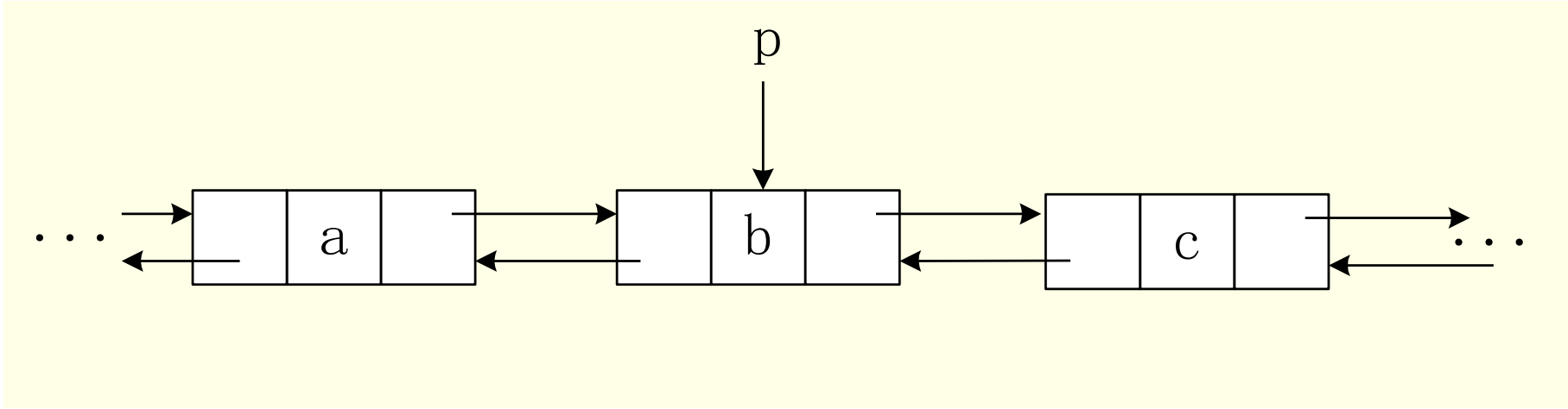
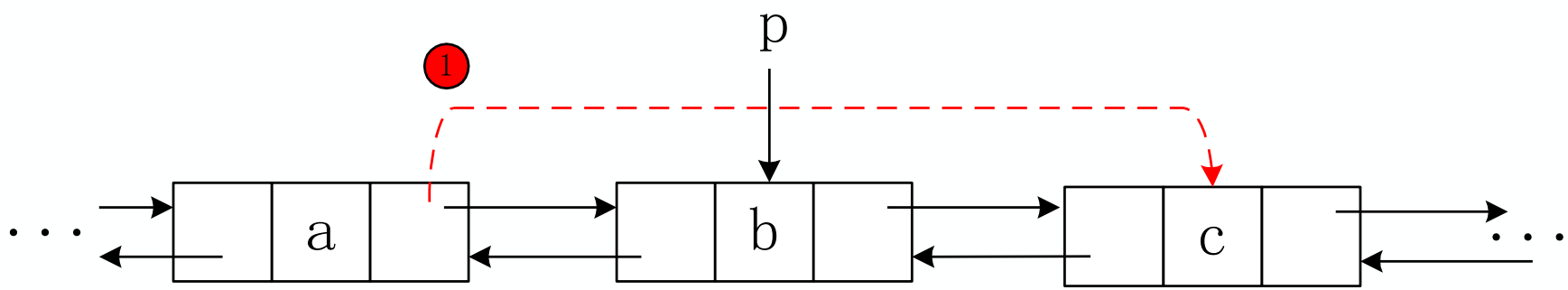
双向链表的删除



