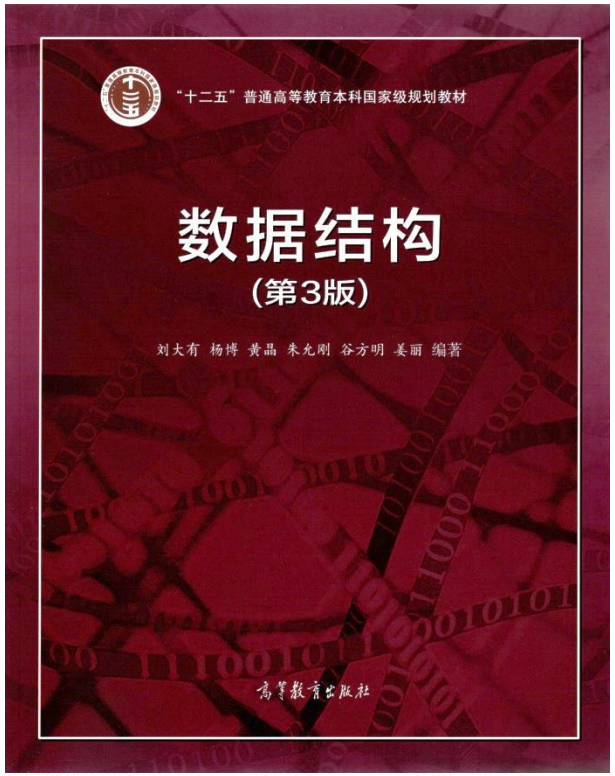
计算机学院王湘浩班

2024级



数据之法 结构之美 算法之道

链表的双指针技巧

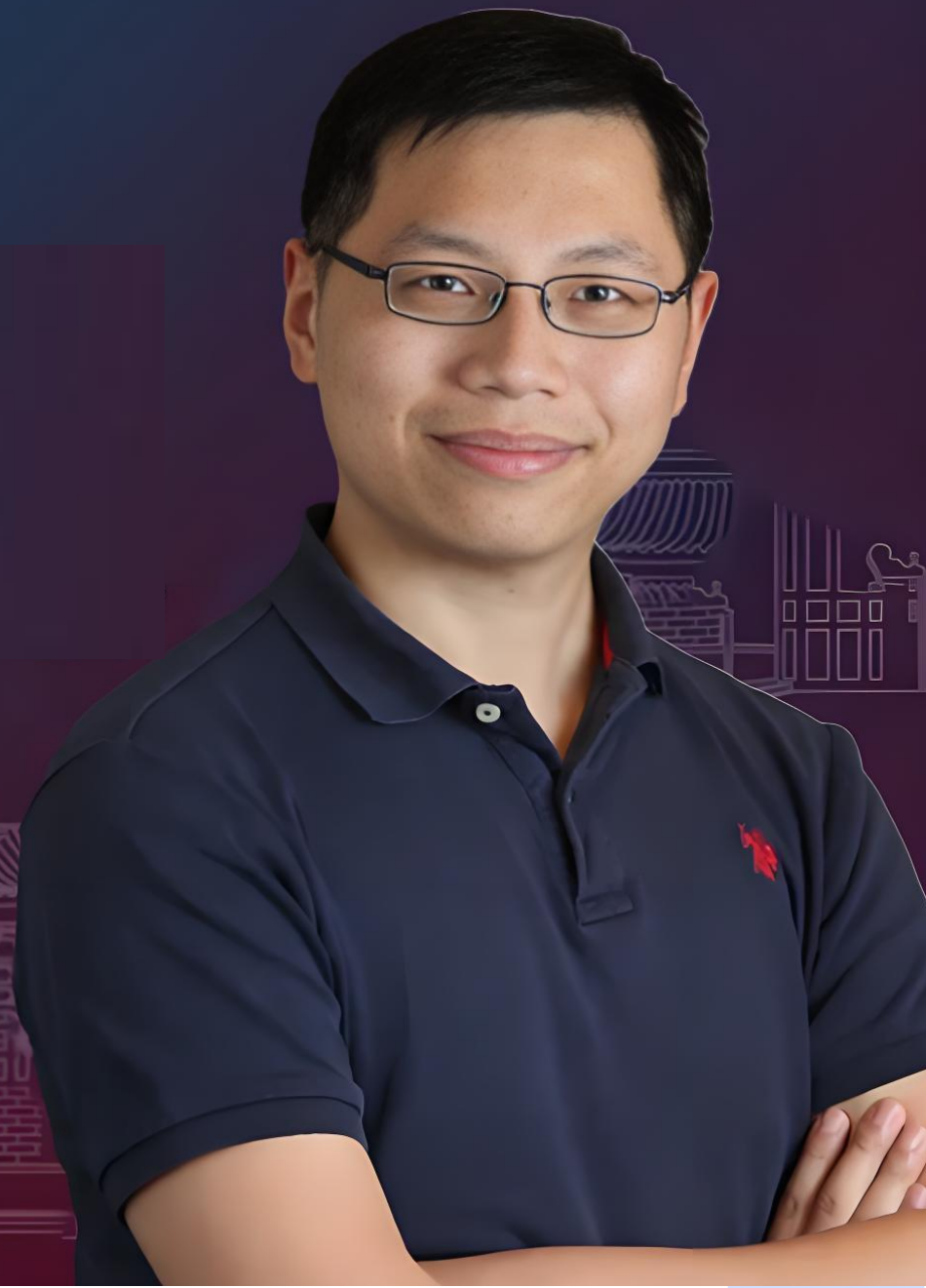
- 找单链表倒数第K个结点
- 找单链表中间结点
- 反转单链表
- 单链表求交点
- 单链表判环
- 其他

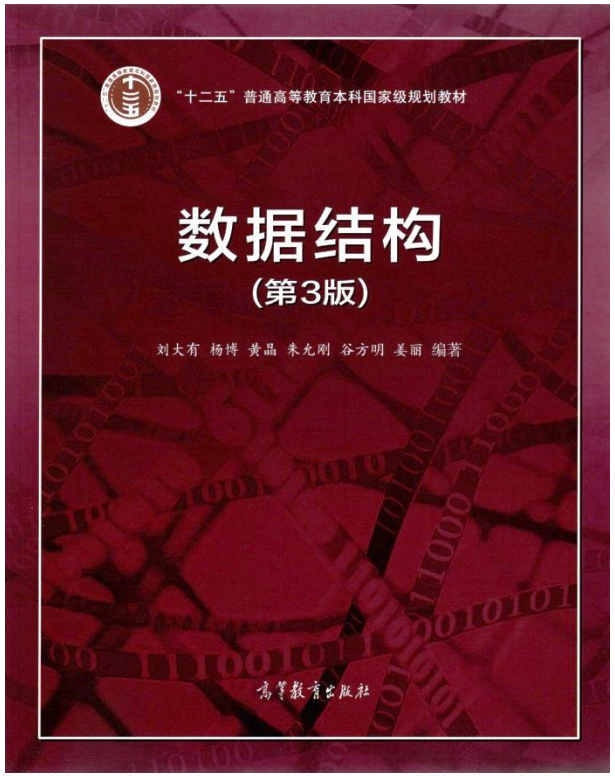
zhuyungang@jlu.edu.cn

Last updated on 2025.2

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

Last updated on 2025.2

zhuyungang@jlu.edu.cn

找单链表倒数第k个结点

B

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。【大厂面试题 LeetCode 0202】

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

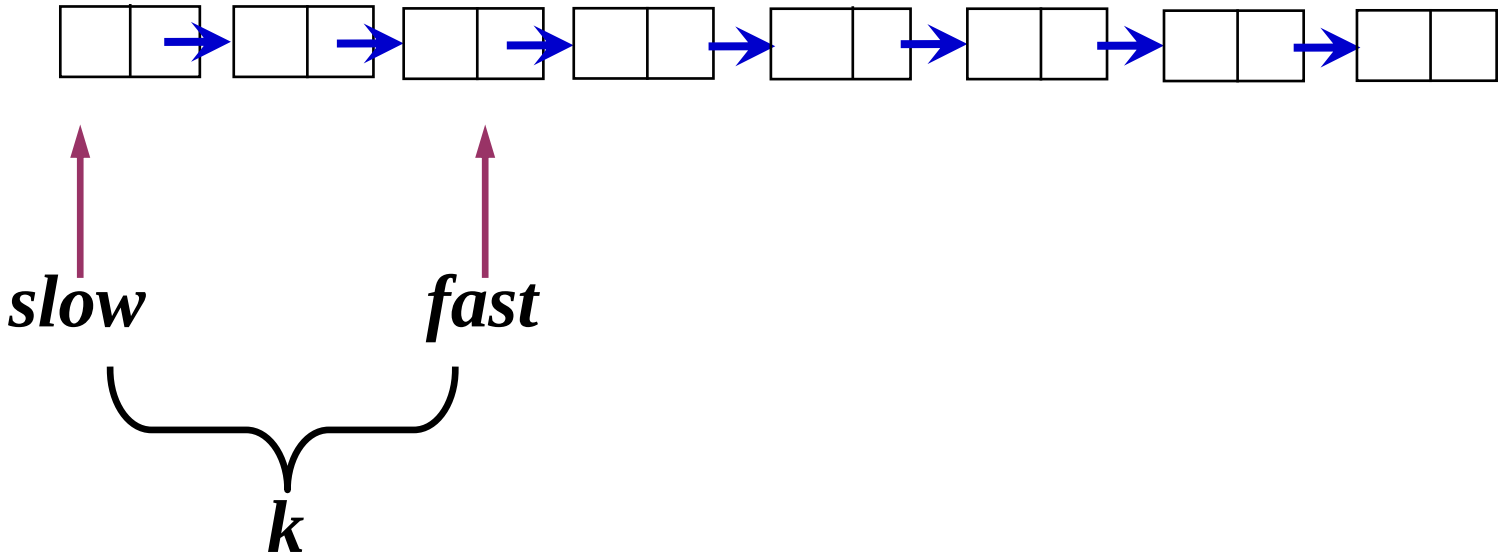
B

- 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$O(n)$

B

- 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*)。



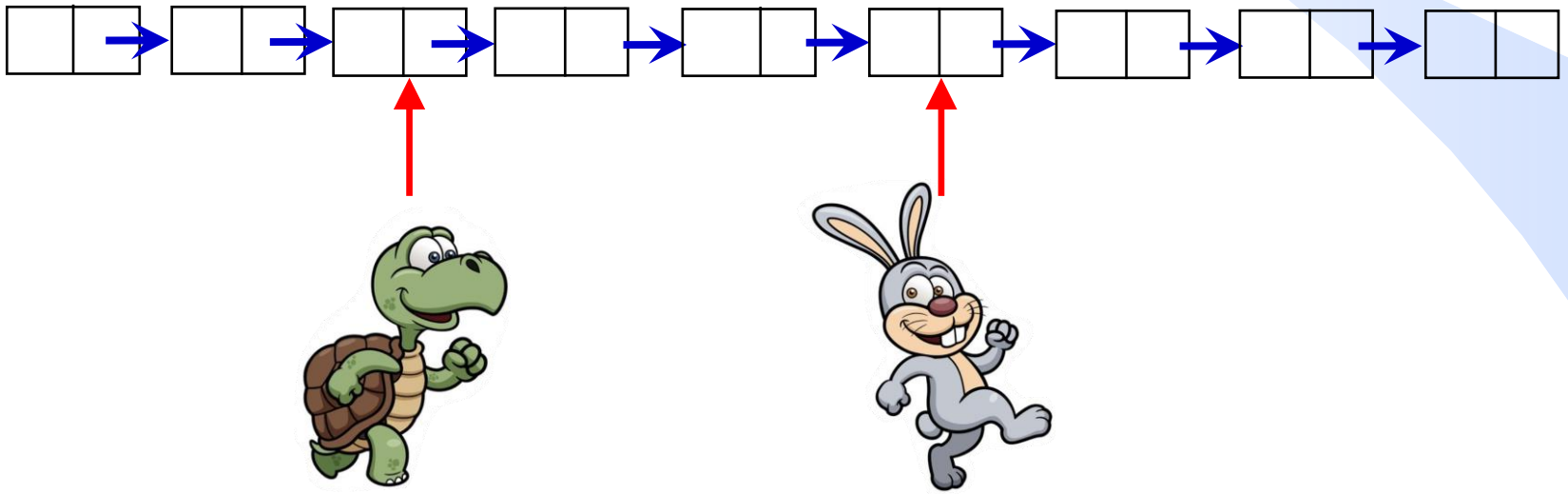
B

```
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;
}
```

