数据结构

（C语言版）（第2版）

线性表

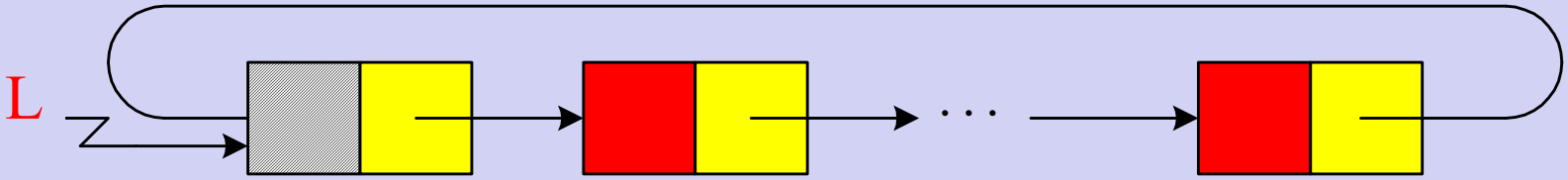
线性表的循环链表表示与实现

主讲教师：汪红松

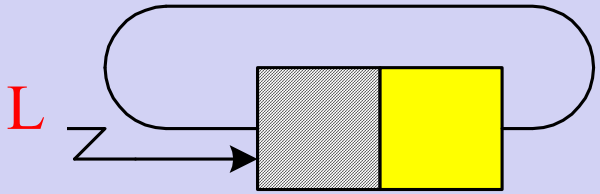
教 学 内 容 Contents

一、循环链表

循环链表——首尾相接的链表。



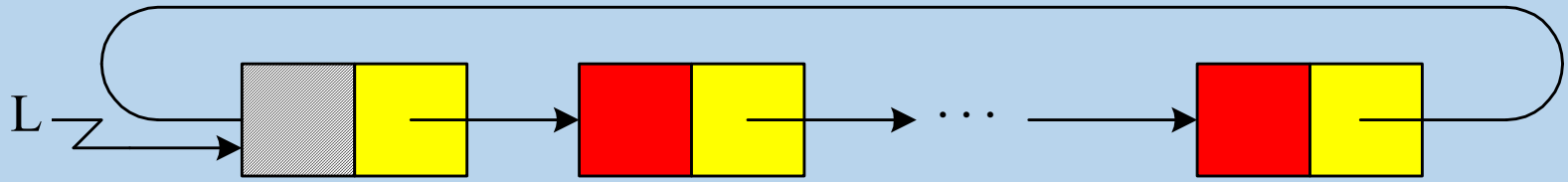
(a) 非空单循环链表



