数据结构（C语言版）（第2版）

线性表

线性表的顺序表表示与实现

主讲教师：汪红松

教 学 内 容 Contents

一、线性表的顺序表示

线性表的顺序表示又称为顺序存储结构或顺序映像。

顺序存储定义 01

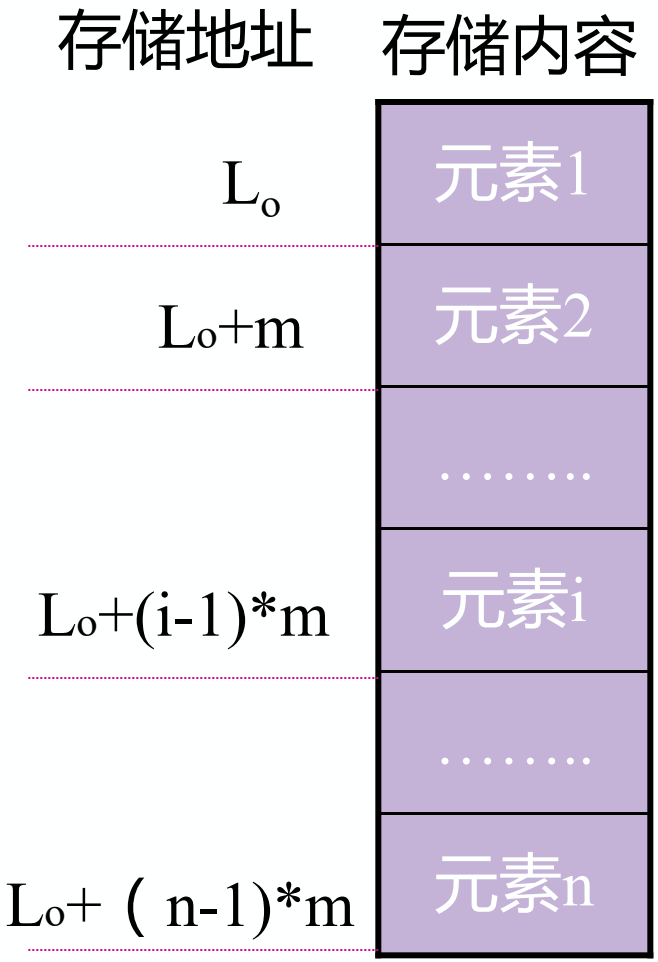
把逻辑上相邻的数据元素存储在物理上相邻的存储单元中的存储结构。

顺序存储方法 02

用一组地址连续的存储单元依次存储线性表的元素，可通过数组V[n]来实现。

一、线性表的顺序表示

1.顺序存储



$Loc(元素i)=L_o+(i-1)*m$

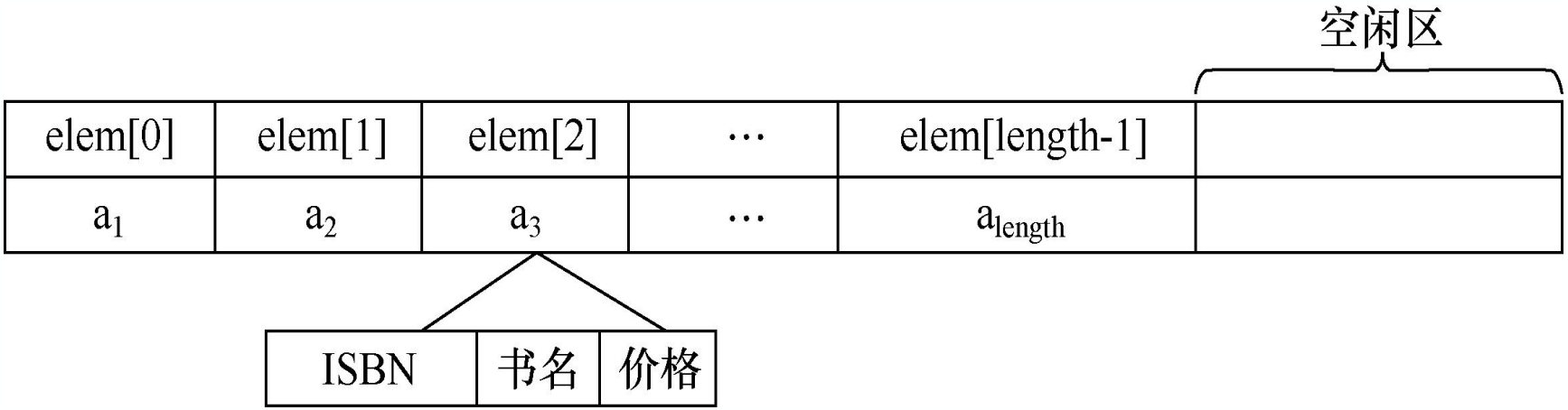
一、线性表的顺序表示

2.顺序表的类型定义

```
#define  MAXSIZE 100     //最大长度
typedef  struct
{
    ElemType  *elem;      //指向数据元素的基地址
   int  length;                  //线性表的当前长度
 }SqList ;
```

