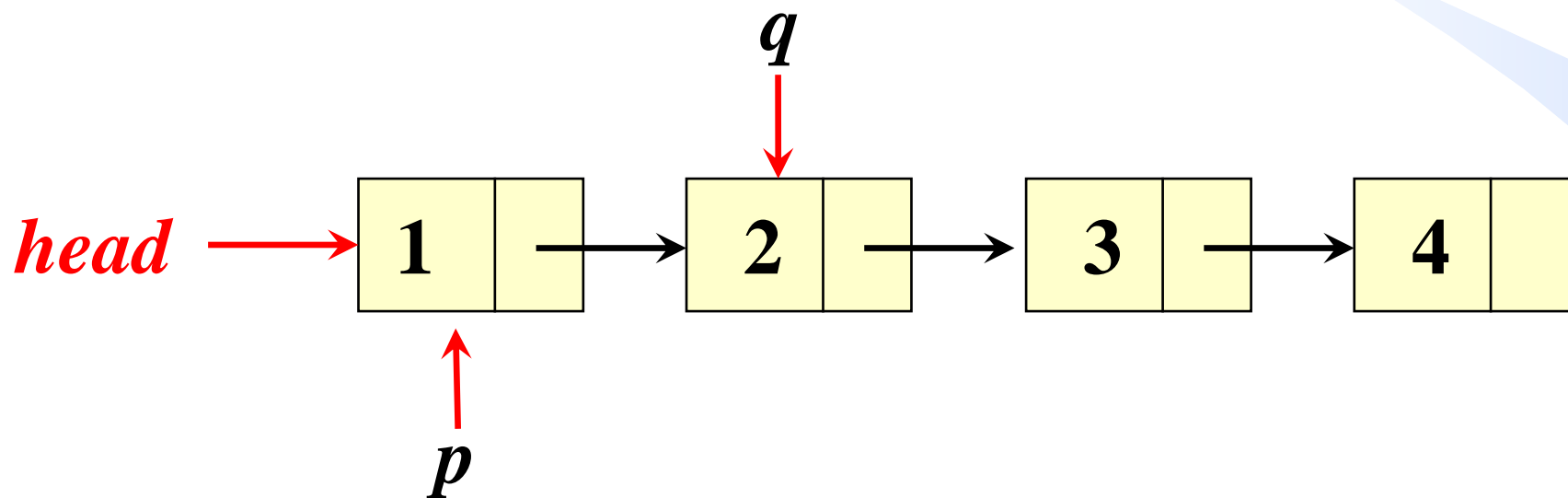


反转单链表

给定单链表的头指针`head`，请设计算法反转链表，并返回反转后的链表头指针。要求算法空间复杂度为 $O(1)$ 且不能只单纯改变结点数据域值，而要对结点的指针进行实际的反转。【大厂面试题[LeetCode 206](#)】



