

第2章 线性表

近3周
上课
内容

- 第2章 线性表
- 第3章 栈和队列
- 第4章 串、数组和广义表

线性结构

(逻辑、存储
和运算)

线性结构的定义:

若结构是非空有限集, 则有且仅有一个开始结点和一个终端结点, 并且所有结点都最多只有一个直接前趋和一个直接后继。

可表示为: (a_1, a_2, \dots, a_n)

线性结构表达式： (a_1, a_2, \dots, a_n)

线性结构的特点：

- ① 只有一个首结点和尾结点；
- ② 除首尾结点外，其他结点只有一个直接前驱和一个直接后继。

简言之，线性结构反映结点间的逻辑关系是一对一的

线性结构包括线性表、堆栈、队列、字符串、数组等等，其中，最典型、最常用的是



线性表

第2章 线性表



教学目标

1. 了解线性结构的特点
2. 掌握顺序表的定义、查找、插入和删除
3. 掌握链表的定义、创建、查找、插入和删除
4. 能够从时间和空间复杂度的角度比较两种存储结构的不同特点及其适用场合

教学内容

2.1 线性表的定义和特点

2.2 案例引入

2.3 线性的类型定义

2.4 线性表的顺序表示和实现

2.5 线性表的链式表示和实现

2.6 顺序表和链表的比较

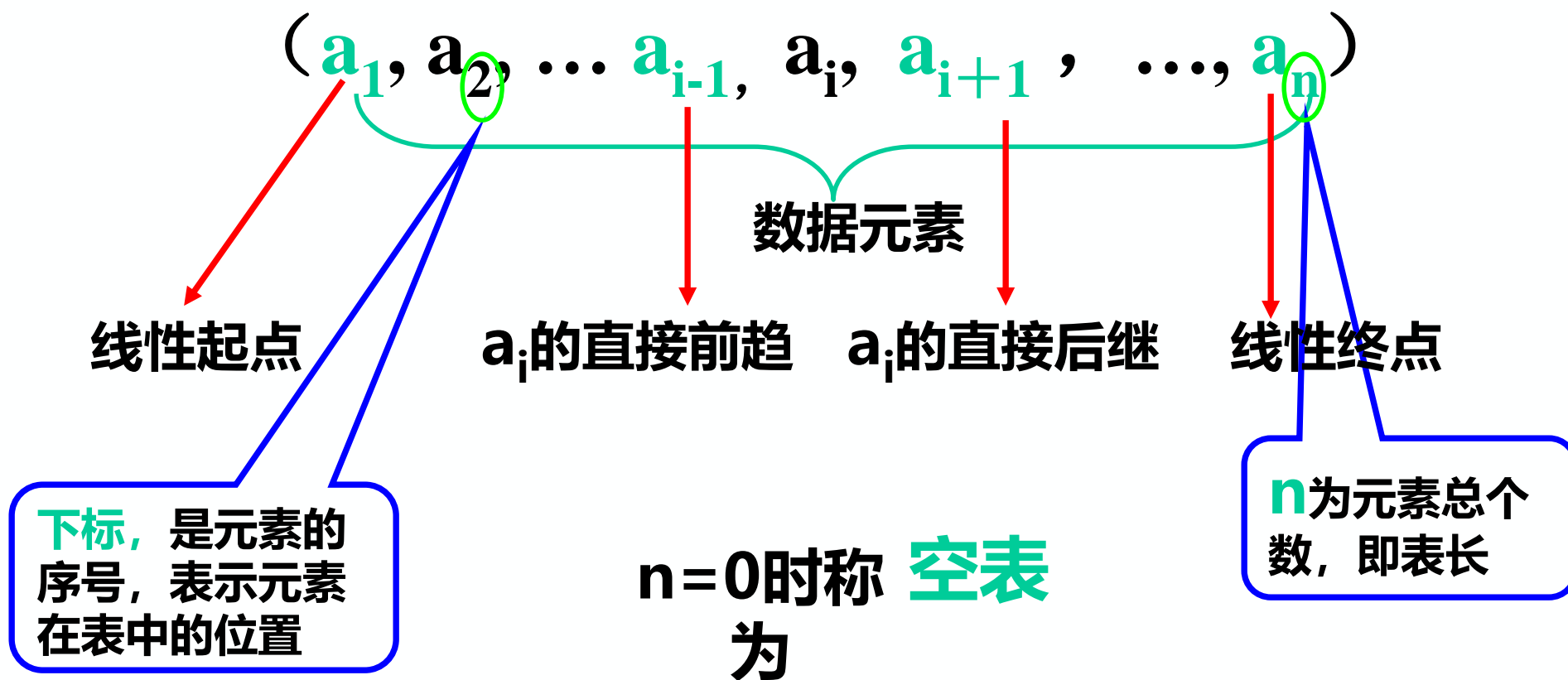
2.7 线性表的应用

2.8 案例分析与实现

2.1 线性表的定义和特点



线性表的定义： 用数据元素的有限序列表示



例1 分析26个英文字母组成的英文表

(A, B, C, D, , Z)

数据元素都是字母； 元素间关系是线性

例2 分析学生情况登记表

学号	姓名	性别	年龄	班级
041810205	于春梅	女	18	04级计算机1班
041810260	何仕鹏	男	20	04级计算机2班
041810284	王 爽	女	19	04级计算机3班
041810360	王亚武	男	18	04级计算机4班
:	:	:	:	:

数据元素都是记录； 元素间关系是线性