3.图书表的顺序存储结构类型定义

图书顺序表



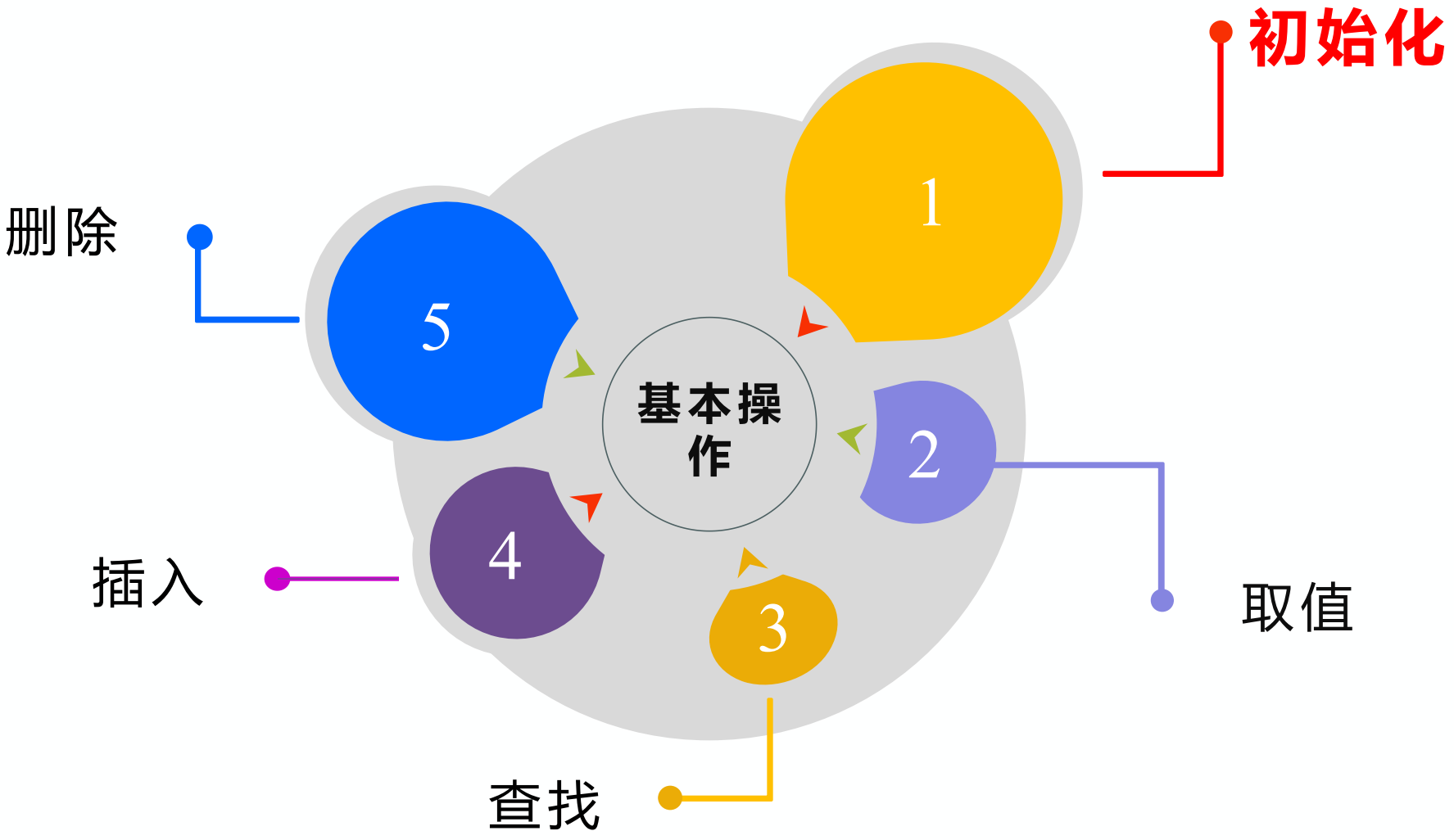
一、线性表的顺序表示

3.图书表的顺序存储结构类型定义

```
#define MAXSIZE 10000      //图书表可能达到的最大长度
typedef struct             //图书信息定义
{
   char no[20];            //图书ISBN
   char name[50];          //图书名字
   float price;            //图书价格
}Book;
typedef struct
{
   Book *elem;             //存储空间的基地址
   int length;             //图书表中当前图书个数
}SqList;                   //图书表的顺序存储结构类型为SqList

SqList L;       //L.elem[i-1]访问表中位置序号为i的图书记录
```

二、线性表的重要基本操作



二、线性表的重要基本操作

初始化线性表L

```
Status InitList_Sq(SqList &L)                    //构造一 个空的顺序表L
{
    L.elem=new ElemType[MAXSIZE];   //为顺序表分配空间
    if(!L.elem) exit(OVERFLOW);          //存储分配失败
    L.length=0;                                    //空表长度为0
    return OK;
}
```