找单链表中间结点

B

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】



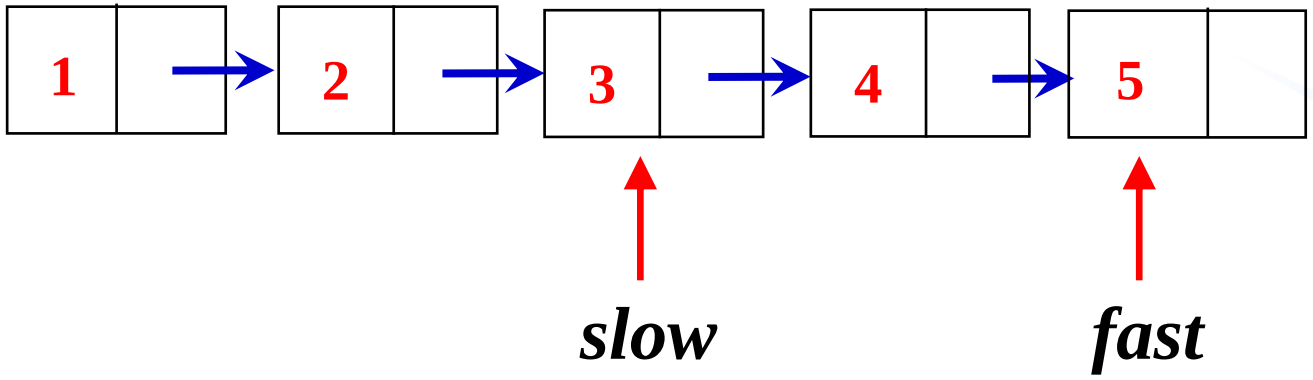
✓维护两个指针fast和slow

✓slow每次移动1步，fast每次移动2步

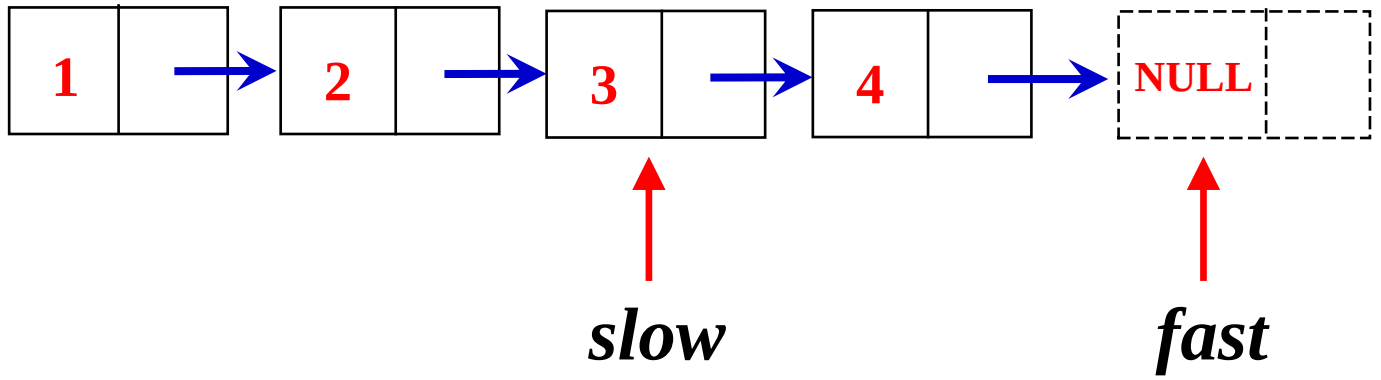
```
slow = slow->next;
fast = fast->next->next;
```

扫描结束的条件

B




```
fast->next==NULL
```



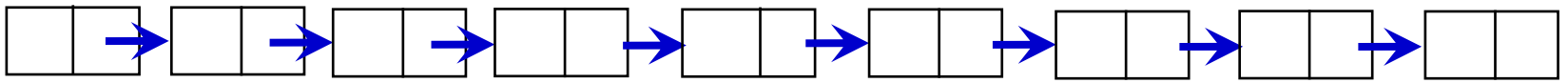

```
fast==NULL
```

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;
}
```

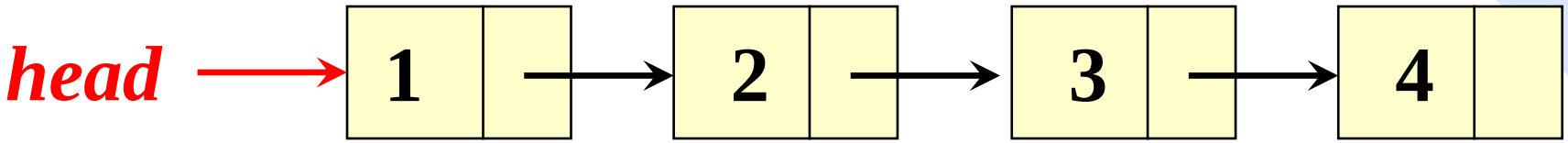
课下思考：对于长度为偶数的链表，若要返回第1个中间结点（即两个中间结点中靠左的那个结点），该如何修改上述代码？

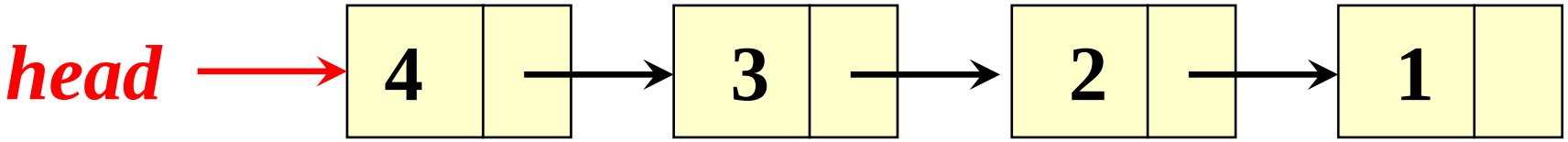


反转单链表

B

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】

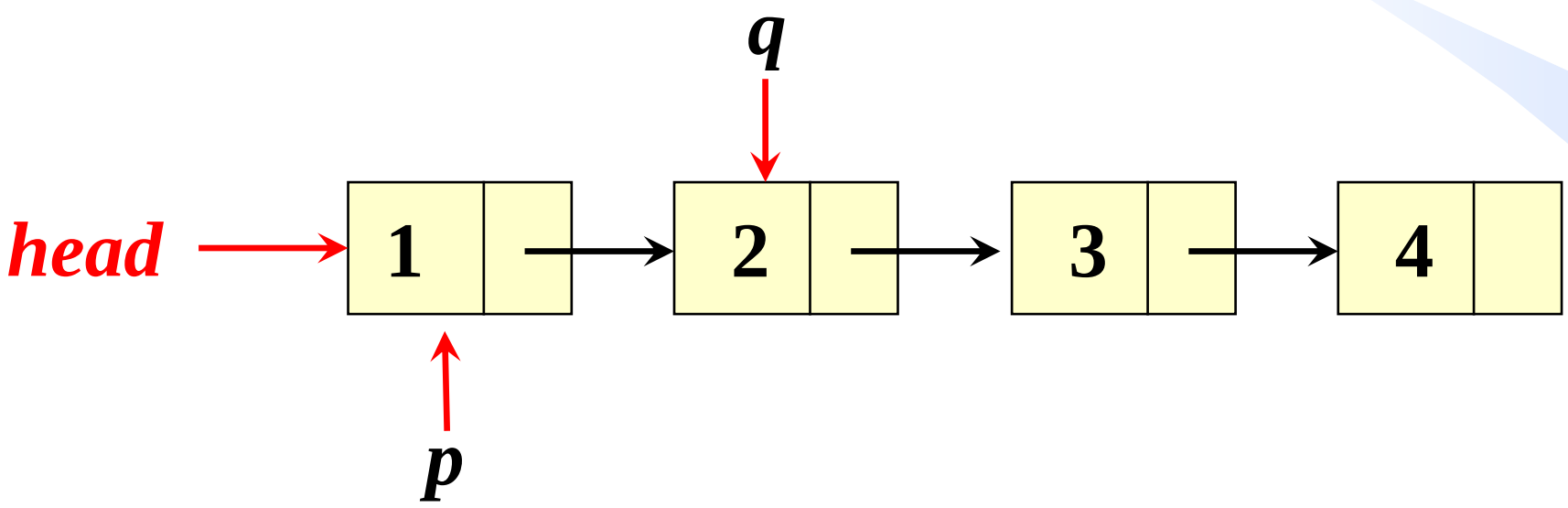




反转单链表

B

```
ListNode *p=head, *q=head->next;
```



空链表不用翻转

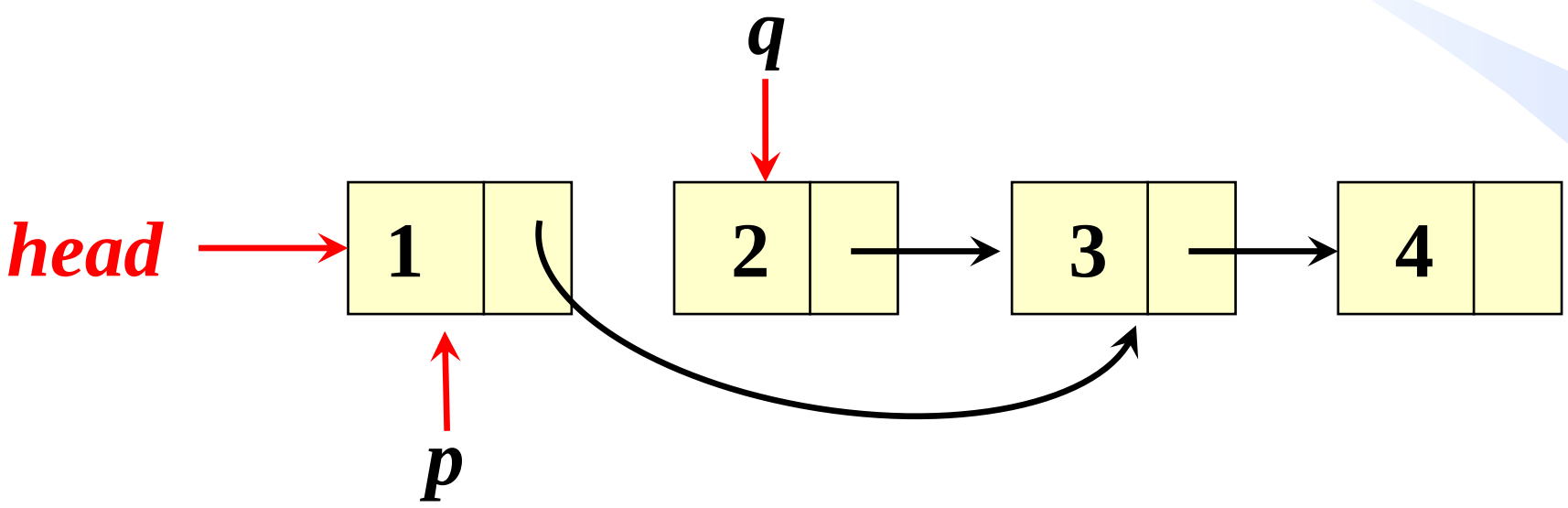
令p指向第1个结点
q指向p的后继

把q从链表中摘出

反转单链表

B