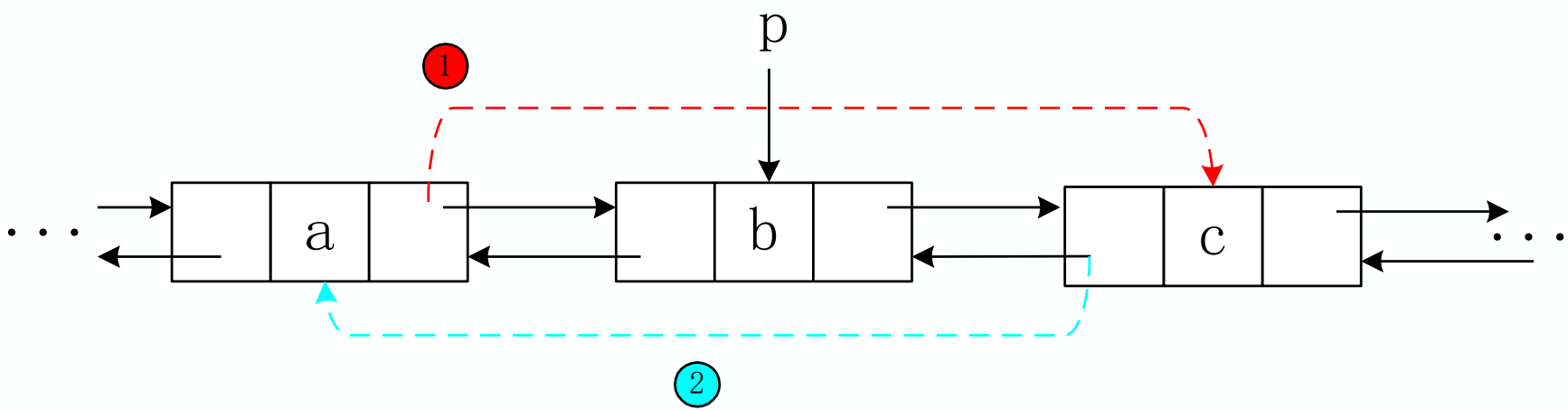
15:54

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;

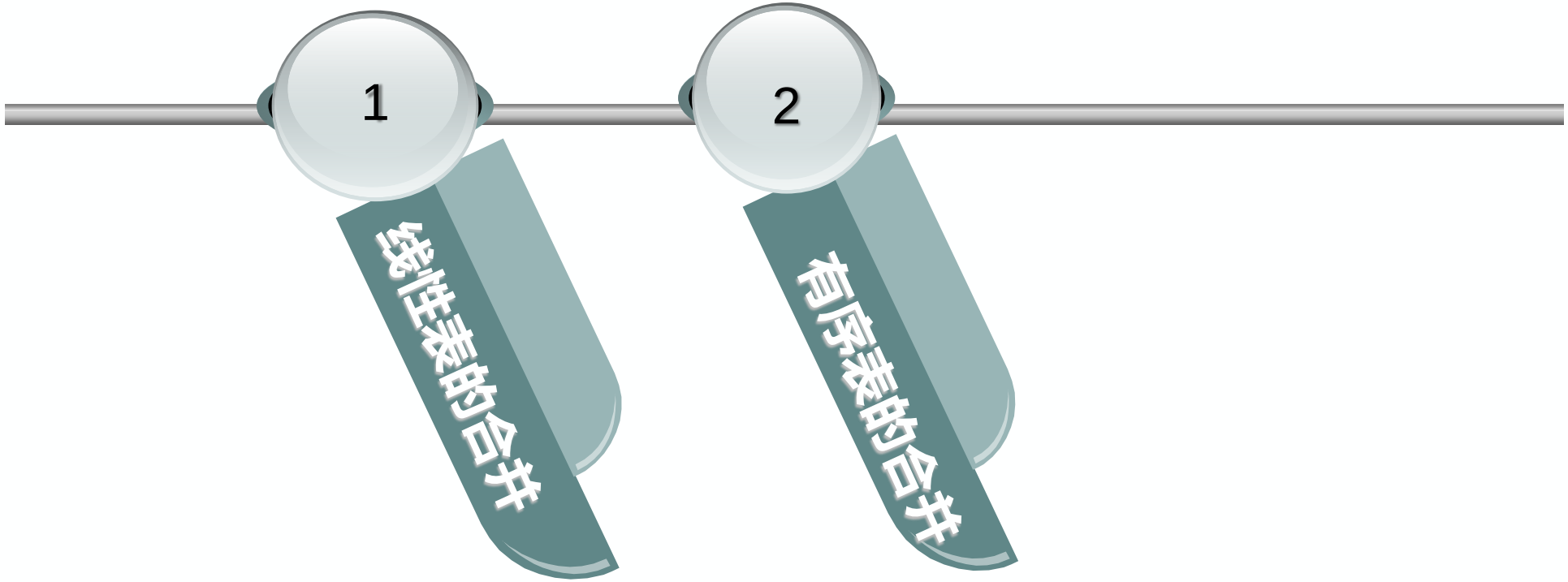
双向链表的删除

```
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;
}
```

2.6 顺序表和链表的比较

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



15:54

2.7.1 线性表的合并

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){
  La_len=ListLength(La);
  Lb_len=ListLength(Lb);
  for(i=1;i<=Lb_len;i++){
     GetElem(Lb,i,e);
     if(!LocateElem(La,e))
        ListInsert(&La,++La_len,e);
  }
}
```

$O(ListLength(LA) \times ListLength(LB))$

2.7.2 有序表的合并

问题描述：

已知线性表La 和Lb中的数据元素按值非递减有序排列,现要求将La和Lb归并为一个新的线性表Lc,且Lc中的数据元素仍按值非递减有序排列.

La=(1 ,7, 8)

Lb=(2, 4, 6, 8, 10, 11)

Lc=(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 ElemType[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
     else *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
```

T(n)= $O(ListLength(LA)+ListLength(LB))$

S(n)= O(n)