二、线性表的重要基本操作

1.取值

获取线性表L中的某个数据元素的内容

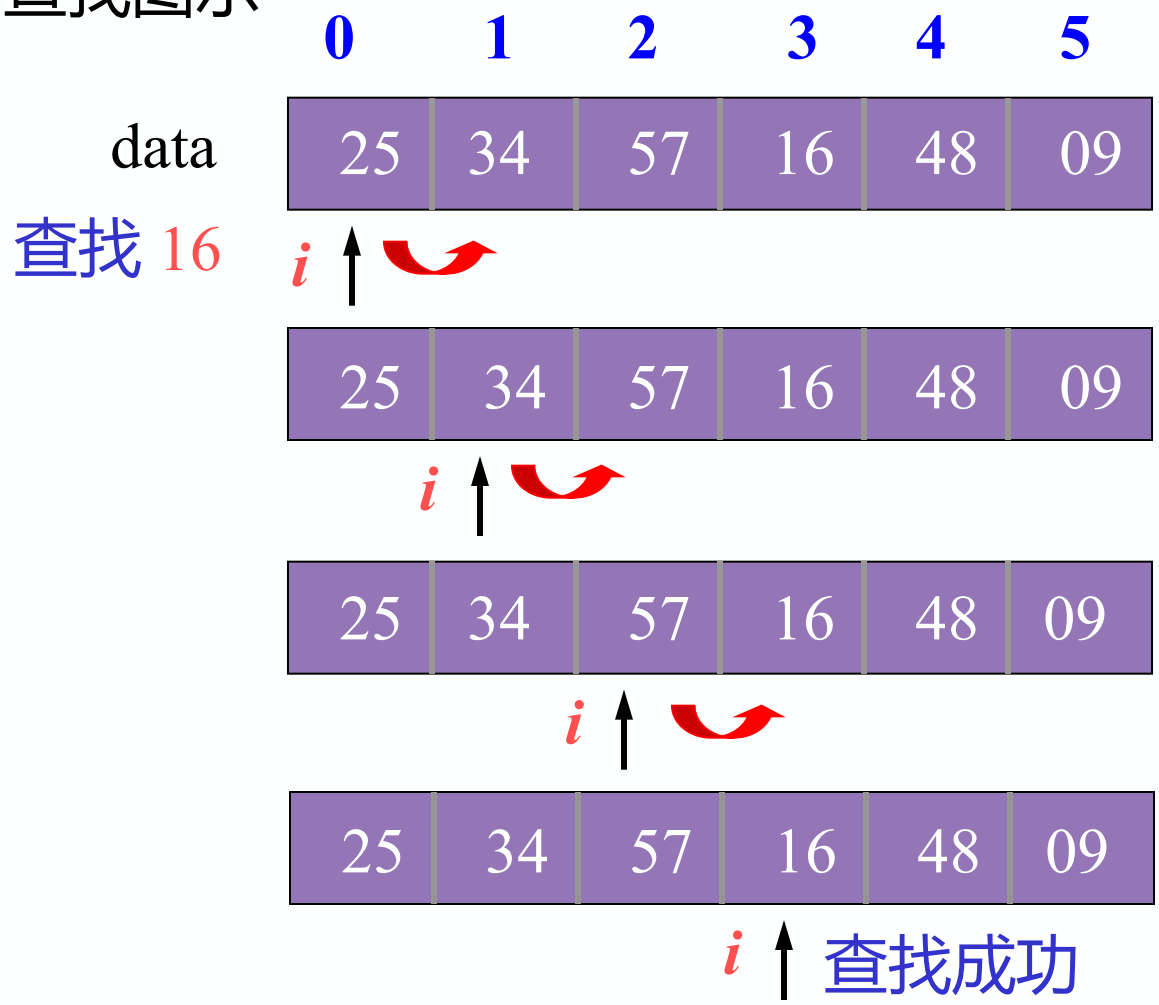
```
int GetElem(SqList L,int i,ElemType &e)
{
  if (i<1||i>L.length) return ERROR;
   //判断i值是否合理，若不合理，返回ERROR
  e=L.elem[i-1];   //第i-1的单元存储着第i个数据
  return OK;
}
```