```
p->next=q->next;
```

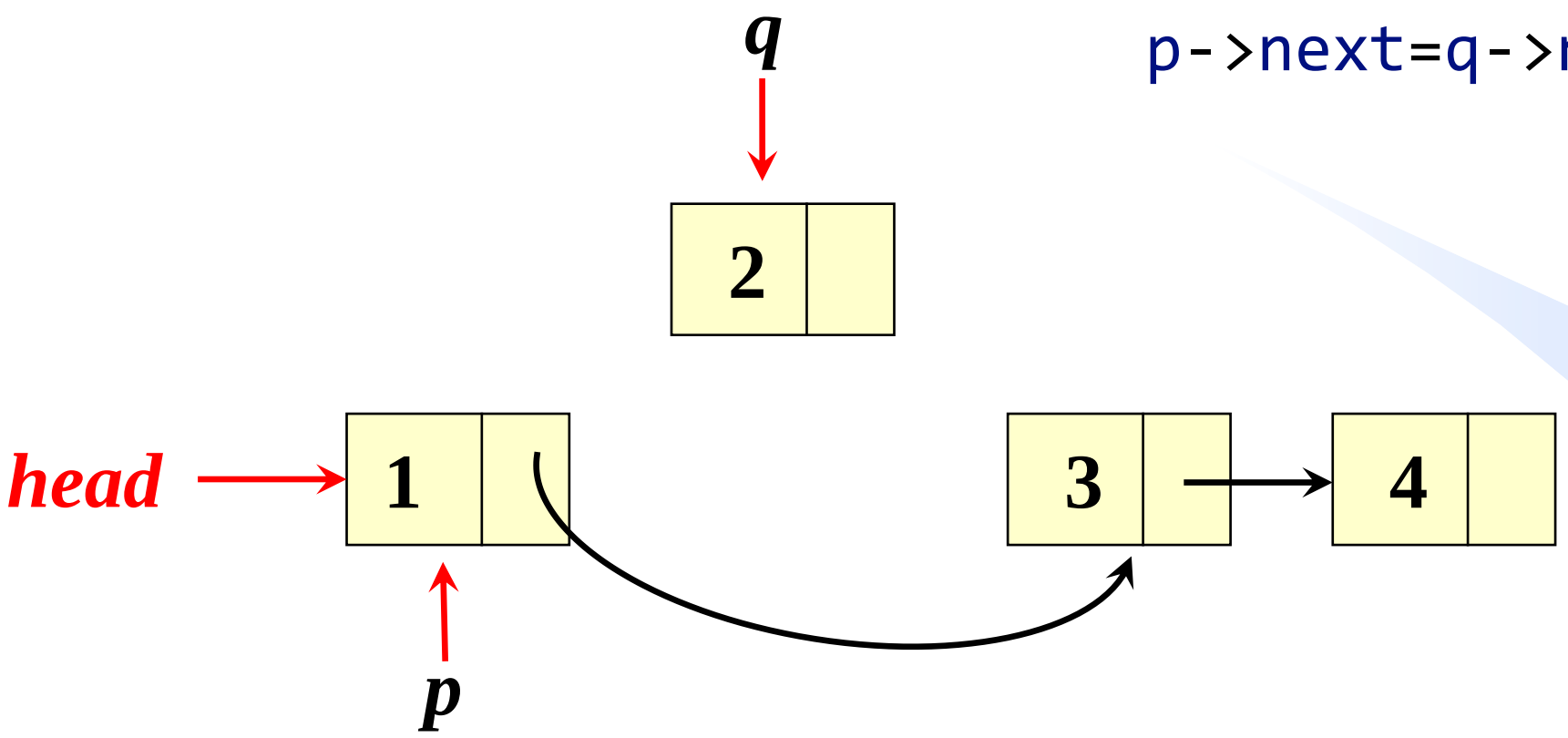


把q从链表中摘出

反转单链表

B

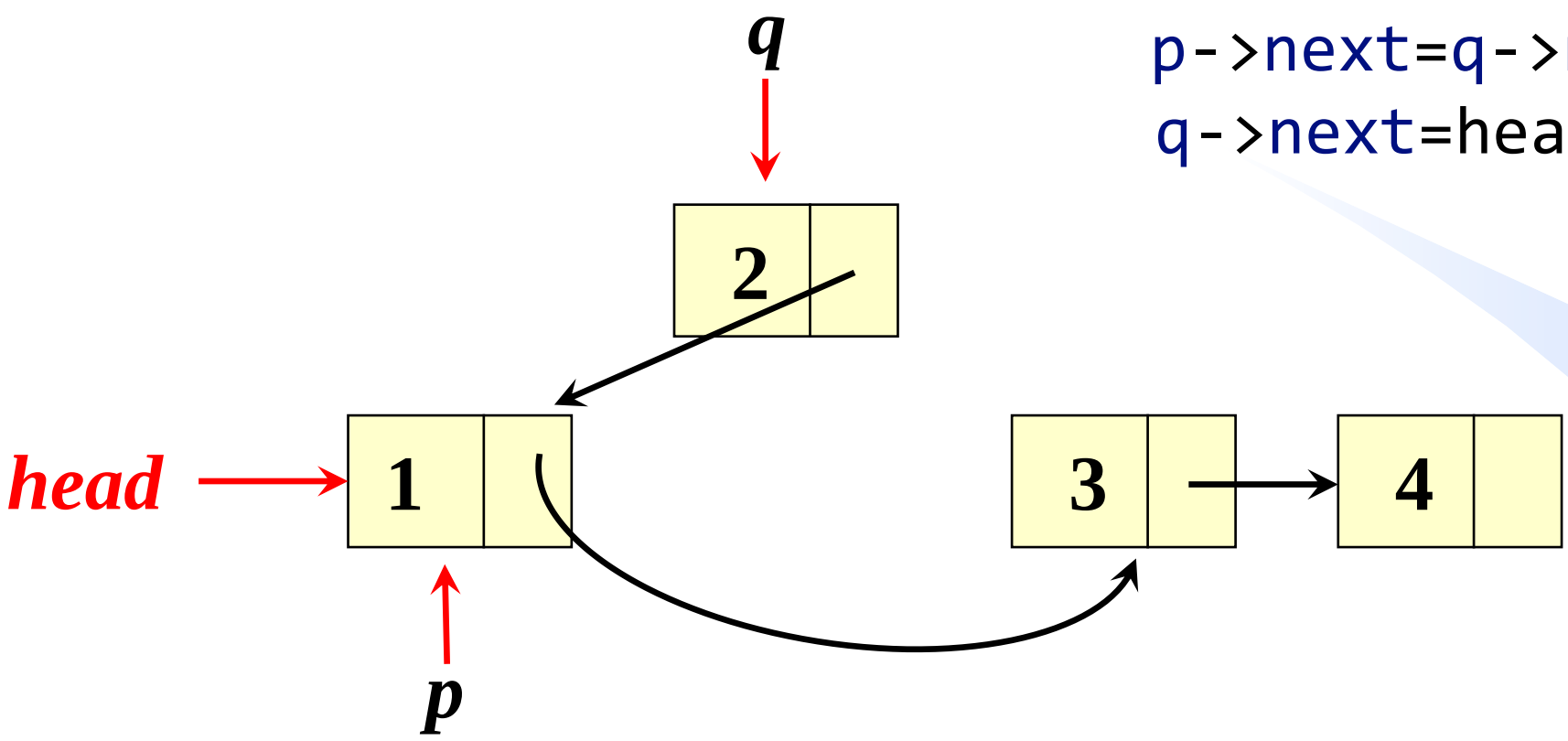
```
p->next=q->next;
```



把q从链表中摘出

反转单链表

B

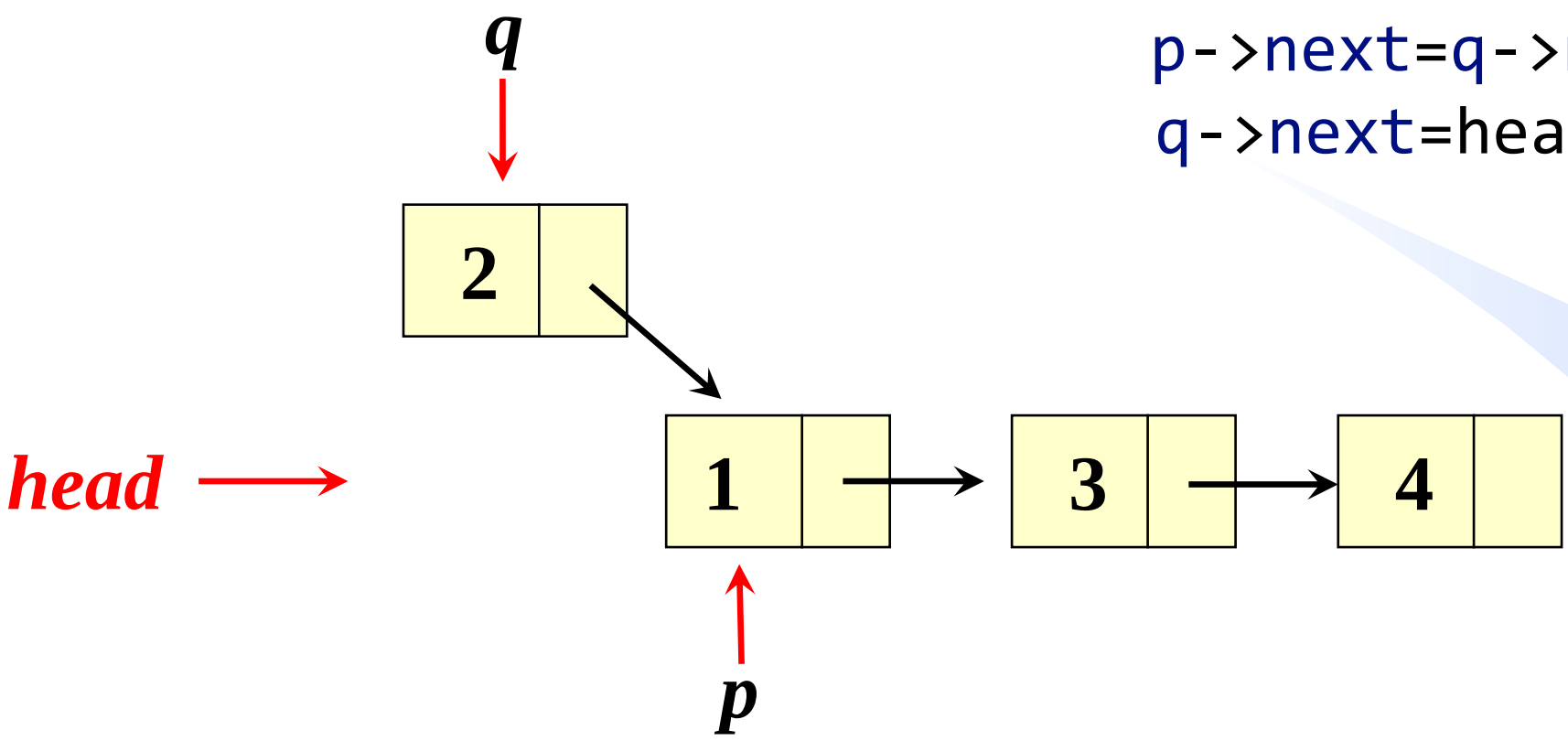