15:54

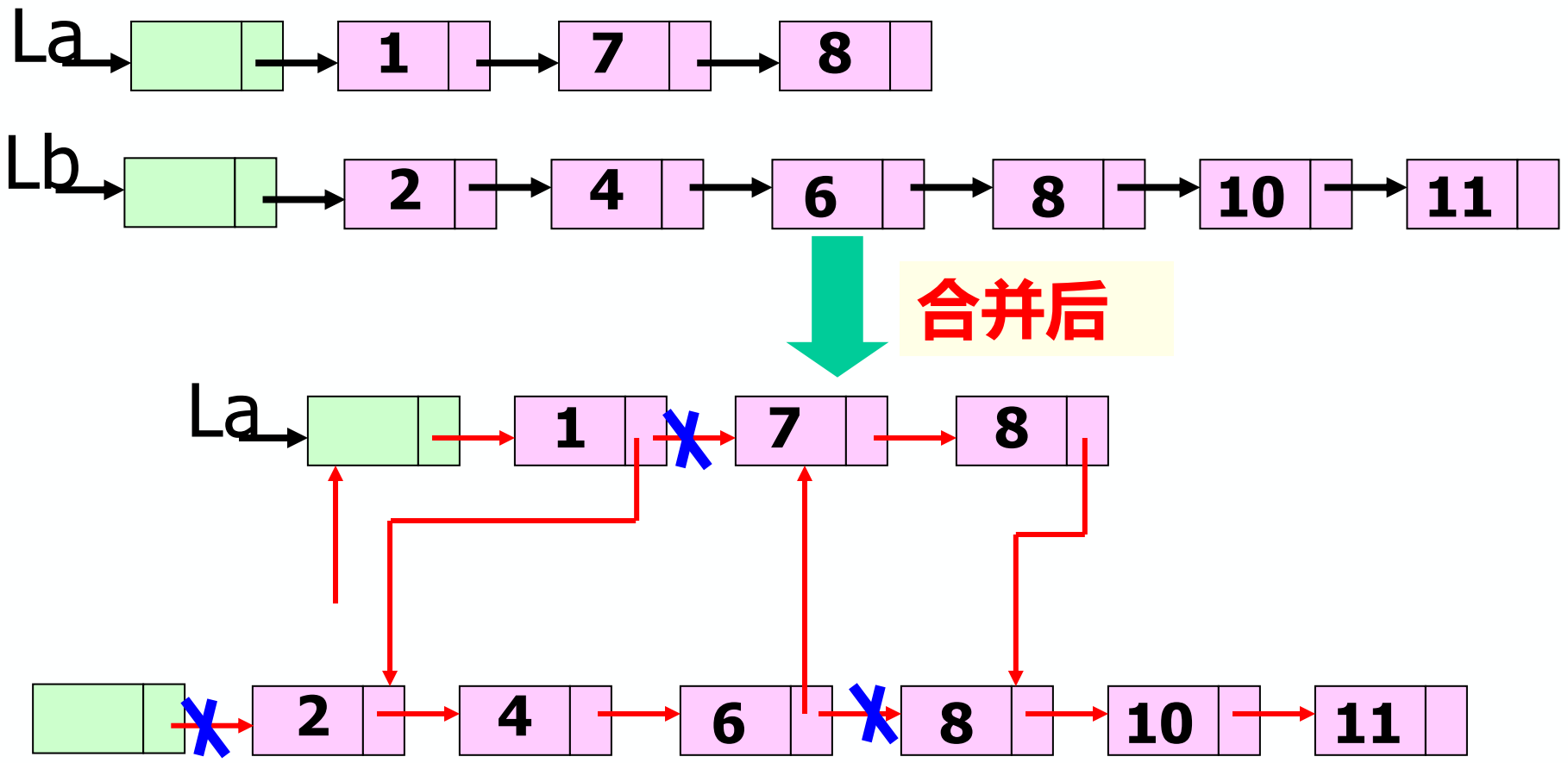
有序链表合并 - - 重点掌握

✓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



【算法步骤】 - 有序的链表合并

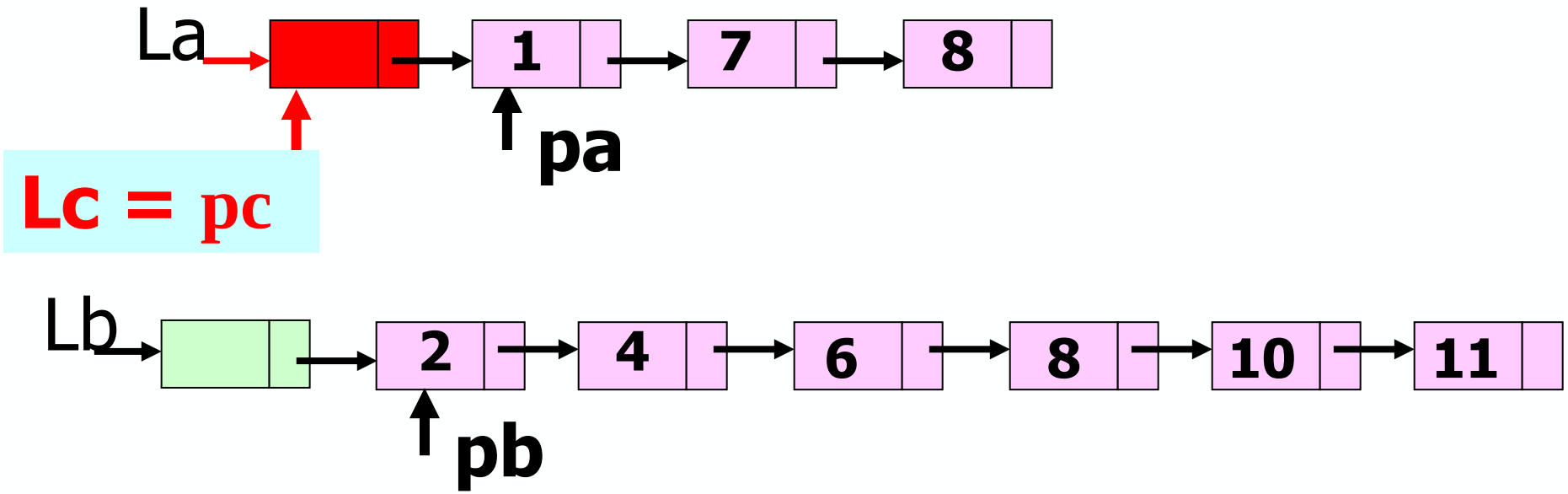