同一线性表中的元素必定具有相同特性

2.2 案例引入



案例2.1：一元多项式的运算

$$P_n(x) = p_0 + p_1x + p_2x^2 + \dots + p_nx^n$$

线性表 $P = (p_0, p_1, p_2, \dots, p_n)$

$$P(x) = 10 + 5x - 4x^2 + 3x^3 + 2x^4$$

数组表示

(每一项的指数*i*隐含在其系数 p_i 的序号中)

指数 (下标 <i>i</i>)	0	1	2	3	4
系数 $p[i]$	10	5	-4	3	2

$$R_n(x) = P_n(x) + Q_m(x)$$



线性表 $R = (p_0 + q_0, p_1 + q_1, p_2 + q_2, \dots, p_m + q_m, p_{m+1}, \dots, p_n)$

稀疏多项式

$$S(x) = 1 + 3x^{10000} + 2x^{20000}$$



案例2.2：稀疏多项式的运算

多项式**非零项**的数组表示

(a) $A(x) = 7 + 3x + 9x^8 + 5x^{17}$

下标i	0	1	2	3
系数 a[i]	7	3	9	5
指数	0	1	8	17

(b) $B(x) = 8x + 22x^7 - 9x^8$

下标i	0	1	2
系数 b[i]	8	22	-9
指数	1	7	8

$$P_n(x) = p_1 x^{e_1} + p_2 x^{e_2} + \dots + p_m x^{e_m} \rightarrow$$



线性表 $P = ((p_1, e_1), (p_2, e_2), \dots, (p_m, e_m))$

- 创建一个**新数组c**
- 分别从头遍历比较a和b的每一项
 - ✓ **指数相同**，对应系数相加，若其和不为零，则在c中增加一个新项
 - ✓ **指数不相同**，则将指数较小的项复制到c中
- 一个多项式已遍历**完毕**时，将另一个剩余项依次复制到c中即可