```
p->next=q->next;
q->next=head;
```

把q插入到表头

反转单链表

B



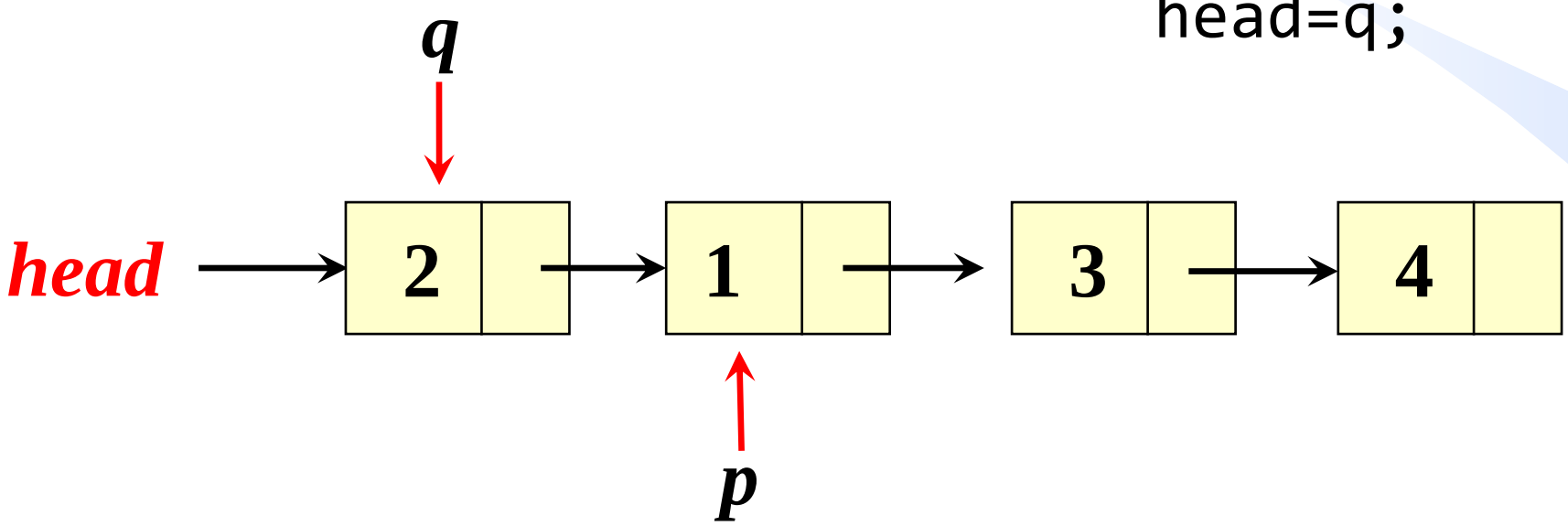

```
p->next=q->next;
q->next=head;
```

把q插入到表头

反转单链表

B

```
p->next=q->next;
q->next=head;
head=q;
```

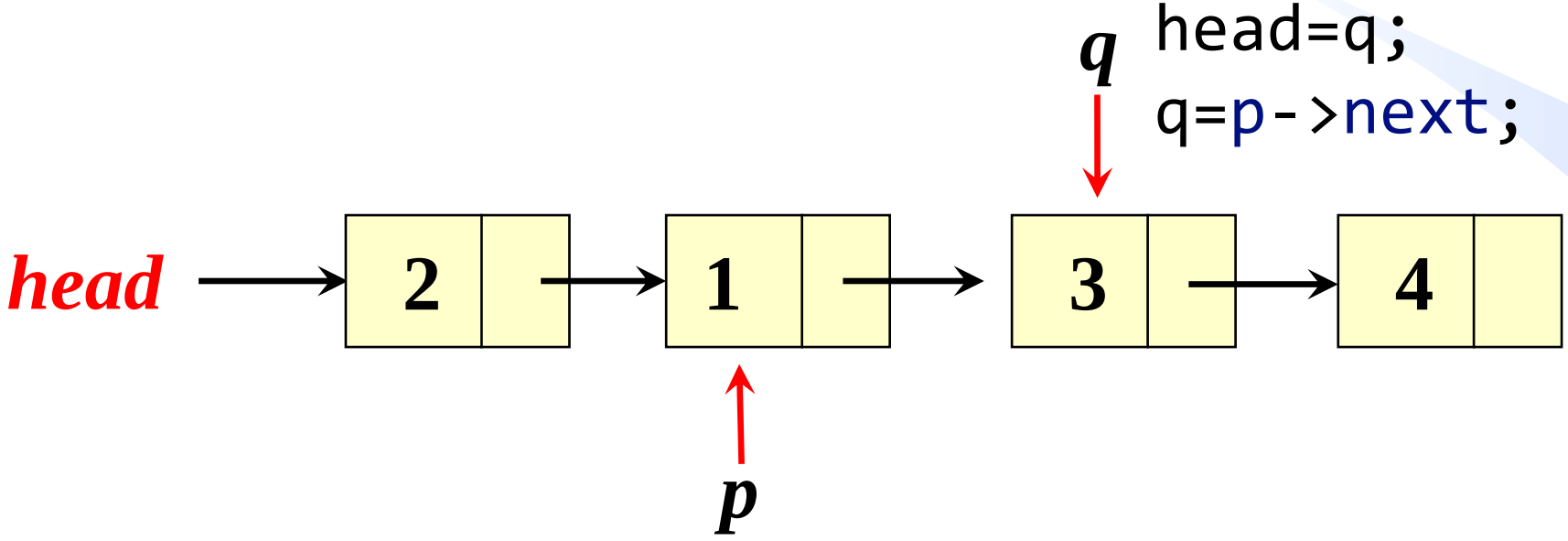


把q插入到表头

反转单链表

B