(b) 空表

L->next=L

一、循环链表



说明

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；

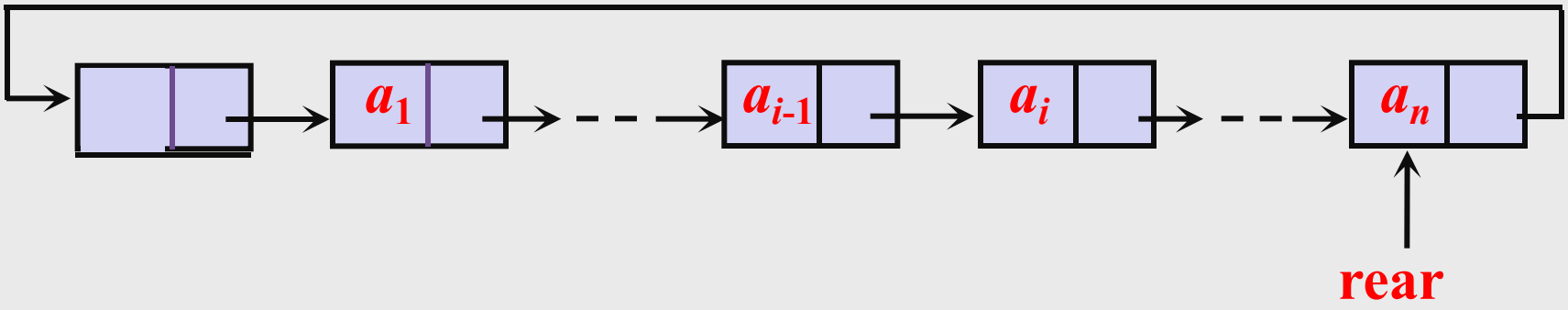
循环链表中没有明显的尾端

如何避免死循环

循环条件：p != L　　或　　p->next != L

循环链表

说明　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？