●顺序存储结构存在问题

✓存储空间分配不灵活

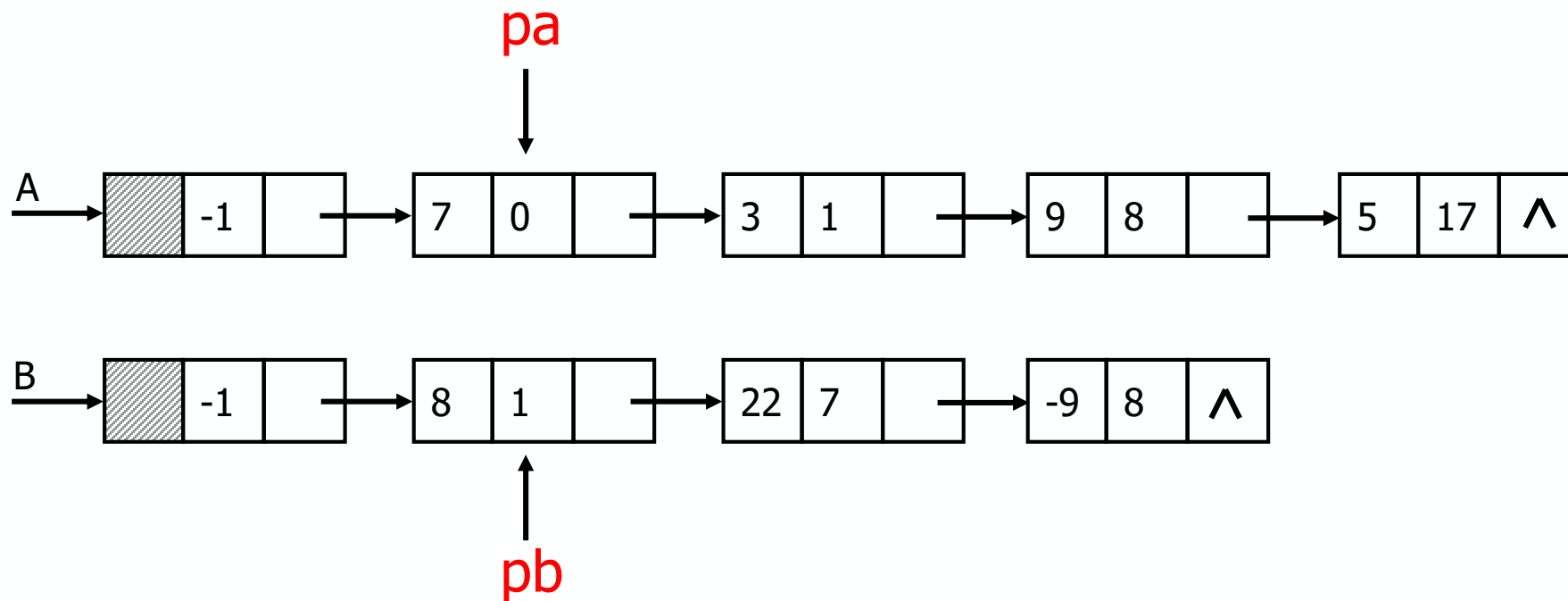
✓运算的空间复杂度高



链式存储结构

$$A_{17}(x)=7+3x+9x^8+5x^{17}$$

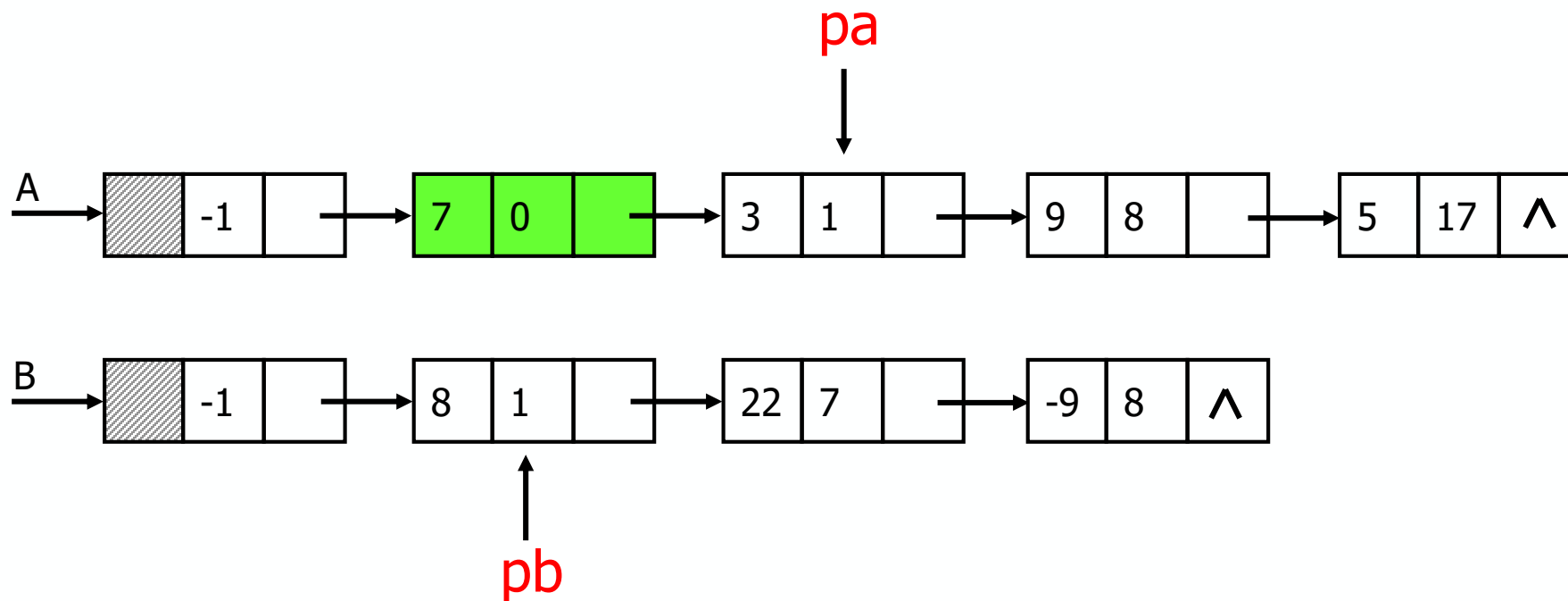
$$B_8(x)=8x+22x^7-9x^8$$



多项式相加

$$A_{17}(x) = 7 + 3x + 9x^8 + 5x^{17}$$

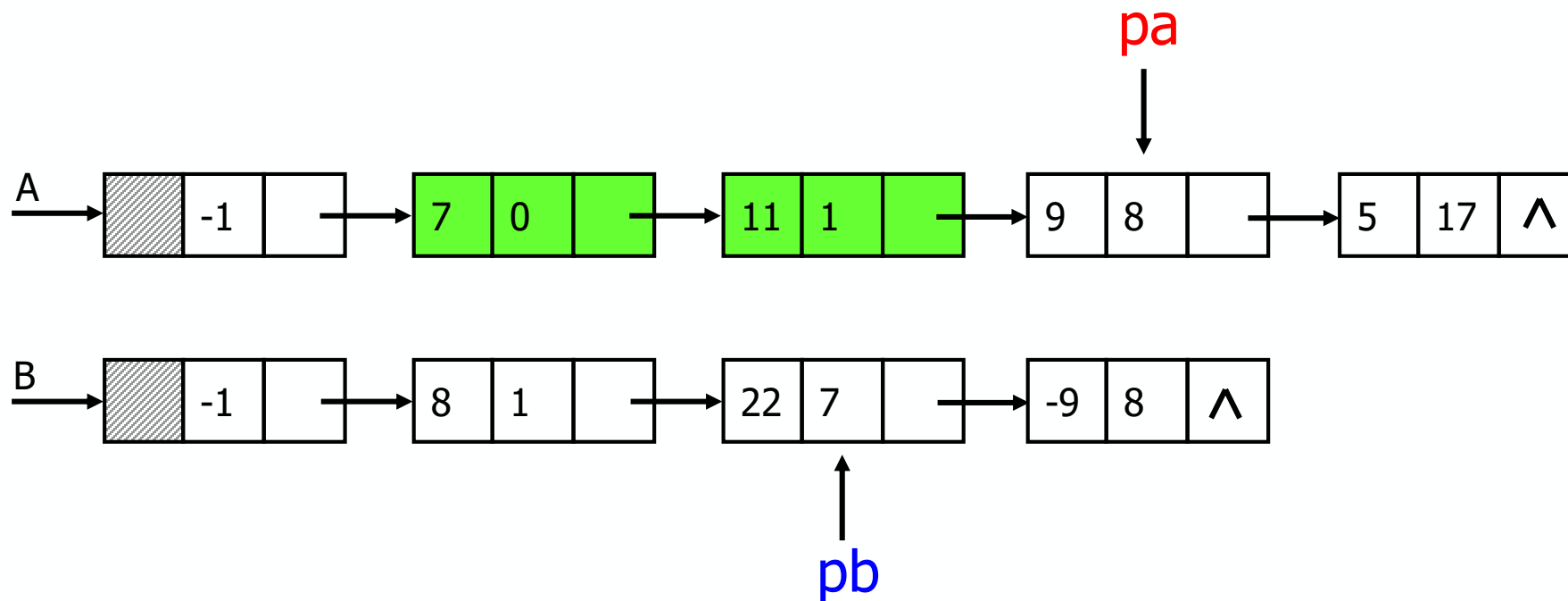