```
p->next=q->next;
q->next=head;
head=q;
q=p->next;
```



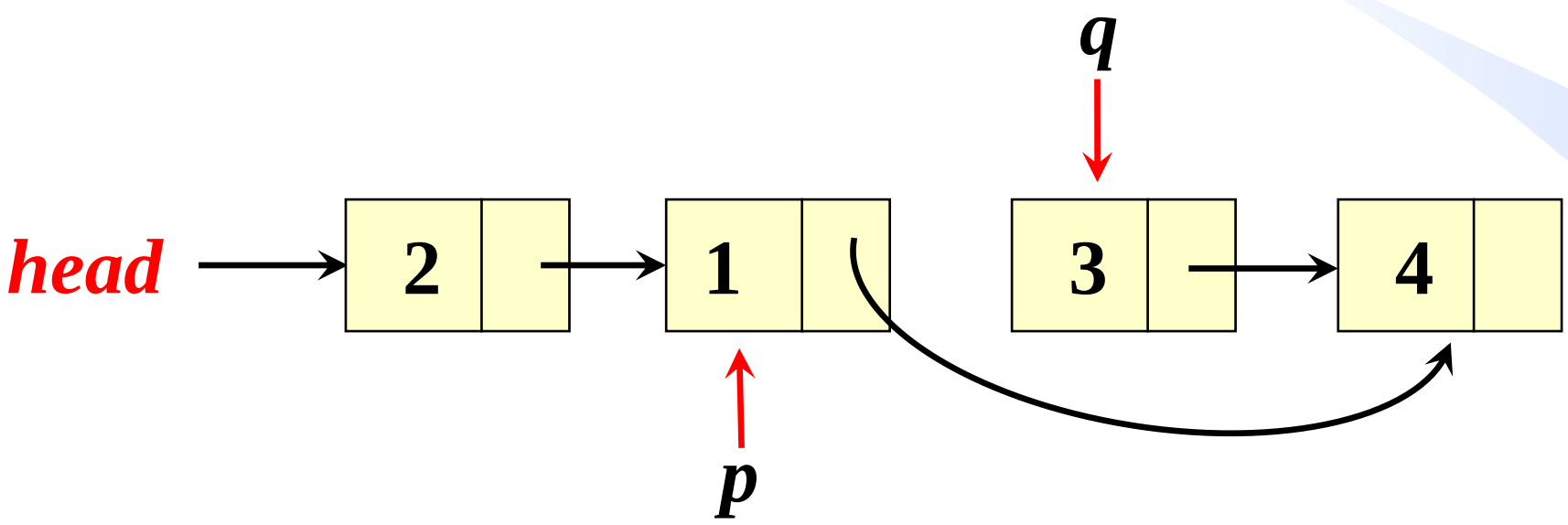
令q指向p的后继

把q从链表中摘出

反转单链表

B

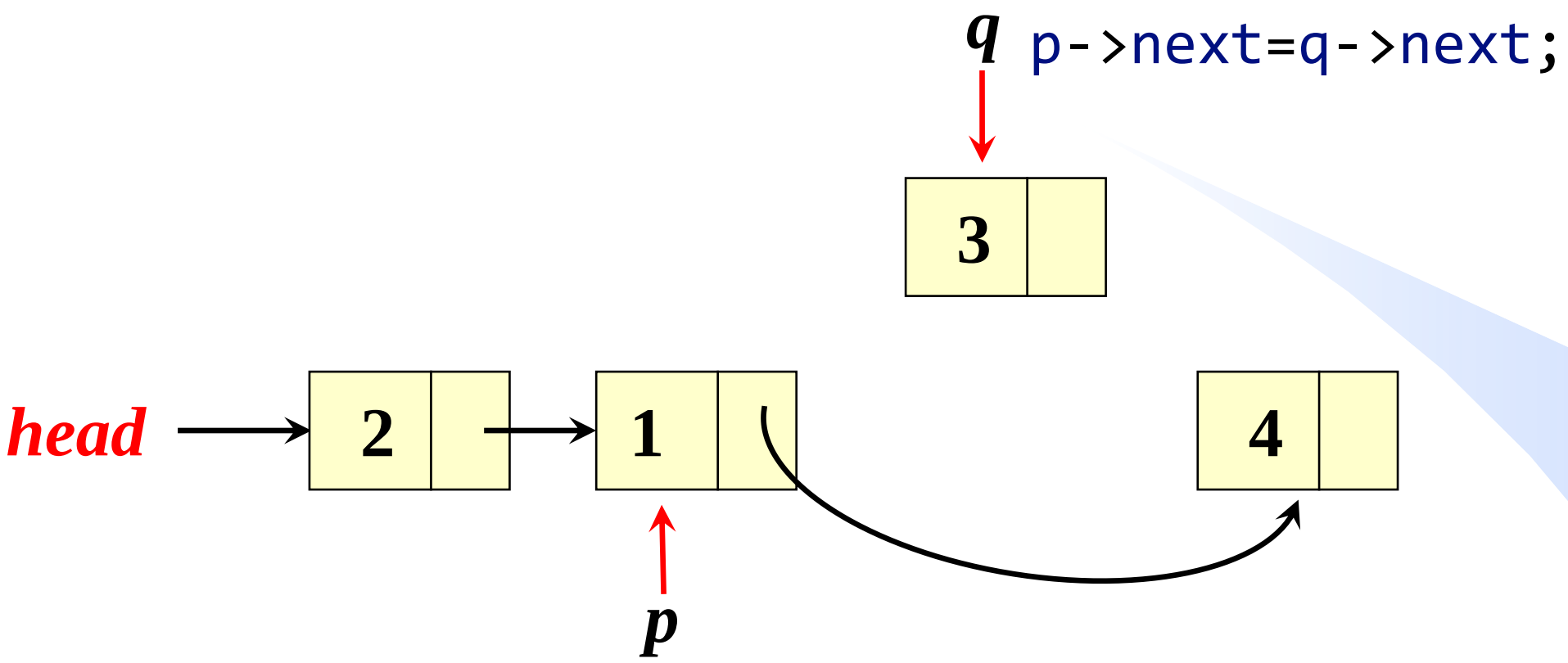
```
p->next=q->next;
```



把q从链表中摘出

反转单链表

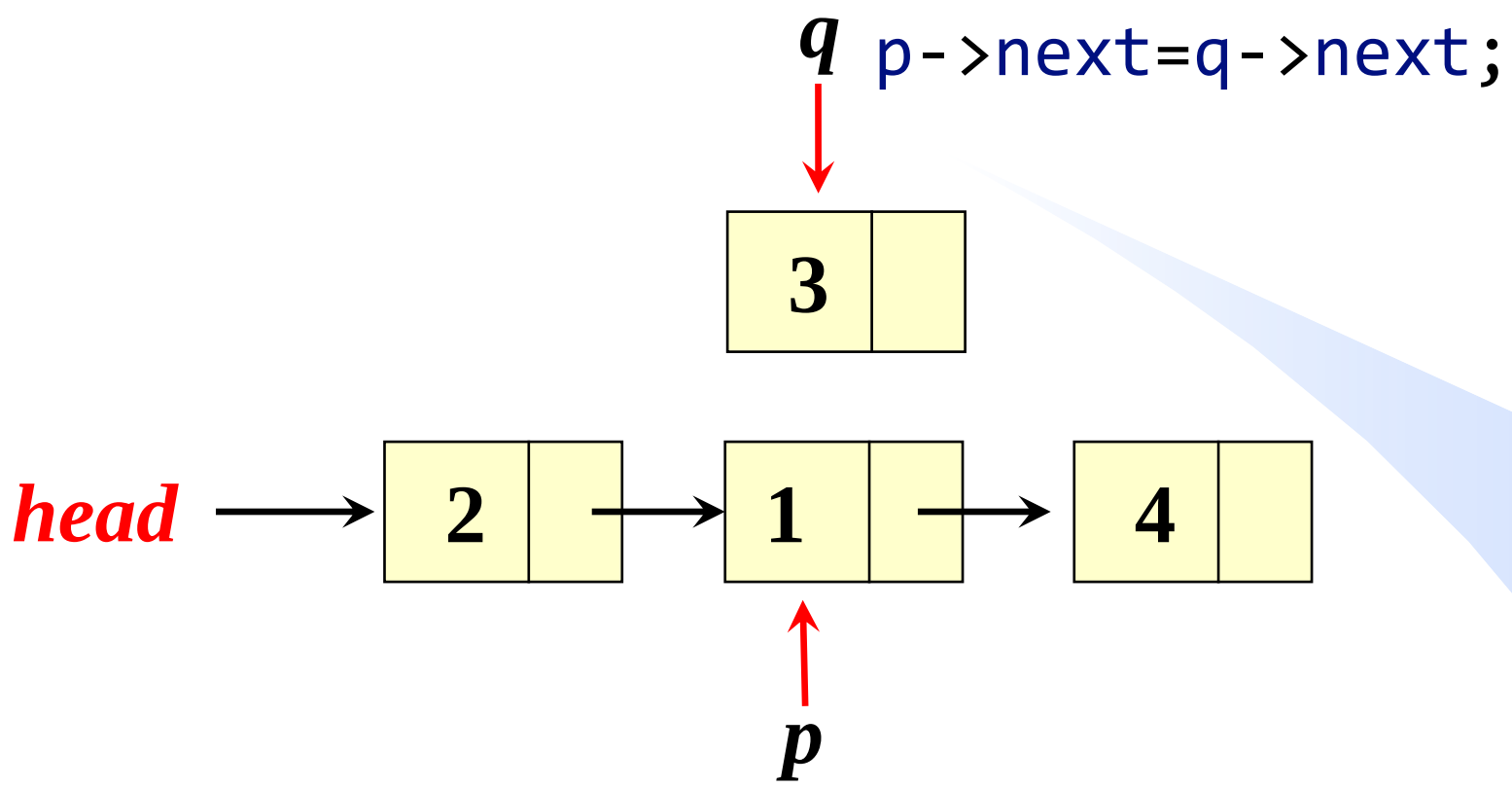
B



把q从链表中摘出

反转单链表

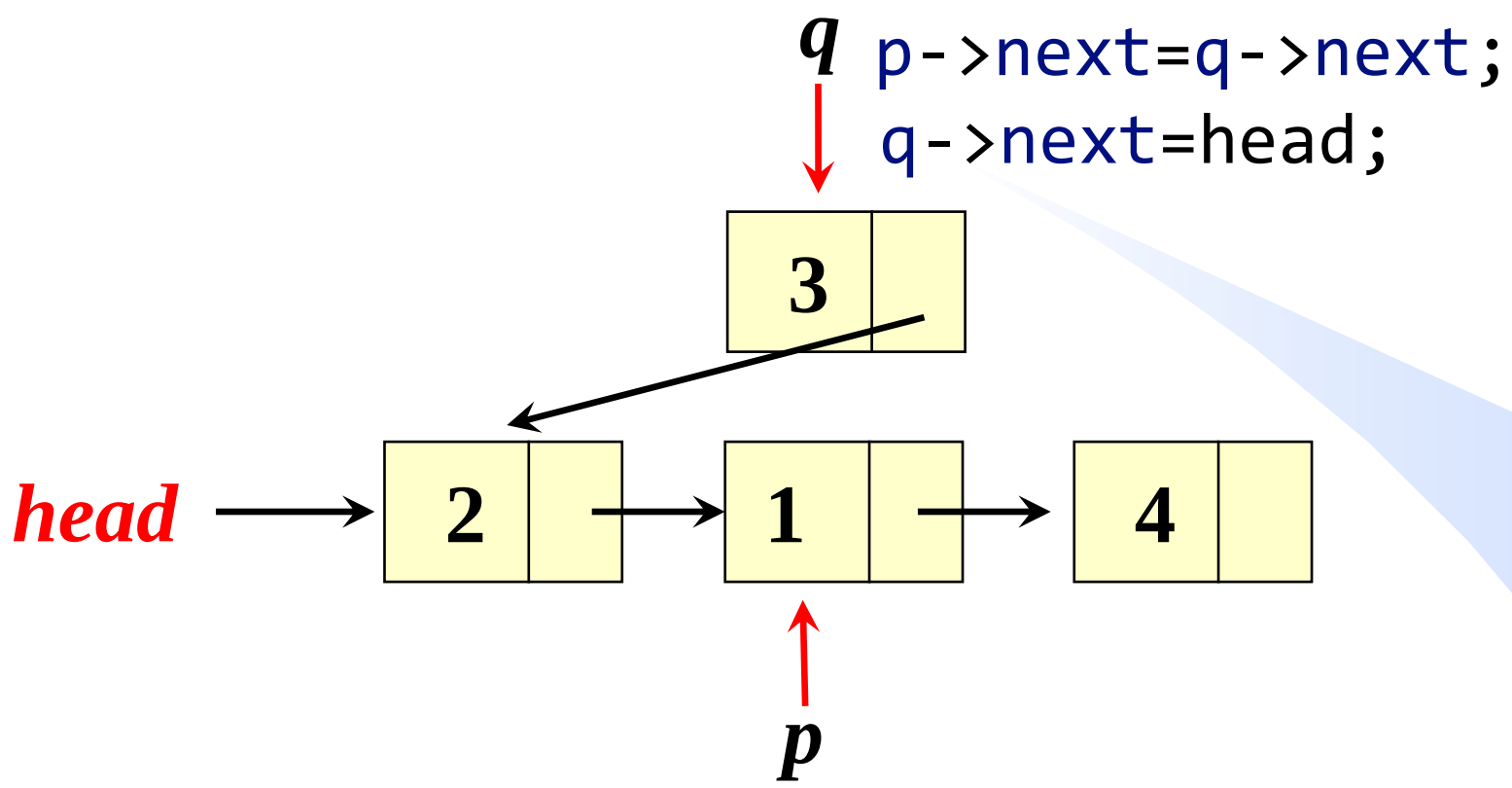
B



把q插入到表头

反转单链表

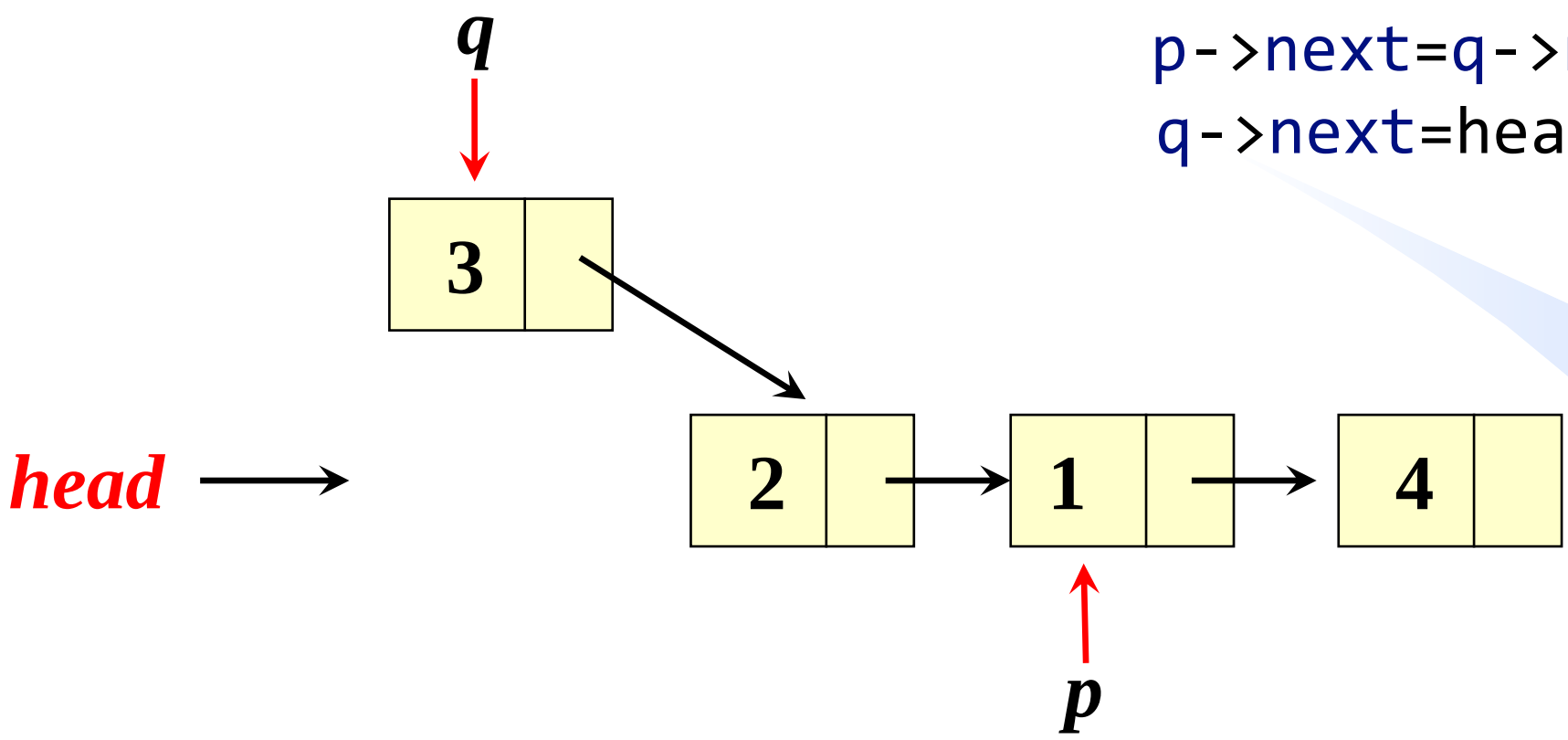
B



把q插入到表头

反转单链表

B



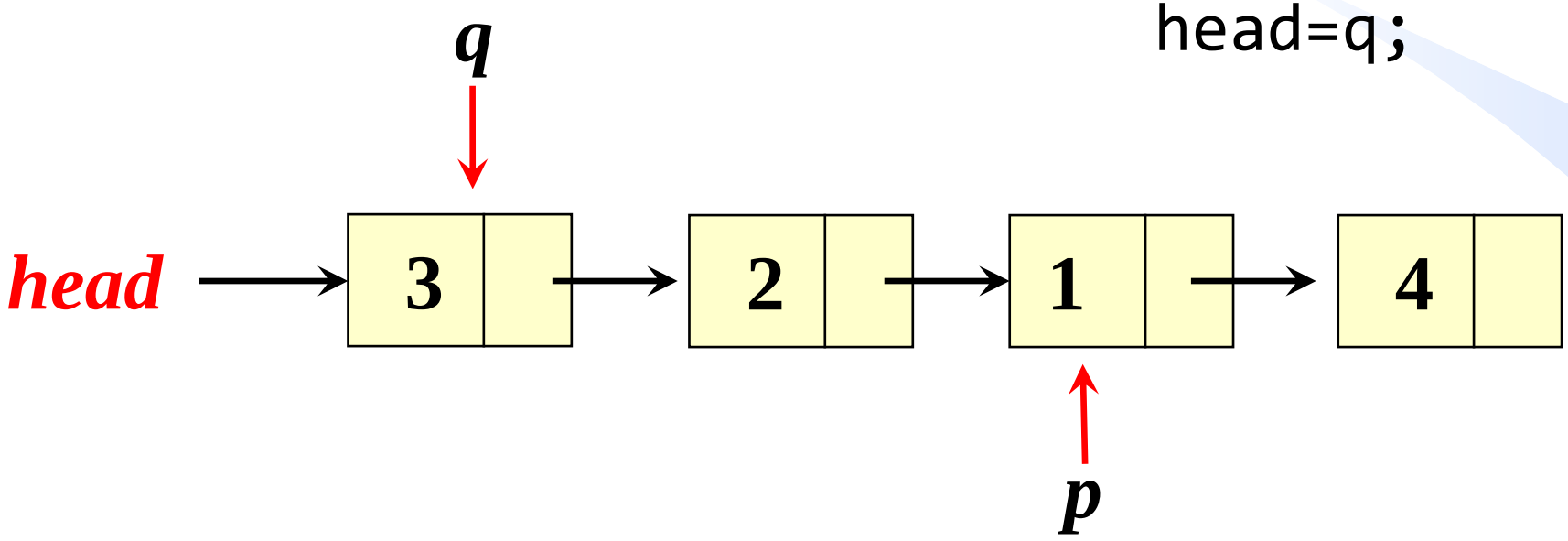