反转单链表

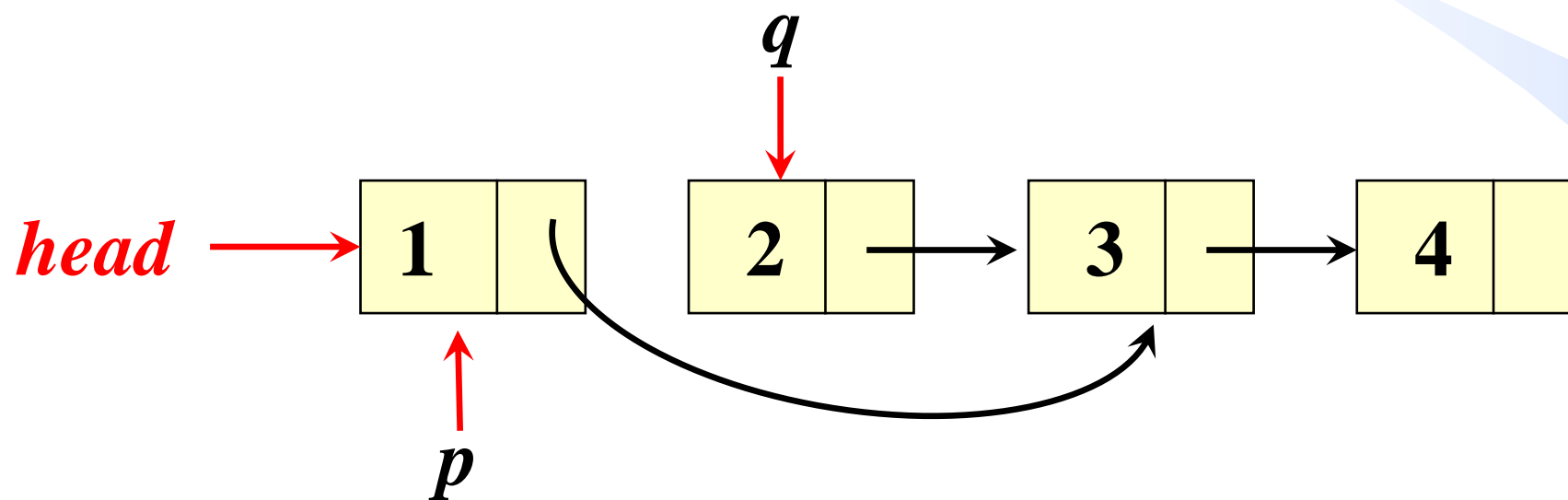


空链表不用翻转

令 p 指向第1个结点
 q 指向 p 的后继

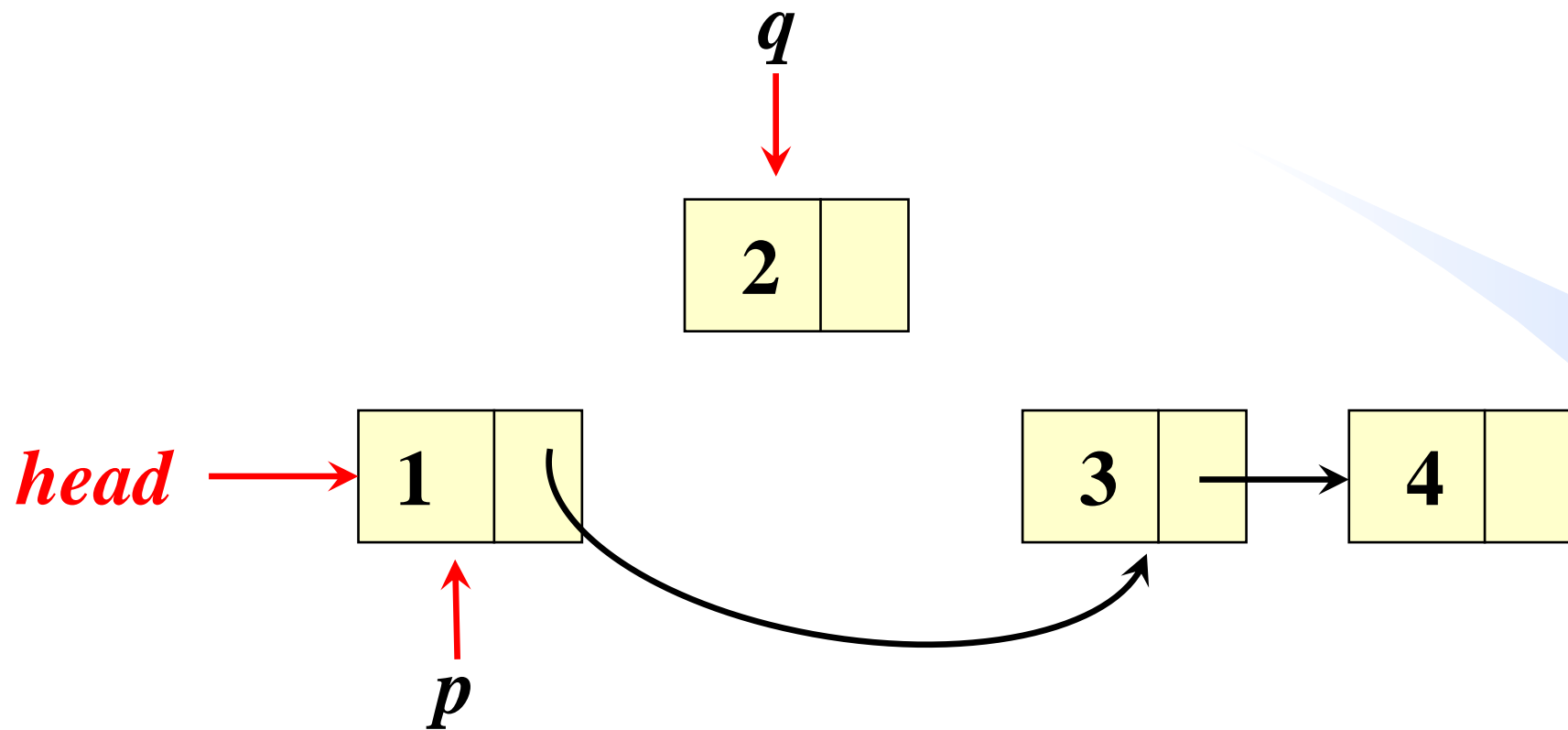
把 q 从链表中摘出

反转单链表



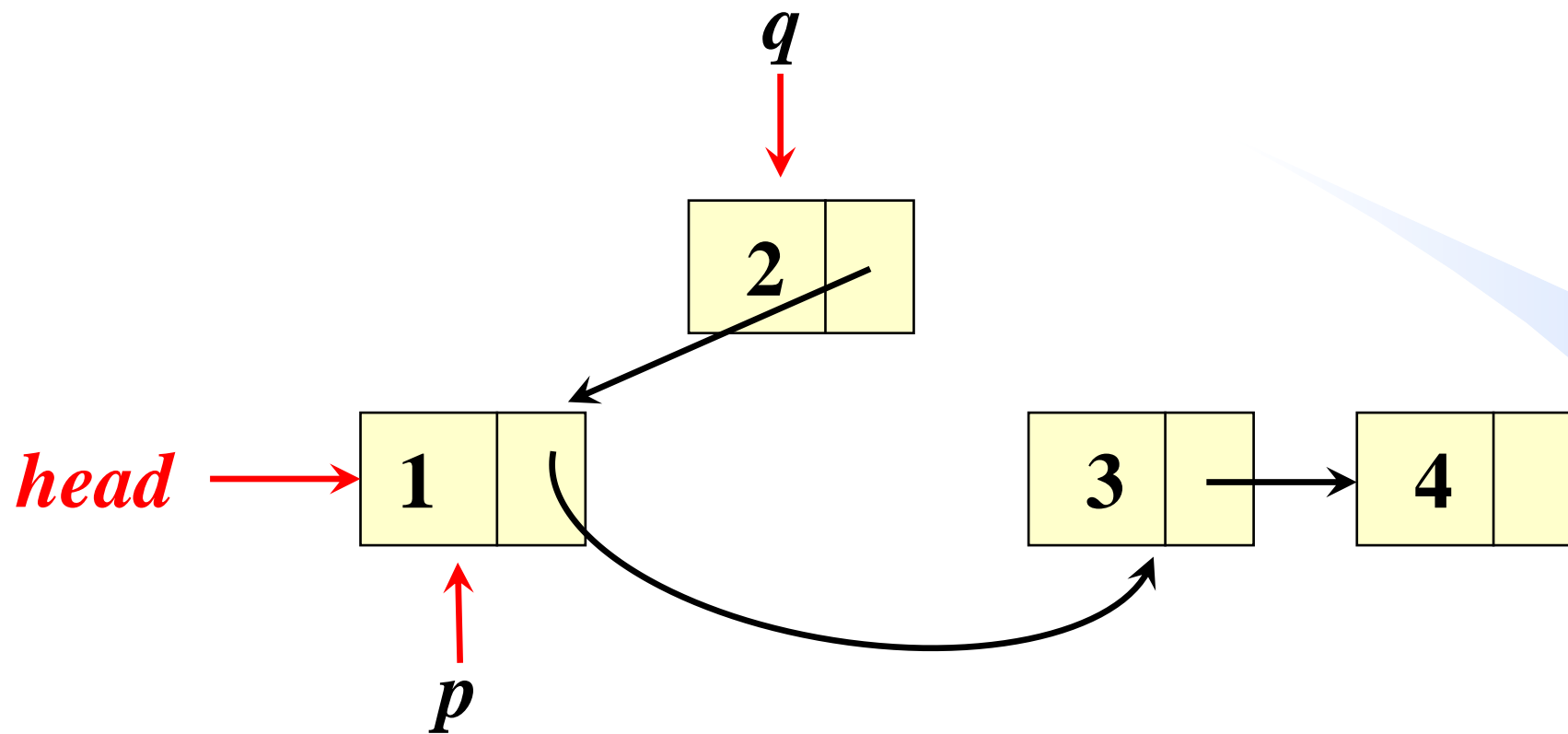
把 q 从链表中摘出

反转单链表



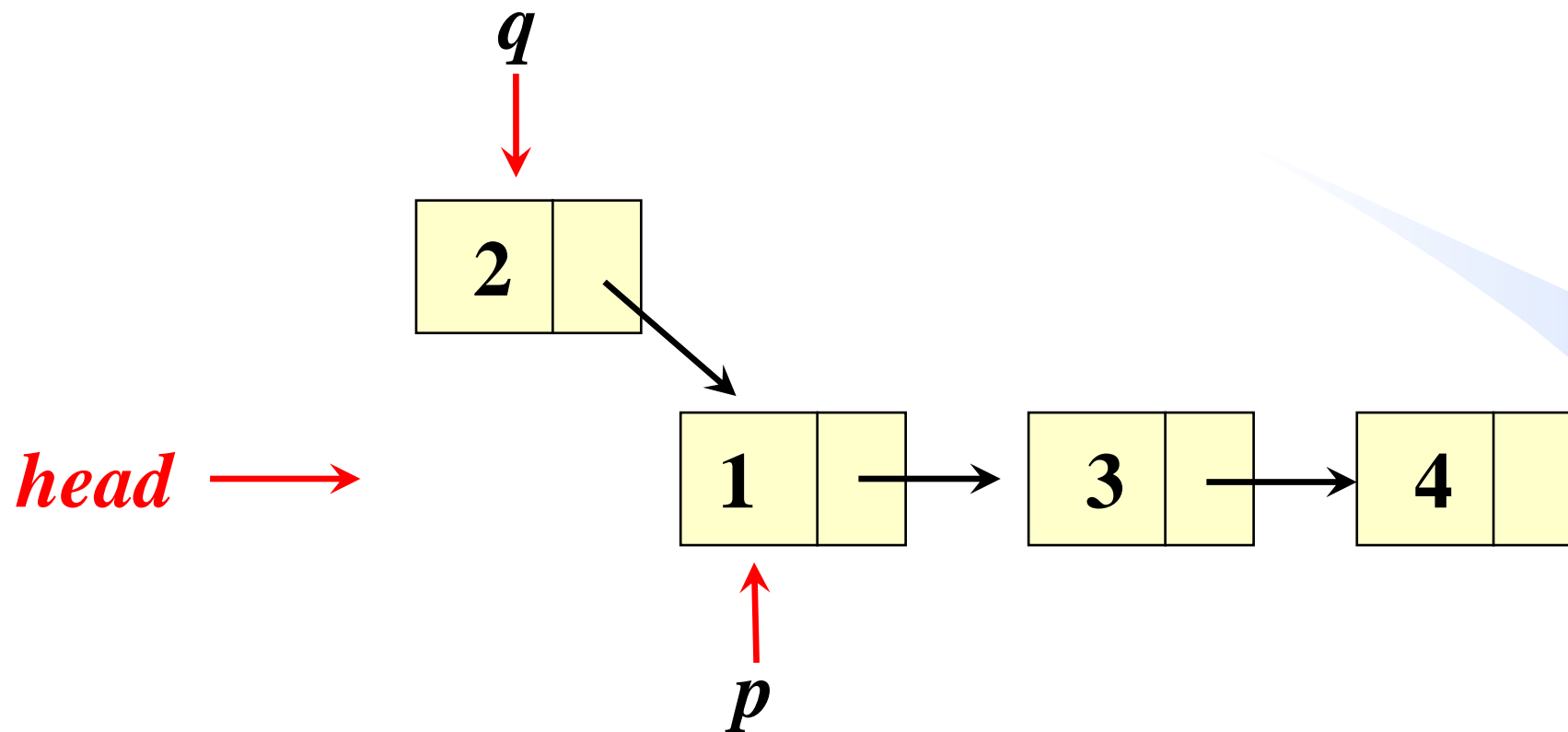
把 q 从链表中摘出

反转单链表



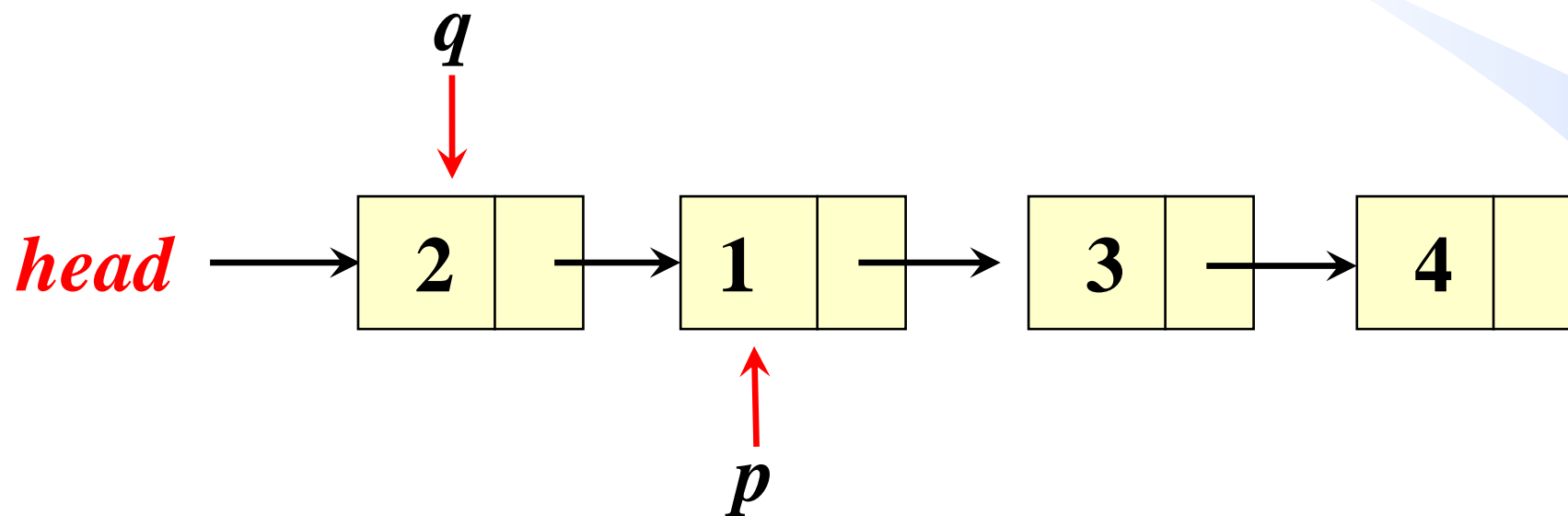
把 q 插入到表头

反转单链表



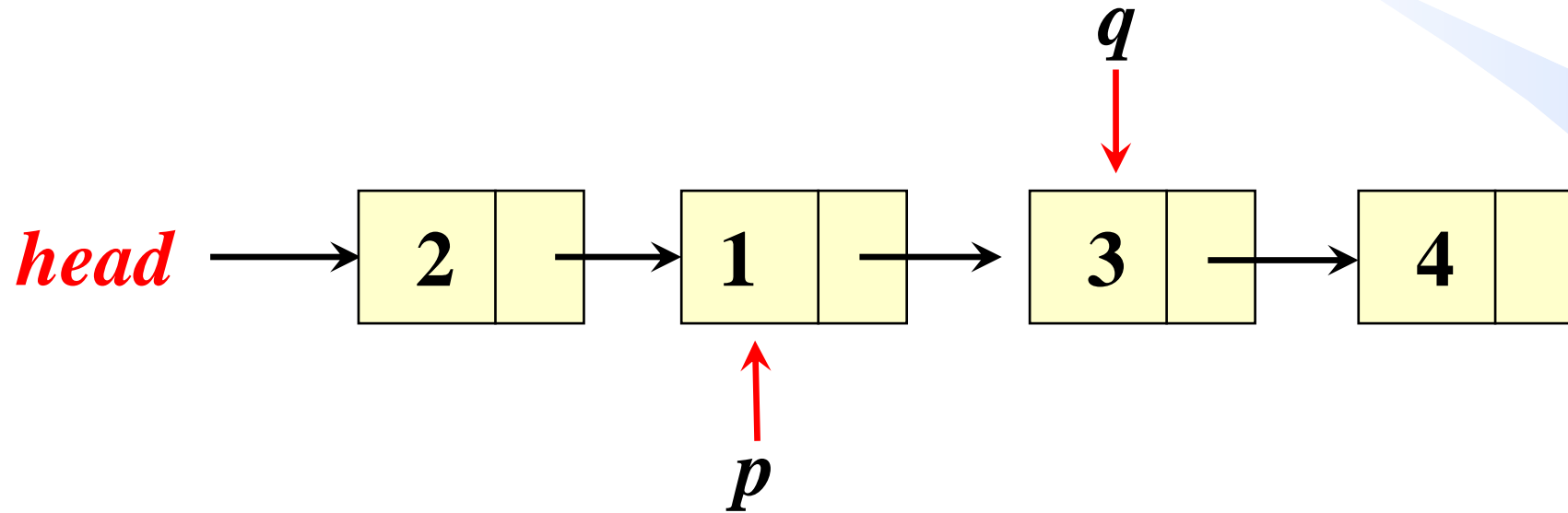
把 q 插入到表头

反转单链表



把 q 插入到表头

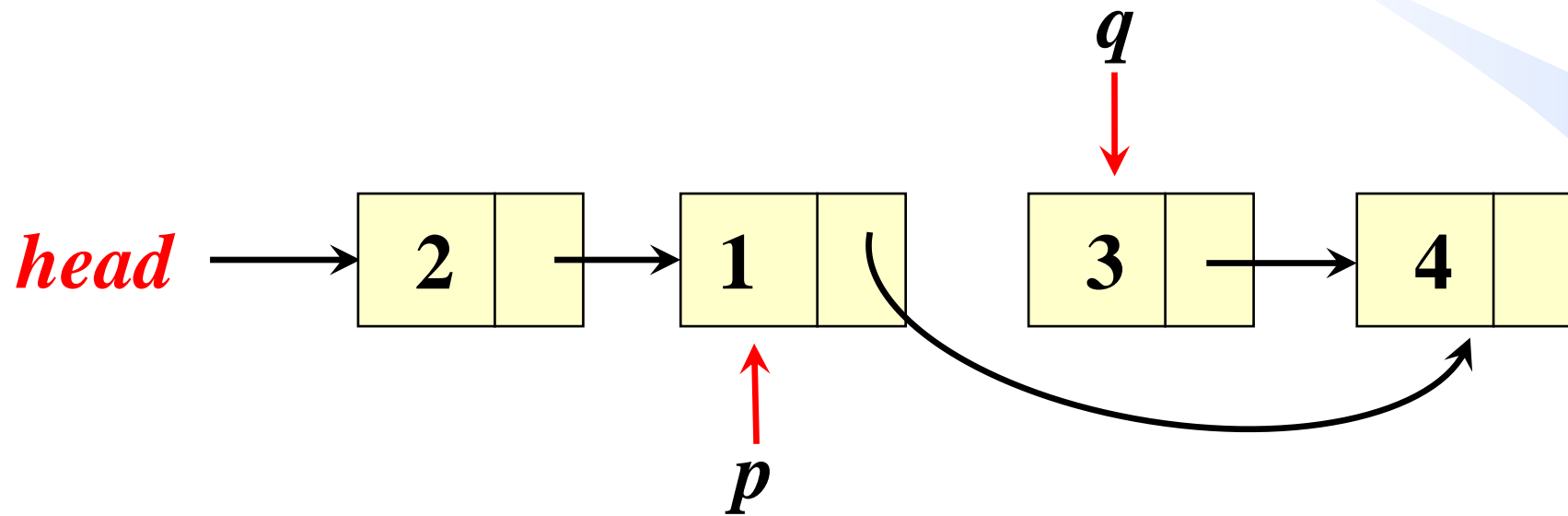
反转单链表



令 q 指向 p 的后继

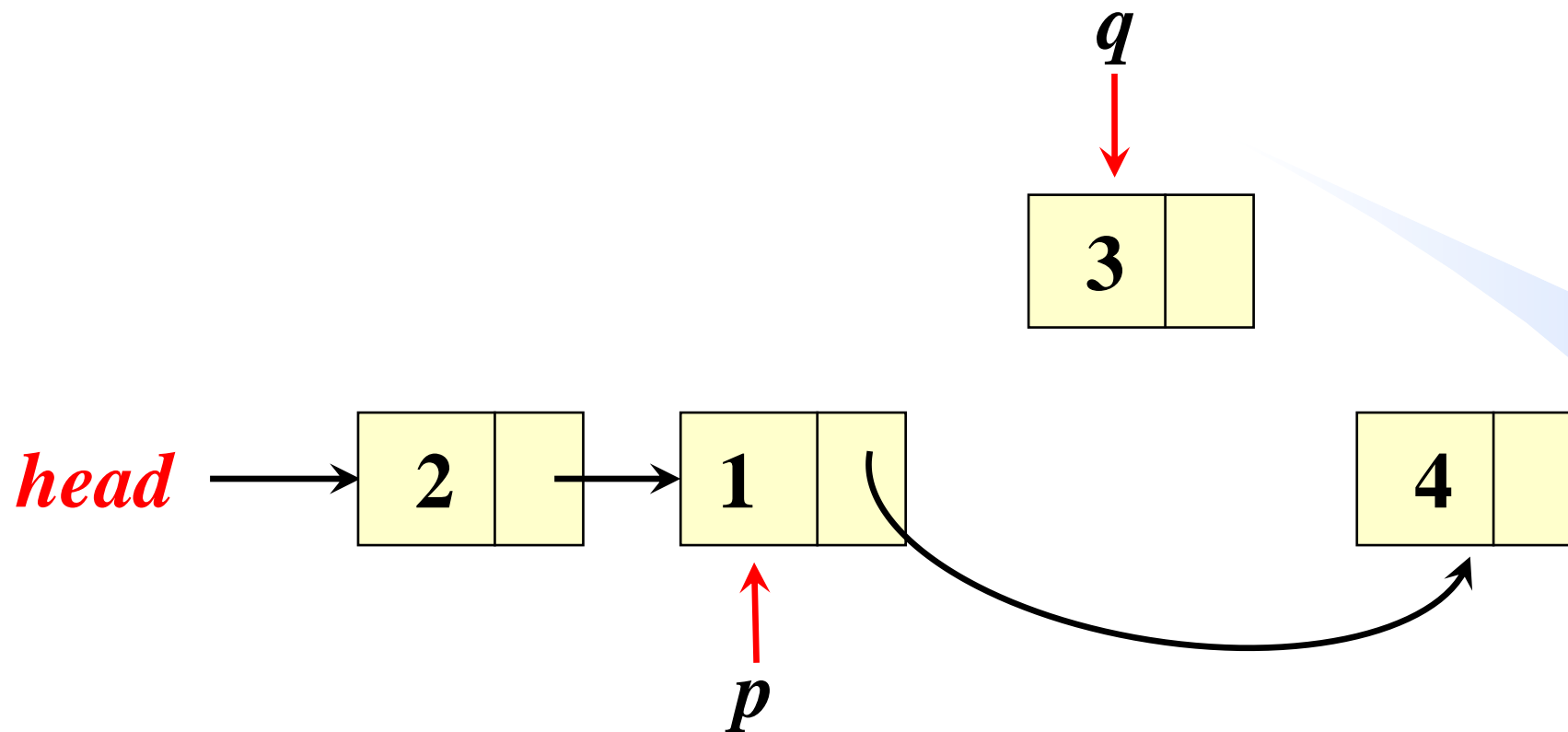
把 q 从链表中摘出

反转单链表



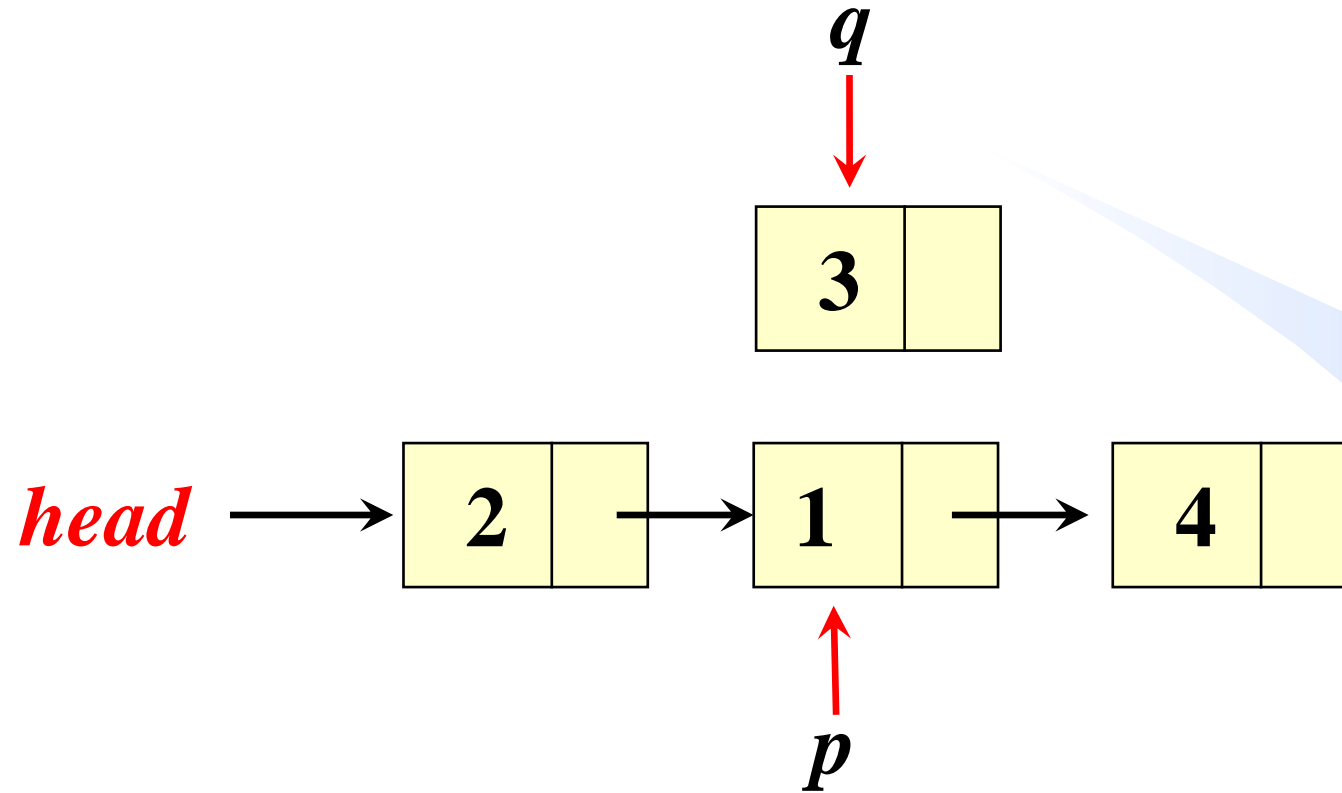
把 q 从链表中摘出

反转单链表



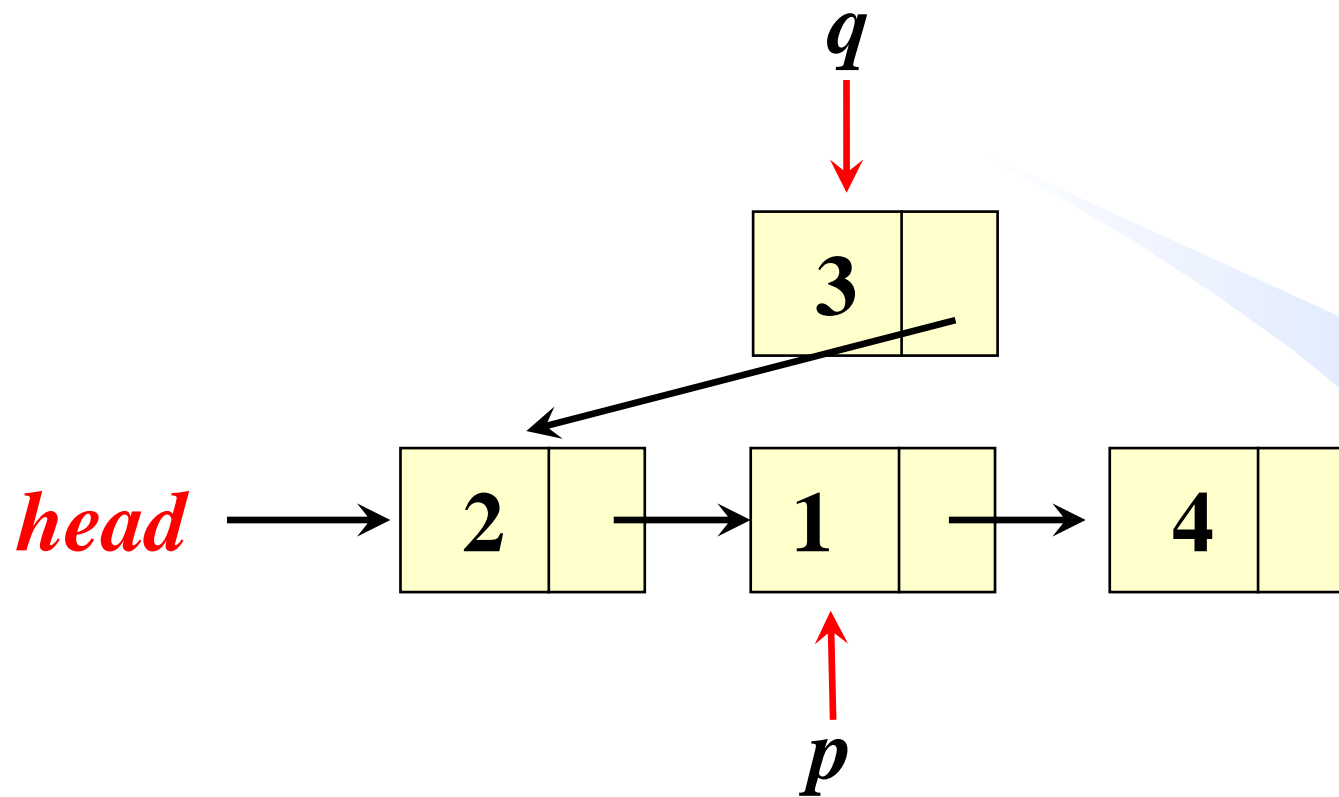
把 q 从链表中摘出

反转单链表



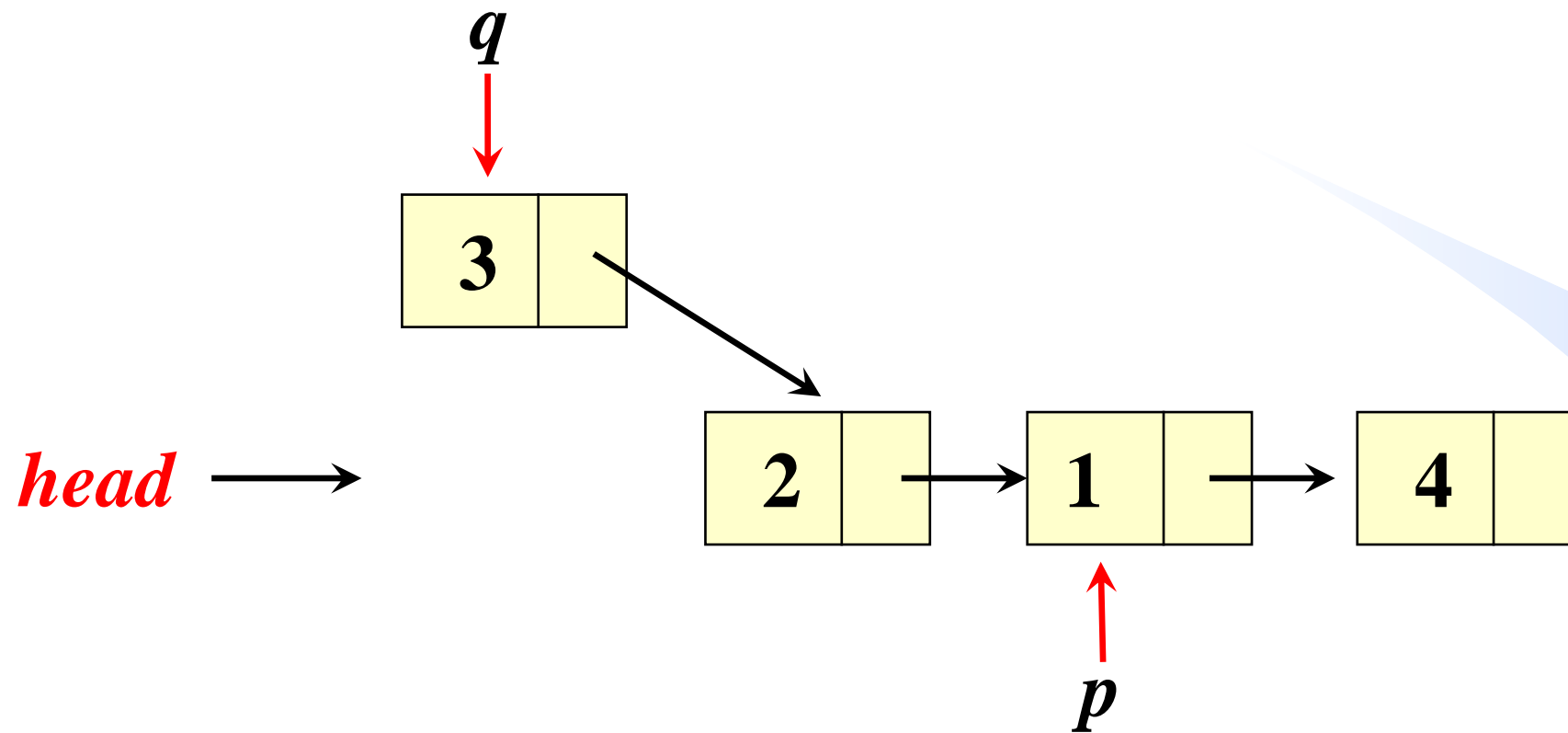
把 q 插入到表头

反转单链表



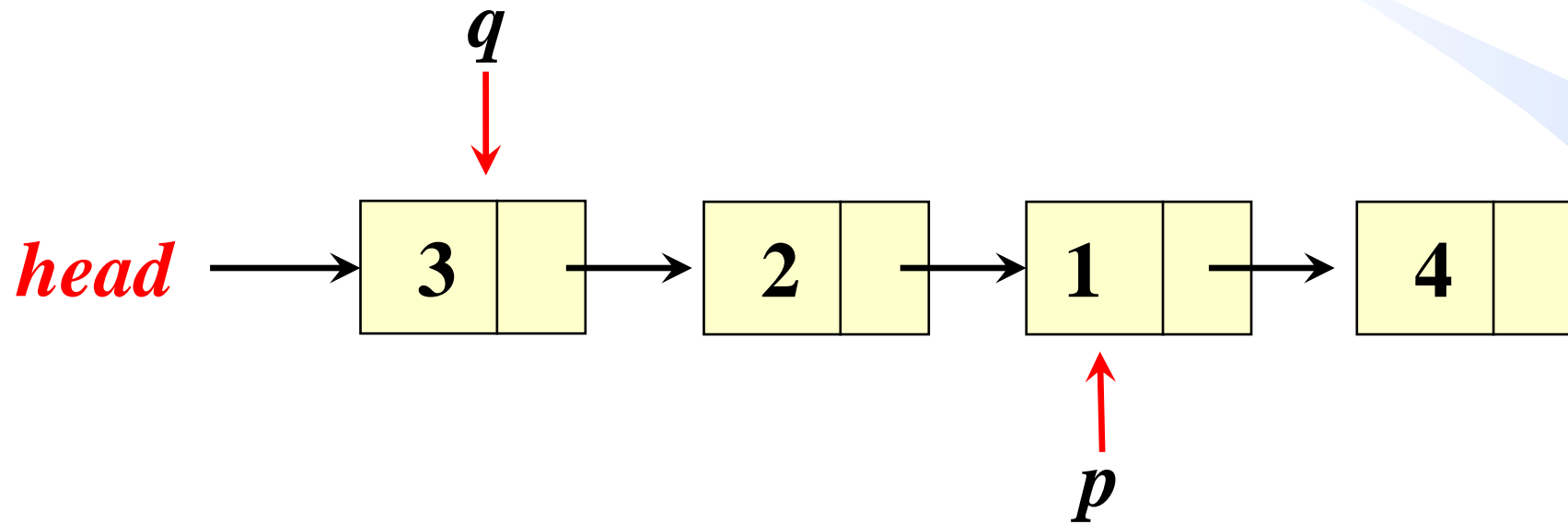
把 q 插入到表头

反转单链表



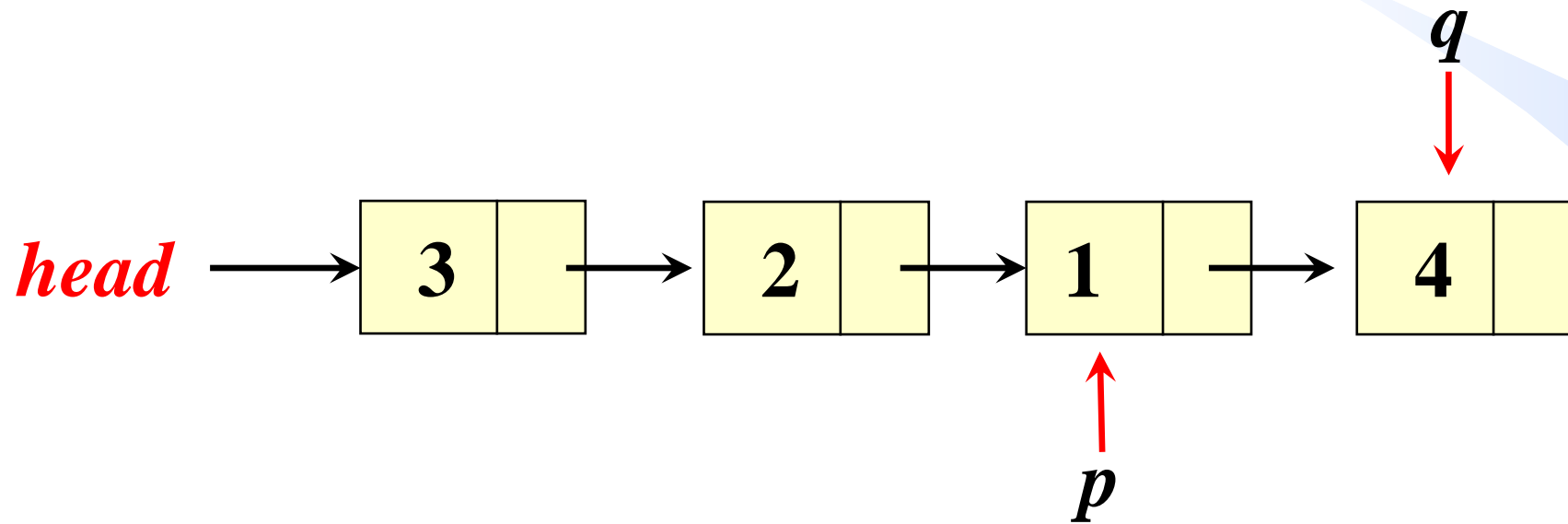
把 q 插入到表头

反转单链表



把 q 插入到表头

反转单链表



令 q 为 p 的后继

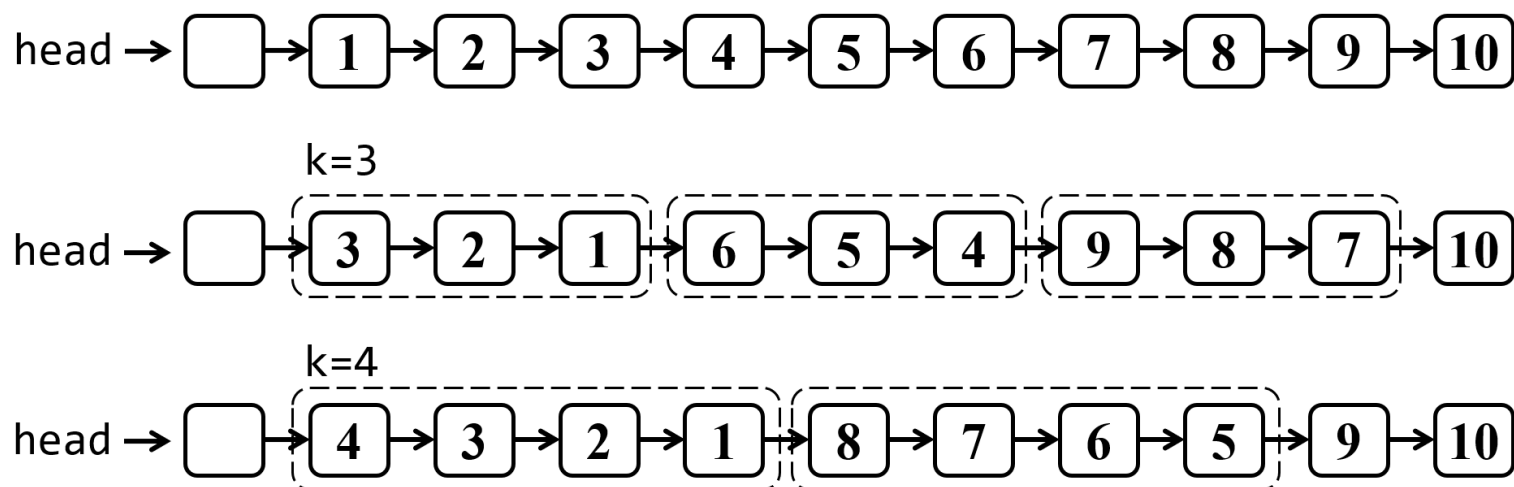
```
ListNode* reverseList(ListNode* head) {  
