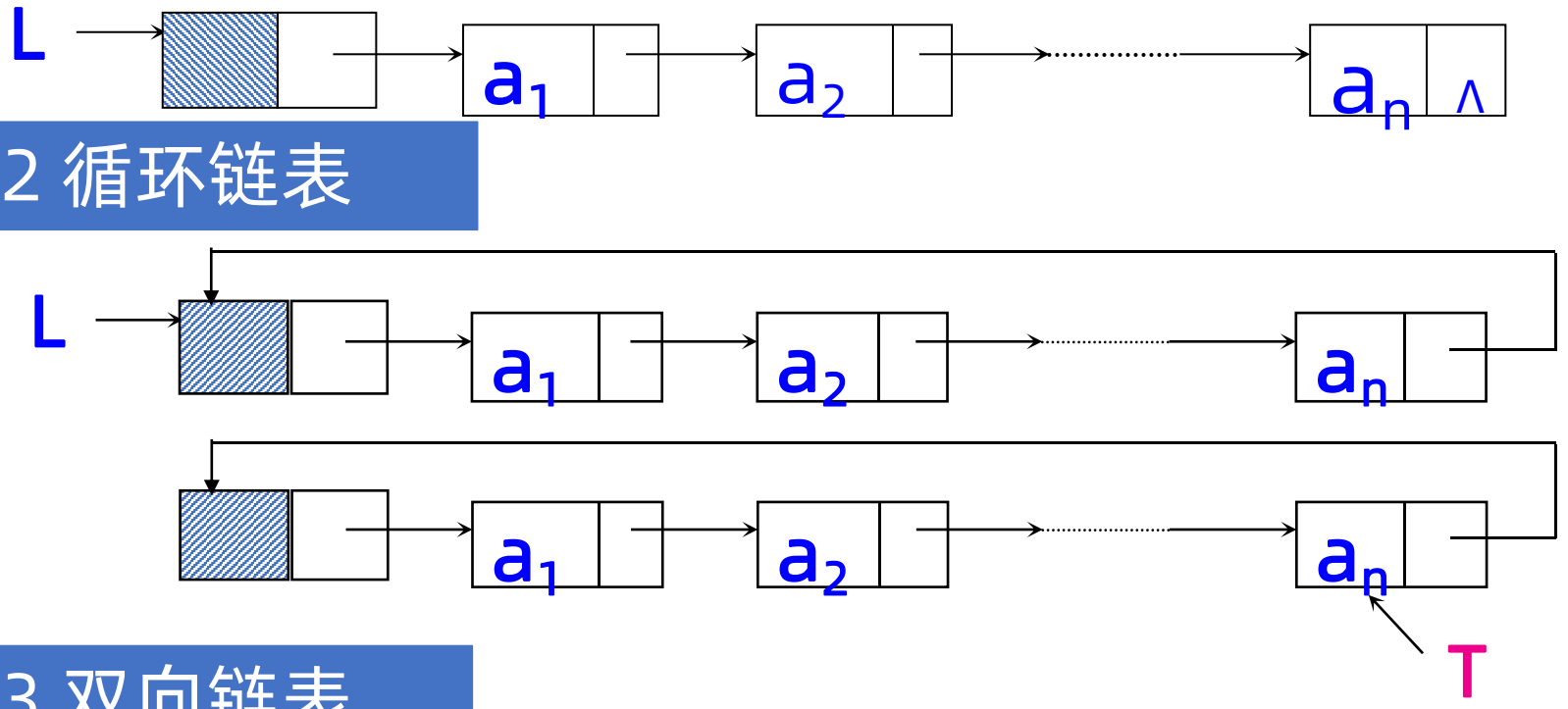
知识回顾：线性表的链式表示和实现

链表

- 2.3.1单链表
- 2.3.2 循环链表
- 2.3.3 双向链表



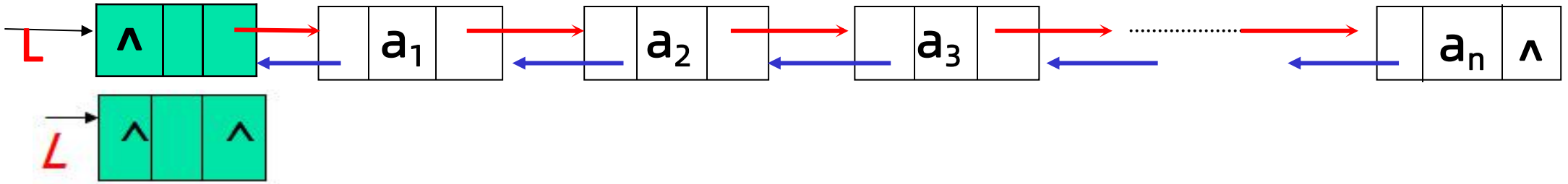
2.3.3 双向链表

刘　芳 LiuFang

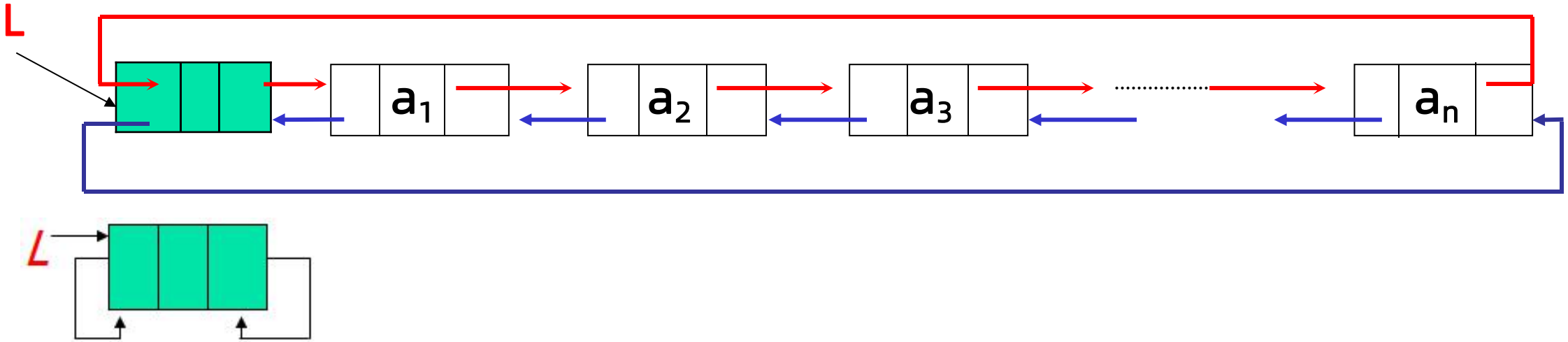
四川師範大學
SICHUAN NORMAL UNIVERSITY

