



石家莊鐵道大學  
SHIJIAZHUANG TIEDAO UNIVERSITY

在线开放课程

线性表

顺序表的基本操作  
——删除算法

主讲：刘辉

# 目录

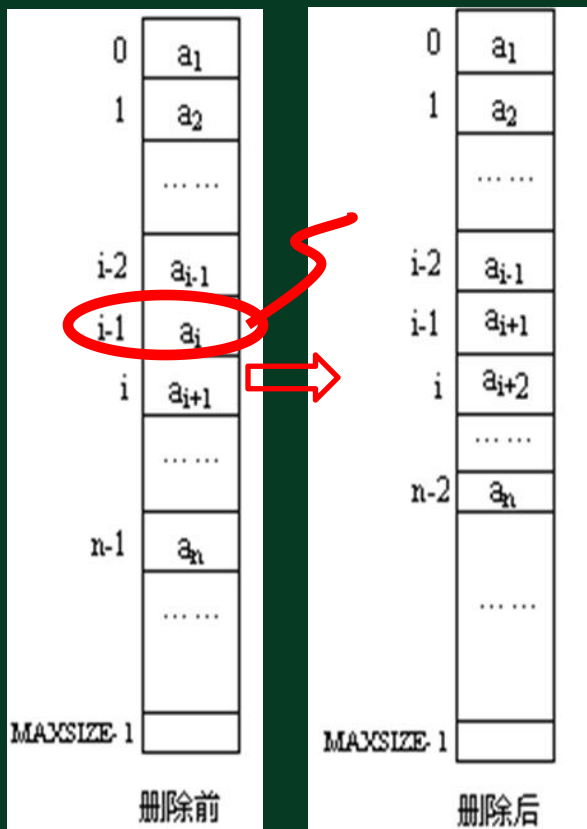
- ◆ 1 顺序表的删除算法描述
- ◆ 2 顺序表的删除算法实现
- ◆ 3 顺序表的删除算法复杂度分析
- ◆ 4 顺序表结构的优缺点分析

# 一、顺序表的删除算法

- 顺序表的删除是指将表中的第  $i$  个元素从线性表中去掉。  
 $i$  的取值范围为： $1 \leq i \leq n$

步骤：

- ① 将  $a_{i+1} \sim a_n$  顺序向上移动。
- ② 修改  $last$  指针值使其仍指向最后一个元素。



# 一、顺序表的删除算法

## 删除算法的示例

删除顺序表中的第4个结点89

	1	25	25	25	25
	2	12	12	12	12
	3	47	47	47	47
删除	4	89	36	36	36
	5	36	↓	14	14
	6	14	14	↑	
	7				
	8				
	9				

删除第 4 个结点，移动  $6-4$  次

删除第  $i$  个结点，移动  $n-i$  次

## 二、顺序表的删除算法实现

### ◆ 删除算法的思想

- (1) 判断删除位置*i* 是否合法（合法值为 $1 \leq i \leq n$ ）。
- (2) 将欲删除的元素保留在*e*中。
- (3) 将第*i*+1至第*n* 位的元素依次向前移动一个位置。
- (4) 表长减1，删除成功返回OK。

## 二、顺序表的删除算法实现

### ◆ 删除算法实现

- 删除第*i*个元素，并用*e*返回其值

```
Status ListDelete_Sq(SqList &L, int i, ElemType &e){
```

```
    if ((i<1)||i>L.length) return ERROR; //i值不合法
    e=L.elem[i-1]; //将欲删除的元素保留在e中
    for (j=i;j<=L.length-1;j++)
        L.elem[j-1]=L.elem[j]; //被删元素之后的元素前移
    --L.length; //表长减1
    return OK;
```

```
}
```

```
#define MAXSIZE 100
typedef struct {
    ElemType *elem;
    int length
}SqList;
```

## 二、顺序表的删除算法实现

```
typedef struct {  
    ElemType *elem;  
    int length;  
    int listsize;  
} sqList;
```

### ◆ 如何调用顺序表描述的这些算法？


```
void main()  
{  
    int i,l,e;  
    sqList a;  
    InitList(&a);  
    printf("Please input array 5 number:");  
    for(i=0;i<5;i++)  
        scanf("%d",&a.elem[i]);  
    a.length=i;  
    printf("\nInput the delete element:");  
    scanf("%d",&i);  
    if(DeleteList(&a,i))  
    {  
        printf("delete succeed!\n");  
        for(i=0;i<a.length;i++)  
            printf("%d ",a.elem[i]);  
        printf("\n");  
    }  
    else  
        printf("Can't find the delete data!\n");  
}
```

```
Please input array 5 number:3 -12 45 4 56  
  
Input the delete element:4  
delete succeed!  
3 -12 45 56  
Press any key to continue
```

## 三、顺序表的删除算法复杂度分析

### ◆ 算法时间主要耗费在移动元素的操作上

- 若删除尾结点，则根本无需移动（**移动次数最少**）；
- 若删除首结点，则表中 $n-1$ 个元素全部前移（**移动次数最多**）；
- 若要考虑在各种位置删除（共 $n$ 种可能）的平均移动次数，该如何计算？


$$AMN = \frac{1}{n} \sum_{i=1}^n (n-i) = \frac{1}{n} \frac{(n-1)n}{2} = \frac{n-1}{2}$$



## 三、顺序表的删除算法复杂度分析

- ◆ 顺序表基本算法的复杂度分析
  - 时间复杂度：（查找、插入、删除）  
算法的平均时间复杂度均为 $O(n)$
  - 空间复杂度： $S(n)=O(1)$ （没有占用  
辅助空间）

## 四、顺序表结构的优缺点分析

### ◆ 顺序表结构的特点

- 利用数据元素的存储位置表示线性表中相邻数据元素之间的前后关系，即线性表的逻辑结构与存储结构一致
- 在访问线性表时，可以快速地计算出任何一个数据元素的存储地址。因此可以近似地认为，访问每个元素所花时间相等

这种存取元素的方法被称为**随机存取法**

# 四、顺序表结构的优缺点分析

## ◆ 顺序表的优缺点分析

### • 优点

- ◆ 存储密度大（结点本身所占存储量/结点结构所占存储量）
- ◆ 可以随机存取表中任一元素

### • 缺点

- ◆ 在插入、删除某一元素时，需大量移动元素
  - ◆ 浪费存储空间
  - ◆ 属于静态存储形式，数据元素的个数不能自由扩充
- ➡ 为解决这些缺点引入 **链表**

# 小结

- 掌握顺序表的删除算法
- 掌握顺序表删除算法的效率分析
- 了解顺序表存储结构的优缺点

# 谢谢！