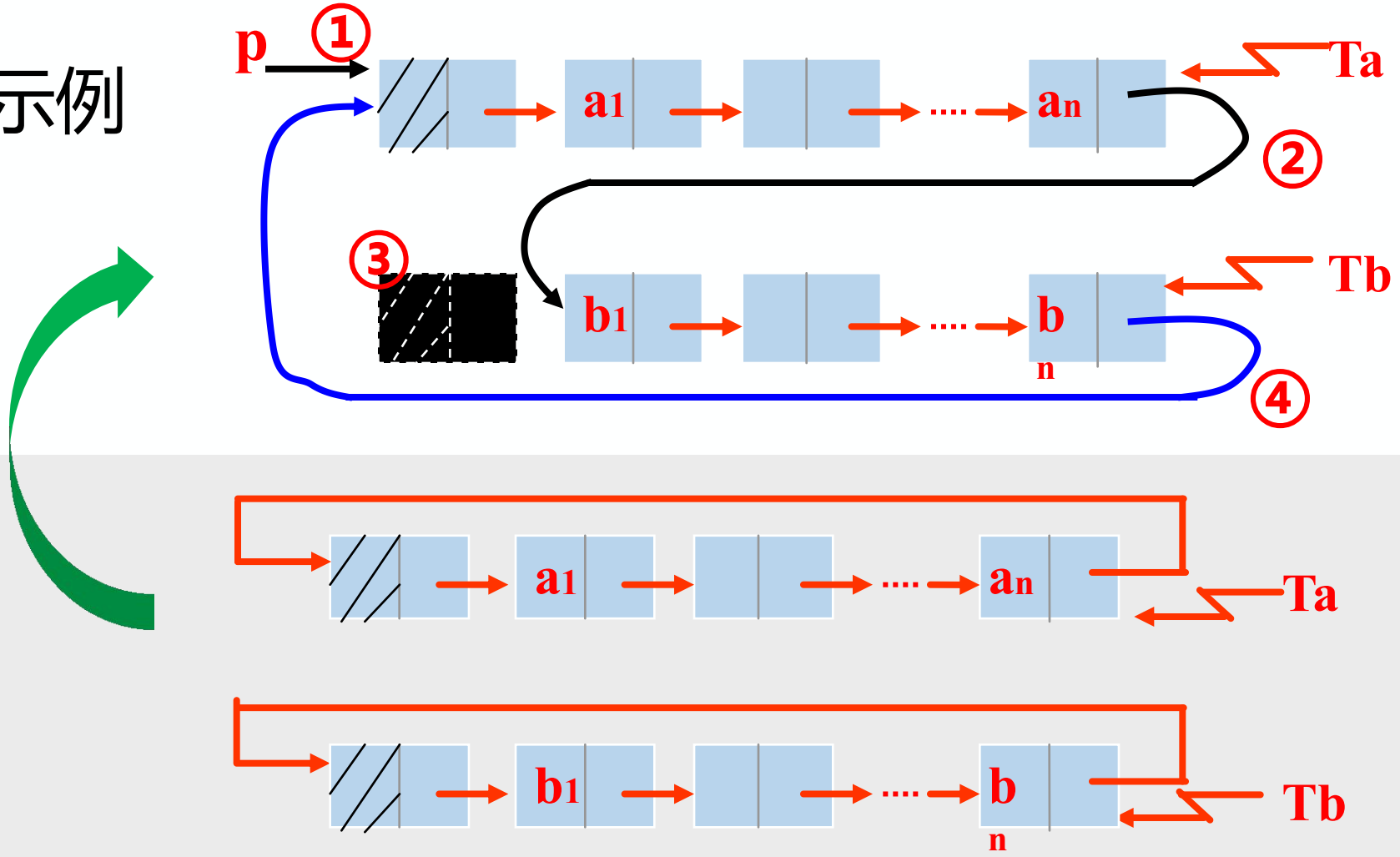
开始结点：rear->next->next

终端结点：rear

一、循环链表

循环链表的合并

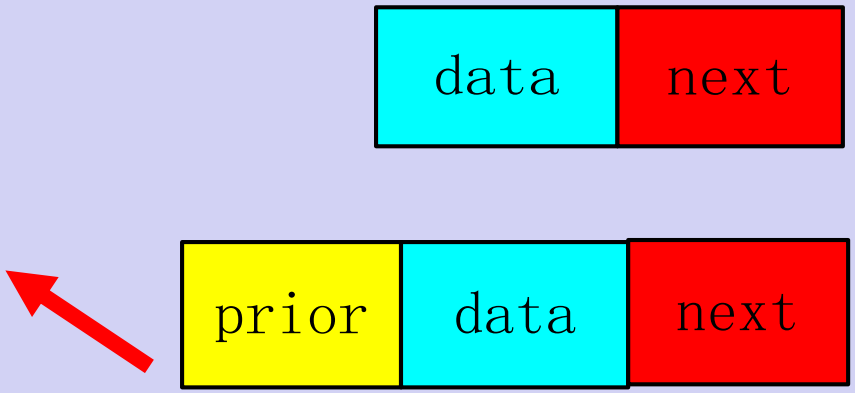
示例



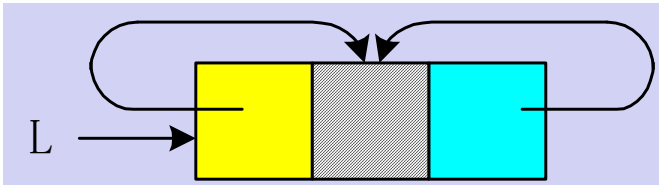
二、双向链表

双链表——有两个指针域的链表。