$$B_8(x) = 8x + 22x^7 - 9x^8$$



多项式相加

$$A_{17}(x) = 7 + 3x + 9x^8 + 5x^{17}$$

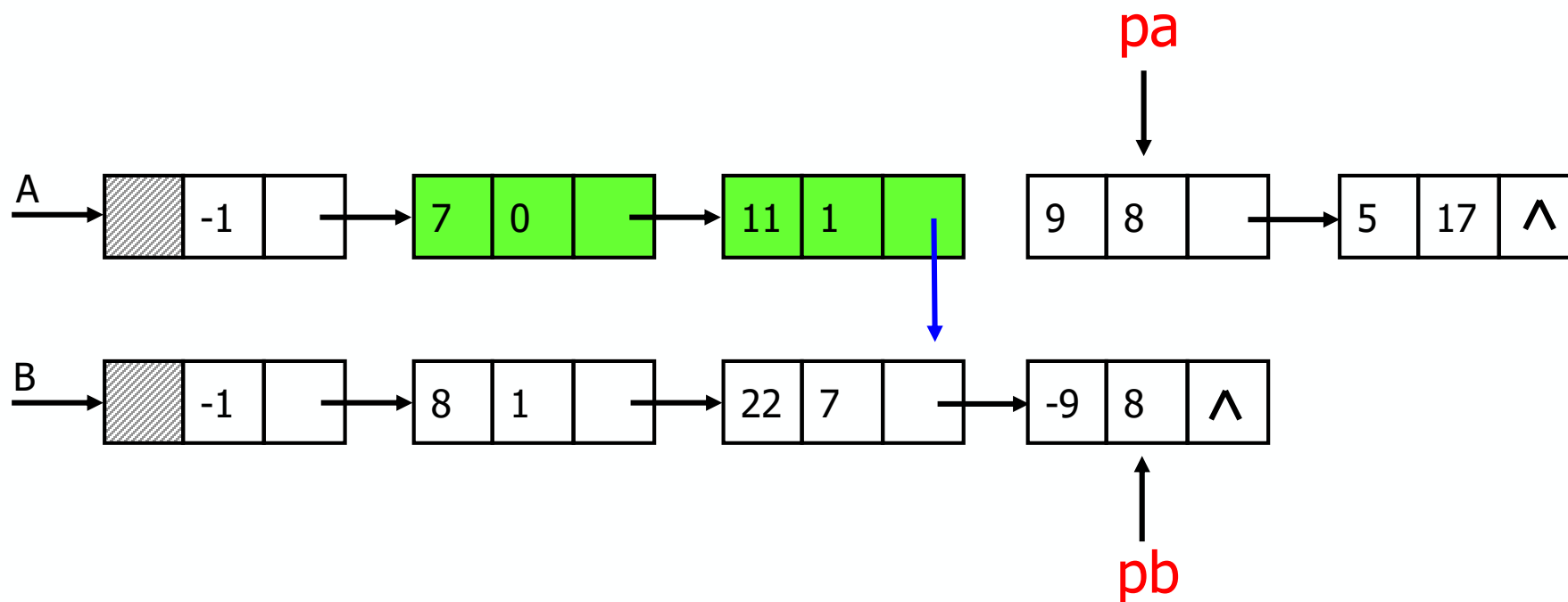
$$B_8(x) = 8x + 22x^7 - 9x^8$$



多项式相加

$$A_{17}(x)=7+3x+9x^8+5x^{17}$$

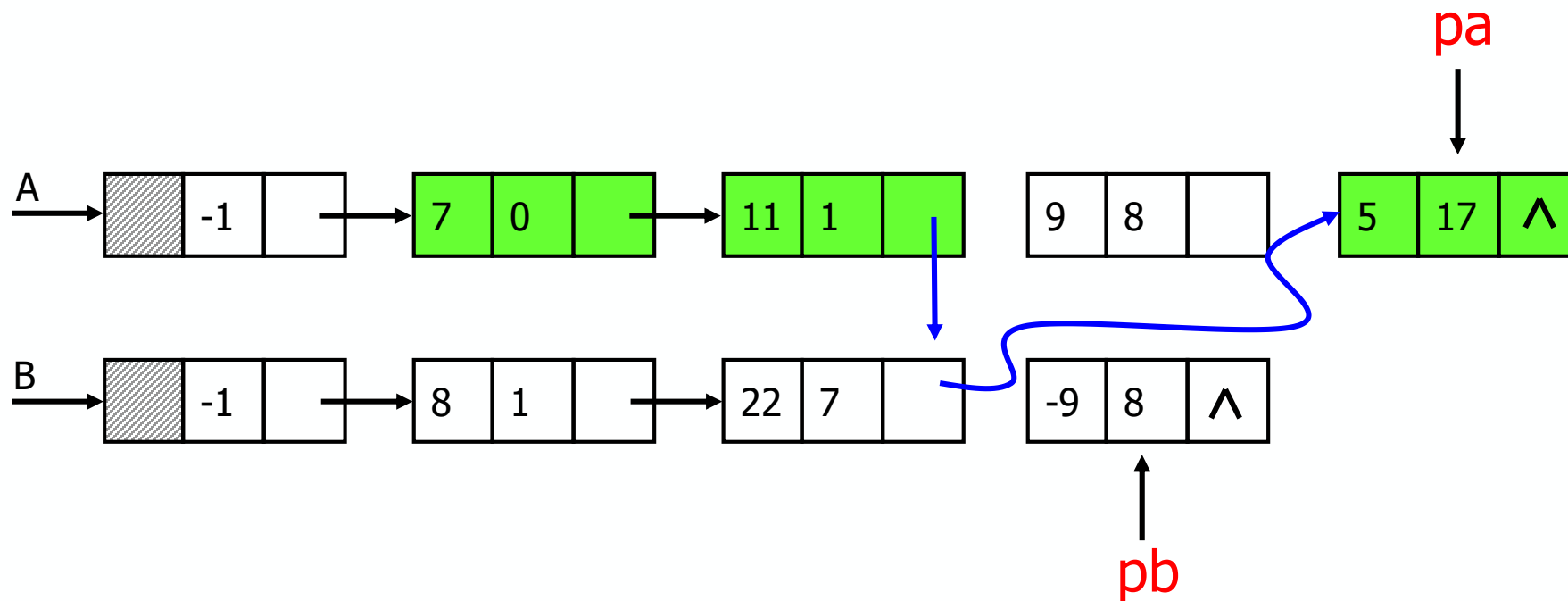
$$B_8(x)=8x+22x^7-9x^8$$