(1)Lc指向La

(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

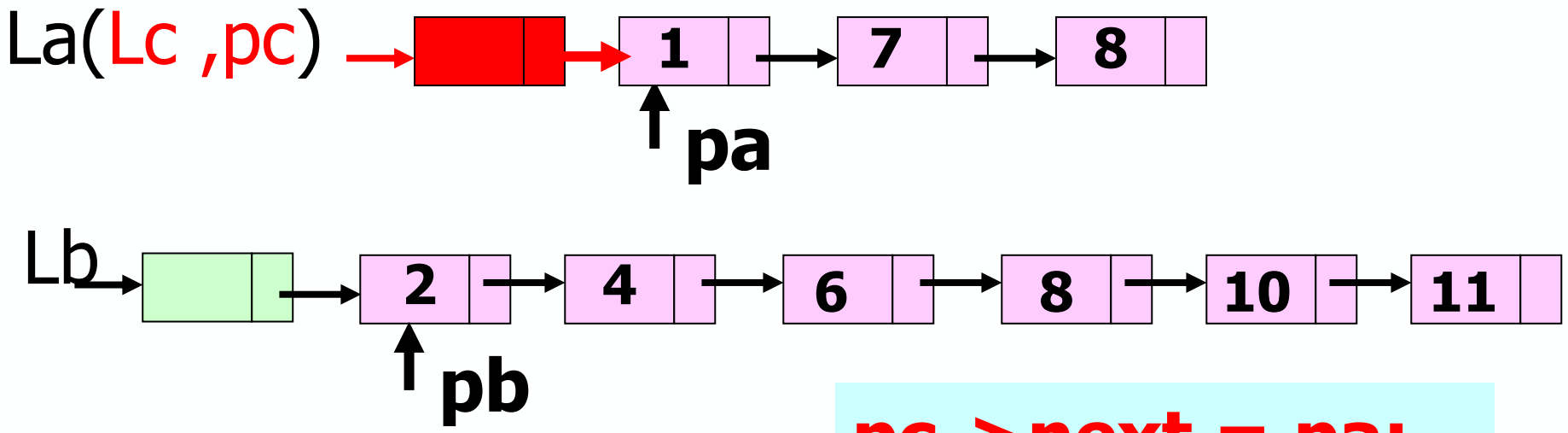
(3) 继续将 La 或 Lb 其中一个表的剩余结点插入在 Lc 表的最后

(4) 释放 Lb 表的表头结点

