

点此获取更多资源

## 第04讲 线性表的循环链表和双向链表



六星教育首席架构师：Vico老师  
官方助理冰芯老师QQ：1930070991

2. 1 线性表的定义和特点

2. 2 案例引入

2. 3 线性的类型定义

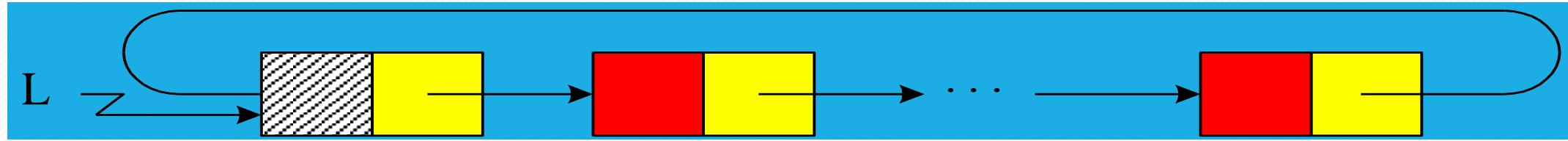
2. 4 线性表的顺序表示和实现

2. 5 线性表的循环链表和双向链表

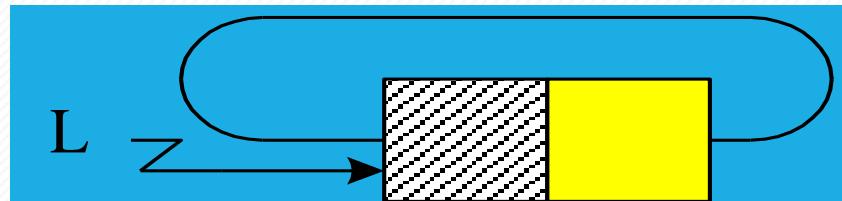
2. 6 顺序表和链表的比较

2. 7 线性表的应用

## 2.5.3 循环链表

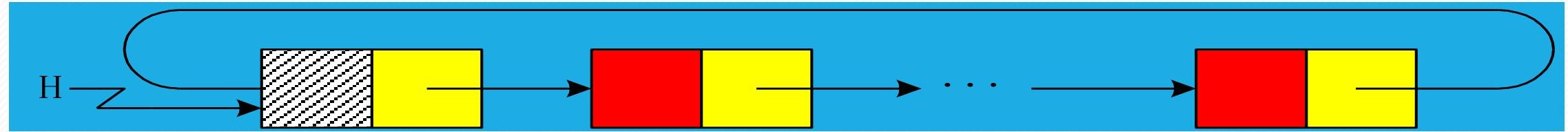


(a) 非空单循环链表



(b) 空表

$L \rightarrow \text{next} = L$



从循环链表中的任何一个结点的位置都可以找到其他所有结点，而单链表做不到；



循环链表中没有明显的尾端



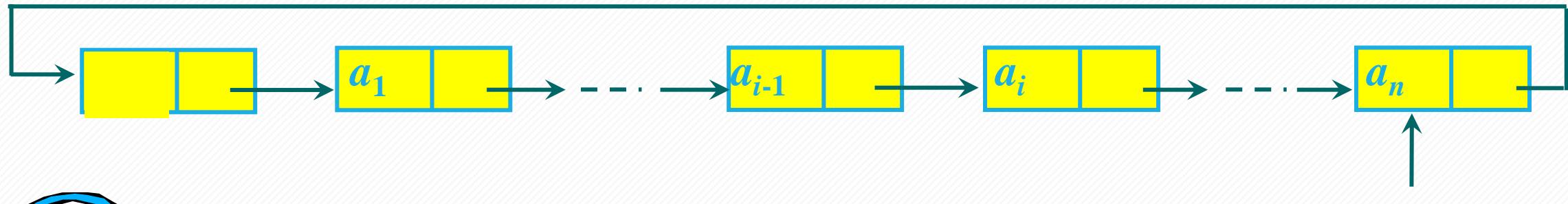
如何避免死循环

循环条件:  $p \neq \text{NULL} \rightarrow p \neq L$

$p \rightarrow \text{next} \neq \text{NULL} \rightarrow p \rightarrow \text{next} \neq L$



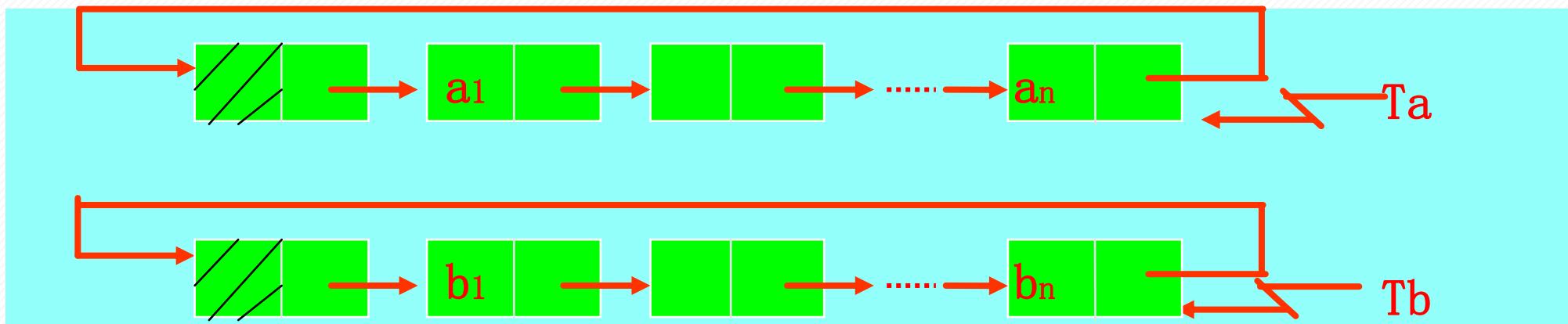
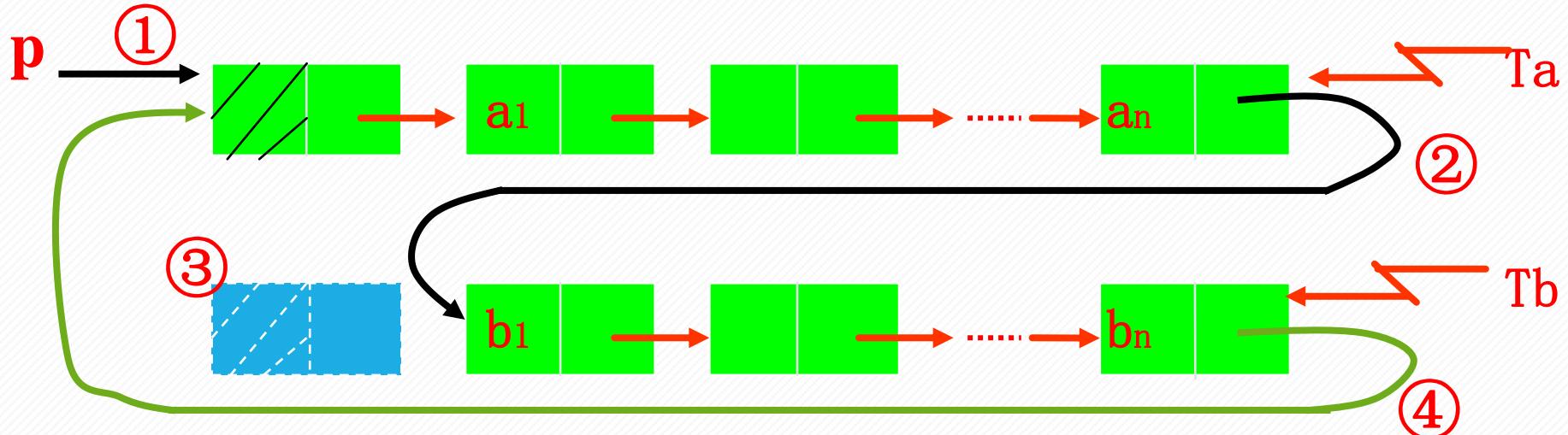
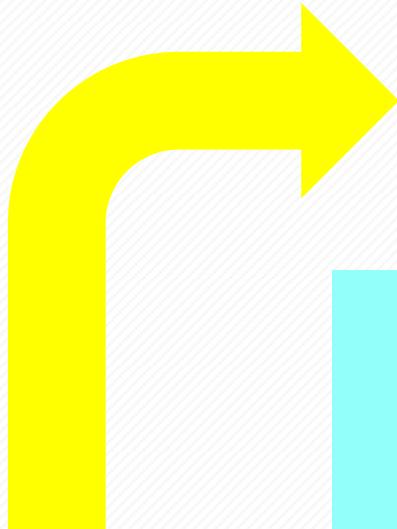
对循环链表，有时不给出头指针，而给出尾指针  
可以更方便的找到第一个和最后一个结点