    if(head==nullptr) return head;  
    ListNode *p=head, *q=head->next;  
    while(q!=nullptr){  
        p->next=q->next;  
        q->next=head;  
        head=q;  
        q=p->next;  
    }  
    return head;  
}
```

Talk is cheap,
show me the code



课下思考

(20分) 给定一个包含哨位结点的单链表和参数 k ，请设计一个算法，对该链表每 k 个结点为一组进行反转，剩余结点不足一组时无需反转。例如下图给出了一个包含10个结点的链表当 $k=3$ 和 $k=4$ 的反转示例。



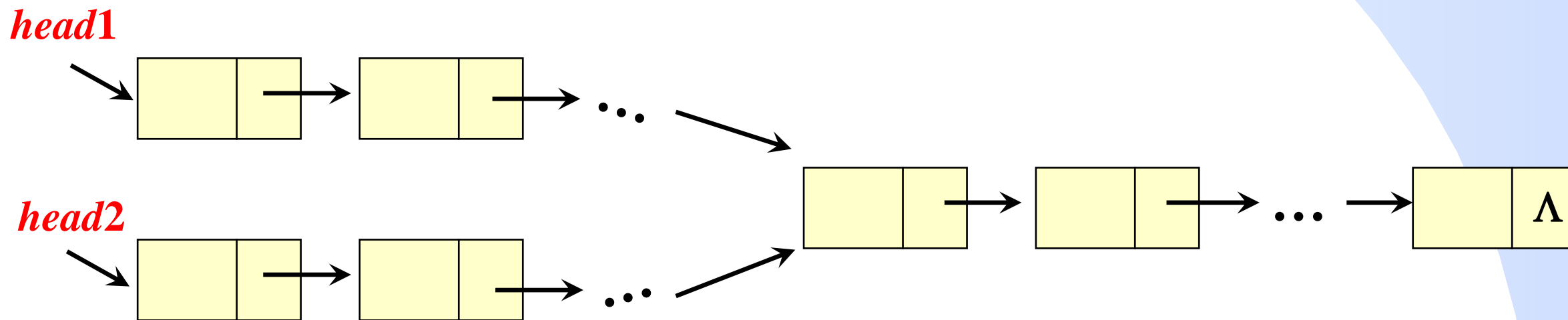
即设计实现如下函数：

```
ListNode* Reverse(ListNode* Head, int k)
```

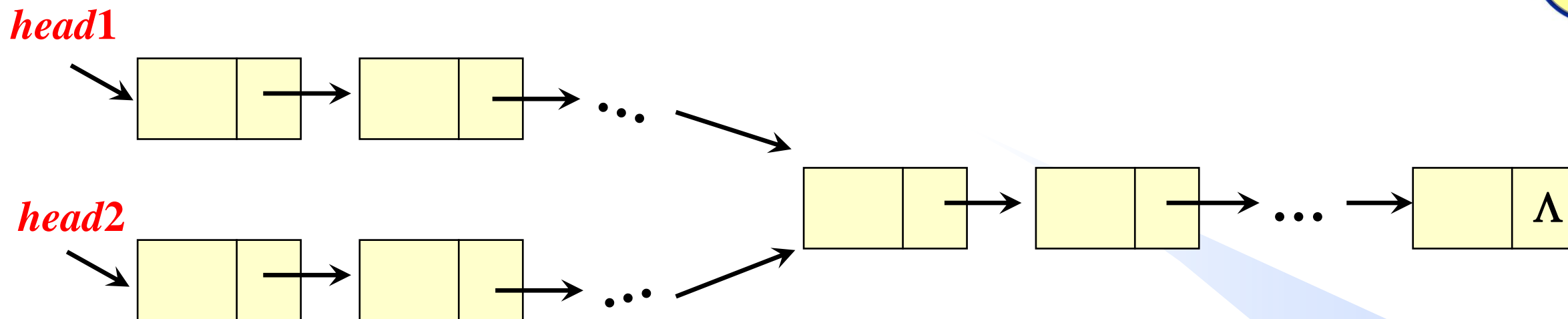
其中Head为单链表的头指针，函数返回值为反转后链表的头指针。函数中可调用你自定义的其他函数。要求算法空间复杂度为 $O(1)$ 且不能只单纯改变结点数据域值，而要对结点的指针进行实际的反转。【吉林大学计软两院22级期末考试题】

链表相交问题

给定两个单链表的头指针 $head1$ 和 $head2$ ，设计一个算法判断这两个链表是否相交,如果相交则返回第一个交点，要求时间复杂度为 $O(L_1+L_2)$ ， L_1 、 L_2 分别为两个链表的长度。为了简化问题，这里我们假设两个链表均不含有环【大厂面试题 [LeetCode 160](#)】



链表相交问题



方案1:

FOR \forall 结点 $i \in$ 链表1 **DO**

(

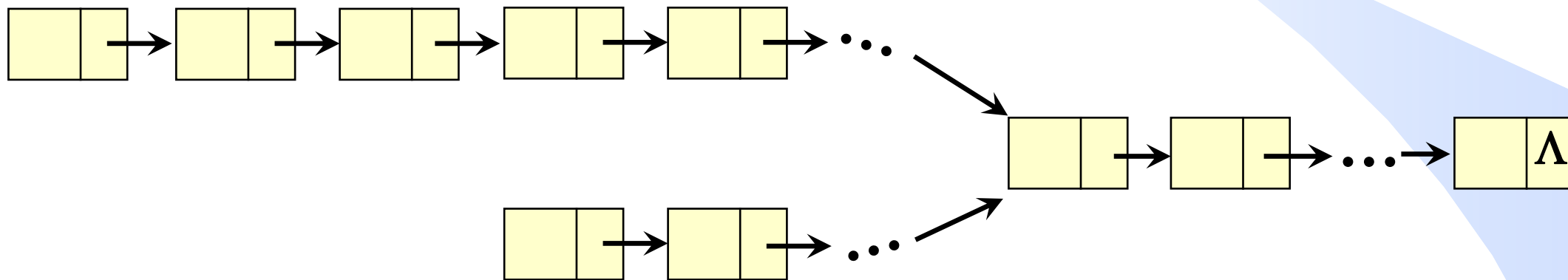
判断“结点 i ”是否在链表2中.

)

时间复杂度: $O(L_1 * L_2)$

链表相交问题——思路