```
p->next=q->next;
q->next=head;
```

把q插入到表头

反转单链表

B

```
p->next=q->next;
q->next=head;
head=q;
```

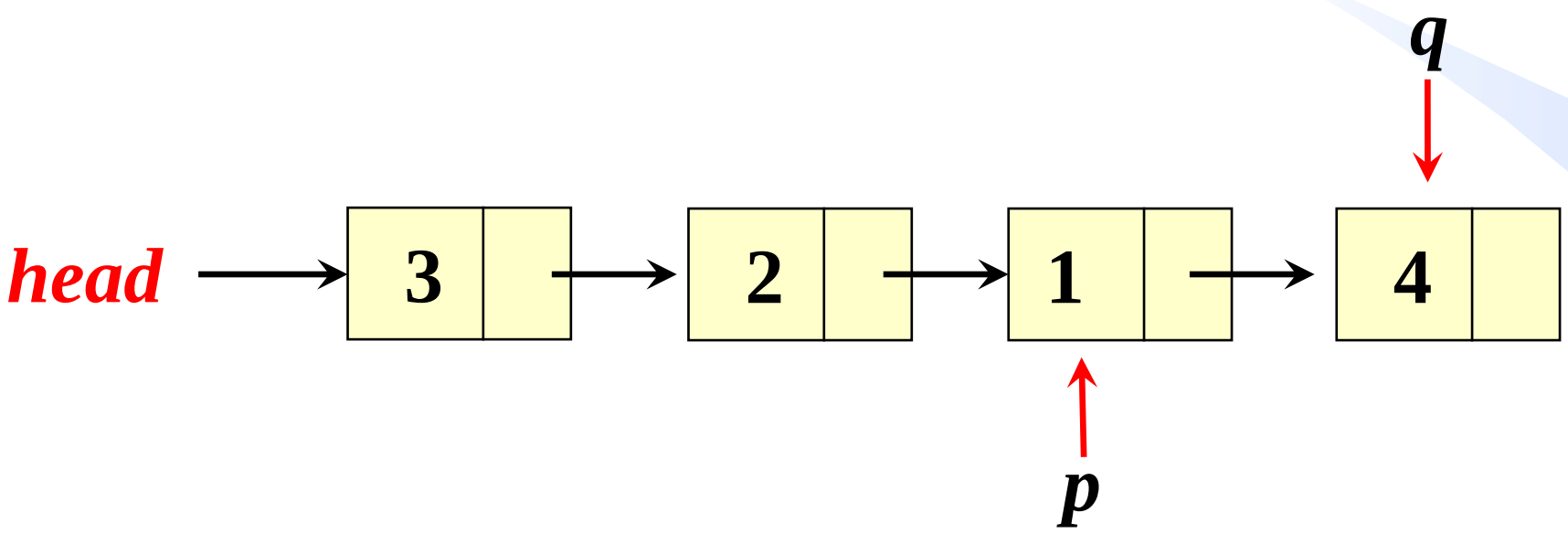


把q插入到表头

反转单链表

B

```
q=p->next;
```



令q为p的后继

B

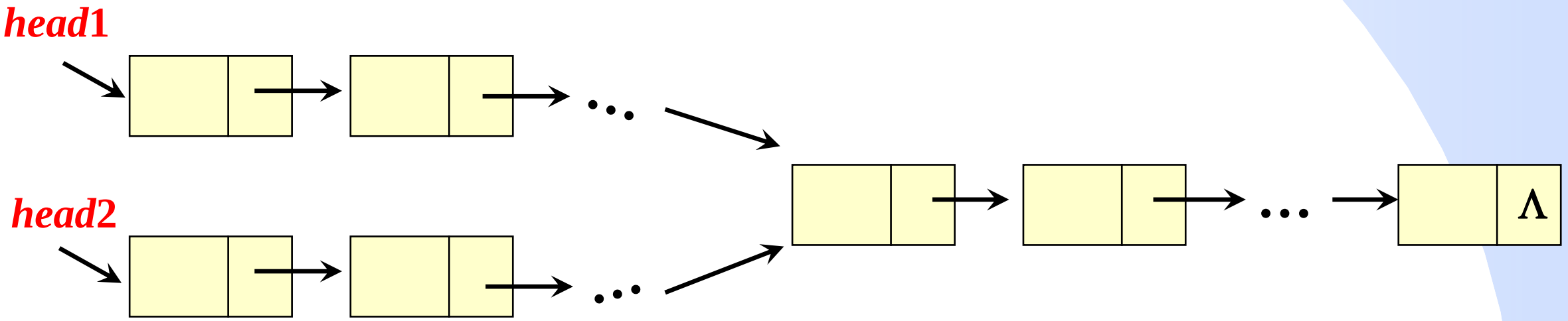
```
ListNode* reverseList(ListNode* head) {
        if(head==nullptr) return head;
        ListNode *p=head, *q=head->next;
        while(q!=nullptr){
            p->next=q->next;
            q->next=head;
            head=q;
            q=p->next;
        }
        return head;
    }
```

Talk is cheap,
show me the code

链表相交问题

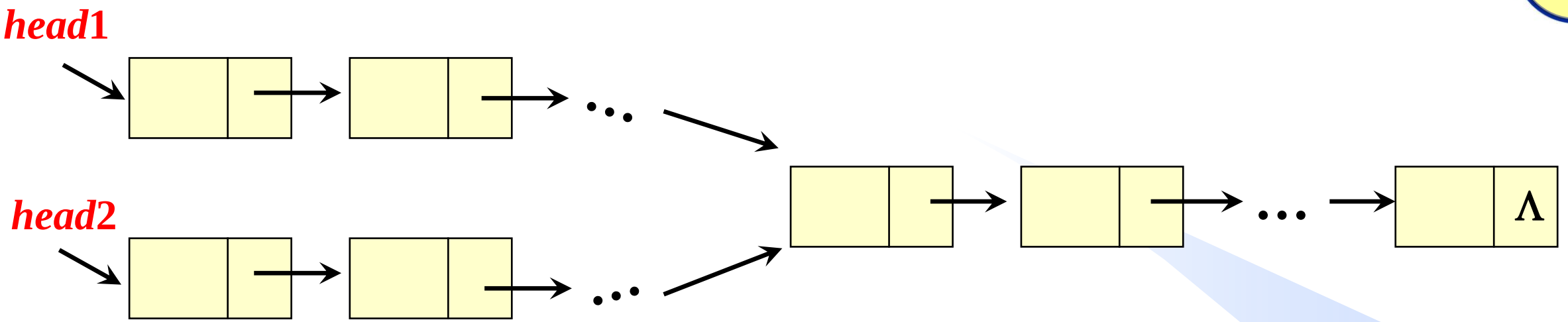
B

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

B



方案1：

FOR ∀结点i ∈ 链表1 DO

(

判断“结点i”是否在链表2中.

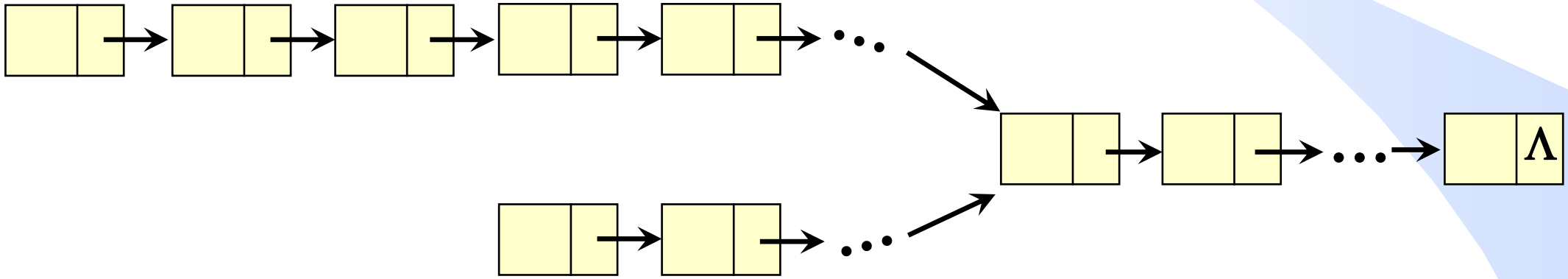
)

时间复杂度：$O(L_1*L_2)$

链表相交问题——思路

B

- 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*p*1指向较长的那个链表，*p*2指向较短的那个链表，*p*1先向后移动|*L*1-*L*2|步，使*p*1和*p*2“对齐”，然后*p*1和*p*2同时并行向右移动，每移动一步比较*p*1和*p*2是否相等，当二者首次相等时，其指向的结点即为交点。

B

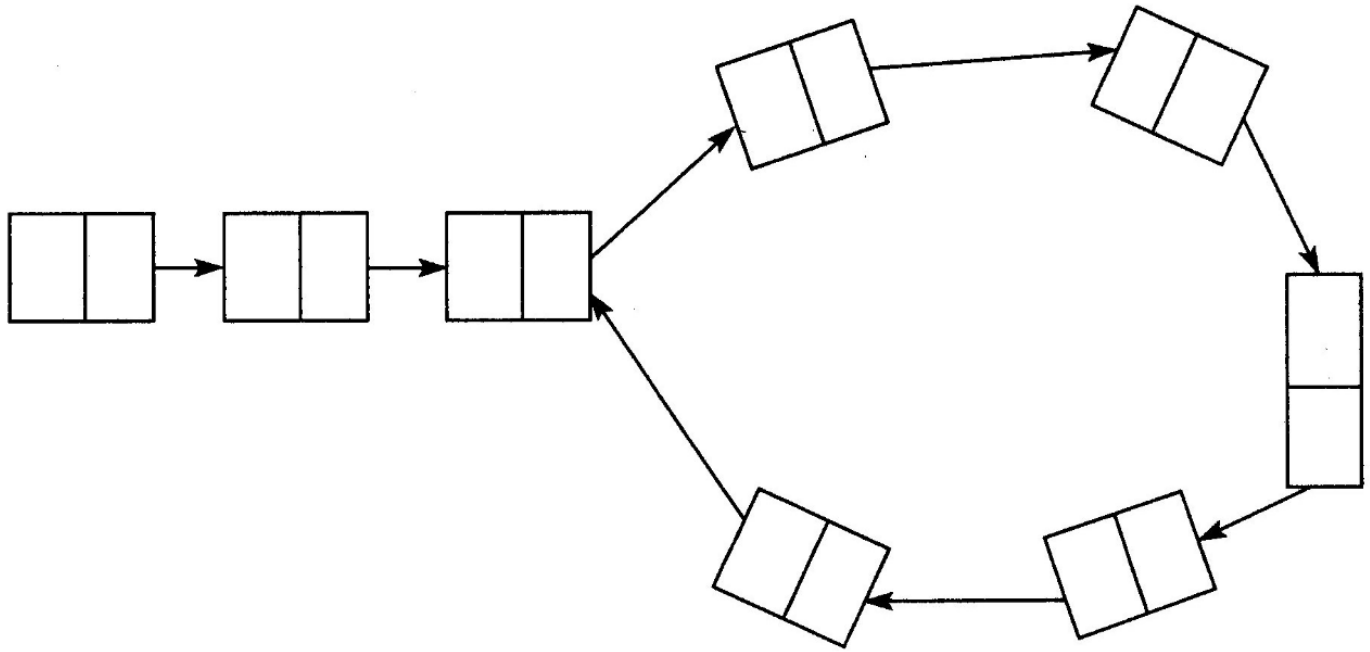
```
ListNode* cross(ListNode *head1, ListNode *head2) {
      //若两个链表相交，返回交点指针，否则返回NULL
      if (head1==NULL || head2==NULL) return NULL;
      if (head1==head2) return head1;
      ListNode *p1=head1,*p2=head2;
      int L1=1,L2=1;
      while (p1->next!=NULL) { L1++;  p1=p1->next;}
      while (p2->next!=NULL)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