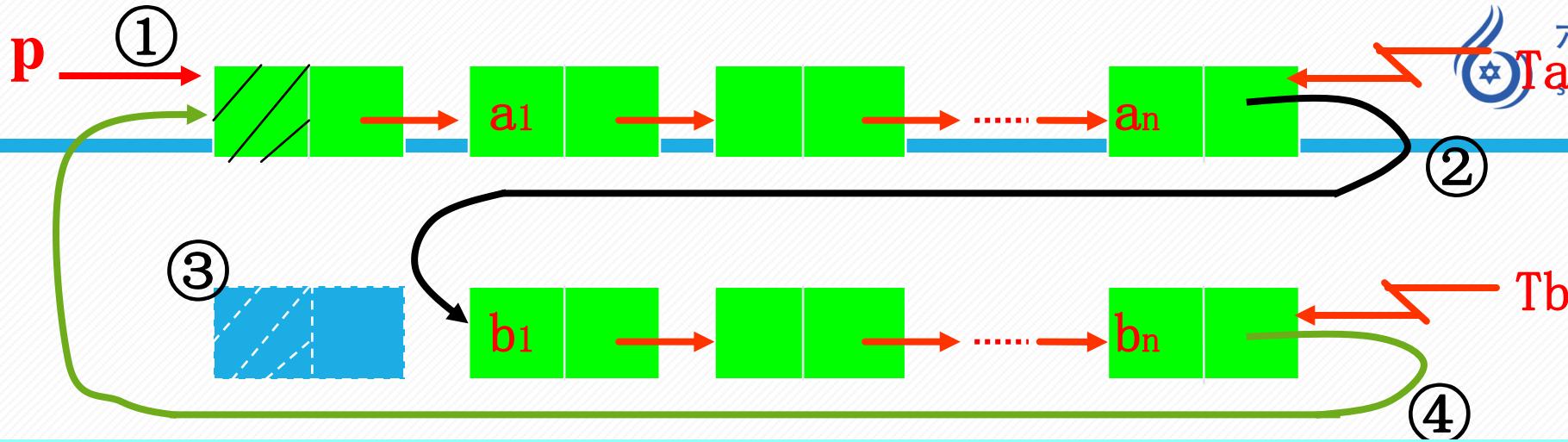


如何查找开始结点和终端结点？

开始结点： `rear->next->next`  
终端结点： `rear`

# 循环链表的合并





**LinkList Connect(LinkList Ta, LinkList Tb)**

{//假设Ta、 Tb都是非空的单循环链表

**p=Ta->next;**

**Ta->next=Tb->next->next;**

**delete Tb->next;**

**Tb->next=p;**

**return Tb;**

}

//①p存表头结点

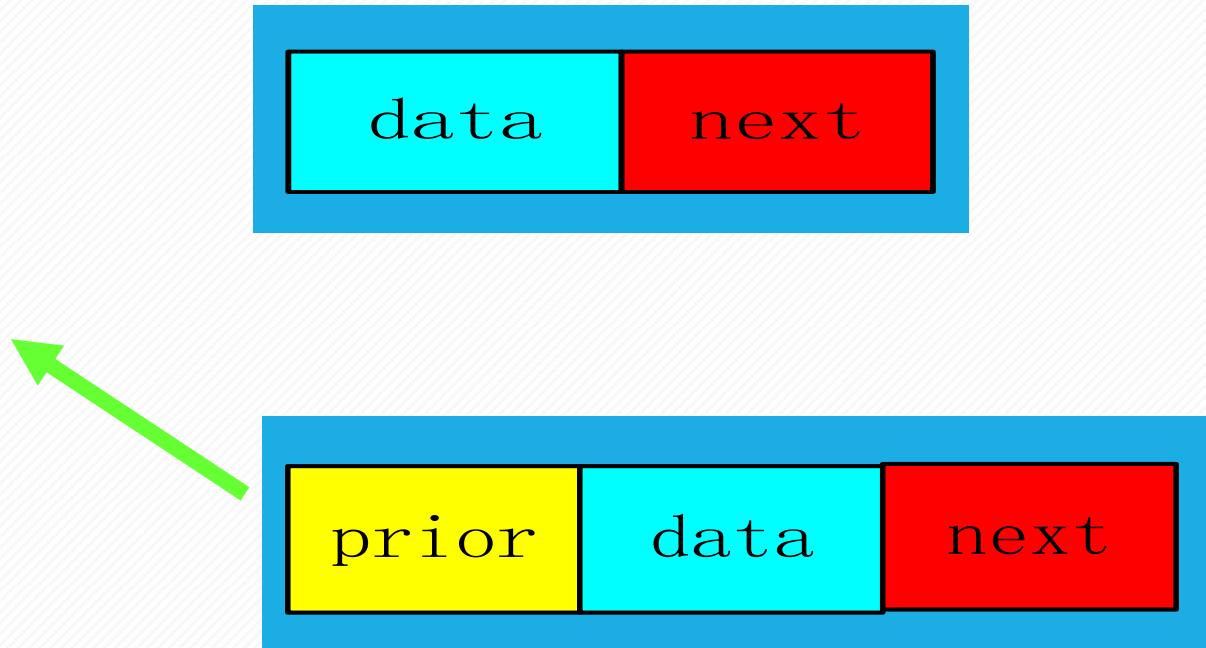
//②Tb表头连结Ta表尾

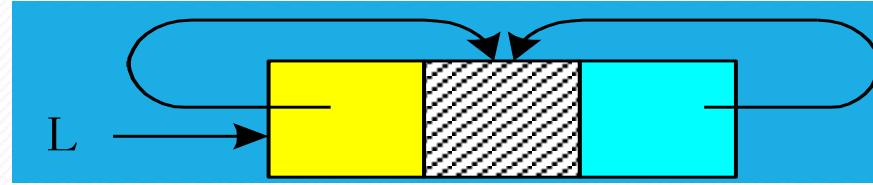
//③释放Tb表头结点

//④修改指针

## > 2.5.4 双向链表

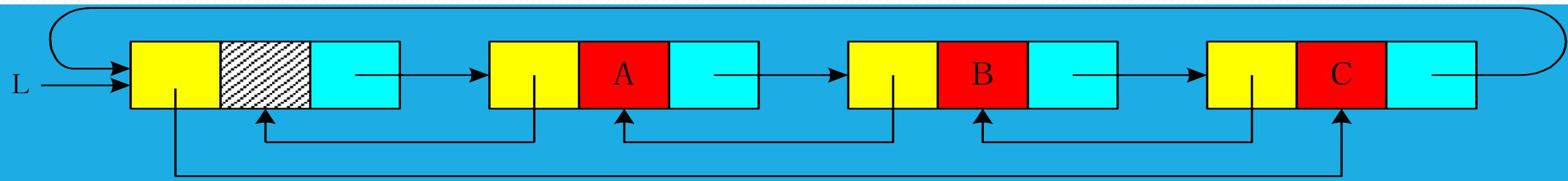
```
typedef struct DuLNode{  
    ElemtType  data;  
    struct DuLNode *prior;  
    struct DuLNode *next;  
}DuLNode, *DuLinkList
```