- **是否相交？** 如果两个链表相交，则最后一个结点一定是共有的，可以分别遍历2个链表，记录其最后一个结点和链表长度。若2个链表最后一个结点相等，则相交，否则不相交。



- **找交点？** 用指针 $p1$ 指向较长的那个链表， $p2$ 指向较短的那个链表， $p1$ 先向后移动 $|L1-L2|$ 步，使 $p1$ 和 $p2$ “对齐”，然后 $p1$ 和 $p2$ 同时并行向右移动，每移动一步比较 $p1$ 和 $p2$ 是否相等，当二者首次相等时，其指向的结点即为交点。

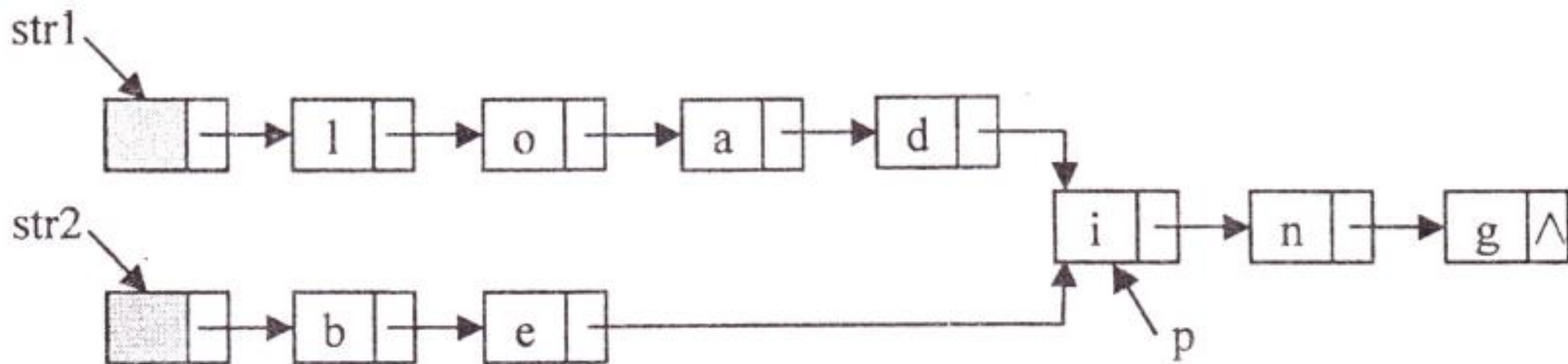

```
ListNode* cross(ListNode *head1, ListNode *head2) {  
    //若两个链表相交，返回交点指针，否则返回NULL  
    if (head1==NULL || head2==NULL) return NULL;  
    if (head1==head2) return head1;  
    ListNode *p1=head1,*p2=head2;  
    int L1=1,L2=1;  
    while (p1->next!=NULL) { L1++; p1=p1->next;}  
    while (p2->next!=NULL) { L2++; p2=p2->next;}  
    if (p1!=p2) return NULL; //不相交  
    if (L1>=L2) { p1=head1; p2=head2;}  
    else { p1=head2; p2=head1;}  
    for(int i=0; i<abs(L1-L2); i++) p1=p1->next; //p1和p2对齐  
    while (p1!=p2) { p1=p1->next; p2=p2->next;}  
    return p1;  
}
```

Talk is cheap,
show me the code



考研题全国卷

(13 分) 假定采用带头结点的单链表保存单词，当两个单词有相同的后缀时，则可共享相同的后缀存储空间。例如，“loading”和“being”的存储映像如下图所示。