有序链表合并（初始化）

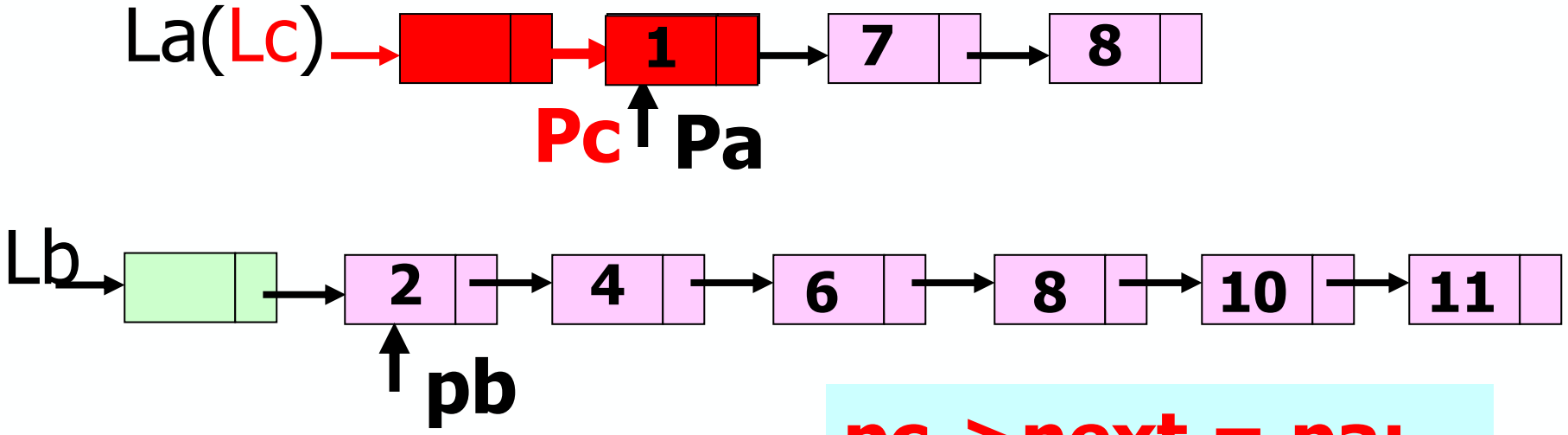


有序链表合并(pa->data < = pb->data)



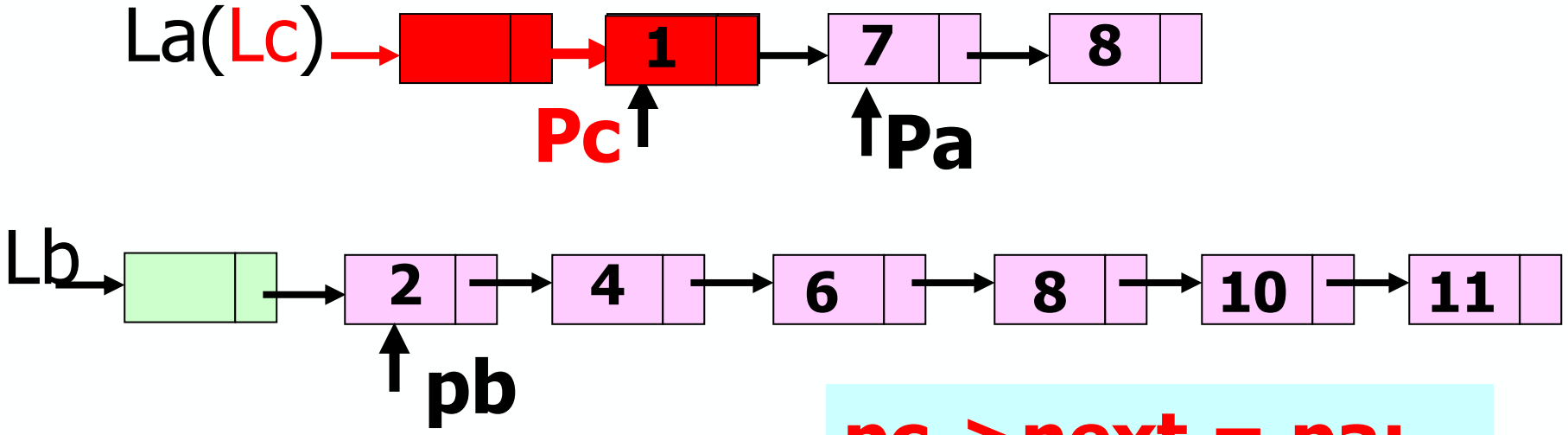
pc->next = pa;

有序链表合并(pa->data < = pb->data)