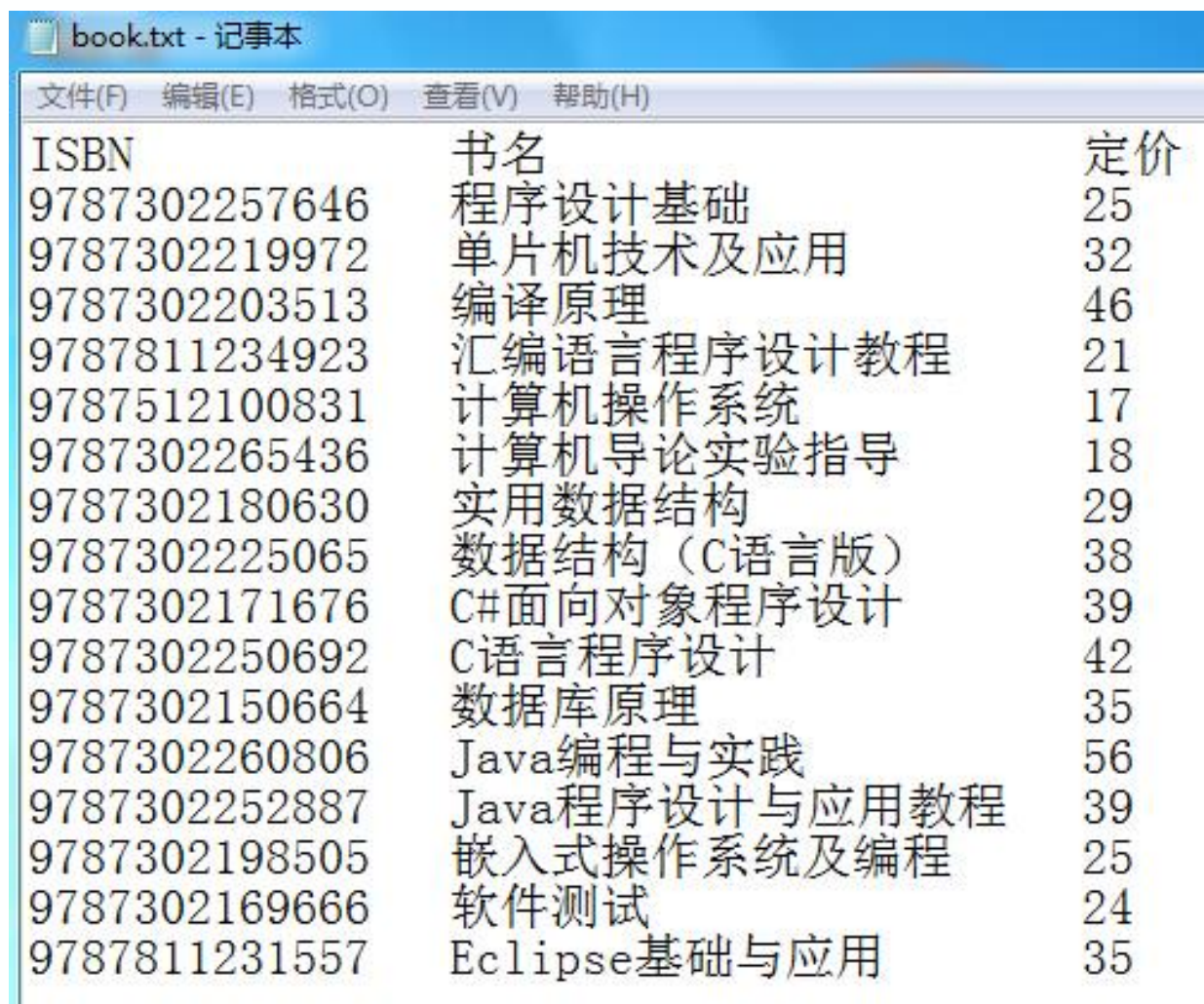


多项式相加

$$A_{17}(x) = 7 + 3x + 9x^8 + 5x^{17}$$

$$B_8(x) = 8x + 22x^7 - 9x^8$$





book.txt - 记事本

ISBN	书名	定价
9787302257646	程序设计基础	25
9787302219972	单片机技术及应用	32
9787302203513	编译原理	46
9787811234923	汇编语言程序设计教程	21