双向链表结点结构及示意图

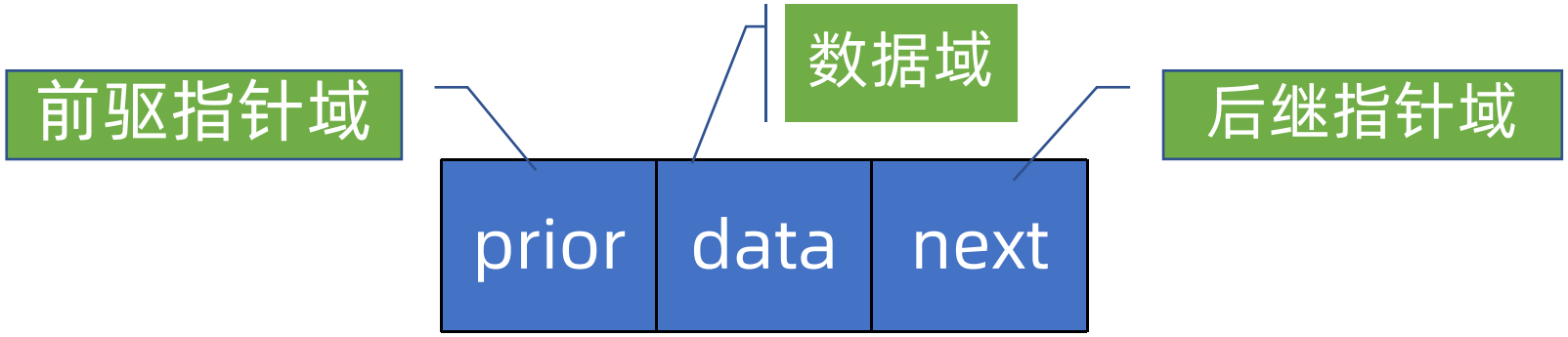
- 带头结点的双向链表



- 带头结点的双向循环链表



双向链表的结点结构及类型定义




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

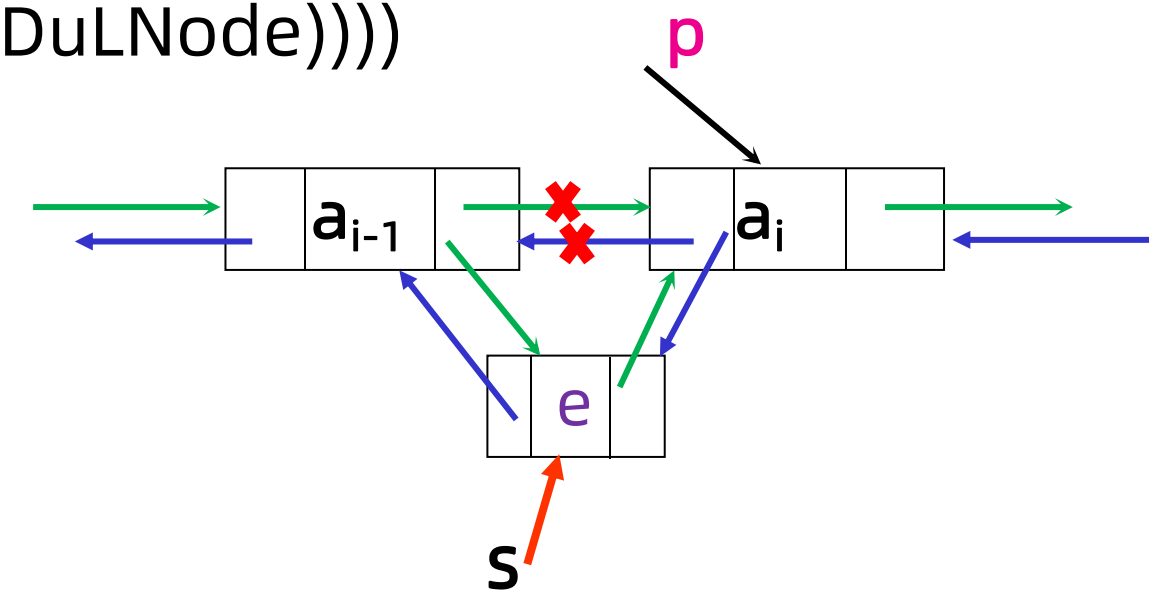
双向（循环）链表的基本运算

1. 建立 InitList(&L)
2. 输出 OutputList(L)
3. 求长度 ListLength(L)
4. 查找 LocateElem(L,e)
   GetElem(L,i,&e)
5. 插入 ListInsert(&L,i,e)
6. 删除 ListDelete(&L,i,&e)
7. 判空 ListEmpty(L)
8. 表的合并 MergeList(La,Lb,&Lc)
9. 清空表 ClearList(&L)
10. 销毁表 DestroyList(&L)