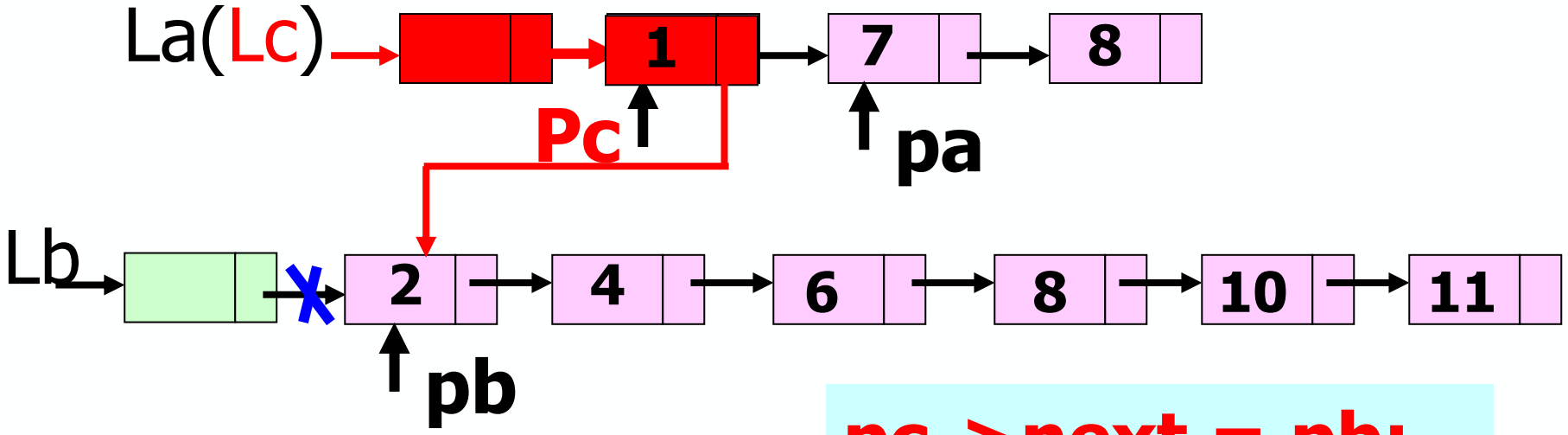

```
pc->next = pa;
pc= pa;
```

有序链表合并(pa->data < = pb->data)



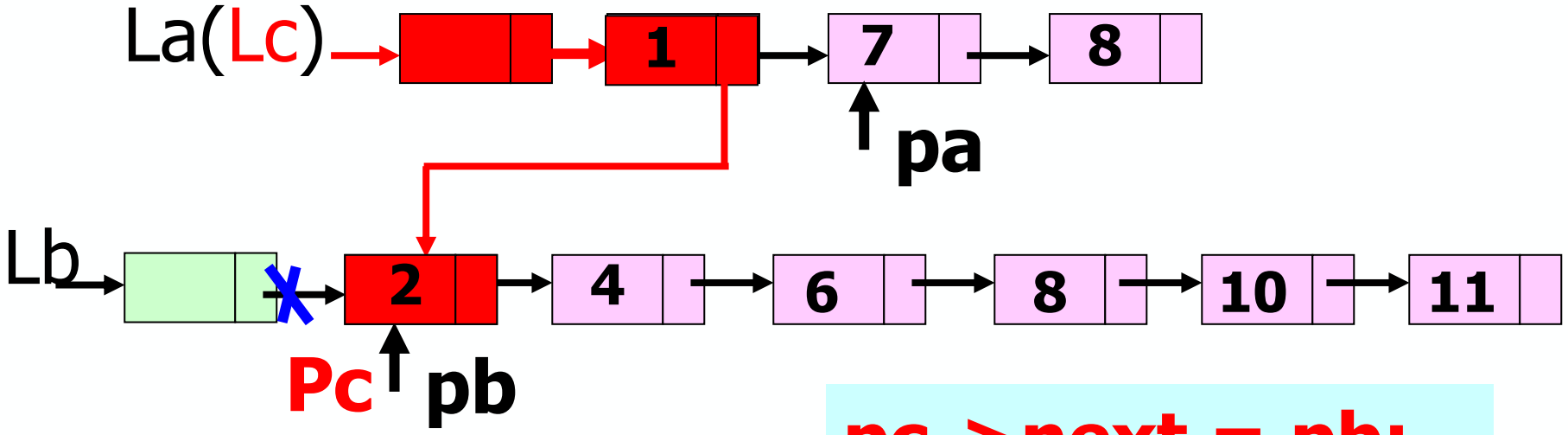

```
pc->next = pa;
pc= pa;
pa = pa->next;
```

有序链表合并(pa->data >pb->data)



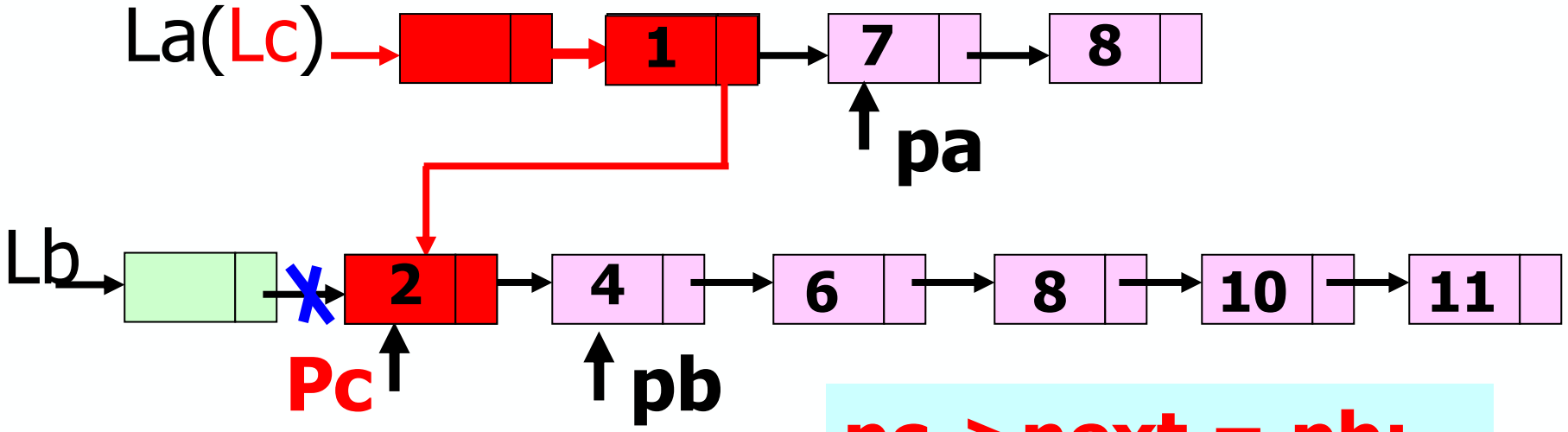
pc->next = pb;