(a) 空双向循环链表

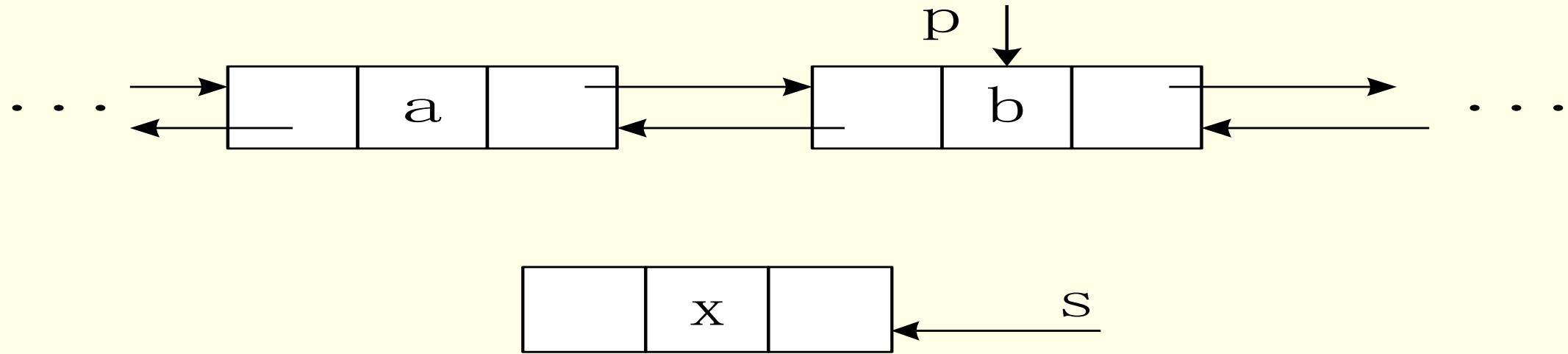
$L->next=L$



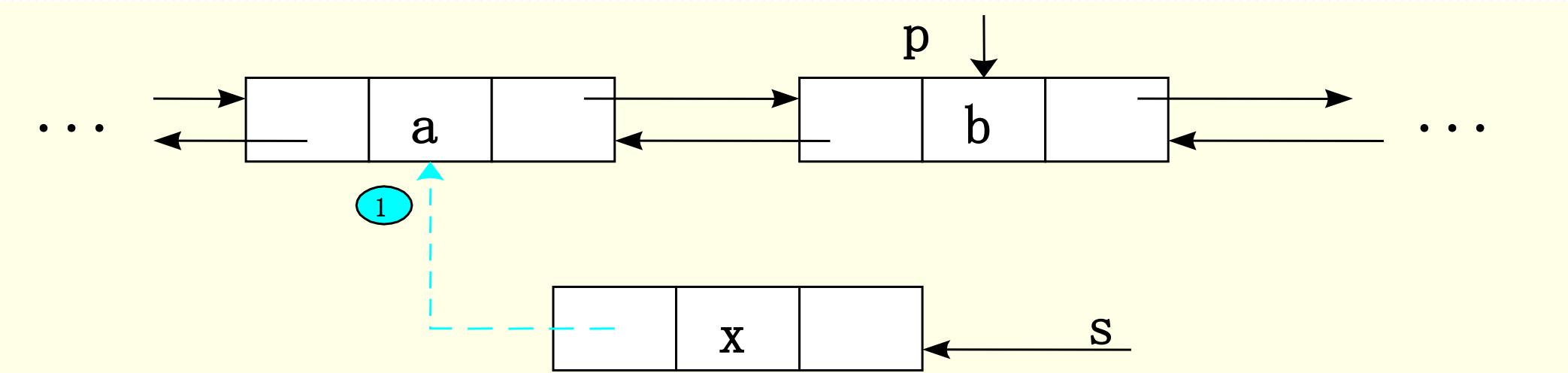
(b) 双向循环链表

$c->next->prior$      $c->prior->next$      $c$

# 双向链表的插入

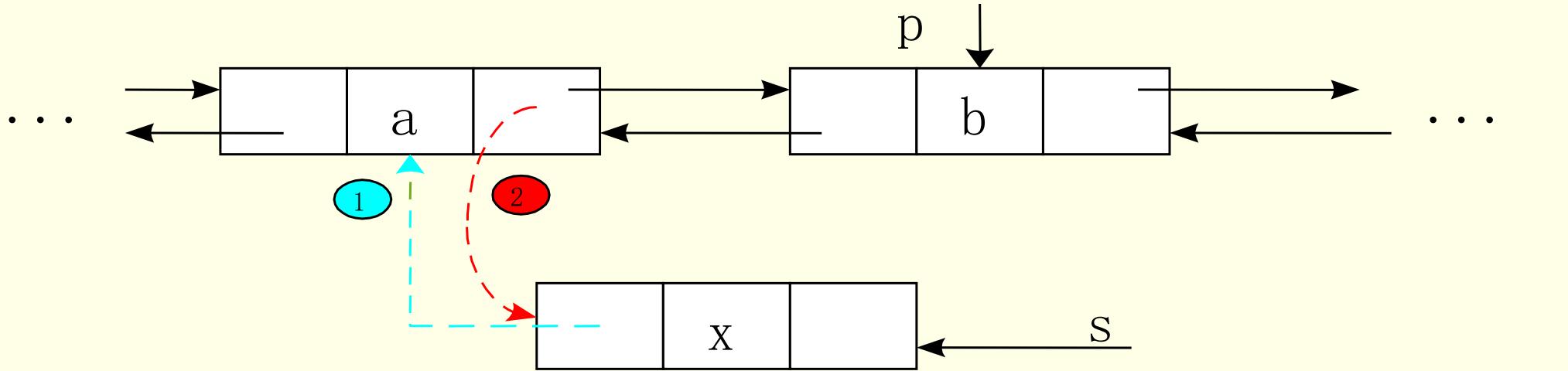


# 双向链表的插入



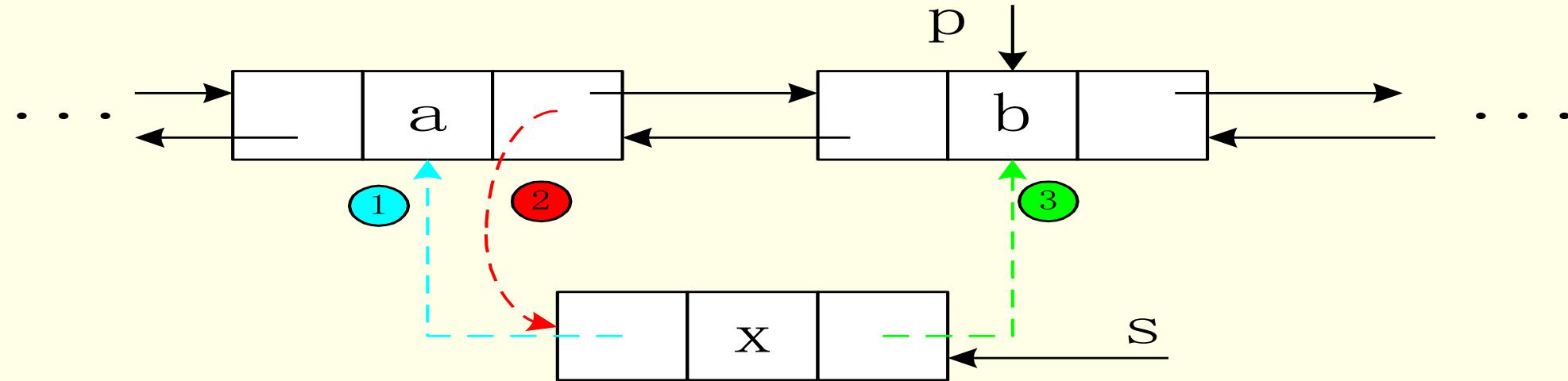
1.  $s->prior=p->prior;$

# 双向链表的插入



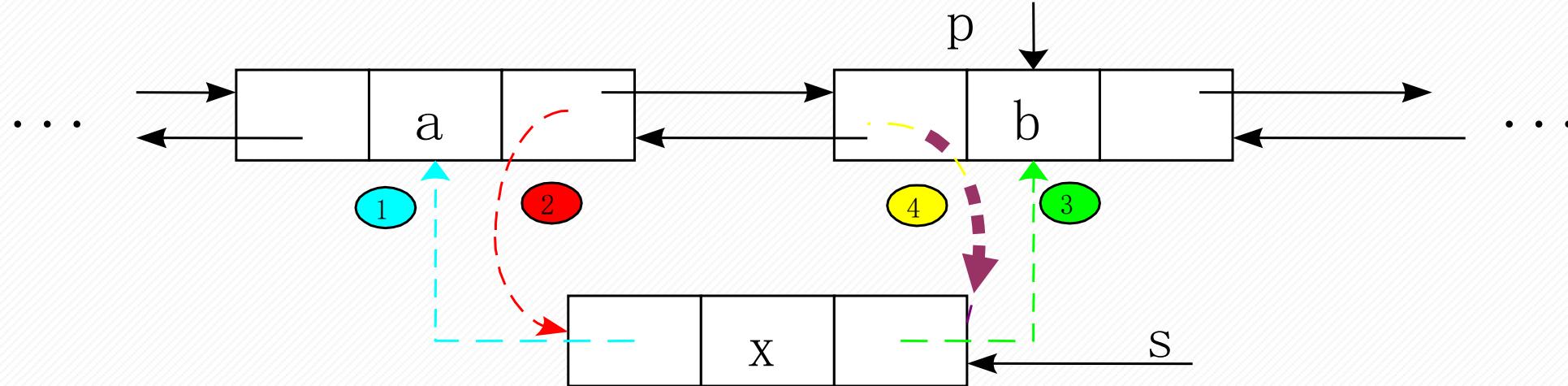
1. ***s->prior=p->prior;***
2. ***p->prior->next=s;***

# 双向链表的插入



1. ***s->prior=p->prior;***
2. ***p->prior->next=s;***