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

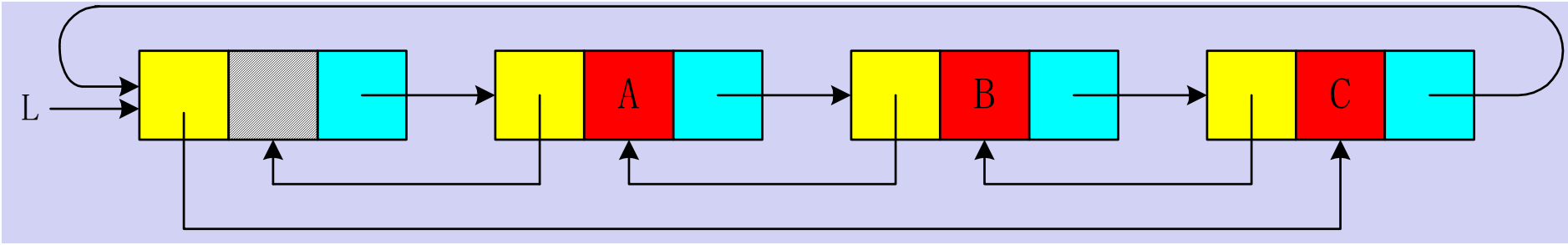


二、双向链表



L->next=L

(a) 空双向循环链表

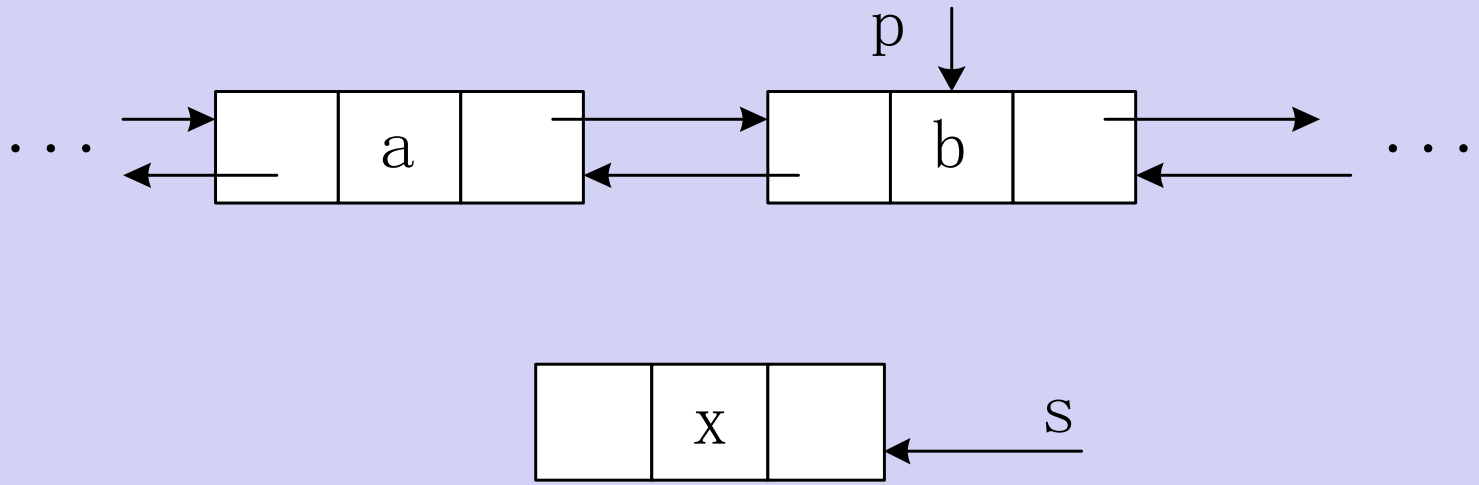


(b) 双向循环链表

d->next->prior = d->prior->next = d

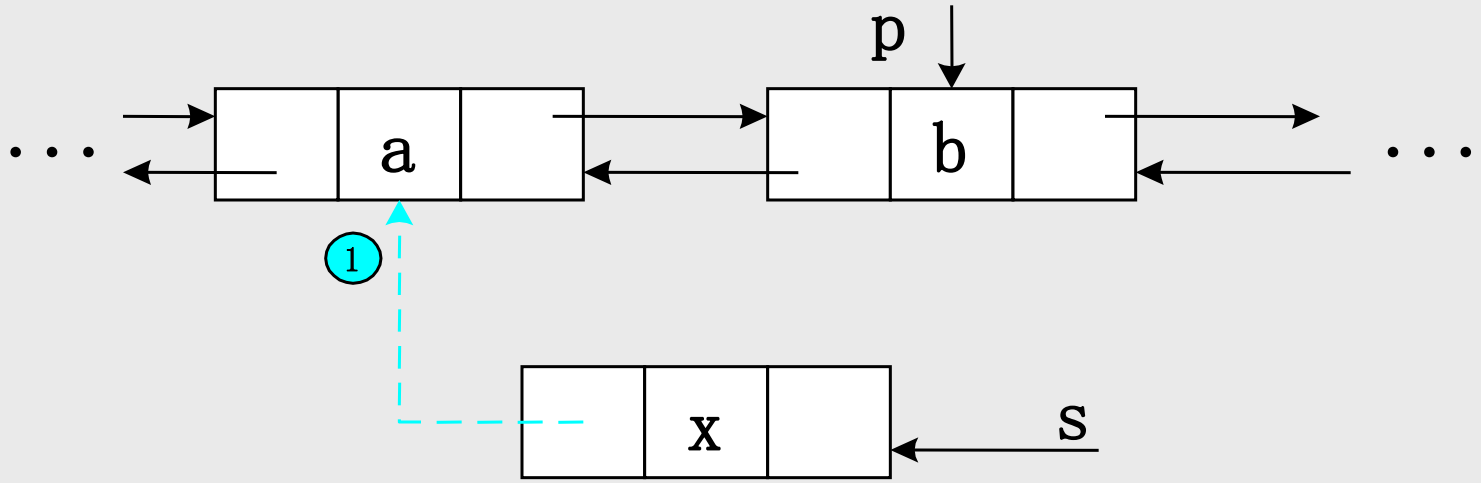