有序链表合并(pa->data >pb->data)



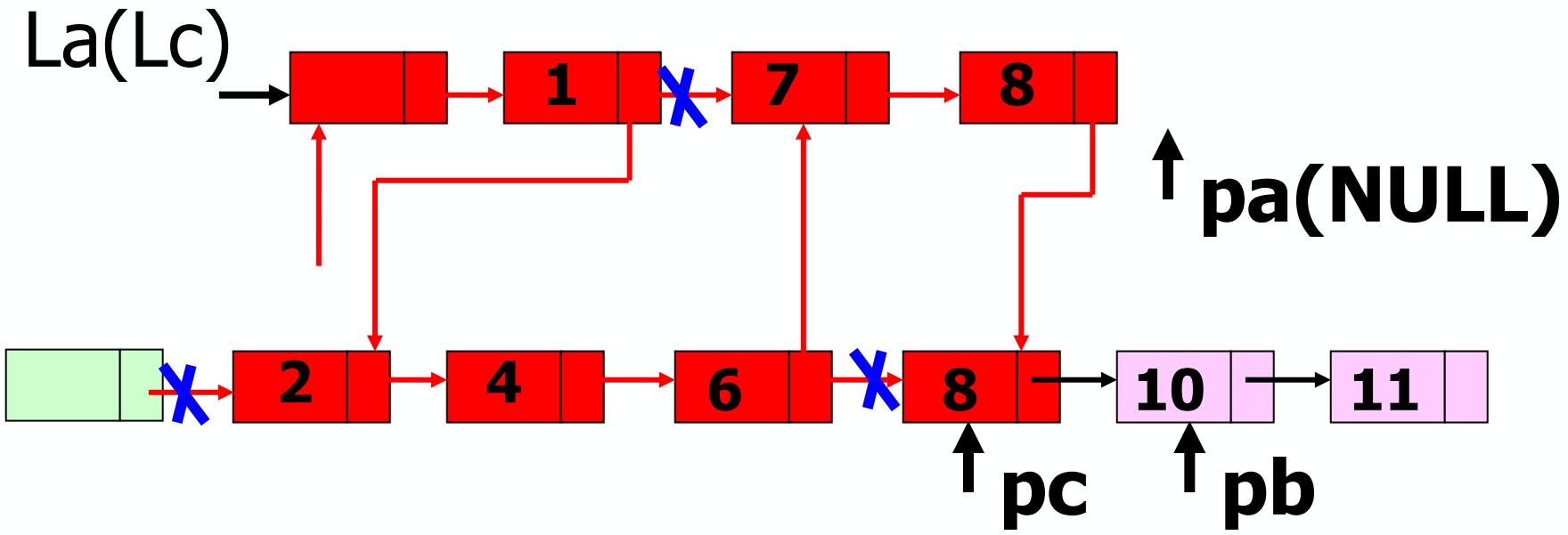

```
pc->next = pb;
pc= pb;
```

有序链表合并(pa->data >pb->data)