3. ***s->next=p;***

# 双向链表的插入

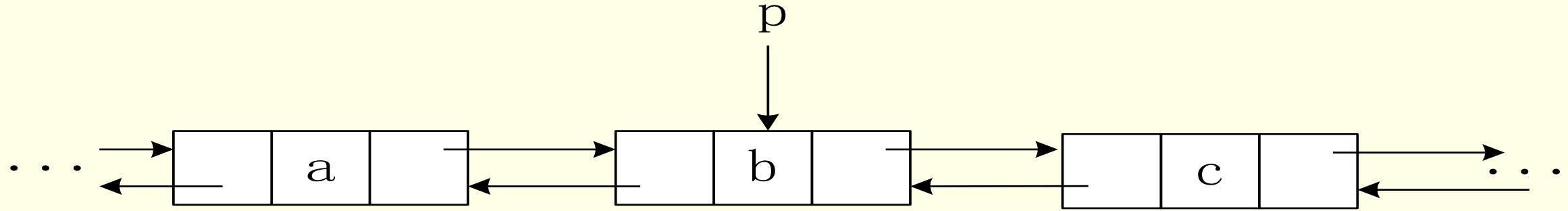


1. ***s->prior=p->prior;***
2. ***p->prior->next=s;***
3. ***s->next=p;***
4. ***p->prior=s;***

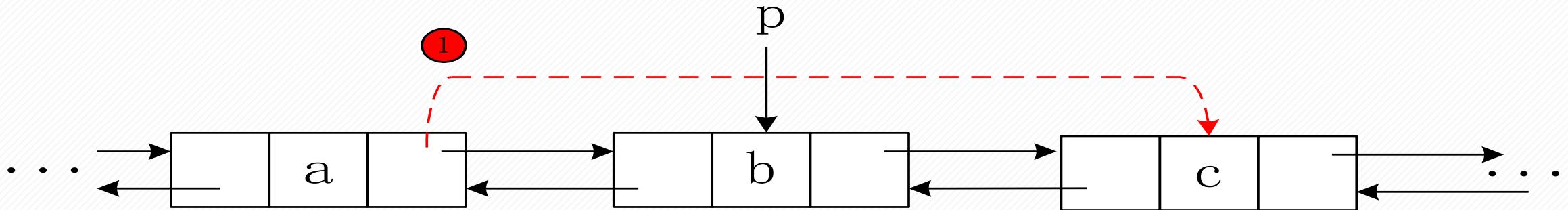
# 双向链表的插入

```
Status ListInsert_DuL(DuLinkList &L,int i,ElemType e){  
    if(!(p=GetElemP_DuL(L,i))) return ERROR;  
    s= new DuListNode;  
    s->data=e;  
    s->prior=p->prior;  
    p->prior->next=s;  
    s->next=p;  
    p->prior=s;  
    return OK;  
}
```