二、双向链表

1.双向链表的插入



二、双向链表

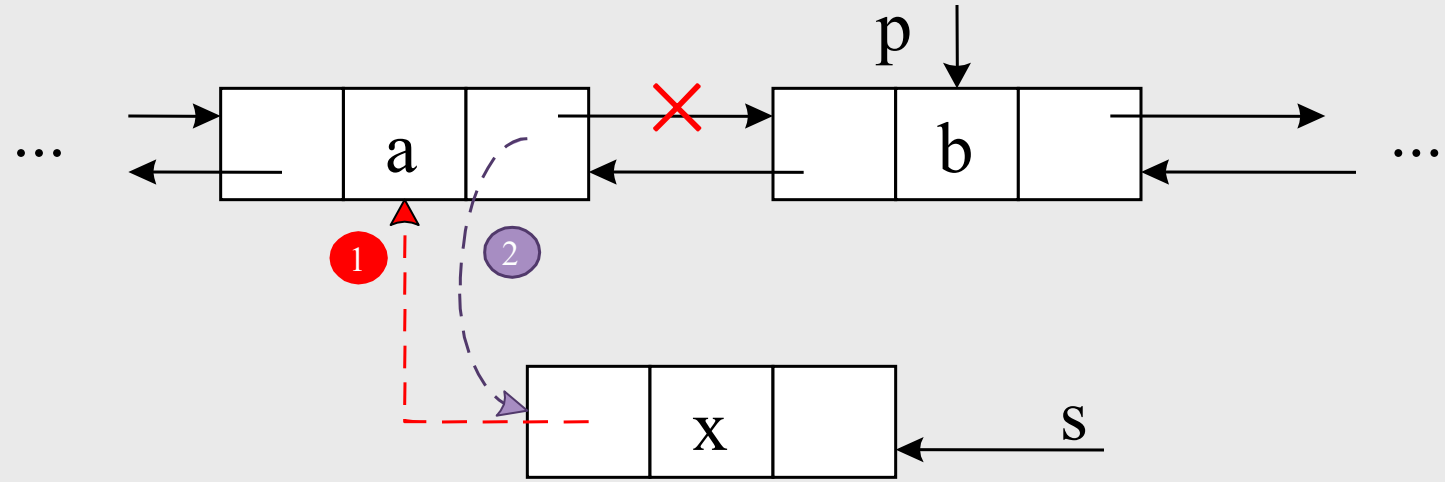
1.双向链表的插入



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

二、双向链表

1.双向链表的插入

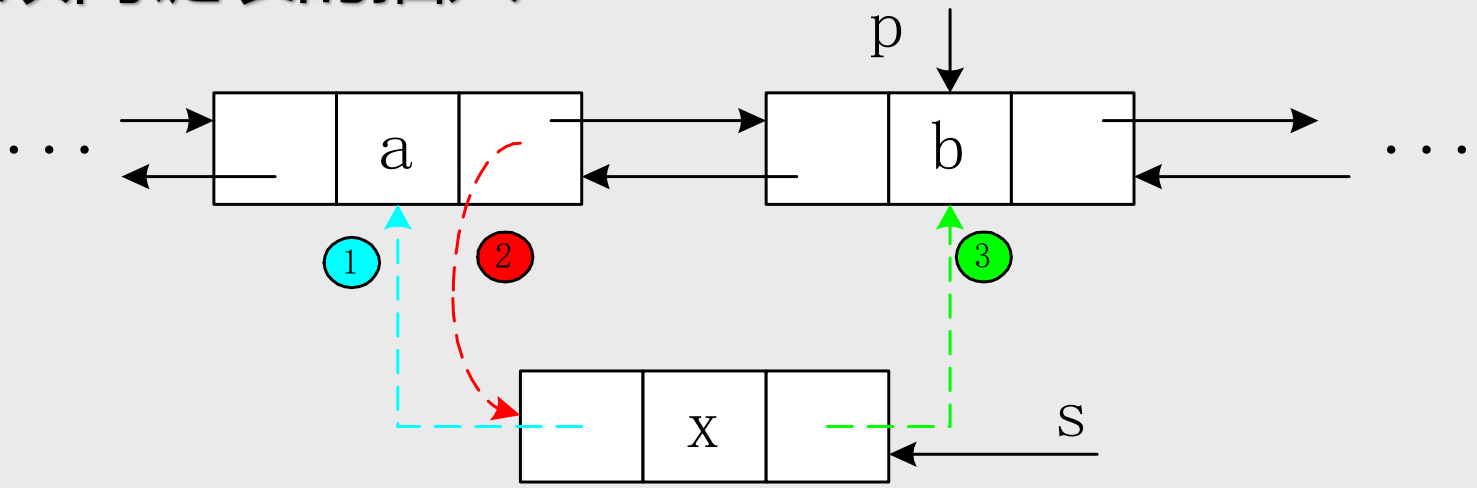


1. s->prior=p->prior;