随机存取

二、线性表的重要基本操作

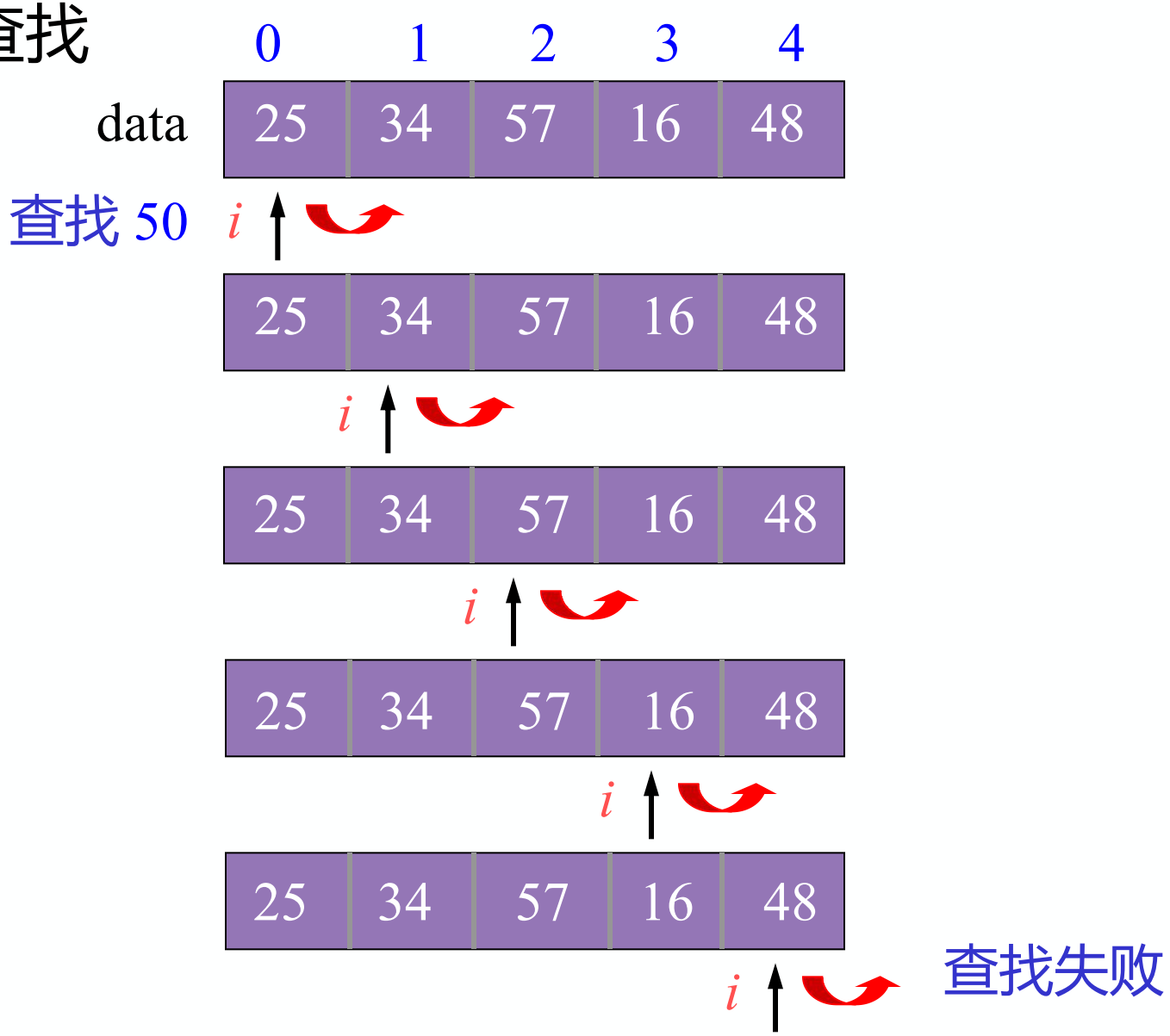
2.查找

顺序查找图示



二、线性表的重要基本操作

2.查找



二、线性表的重要基本操作

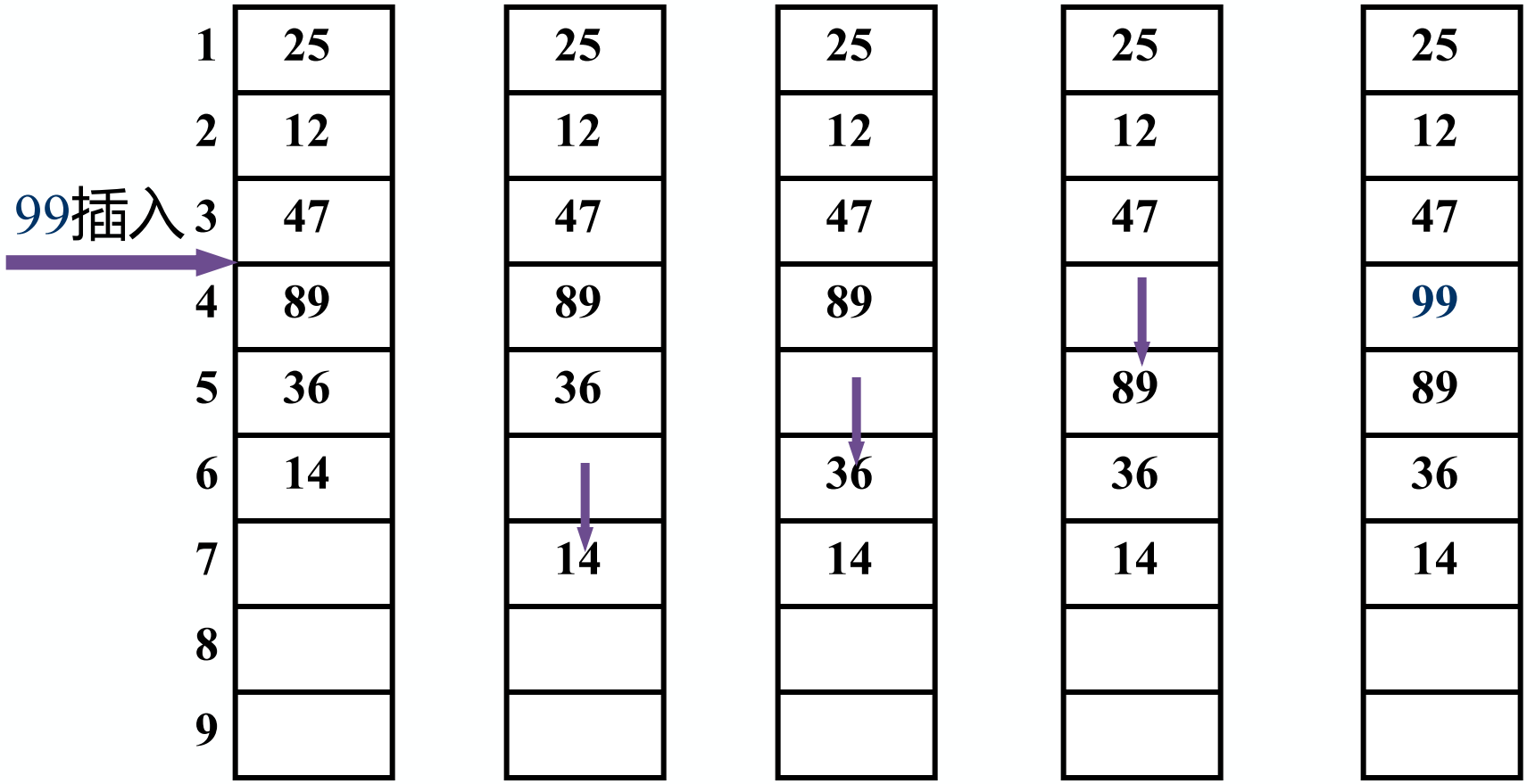
2.查找

在线性表L中查找值为e的数据元素

```
int LocateELem(SqList L,ElemType e)
{
    for (i=0;i< L.length;i++)
          if (L.elem[i]==e) return i+1;
     return 0;
}
```