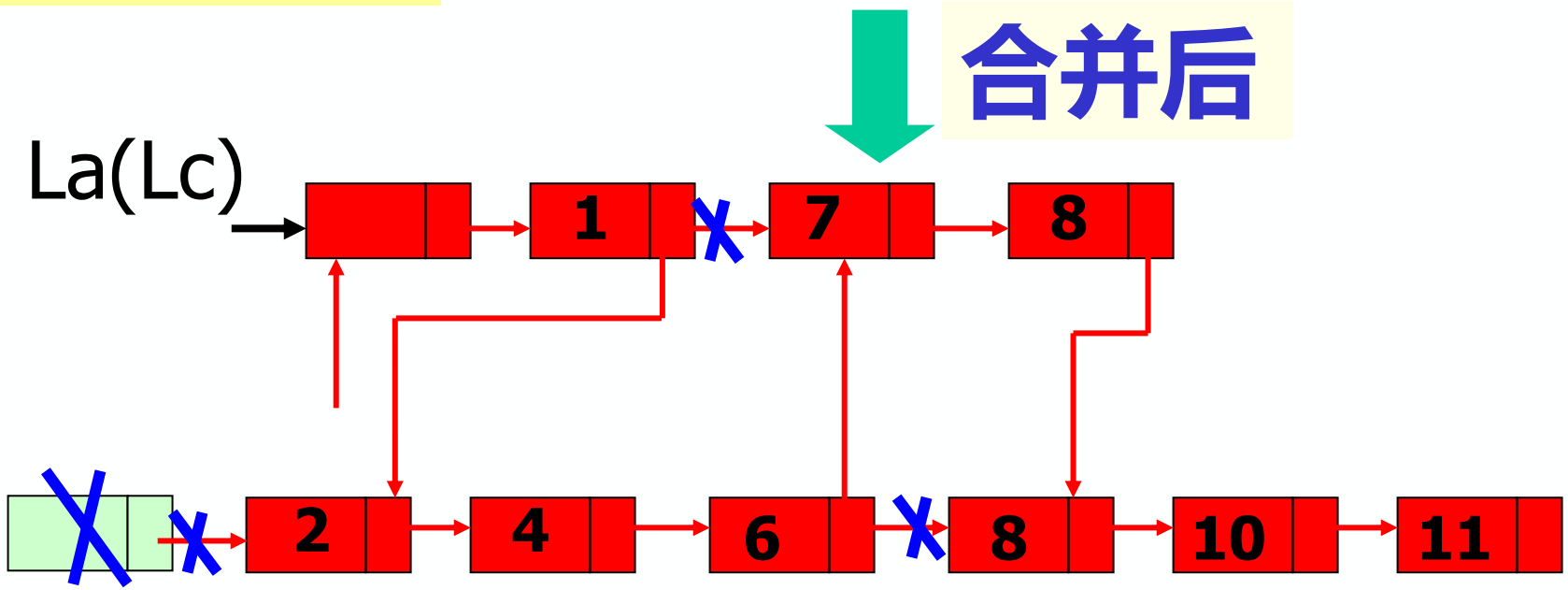

```
pc->next = pb;
pc= pb;
pb =pb->next;
```

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

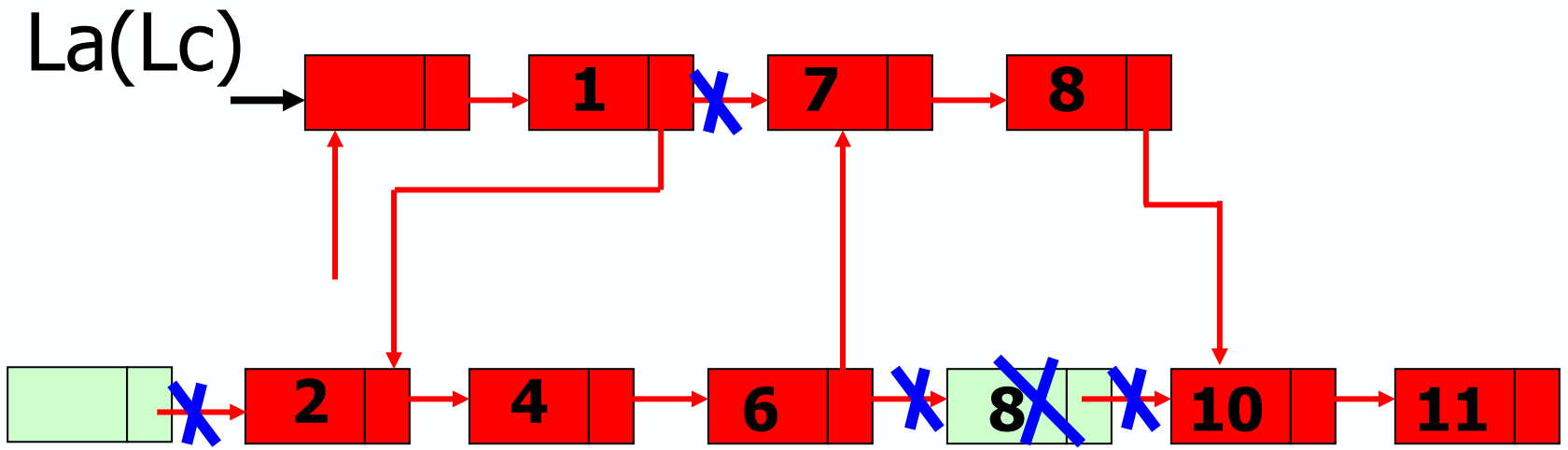
```
void MergeList_L(LinkList &La,LinkList &Lb,LinkList &Lc){
  pa=La->next;  pb=Lb->next;
  pc=Lc=La;            //用La的头结点作为Lc的头结点
  while(pa && pb){
    if(pa->data<=pb->data){ pc->next=pa;pc=pa;pa=pa->next;}
    else{pc->next=pb; pc=pb; pb=pb->next;}
  pc->next=pa?pa:pb;    //插入剩余段
  delete Lb;  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

T(n)= $O(ListLength(LA)+ListLength(LB))$

S(n)= O(1)

15:54

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。