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

二、双向链表

1.双向链表的插入



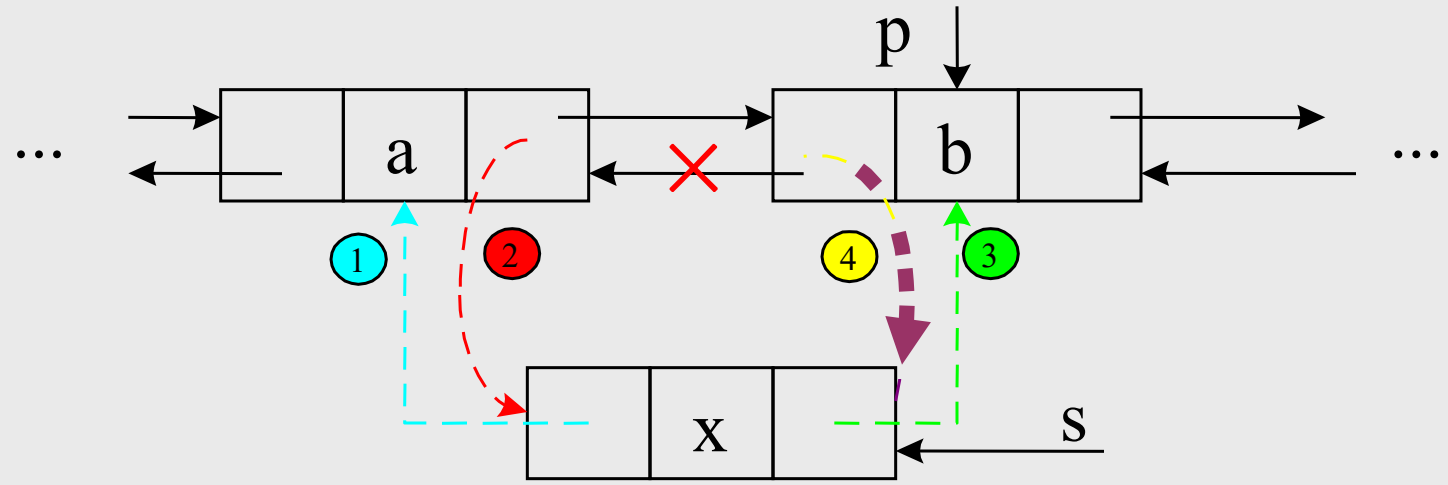
1. s->prior=p->prior;

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

3. s->next=p;

二、双向链表

1.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;