查找算法的时间复杂度为O(n)。

二、线性表的重要基本操作　3.插入

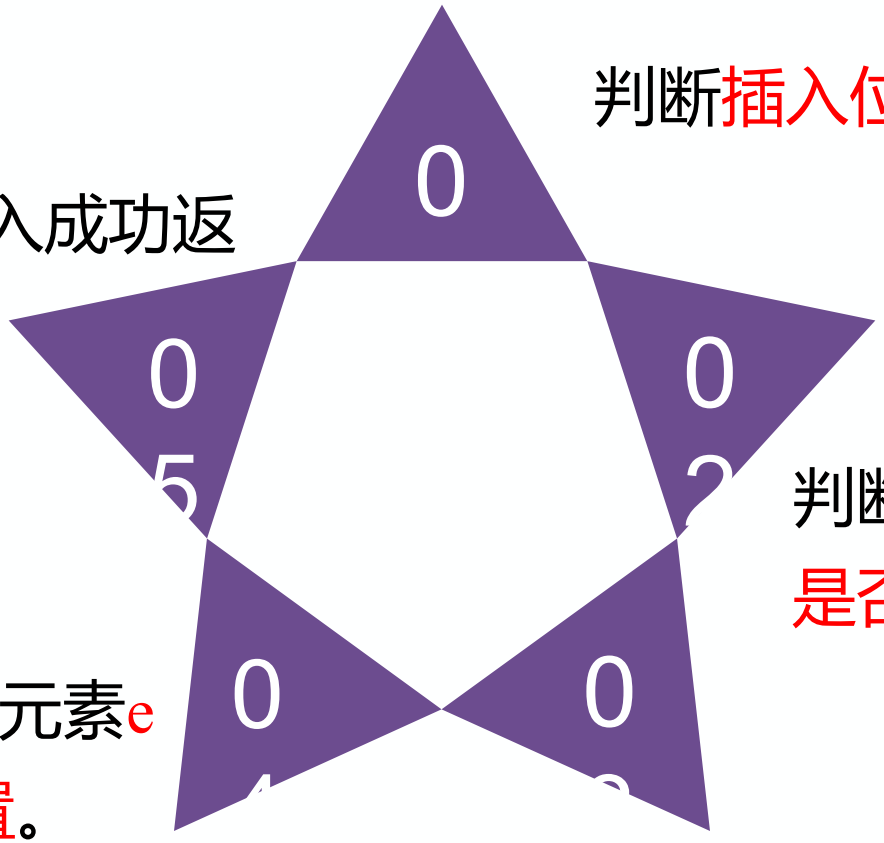


插第 4 个结点之前，移动 6 - 4+1 次

插在第 i 个结点之前，移动 n-i+1 次

二、线性表的重要基本操作

4.算法步骤



判断插入位置i 是否合法。

判断顺序表的存储空间是否已满。

将第n至第i 位的元素依次向后移动一个位置，空出第i个位置。

将要插入的新元素e放入第i个位置。

表长加1，插入成功返回OK。

在线性表L中第i个数据元素之前插入数据元素e

```
Status ListInsert_Sq(SqList &L,int i ,ElemType e)
{
  if(i<1 || i>L.length+1) return ERROR;          //i值不合法
  if(L.length==MAXSIZE) return ERROR;    //当前存储空间已满
  for(j=L.length-1;j>=i-1;j--)
     L.elem[j+1]=L.elem[j];    //插入位置及之后的元素后移
   L.elem[i-1]=e;                    //将新元素e放入第i个位置
   ++L.length;                        //表长增1
   return OK;
}
```