和单(单循环)链表类似
只不过可以顺着两个方向访问。
注意操作时保持链的完整性

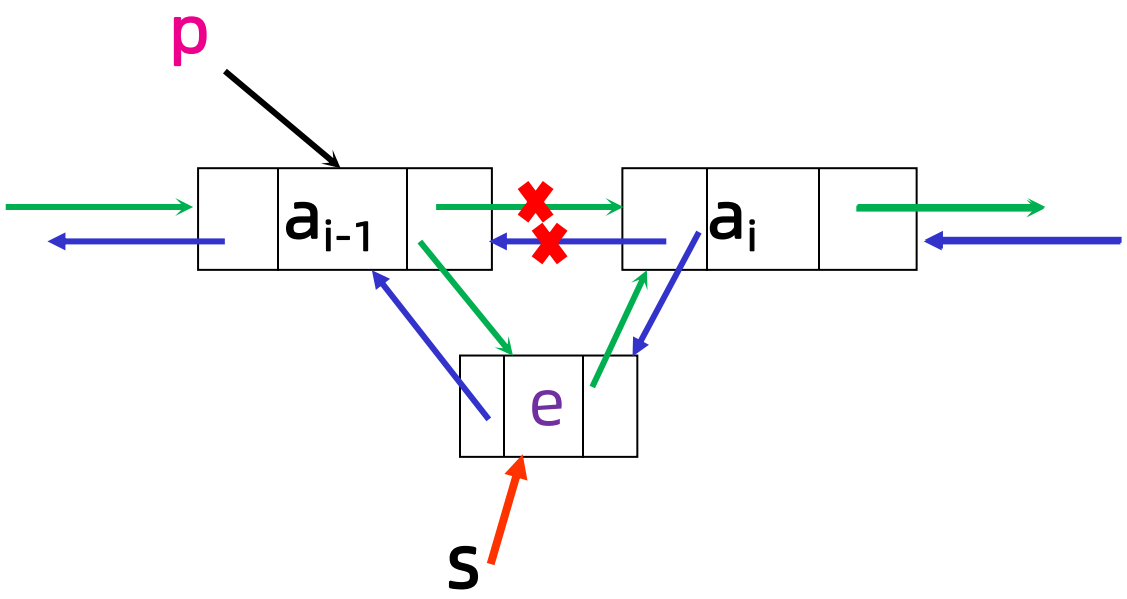
双向链表的插入

```
Status ListInsert_DUL(DuLinkList &L,int i,ElemType e){
    //在L中确定插入位置，使得指针p指向第i个结点
    //若i不合法，则返回ERROR
    if (!(s=(DuLinkList)malloc(sizeof(DuLNode))))
        return ERROR;
    s->data=e;
    s->prior=p->prior ;
    p->prior->next=s  ;
    s->next=p  ;    p->prior=s ;
    return OK
}
```



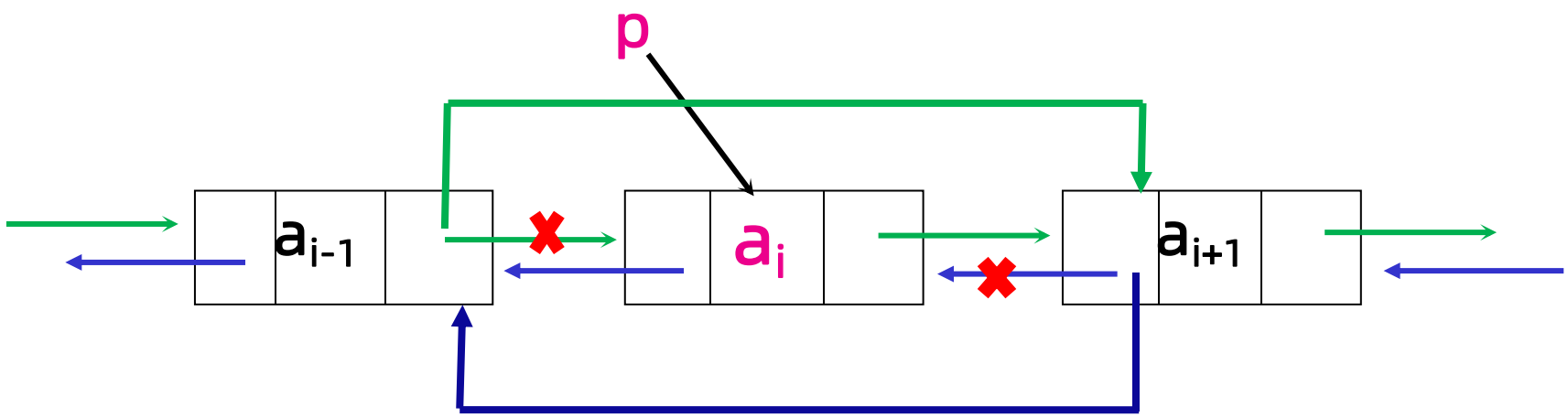
双向链表的插入

- 思考：若指针定位在第i-1个结点，如何在p的后面插入新的结点，并保持链的完整性。




```
s->next=p->next ;
p->next->prior=s;
s->prior=p ;
p->next=s;
```