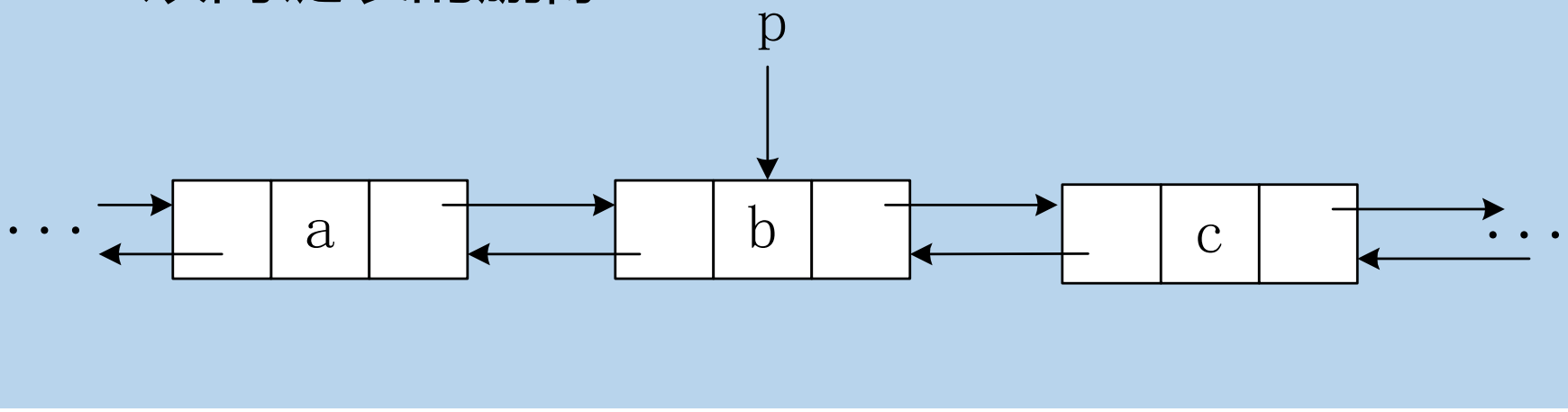
二、双向链表

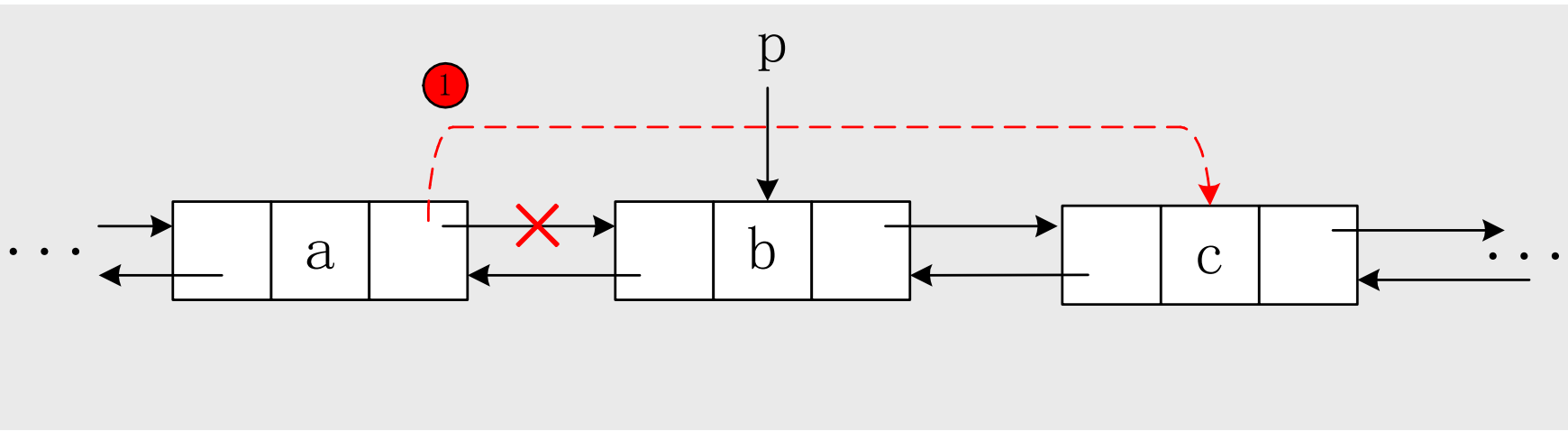
1.双向链表的插入

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

二、双向链表

2.双向链表的删除

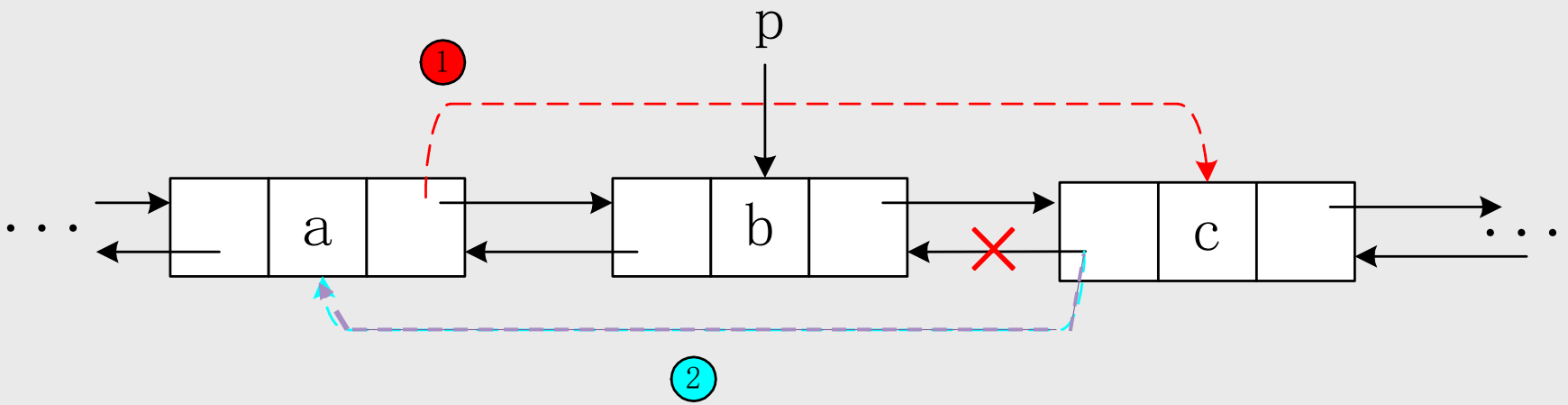




1. p->prior->next=p->next;

二、双向链表

2.双向链表的删除



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

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

二、双向链表

2.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;
}
```