单链表判环问题

B

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

B

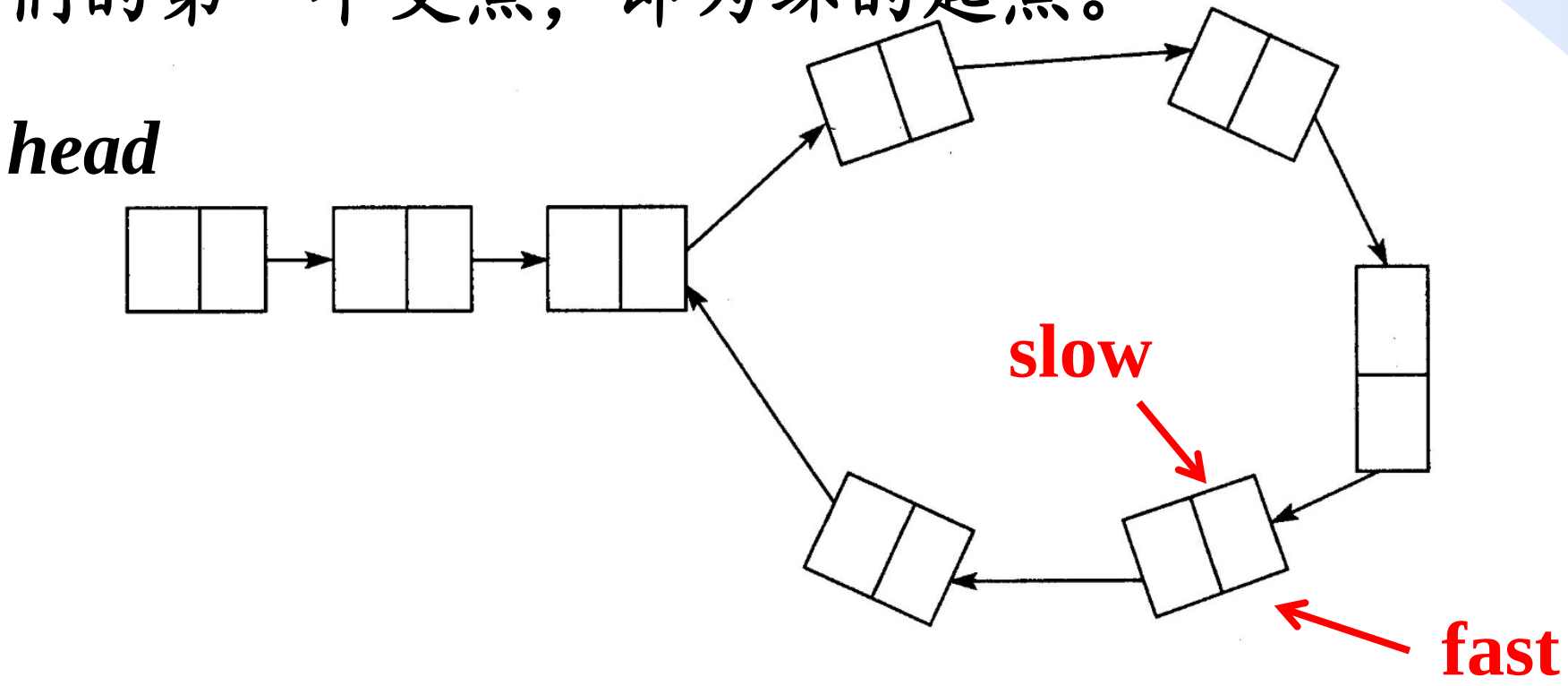
- 使用两个指针slow和fast从链表头开始遍历，slow每次前进1步，fast每次前进2步。
- 如果含有环，fast和slow必然会在某个时刻相遇（fast==slow），好比在环形跑道上赛跑时运动员的套圈。
- 如果遍历过程中，最终fast达到NULL，则说明无环。

```
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

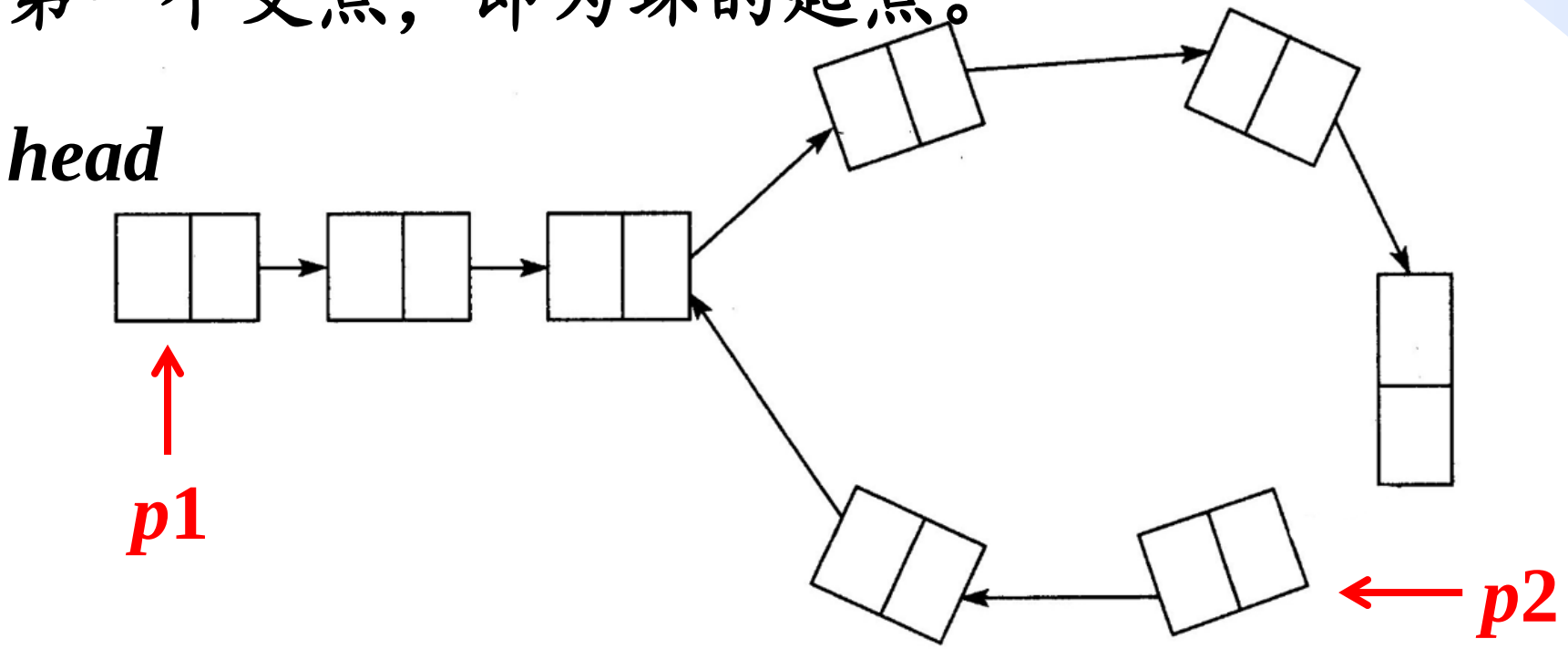
B

从相遇结点（fast==slow所对应的结点）处断开环，令p1←head，p2←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

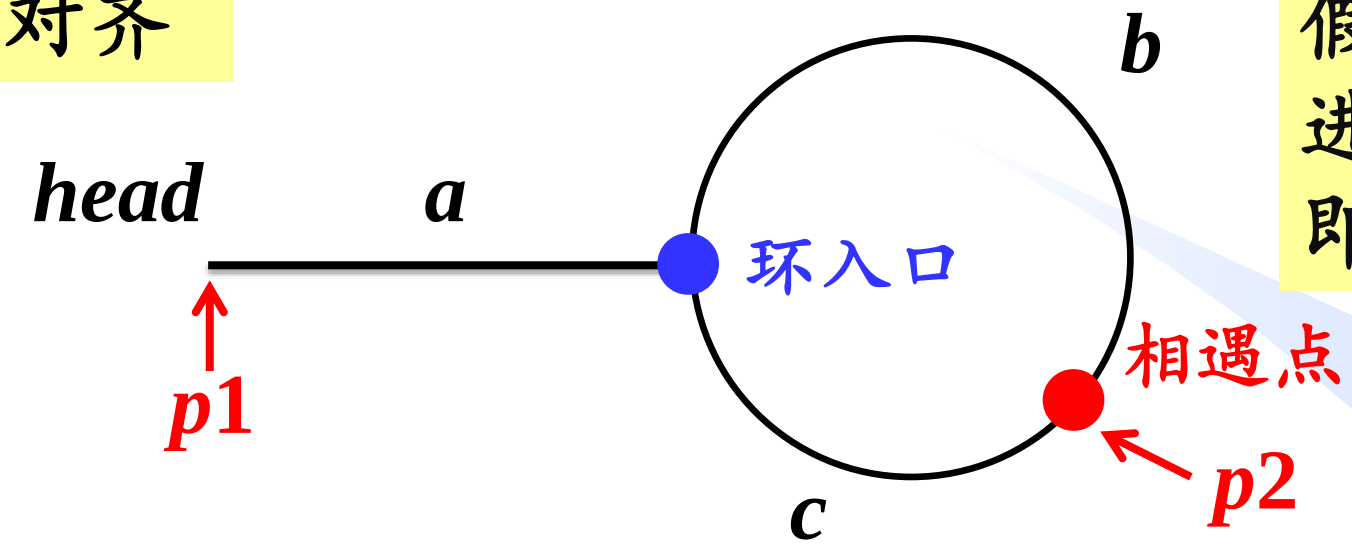
从相遇结点（*fast*==*slow*所对应的结点）处断开环，令*p*1←*head*, *p*2←*fast*。此时，原单链表可以看作两条单链表，一条从*p*1开始，另一条从*p*2开始，结合链表相交问题，找到它们的第一个交点，即为环的起点。



单链表判环问题

B

p1和p2无需对齐



假定fast指针进入环1圈后即与slow相遇

2*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

B

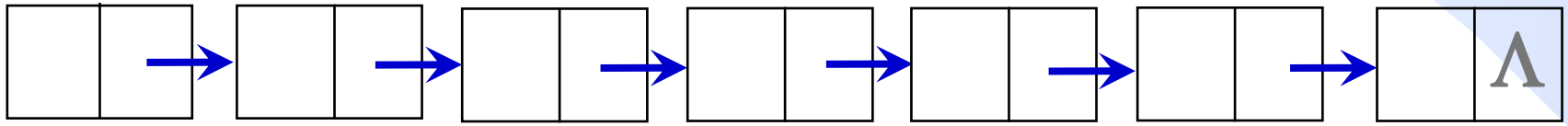
```
ListNode* detectCycleEntrance(ListNode *head) {
        ListNode *slow = head, *fast = head;
        while (fast != NULL && fast->next != NULL){
            fast = fast->next->next;
            slow = slow->next;
            if (fast == slow) {  //有环，找环的入口
                ListNode *p1 = head, *p2=fast;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