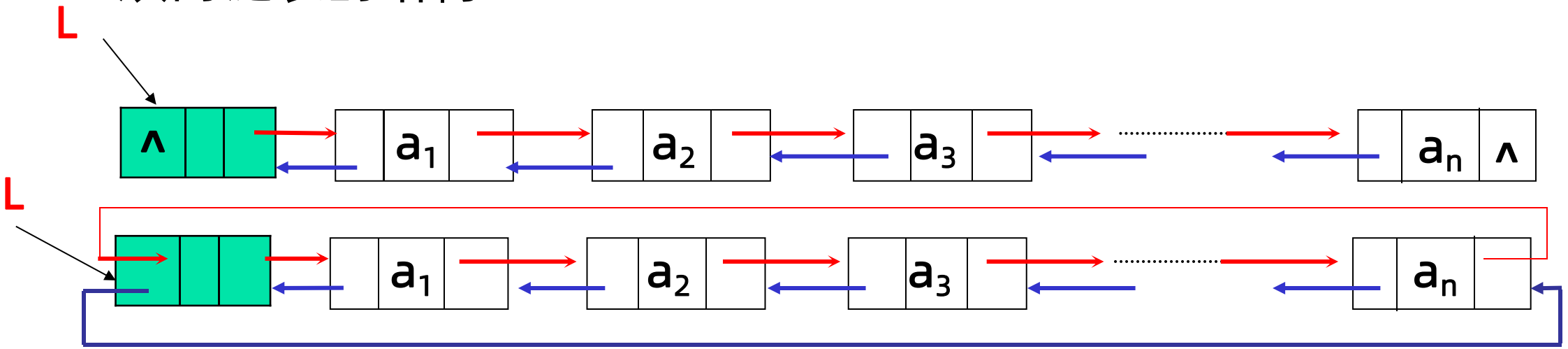
双向链表的删除




```
e=p->data;
p->prior->next =p->next;
p->next->prior= p->prior;
free(p) ;
```

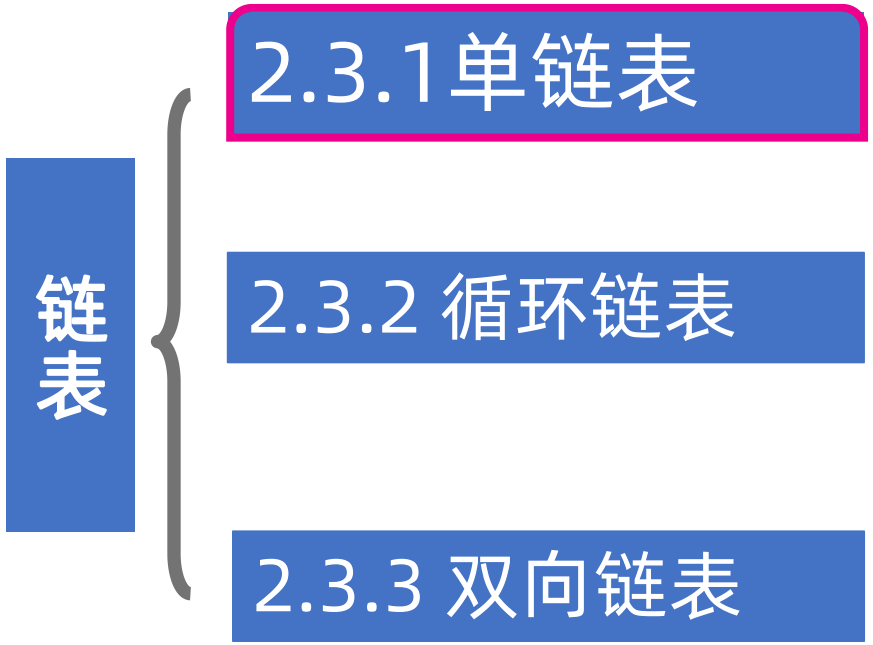
本节要点

- 双向链表的结构



- 双向链表的运算（建立、查找、插入、删除、……）

小结：线性表的链式表示和实现



四川師範大學
SICHUAN NORMAL UNIVERSITY

感谢聆听

业精于勤,荒于嬉;行成于思,毁于随.