## 双向链表的删除

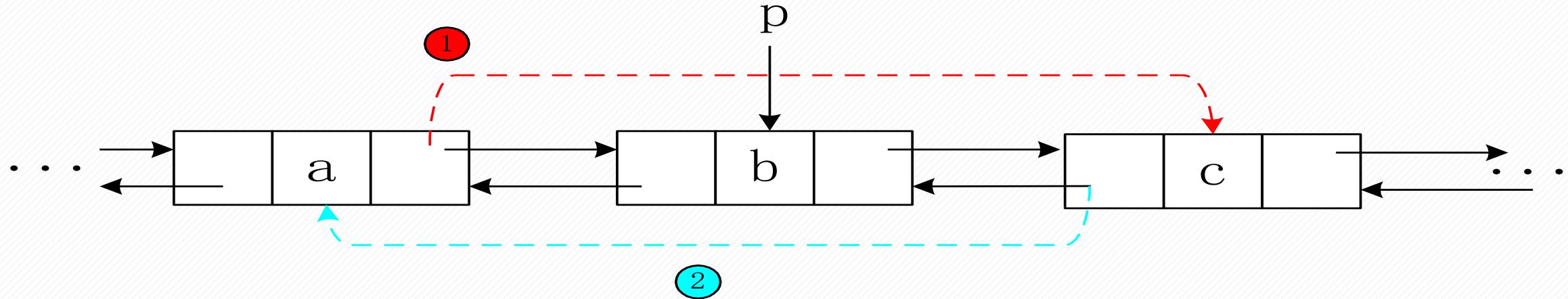


## 双向链表的删除



1.  **$p->prior->next = p->next;$**

## 双向链表的删除



1.  $p->prior->next=p->next;$
2.  $p->next->prior=p->prior;$

## 双向链表的删除

```
Status ListDelete_DuL(DuLinkList &L,int i, ElemType &e){  
    if(!(p=GetElemP_DuL(L,i)))    return ERROR;  
    e= p->data;  
    p->prior->next=p->next;  
    p->next->prior=p->prior;  
    delete p;  
    return OK;  
}
```

存储结构 比较项目		顺序表	链表
空间	存储空间	预先分配，会导致空间闲置或溢出现象	动态分配，不会出现存储空间闲置或溢出现象
	存储密度	不用为表示结点间的逻辑关系而增加额外的存储开销，存储密度等于1	需要借助指针来体现元素间的逻辑关系，存储密度小于1
时间	存取元素	随机存取，按位置访问元素的时间复杂度为 $O(1)$	顺序存取，按位置访问元素时间复杂度为 $O(n)$
	插入、删除	平均移动约表中一半元素，时间复杂度为 $O(n)$	不需移动元素，确定插入、删除位置后，时间复杂度为 $O(1)$
适用情况		① 表长变化不大，且能事先确定变化的范围 ② 很少进行插入或删除操作，经常按元素位置序号访问数据元素	① 长度变化较大 ② 频繁进行插入或删除操作

## > 2.7

# 线性表的应用

1

2

线性表的合并

有序表的合并

问题描述：

假设利用两个线性表La和Lb分别表示两个集合A和B, 现要求一个新的集合

$$A = A \cup B$$

$$La = (7, 5, 3, 11)$$

$$Lb = (2, 6, 3)$$

$$La = (7, 5, 3, 11, 2, 6)$$

## 【算法步骤】

依次取出Lb 中的每个元素，执行以下操作：

在La中查找该元素

如果找不到，则将其插入La的最后

## 【算法描述】

```
void union(List &La, List Lb){  
    m=ListLength(La);  
    n=ListLength(Lb);  
    for(i=1;i<=n;i++){  
        GetElem(Lb,i,e);  
        if(!LocateElem(La,e))  
            ListInsert(La,++m,e);  
    }  
}
```

## 2.7.2

## 有序表的合并

问题描述：

已知线性表L<sub>a</sub> 和L<sub>b</sub>中的数据元素按值非递减有序排列，现要求将L<sub>a</sub>和L<sub>b</sub>归并为一个新的线性表L<sub>c</sub>, 且L<sub>c</sub>中的数据元素仍按值非递减有序排列。

$$L_a = (1, 7, 8)$$

$$L_b = (2, 4, 6, 8, 10, 11)$$

$$L_c = (1, 2, 4, 6, 7, 8, 8, 10, 11)$$

## 【算法步骤】 - 有序的顺序表合并

- (1) 创建一个空表Lc
- (2) 依次从 La 或 Lb 中“摘取”元素值较小的结点插入到 Lc 表的最后，直至其中一个表变空为止
- (3) 继续将 La 或 Lb 其中一个表的剩余结点插入在 Lc 表的最后

# 【算法描述】 - 有序的顺序表合并

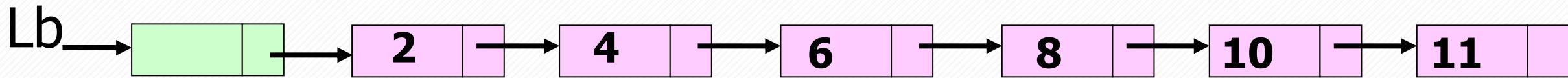
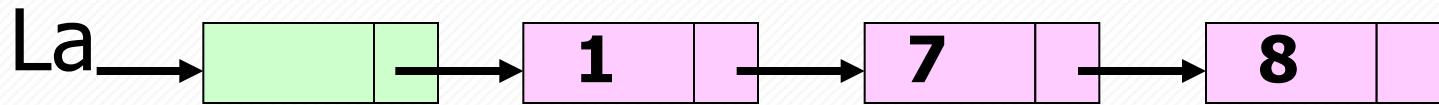
```
void MergeList_Sq(SqList LA,SqList LB,SqList &LC){  
    pa=LA.elem; pb=LB.elem;      //指针pa和pb的初值分别指向两个表的第一个元素  
    LC.length=LA.length+LB.length; //新表长度为待合并两表的长度之和  
    LC.elem=new ElemtType[LC.length]; //为合并后的新表分配一个数组空间  
    pc=LC.elem;                  //指针pc指向新表的第一个元素  
    pa_last=LA.elem+LA.length-1;  //指针pa_last指向LA表的最后一个元素  
    pb_last=LB.elem+LB.length-1;  //指针pb_last指向LB表的最后一个元素  
    while(pa<=pa_last && pb<=pb_last){ //两个表都非空  
        if(*pa<=*pb) *pc++=*pa++;           //依次“摘取”两表中值较小的结点  
        *pc++=*pb++; }  
        while(pa<=pa_last) *pc++=*pa++; //LB表已到达表尾  
        while(pb<=pb_last) *pc++=*pb++; //LA表已到达表尾  
    }//MergeList_Sq
```