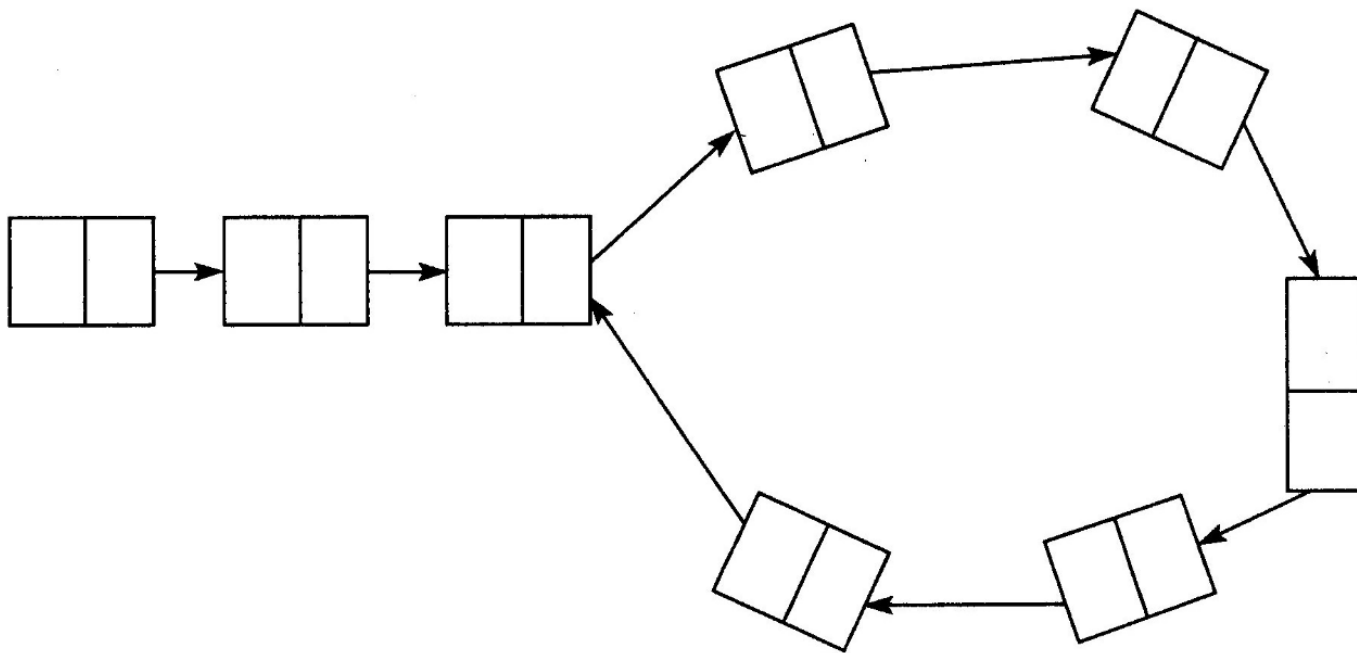
设 $str1$ 和 $str2$ 分别指向两个单词所在单链表的头结点，链表结点结构为

data	next
------	------

，请设计一个时间上尽可能高效的算法，找出由 $str1$ 和 $str2$ 所指的两个链表共同后缀的起始位置（如图中字符 i 所在结点的位置 p ）。

单链表判环问题

编写算法判断一个单链表中是否含有环。如果有环的话，找出从头结点进入环的第一个结点。【大厂面试题[LeetCode 141](#)、[LeetCode 142](#)】



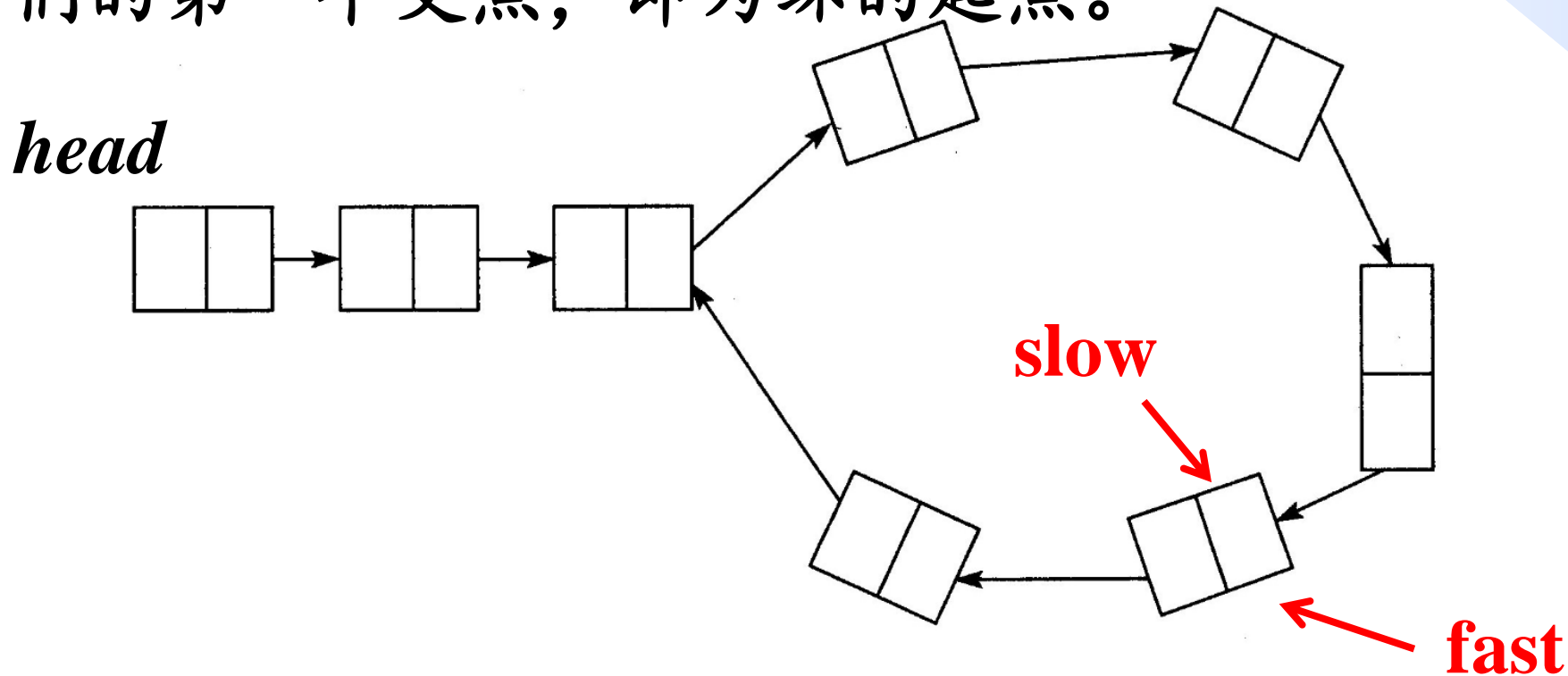
单链表判环问题

- 使用两个指针slow和fast从链表头开始遍历，slow每次前进1步，fast每次前进2步。
- 如果含有环，fast和slow必然会在某个时刻相遇（fast==slow），好比在环形跑道上赛跑时运动员的套圈。
- 如果遍历过程中，最终fast达到NULL，则说明无环。

```
bool hasCycle(ListNode *head) {  
    ListNode *slow = head, *fast = head;  
    while (fast != NULL && fast->next != NULL){  
        slow = slow->next;  
        fast = fast->next->next;  
        if (fast == slow) return true;  
    }  
    return false;  
}
```

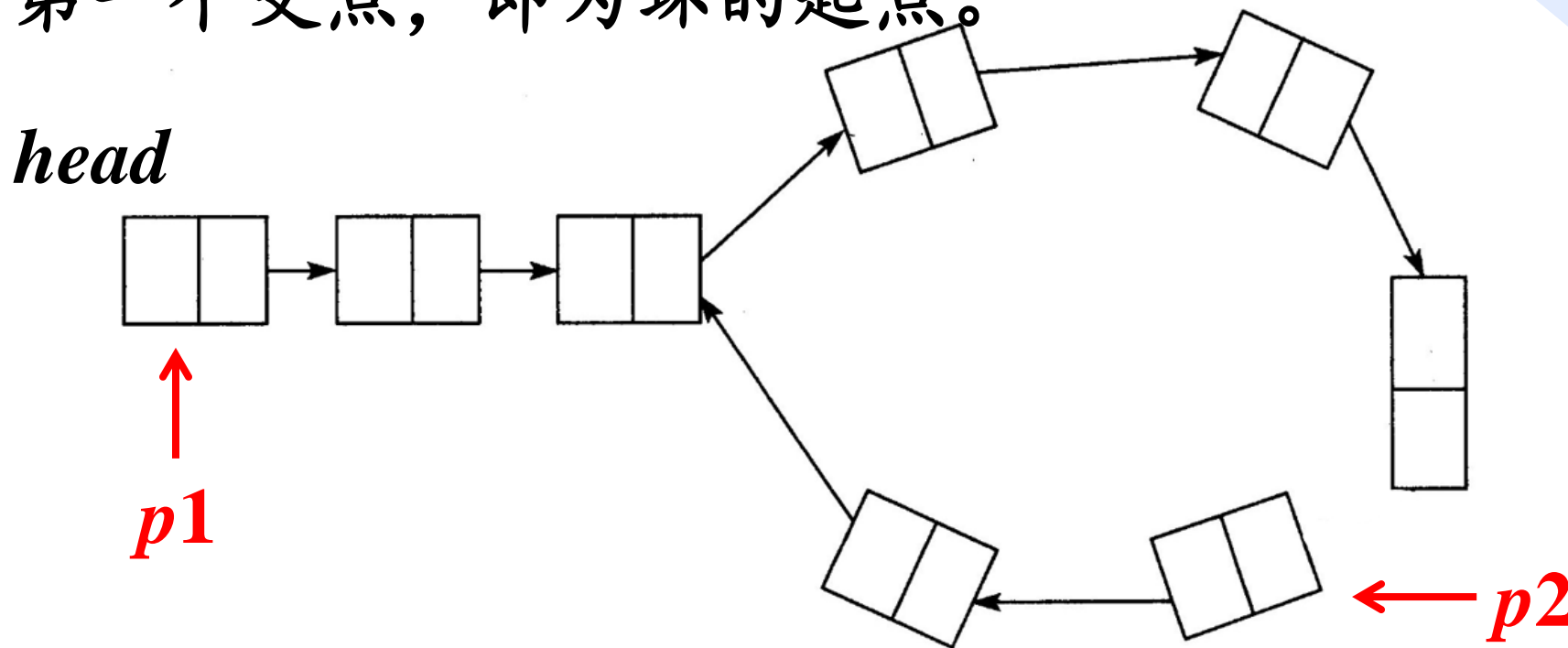
单链表判环问题

从相遇结点（ $\text{fast} == \text{slow}$ 所对应的结点）处断开环，令 $\text{p1} \leftarrow \text{head}$, $\text{p2} \leftarrow \text{fast}$ 。此时，原单链表可以看作两条单链表，一条从 p1 开始，另一条从 p2 开始，结合链表相交问题，找到它们的第一个交点，即为环的起点。



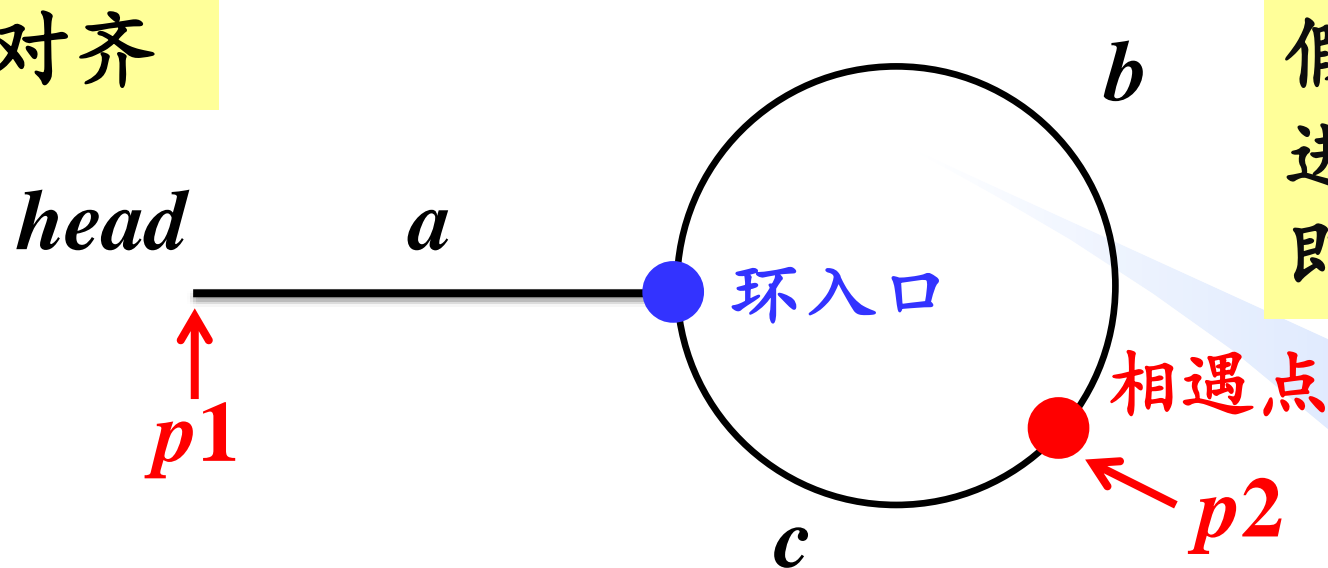
单链表判环问题

从相遇结点（ $fast == slow$ 所对应的结点）处断开环，令 $p1 \leftarrow head, p2 \leftarrow fast$ 。此时，原单链表可以看作两条单链表，一条从 $p1$ 开始，另一条从 $p2$ 开始，结合链表相交问题，找到它们的第一个交点，即为环的起点。