B

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

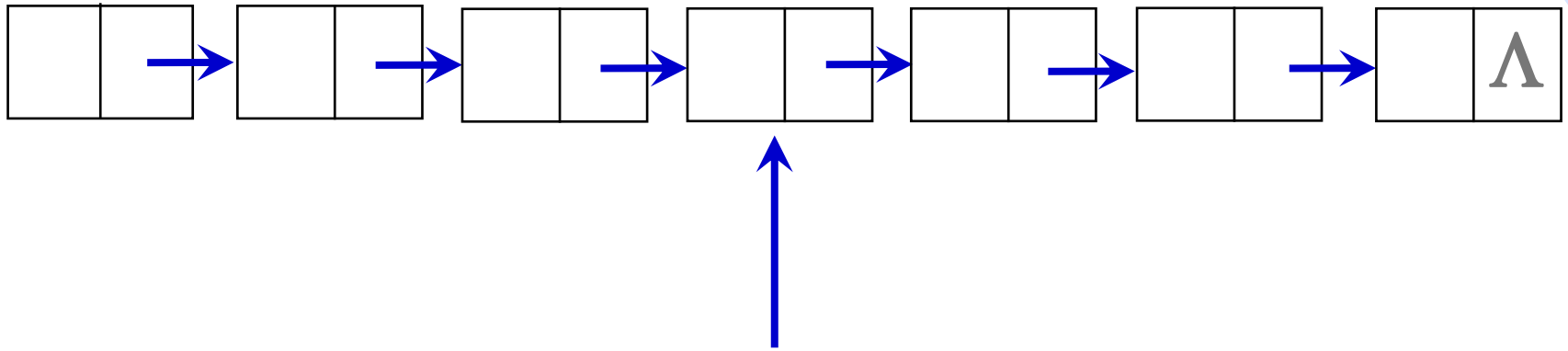
refer

上海自来水来自海上

判断一个单链表是否是回文

B

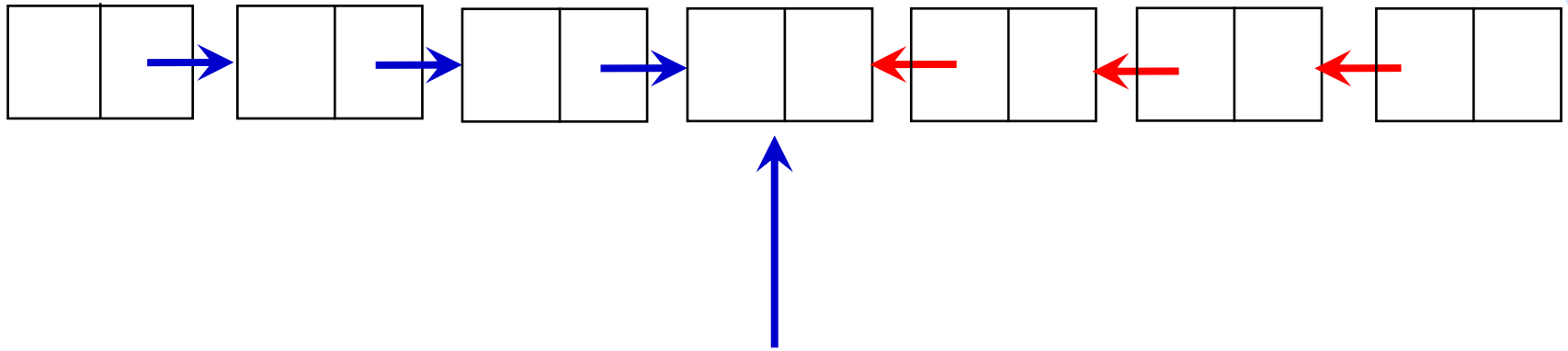
找到中间结点，将后半部分链表反转



判断一个单链表是否是回文

B

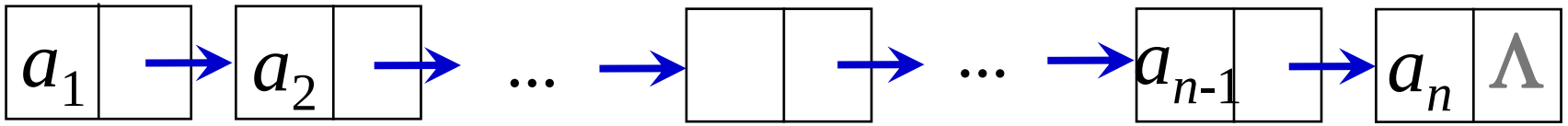
找到中间结点，将后半部分链表反转



重排链表结点

B

设线性表$L=(a_1, a_2, a_3, \ldots, a_{n-2}, a_{n-1}, a_n)$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$L'=(a_1, a_n, a_2, a_{n-1}, a_3, a_{n-2}\ldots)$。【大厂面试题LeetCode 143】



重排链表结点

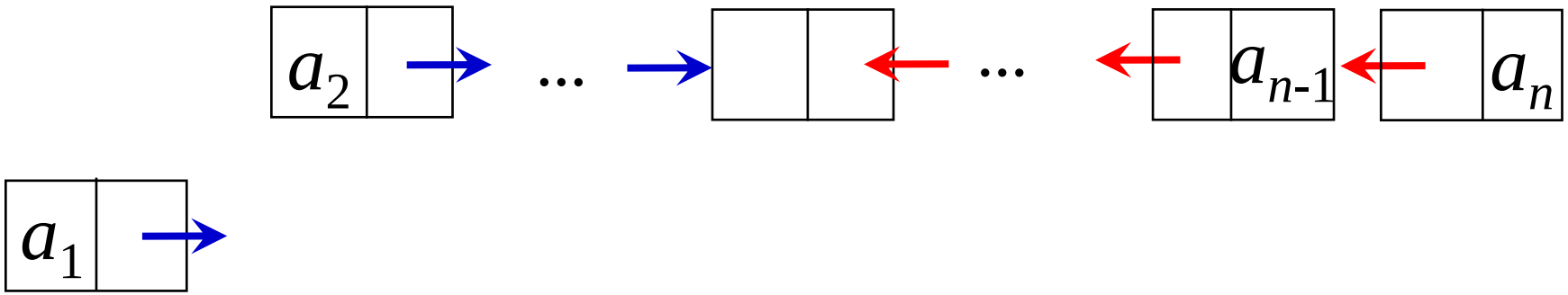
B

设线性表$L=(a_1, a_2, a_3, \ldots, a_{n-2}, a_{n-1}, a_n)$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$L'=(a_1, a_n, a_2, a_{n-1}, a_3, a_{n-2}\ldots)$。【大厂面试题LeetCode 143】

重排链表结点

B

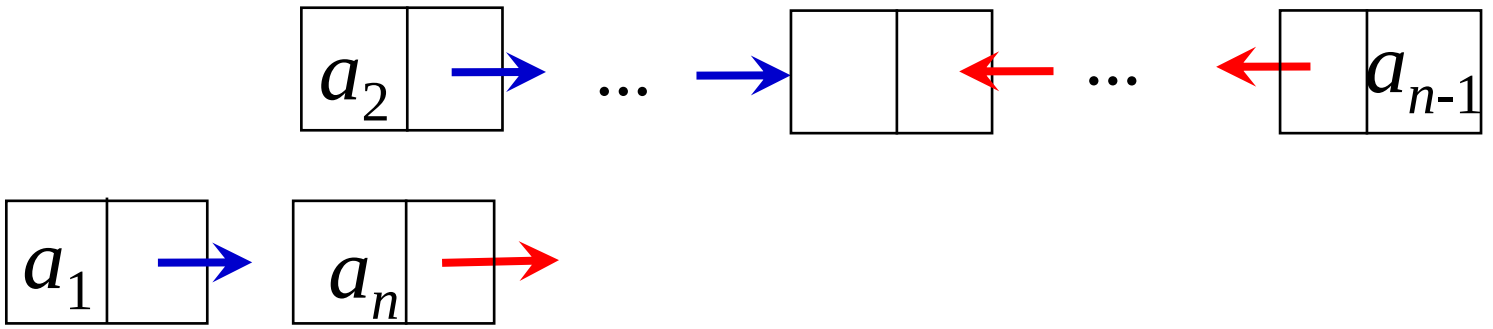
设线性表$L=(a_1, a_2, a_3, \ldots, a_{n-2}, a_{n-1}, a_n)$采用带哨位结点的单链表保存，请设计一个空间复杂度为O(1)且时间上尽可能高效的算法，重新排列L中的各结点，得到线性表$L'=(a_1, a_n, a_2, a_{n-1}, a_3, a_{n-2}\ldots)$。【大厂面试题LeetCode 143】



重排链表结点

B

设线性表$L=(a_1, a_2, a_3, \ldots, a_{n-2}, a_{n-1}, a_n)$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$L'=(a_1, a_n, a_2, a_{n-1}, a_3, a_{n-2}\ldots)$。【大厂面试题LeetCode 143】



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）