实战工程案例1

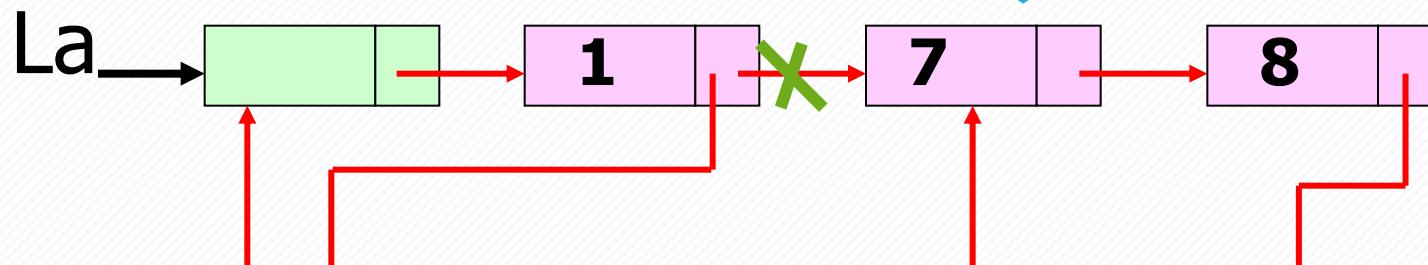
## 有序链表合并 - - **重点掌握**

- ✓ 将这两个有序链表合并成一个有序的单链表。
- ✓ 要求结果链表仍使用原来两个链表的存储空间，不另外占用其它的存储空间。
- ✓ 表中允许有重复的数据。

## 有序链表合并 - - 重点掌握



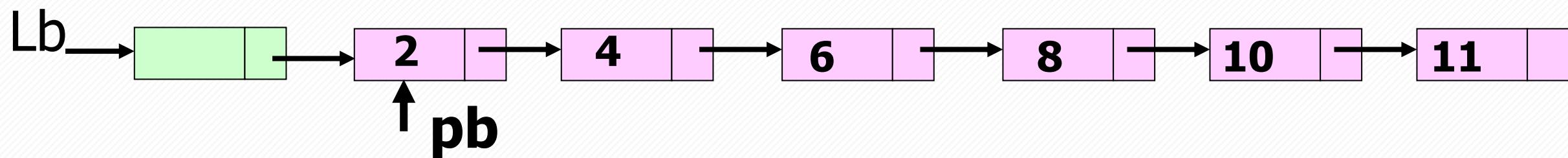
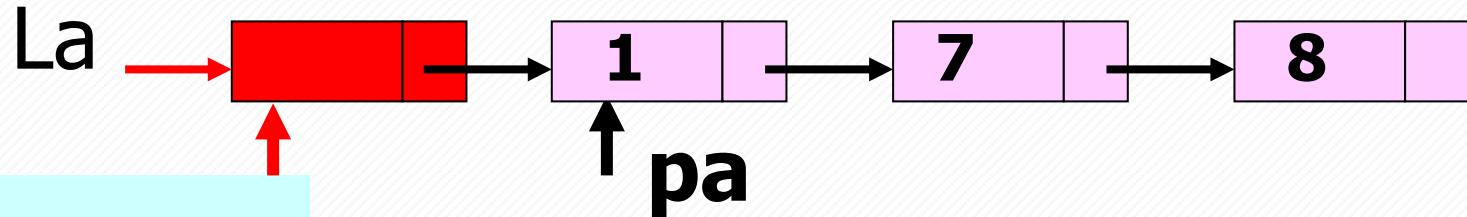
合并后



## 【算法步骤】 - 有序的链表合并

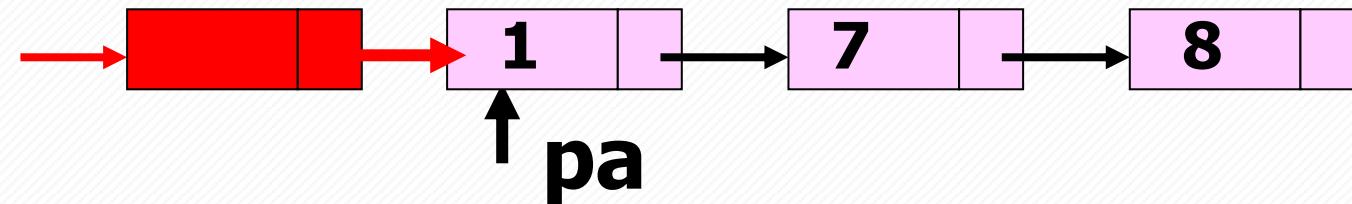
- (1) Lc指向La
- (2) 依次从 La 或 Lb 中“摘取”元素值较小的结点插入到 Lc 表的最后，直至其中一个表变空为止
- (3) 继续将 La 或 Lb 其中一个表的剩余结点插入在 Lc 表的最后
- (4) 释放 Lb 表的表头结点

## 有序链表合并 (初始化)

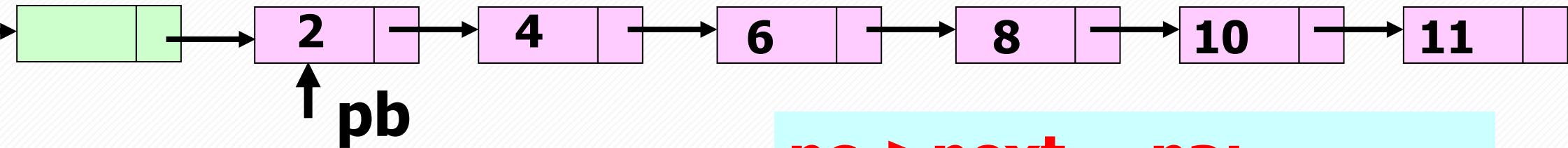


## 有序链表合并( $pa->data \leq pb->data$ )

La(Lc ,pc)

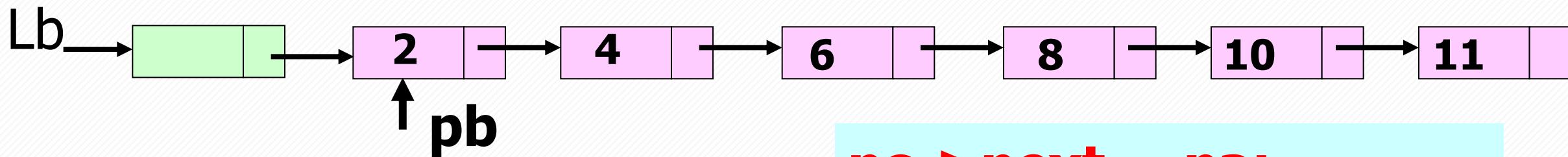
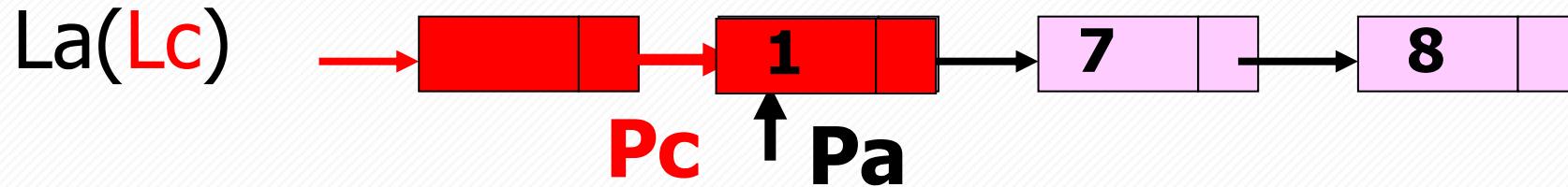