单链表判环问题

$p1$ 和 $p2$ 无需对齐



假定 $fast$ 指针进入环1圈后即与 $slow$ 相遇

$2 * \text{slow}$ 指针走的步数 = fast 指针走的步数

$$2(a+b) = a+b+c+b$$

$$a = c$$

若 $fast$ 在环内走了很多圈才与 $slow$ 相遇，刚才的策略是否还可行？

单链表判环问题

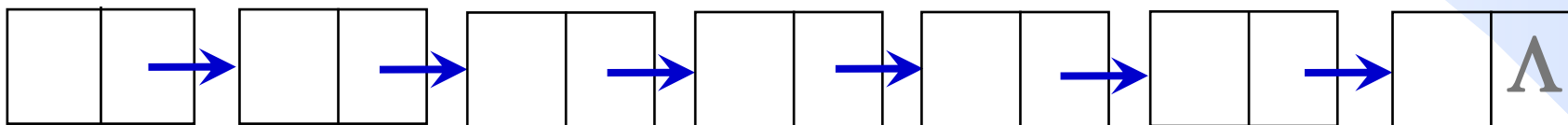
```
ListNode* detectCycleEntrance(ListNode *head) {  
    ListNode *slow = head, *fast = head;  
    while (fast != NULL && fast->next != NULL){  
        fast = fast->next->next;  
        slow = slow->next;  
        if (fast == slow) { //有环, 找环的入口  
            ListNode *p1 = head, *p2=fast;  
            while (p1 != p2) {  
                p1 = p1->next;  
                p2 = p2->next;  
            }  
            return p1;  
        }  
    }  
    return NULL;  
}
```



判断一个单链表是否是回文

要求时间复杂度 $O(n)$ ，空间复杂度 $O(1)$

【大厂面试题 [LeetCode 234](#)】



回文：中文对称

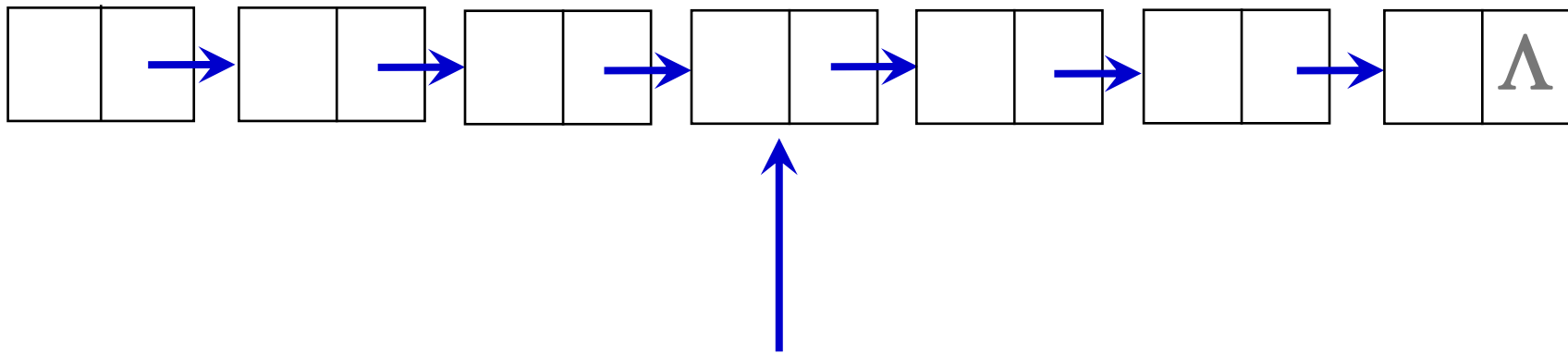
level

refer

上海自来水来自海上

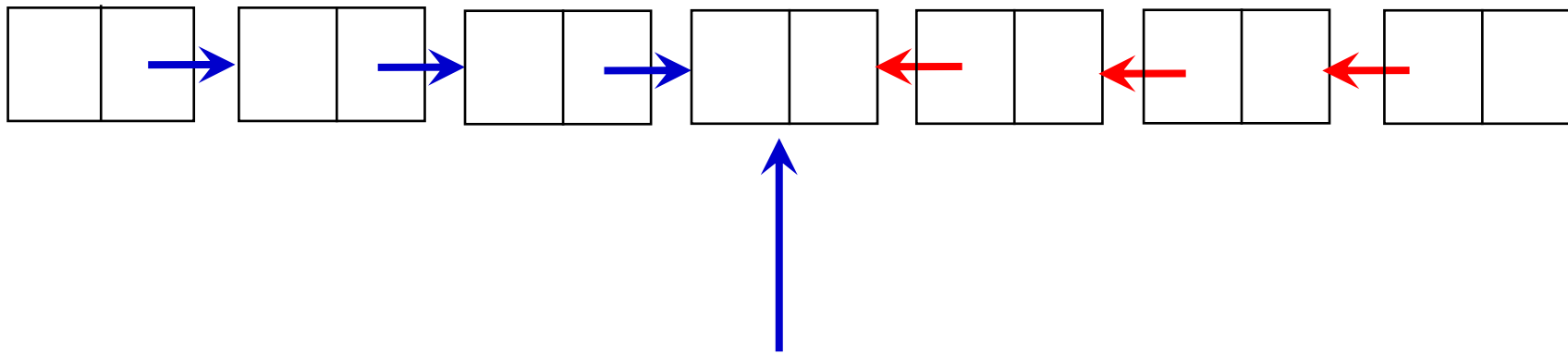
判断一个单链表是否是回文

找到中间结点，将后半部分链表反转



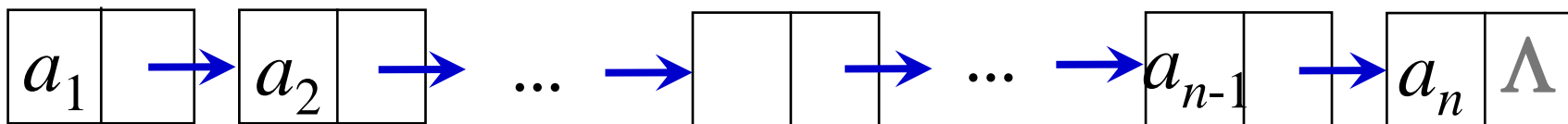
判断一个单链表是否是回文

找到中间结点，将后半部分链表反转



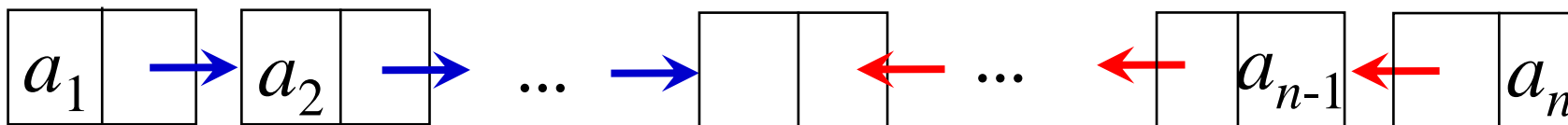
重排链表结点

设线性表 $L=(a_1, a_2, a_3, \dots, a_{n-2}, a_{n-1}, a_n)$ 采用带哨位结点的单链表保存，请设计一个空间复杂度为 $O(1)$ 且时间上尽可能高效的算法，重新排列 L 中的各结点，得到线性表 $L'=(a_1, a_n, a_2, a_{n-1}, a_3, a_{n-2}, \dots)$ 。【2019 年考研题全国卷（13 分），大厂面试题 [LeetCode 143](#)】



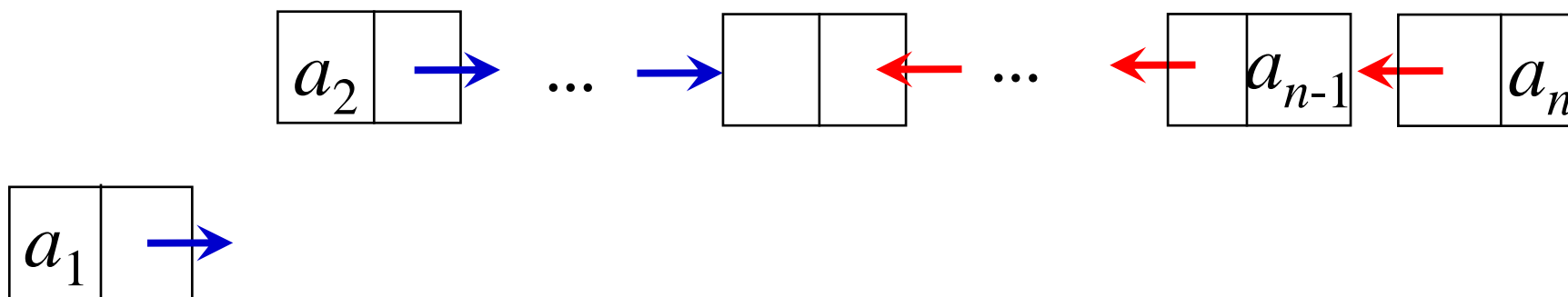
重排链表结点

设线性表 $L=(a_1, a_2, a_3, \dots, a_{n-2}, a_{n-1}, a_n)$ 采用带哨位结点的单链表保存，请设计一个空间复杂度为 $O(1)$ 且时间上尽可能高效的算法，重新排列 L 中的各结点，得到线性表 $L'=(a_1, a_n, a_2, a_{n-1}, a_3, a_{n-2}, \dots)$ 。【2019 年考研题全国卷（13 分），大厂面试题 [LeetCode 143](#)】



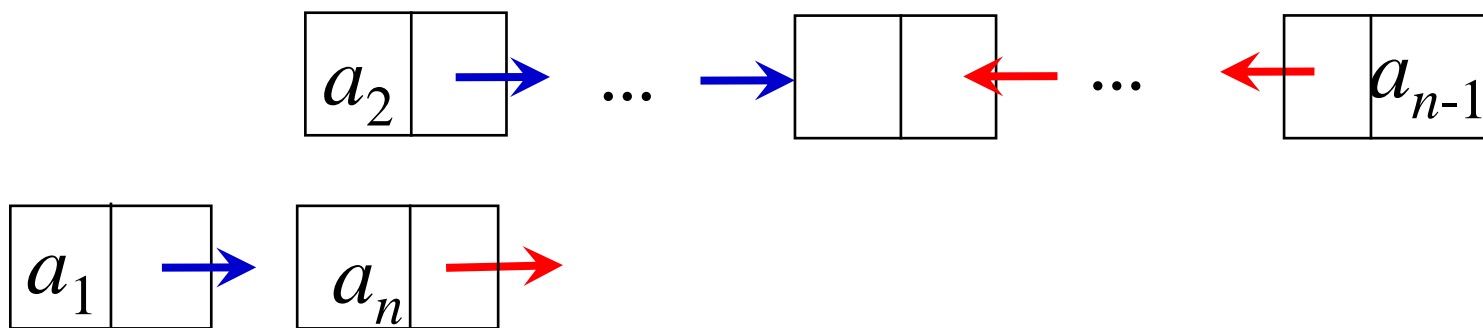
重排链表结点

设线性表 $L=(a_1, a_2, a_3, \dots, a_{n-2}, a_{n-1}, a_n)$ 采用带哨位结点的单链表保存，请设计一个空间复杂度为 $O(1)$ 且时间上尽可能高效的算法，重新排列 L 中的各结点，得到线性表 $L'=(a_1, a_n, a_2, a_{n-1}, a_3, a_{n-2}, \dots)$ 。【2019 年考研题全国卷（13 分），大厂面试题 [LeetCode 143](#)】



重排链表结点

设线性表 $L=(a_1, a_2, a_3, \dots, a_{n-2}, a_{n-1}, a_n)$ 采用带哨位结点的单链表保存，请设计一个空间复杂度为 $O(1)$ 且时间上尽可能高效的算法，重新排列 L 中的各结点，得到线性表 $L'=(a_1, a_n, a_2, a_{n-1}, a_3, a_{n-2}, \dots)$ 。【2019 年考研题全国卷（13 分），大厂面试题 [LeetCode 143](#)】



一元多项式及其操作

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_2 x^2 + a_1 x + a_0$$

一元多项式的链表结点结构如下，每个结点包含两个数据域（系数和指数）和一个链接域。

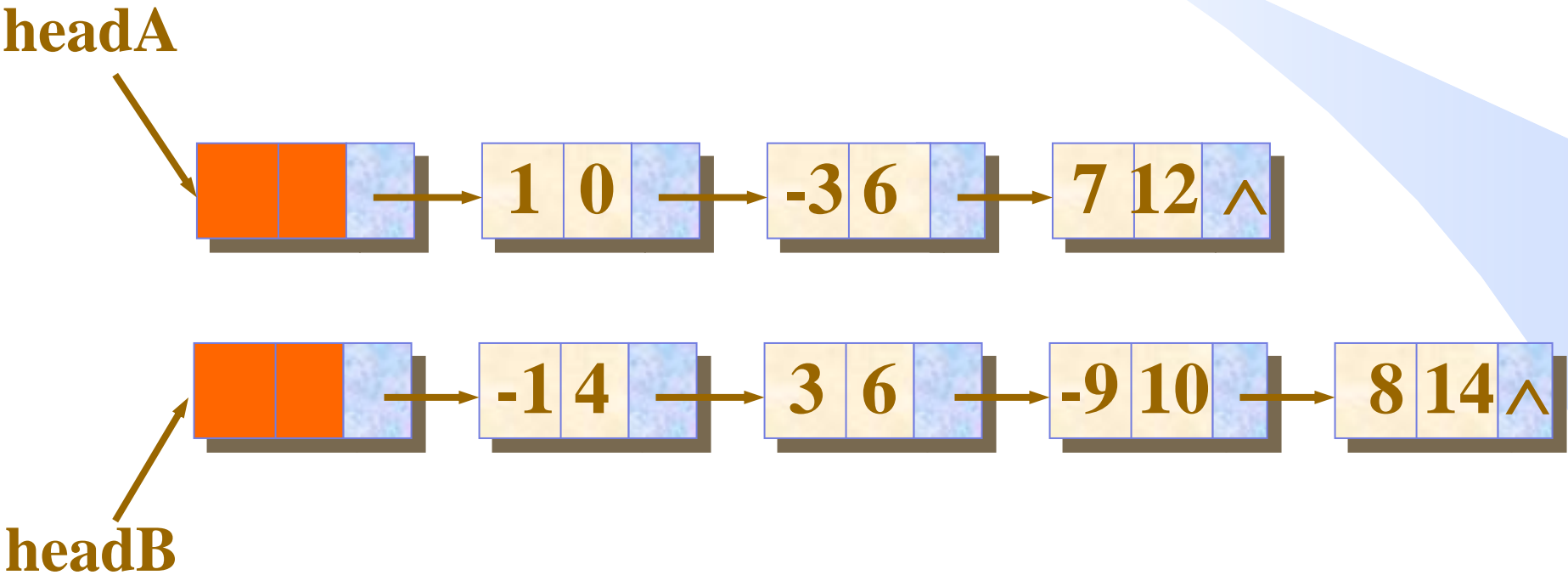
<i>coef</i>	<i>exp</i>	<i>next</i>
-------------	------------	-------------