二、线性表的重要基本操作　　6.算法分析

算法时间主要耗费在移动元素的操作上。

若插入在尾结点之后，则根本无需移动（特别快）；

若元素全部后移（特别慢）；

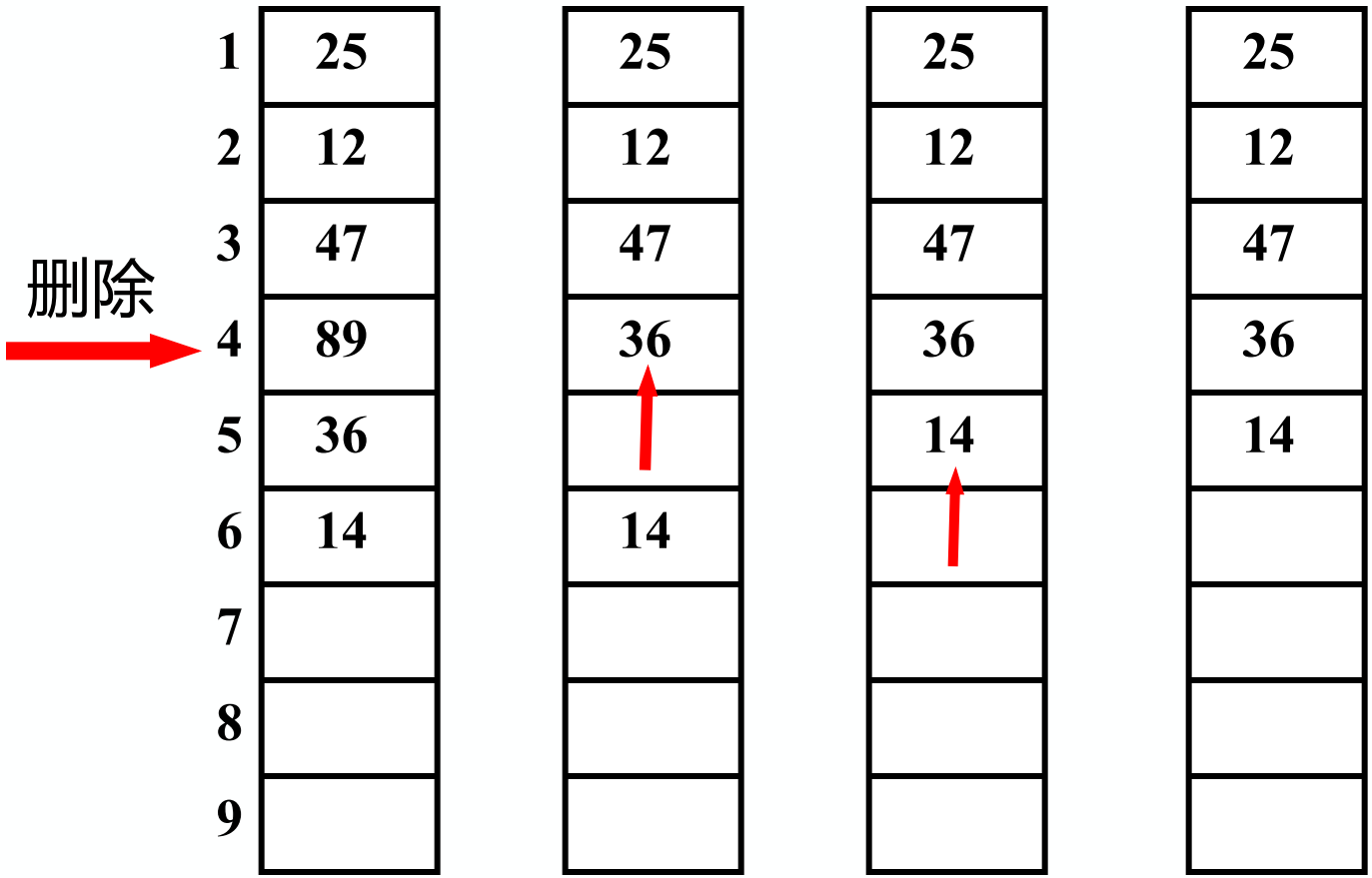
若要考虑在各种位置插入（共n+1种可能）的平均移动次数，该如何计算？

$$\mathrm{AMN} = \frac{1}{n+1}\sum_{i=1}^{n+1}(n-i+1) = \frac{1}{n+1}(n+\cdots+1+0)$$

$$= \frac{1}{(n+1)}\frac{n(n+1)}{2} = \frac{n}{2}$$

$T(n)=O(n)$

二、线性表的重要基本操作　　7.删除



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、线性表的重要基本操作

【算法步骤】

（1）判断删除位置i 是否合法（合法值为1≤i≤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、线性表的重要基本操作　　【算法描述】

将线性表L中第i个数据元素删除

```
Status ListDelete_Sq(SqList &L,int i)
{
  if((i<1)||(i>L.length)) return ERROR;          //i值不合法
  for (j=i;j<=L.length-1;j++)
       L.elem[j-1]=L.elem[j];       //被删除元素之后的元素前移
  --L.length;                                   //表长减1
 return OK;
}
```

二、线性表的重要基本操作　　【算法分析】

算法时间主要耗费在移动元素的操作上

1. 若删除尾结点，则根本无需移动（特别快）；
2. 若删除首结点，则表中n-1个元素全部前移（特别慢）；
3. 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？

$T(n)=O(n)$

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

二、线性表的重要基本操作

查找、插入、删除算法的平均时间复杂度为：O(n)

显然，顺序表操作的空间复杂度

$S(n)=O(1)$

（没有占用辅助空间）。

三、顺序表（顺序存储结构）的特点

（1）

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。

（2）

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。因此可以粗略地认为，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。

三、顺序表（顺序存储结构）的特点

顺序表的优缺点

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，需要移动大量元素。
- ✓ 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。

为克服这一缺点 → 链表

小结

1. 顺序表的表示
2. 顺序表的取值、查找、插入和删除算法实现过程
3. 顺序表的特点