Lb



**pc->next = pa;**

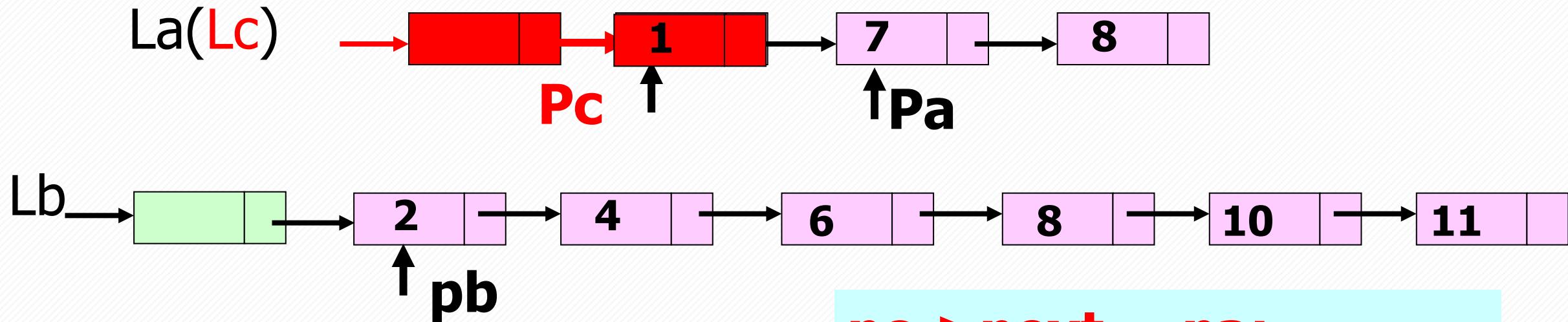
## 有序链表合并( $pa->data \leq pb->data$ )



```
pc->next = pa;  
pc= pa;
```

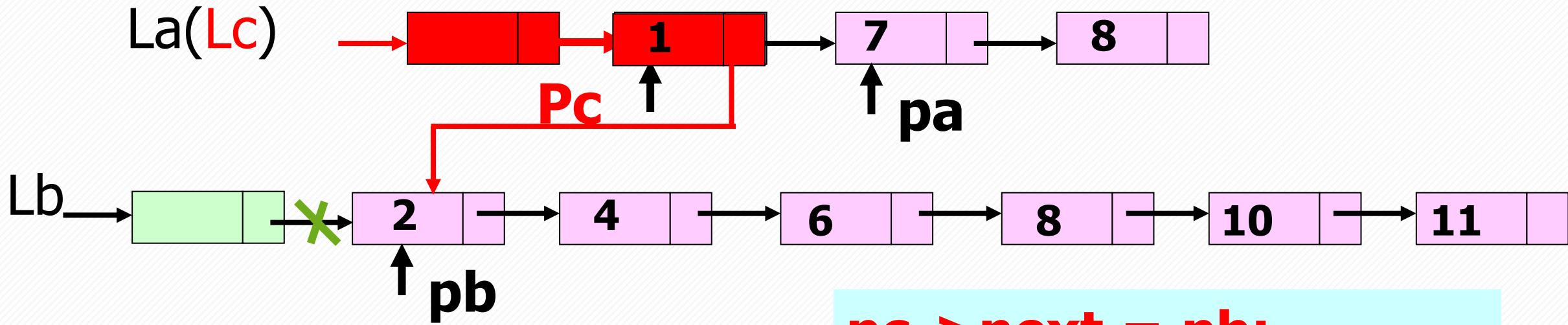


**有序链表合并(  $pa->data <= pb->data$  )**

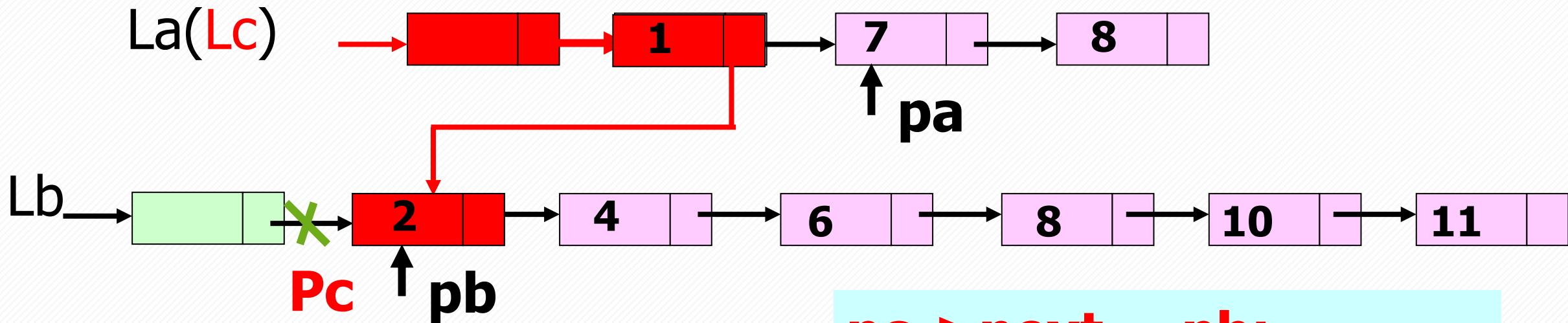


```
pc->next = pa;  
pc= pa;  
pa = pa->next;
```