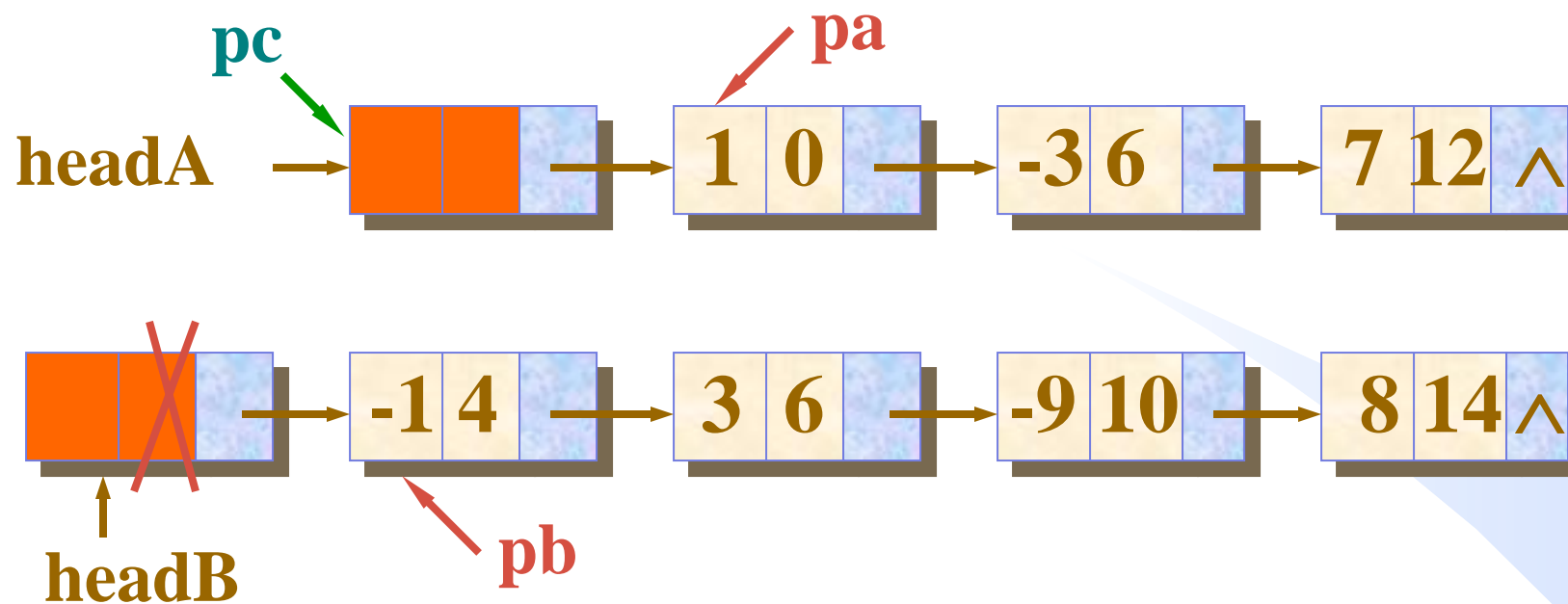
优点：多项式的项数可以动态地增长，不会出现存储溢出问题。
插入、删除方便，不移动元素，只需“穿针引线”。

多项式链表相加示例

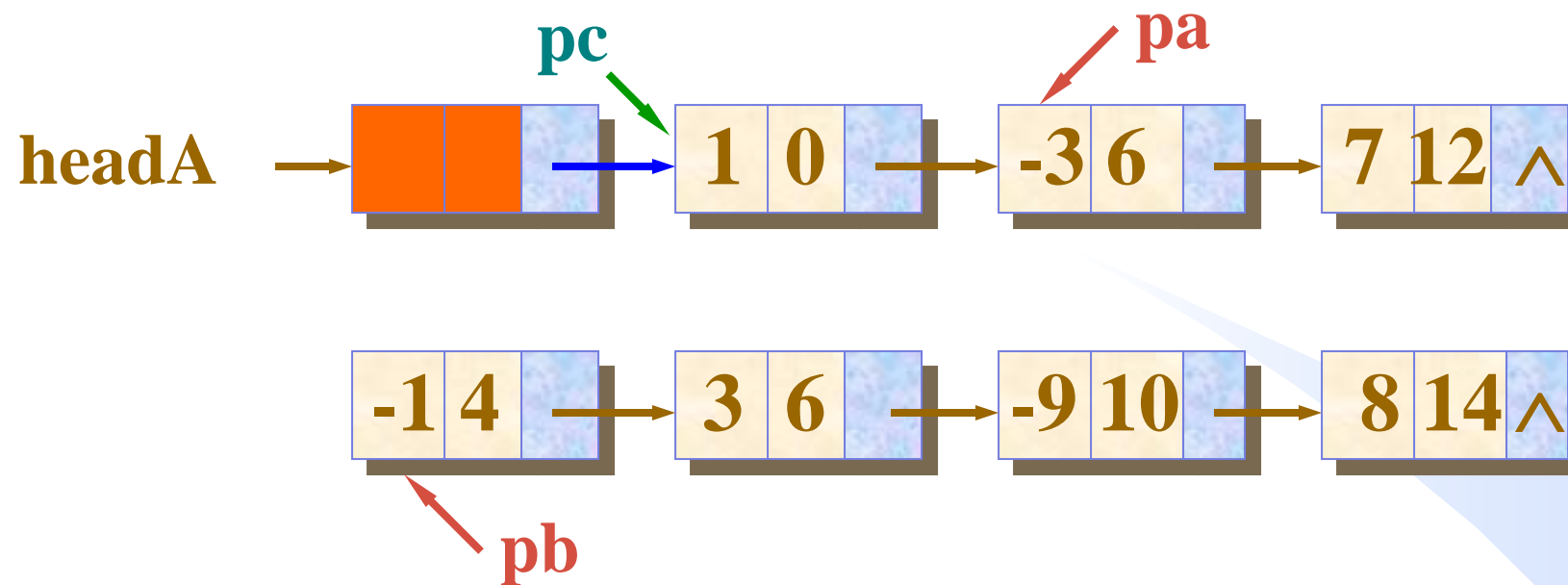
$$A = 1 - 3x^6 + 7x^{12}$$

$$B = -x^4 + 3x^6 - 9x^{10} + 8x^{14}$$

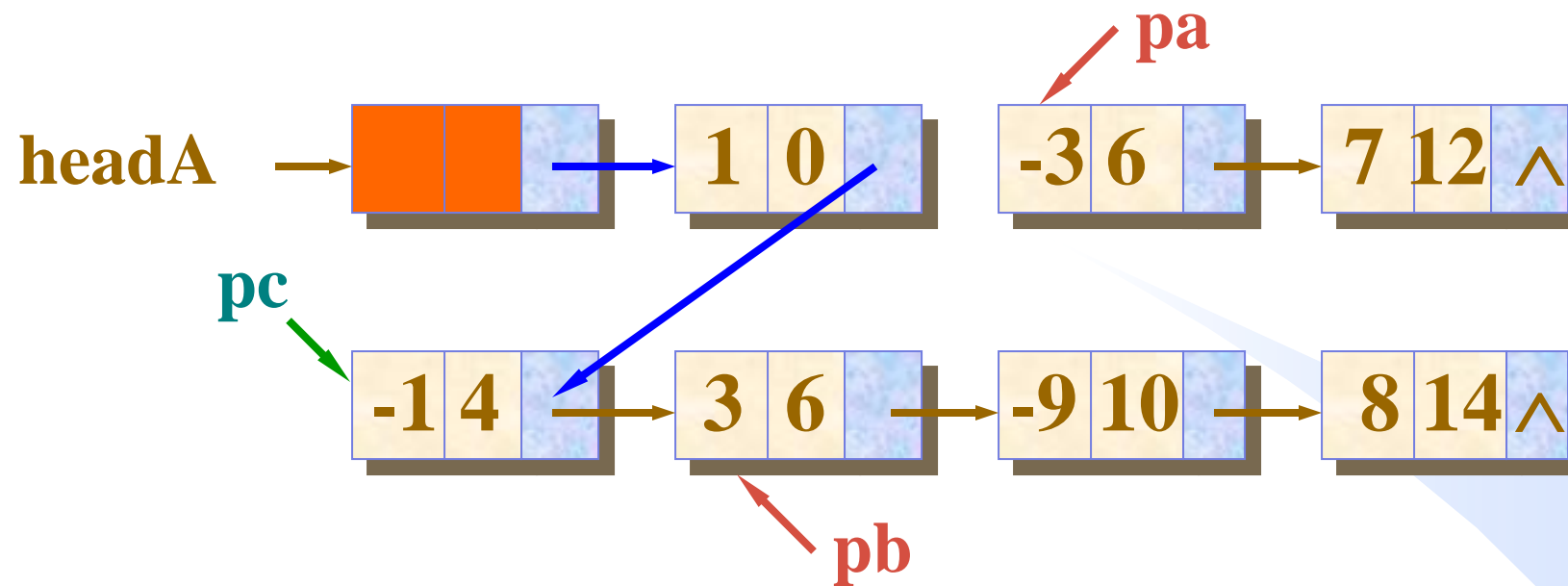




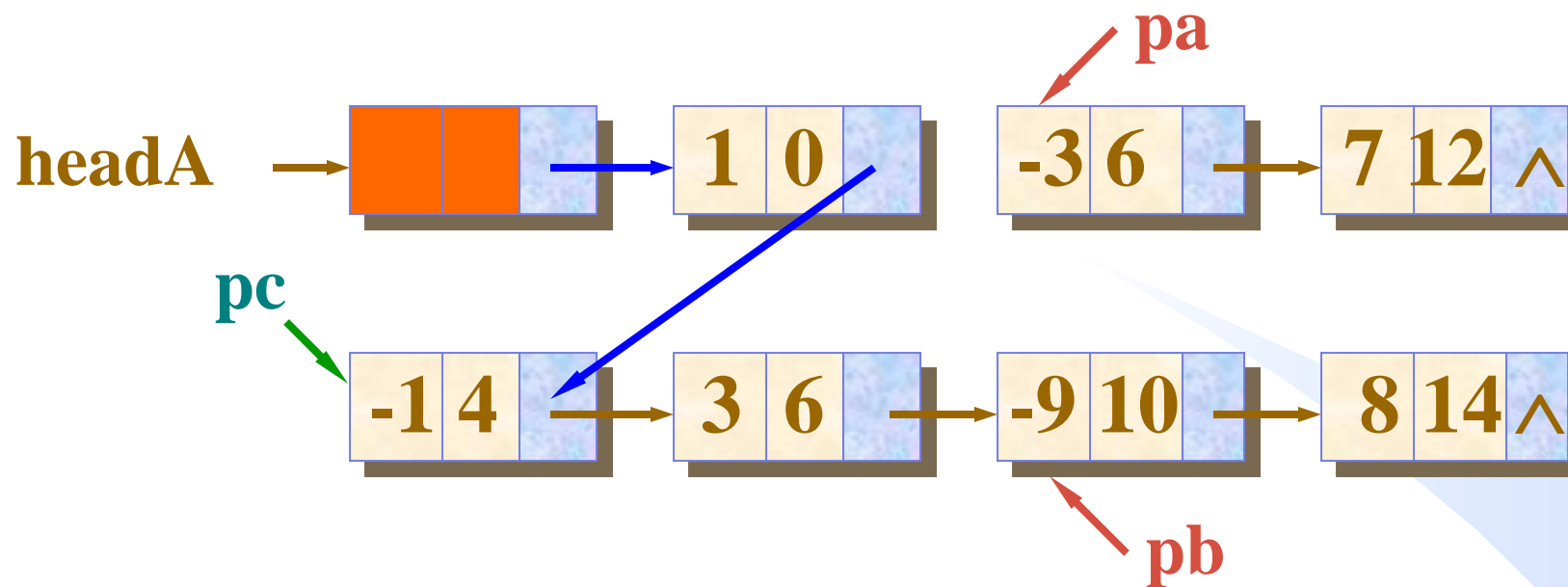
通过指针 pa 扫描链表A，通过指针 pb 扫描链表B
指针 pc 指向当前计算结果的最后一个结点



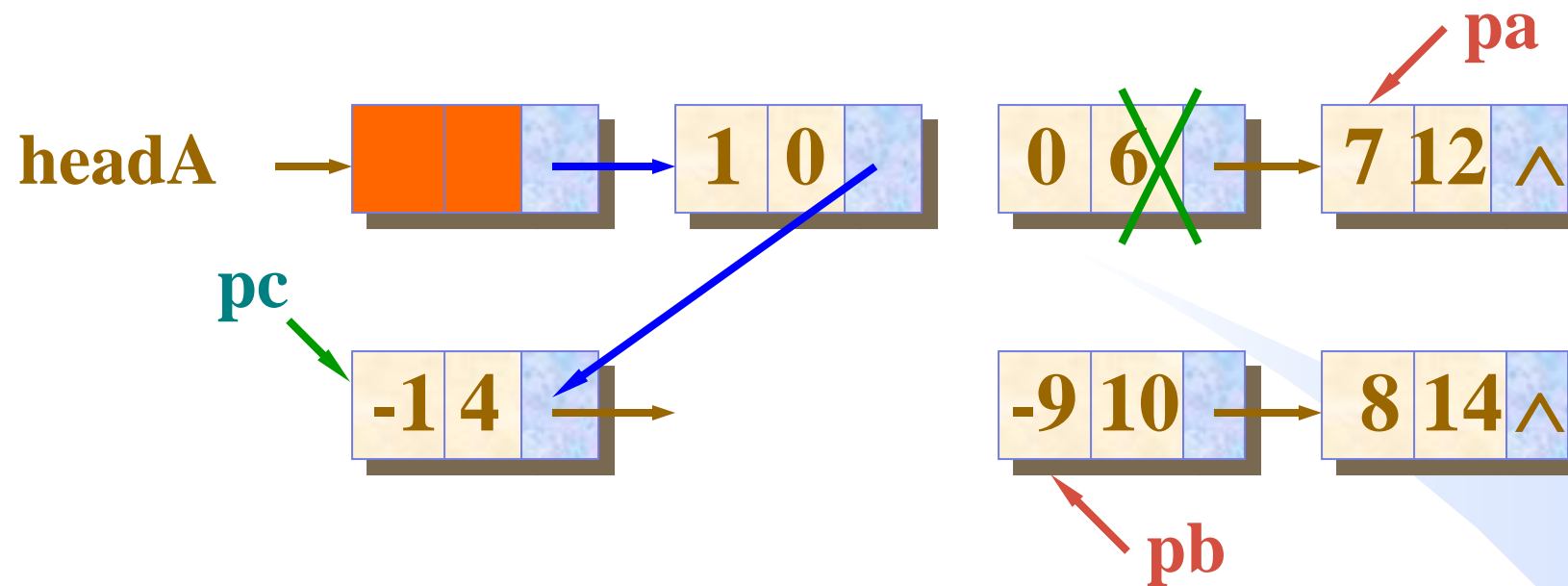
通过指针 pa 扫描链表A，通过指针 pb 扫描链表B
指针 pc 指向当前计算结果的最后一个结点



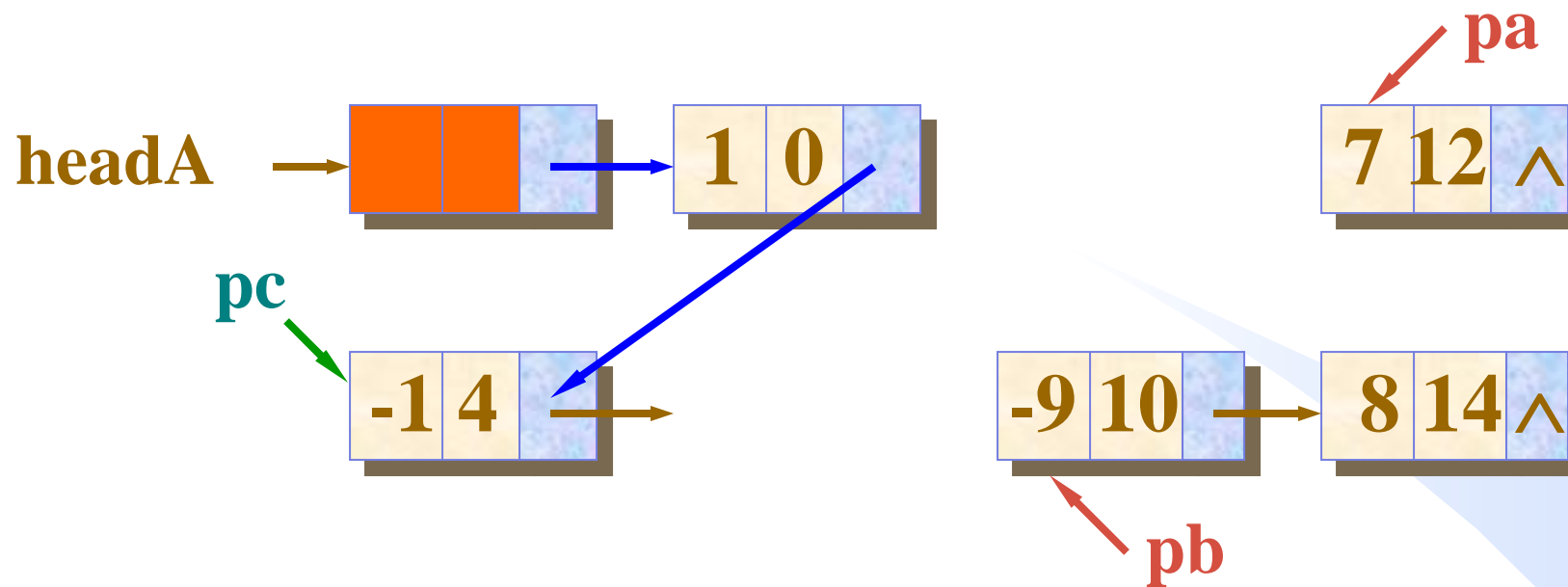
通过指针 pa 扫描链表A，通过指针 pb 扫描链表B
指针 pc 指向当前计算结果的最后一个结点



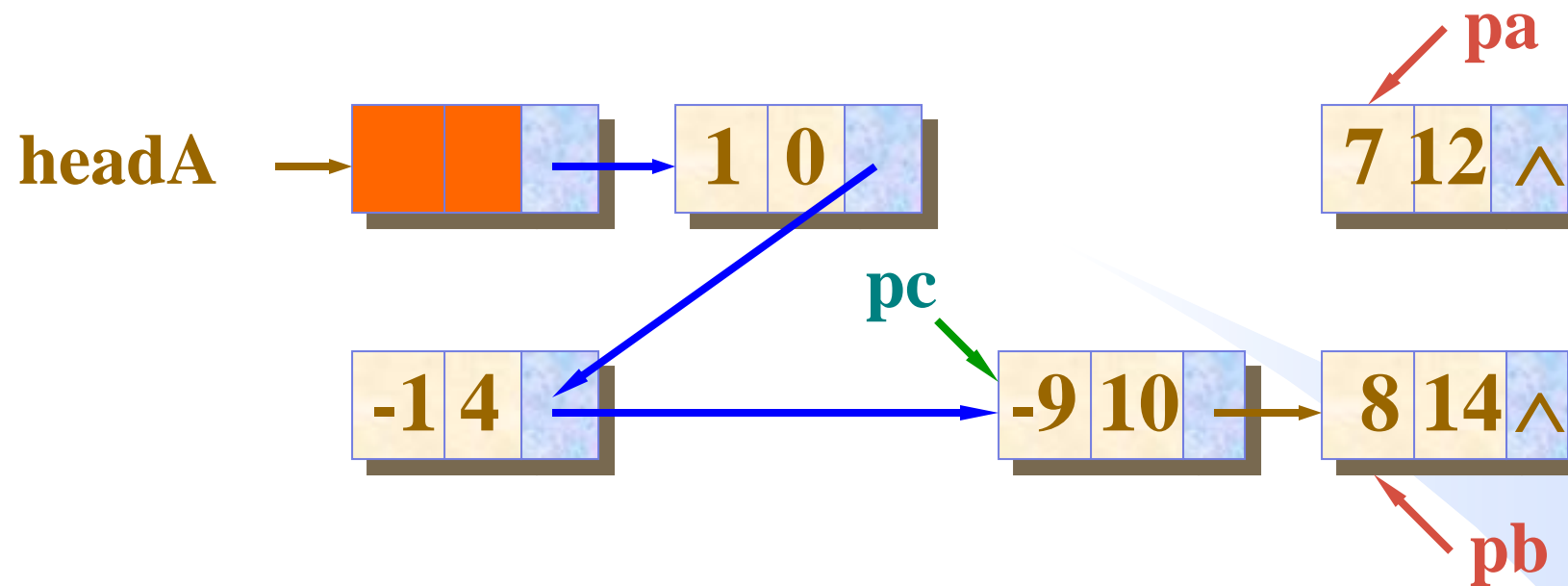
通过指针 pa 扫描链表A，通过指针 pb 扫描链表B
指针 pc 指向当前计算结果的最后一个结点



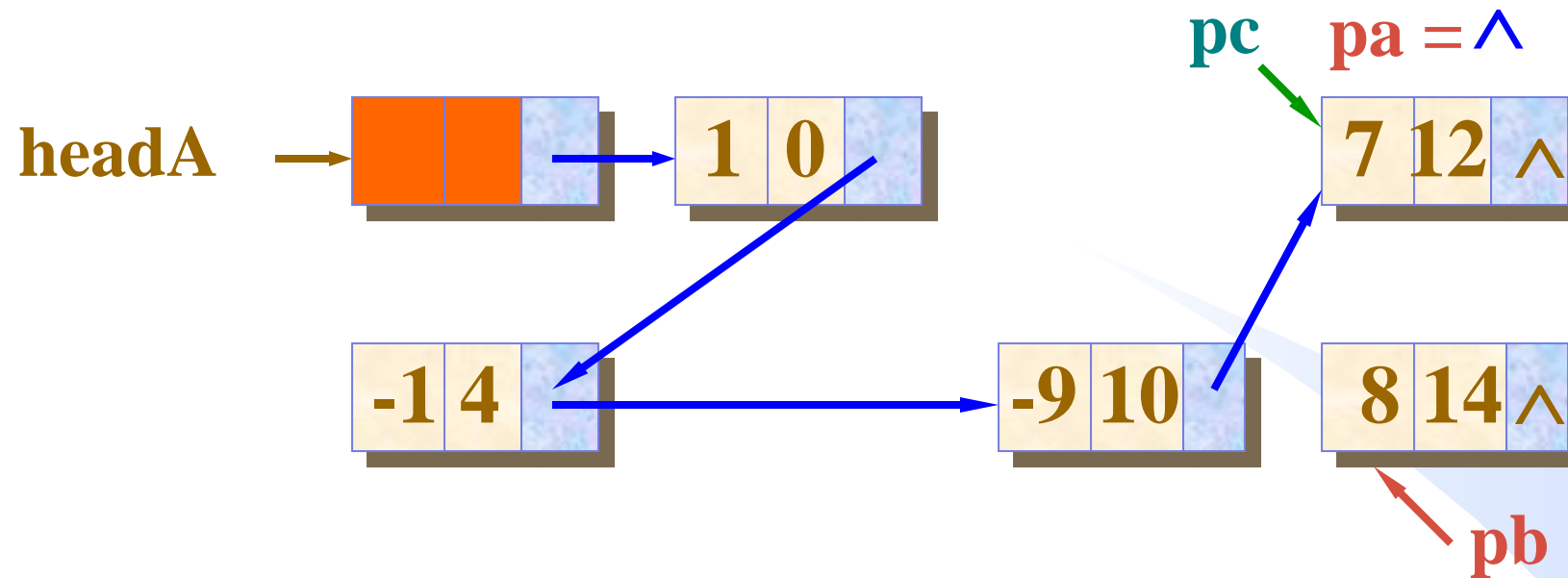
通过指针 pa 扫描链表A，通过指针 pb 扫描链表B
指针 pc 指向当前计算结果的最后一个结点



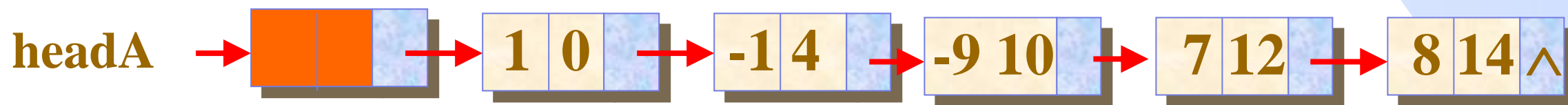
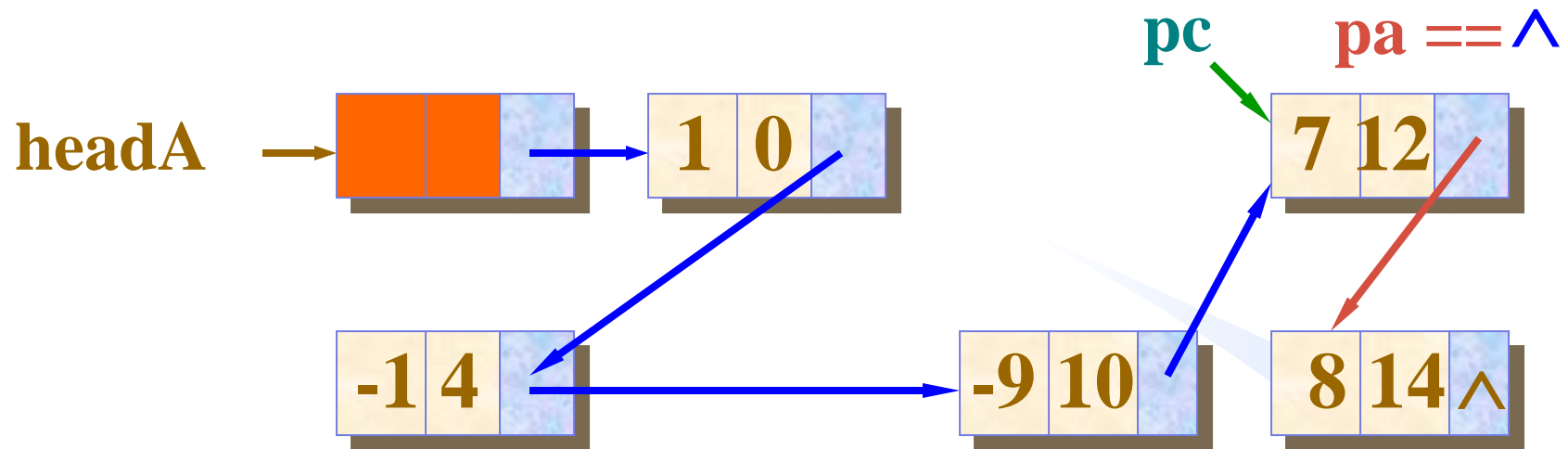
通过指针 pa 扫描链表A，通过指针 pb 扫描链表B
指针 pc 指向当前计算结果的最后一个结点



通过指针 pa 扫描链表A，通过指针 pb 扫描链表B
 指针 pc 指向当前计算结果的最后一个结点



通过指针 pa 扫描链表A，通过指针 pb 扫描链表B
指针 pc 指向当前计算结果的最后一个结点



$$A = 1 - x^4 - 9x^{10} + 7x^{12} + 8x^{14}$$



自愿性质OJ练习题

- ✓ [LeetCode 206](#) (反转链表)
- ✓ [LeetCode 92](#) (反转链表II)
- ✓ [LeetCode 19](#) (删除链表倒数第K个结点)
- ✓ [LeetCode 876](#) (找链表的中间结点)
- ✓ [LeetCode 160](#) (链表相交及交点)
- ✓ [LeetCode 141](#) (链表判环)
- ✓ [LeetCode 142](#) (链表找环的入口)
- ✓ [LeetCode 143](#) (重排链表)
- ✓ [LeetCode 234](#) (判断链表是否是回文)
- ✓ [LeetCode 138](#) (复制带随机指针的链表)