## 有序链表合并( $pa->data > pb->data$ )

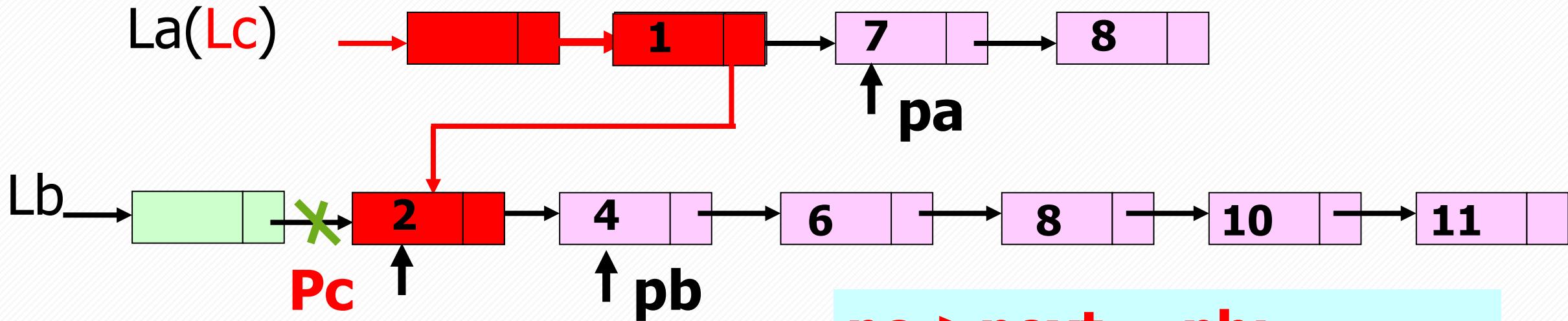


## 有序链表合并( $pa->data > pb->data$ )



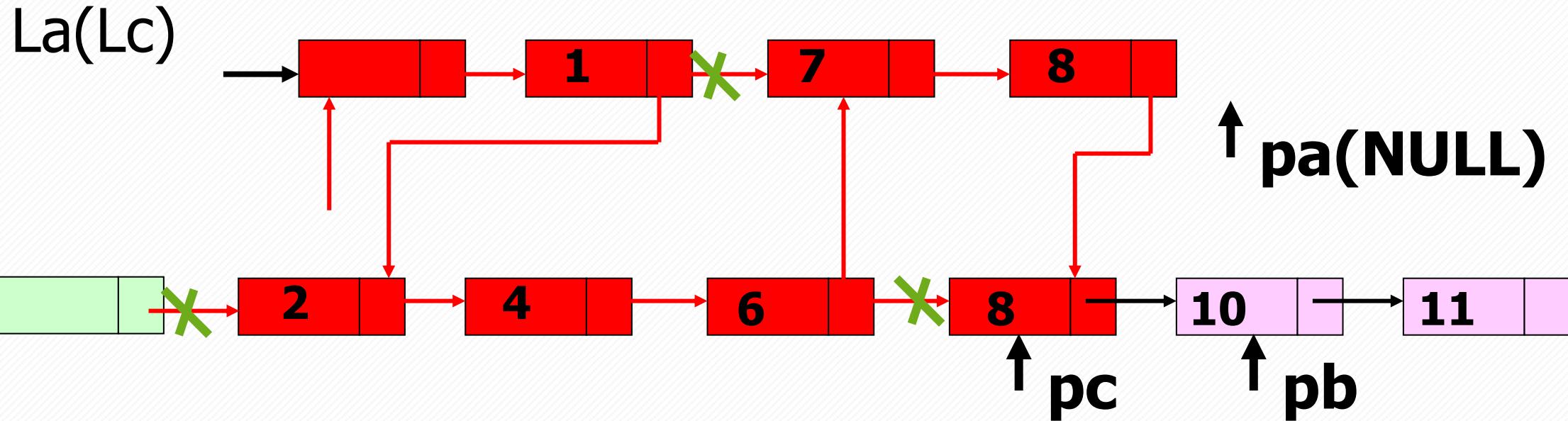
$pc->next = pb;$   
 $pc= pb;$

## 有序链表合并( $pa->data > pb->data$ )



```
pc->next = pb;  
pc= pb;  
pb =pb->next;
```

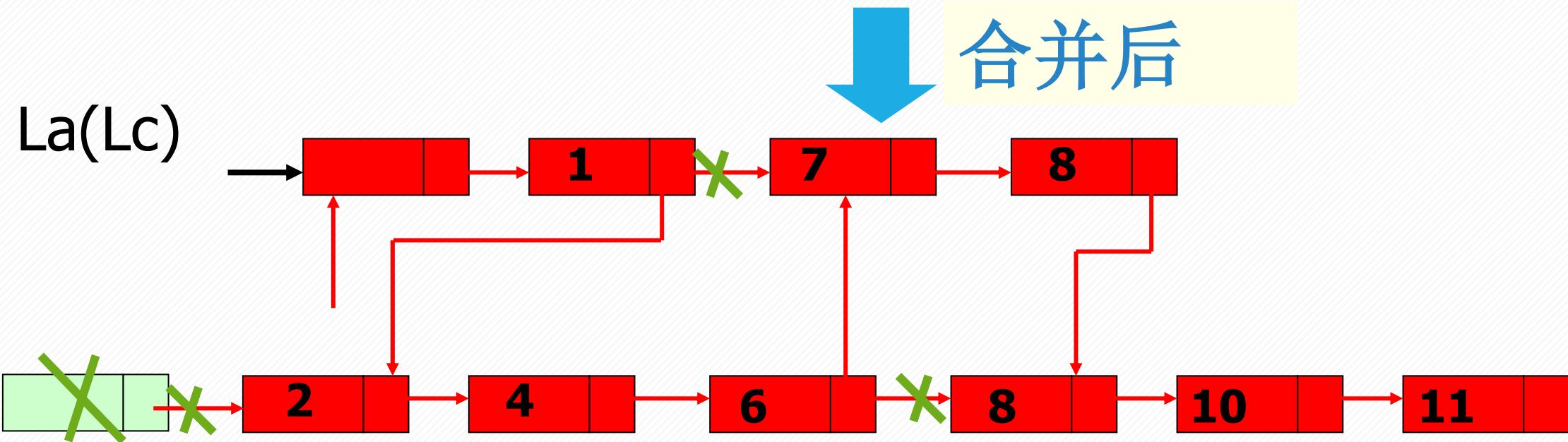
## 有序链表合并



pc-> **next=pa ? pa : pb;**



# 有序链表合并



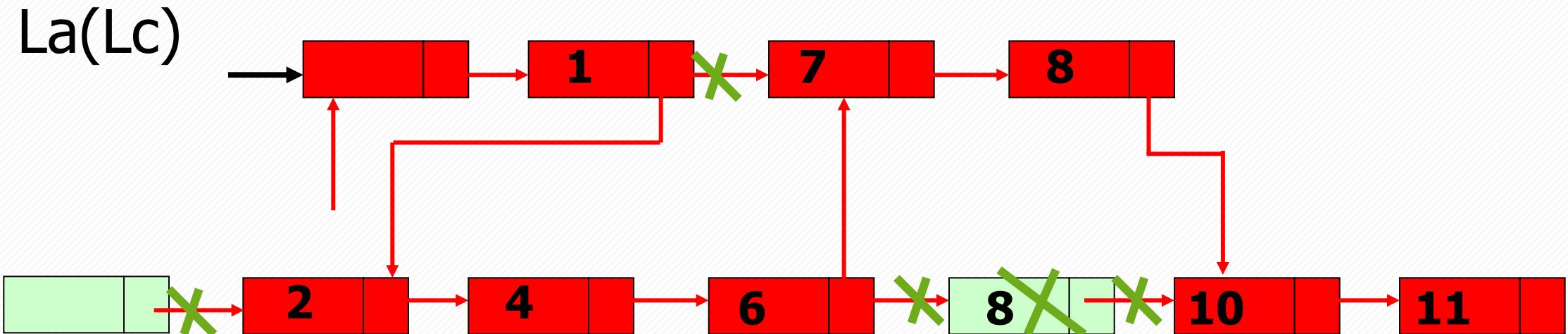
# delete Lb;

## 【算法描述】 - 有序的链表合并

```
void MergeList_L(LinkList &La,LinkList &Lb,LinkList &Lc){  
    pa=La->next; pb=Lb->next;  
    pc=La=La;           //用La的头结点作为Lc的头结点  
    while(pa && pb){  
        if(pa->data<=pb->data){ pc->next=pa;pc=pa;pa=pa->next;}  
        else{pc->next=pb; pc=pb; pb=pb->next;}  
        pc->next=pa?pa:pb; //插入剩余段  
        delete Lb;          //释放Lb的头结点  
    }  
}
```

## 实战工程案例2

思考1：要求合并后的表无重复数据，如何实现？



提示：要单独考虑

**pa->data == pb->data**

## 思考2：将两个非递减的有序链表合并为一个非递增的有序链表，如何实现？

- ✓ 要求结果链表仍使用原来两个链表的存储空间，不另外占用其它的存储空间。
- ✓ 表中允许有重复的数据。

## 【算法步骤】

- (1) Lc指向La
- (2) 依次从 La 或 Lb 中“摘取”元素值较小的结点插入到 Lc 表的表头结点之后，直至其中一个表变空为止
- (3) 继续将 La 或 Lb 其中一个表的剩余结点插入在 Lc 表的表头结点之后
- (4) 释放 Lb 表的表头结点



CimFAX  
传真服务器常用

# 问答

# 将来的你



一定会感激现在拼命的自己



$$\begin{array}{c} b \\ \triangle \\ a \quad c \\ c^2 = a^2 + b^2 \end{array}$$