9787512100831	计算机操作系统	17
9787302265436	计算机导论实验指导	18
9787302180630	实用数据结构	29
9787302225065	数据结构（C语言版）	38
9787302171676	C#面向对象程序设计	39
9787302250692	C语言程序设计	42
9787302150664	数据库原理	35
9787302260806	Java编程与实践	56
9787302252887	Java程序设计与应用教程	39
9787302198505	嵌入式操作系统及编程	25
9787302169666	软件测试	24
9787811231557	Eclipse基础与应用	35

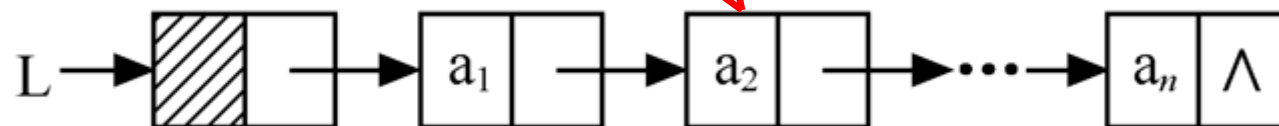
- (1) 查找
- (2) 插入
- (3) 删除
- (4) 修改
- (5) 排序
- (6) 计数

图书顺序表

elem[0]	elem[1]	elem[2]	...	elem[length-1]	空闲区	
a_1	a_2	a_3	...	a_{length}		

ISBN	书名	价格
------	----	----

图书链表





- 线性表中数据元素的类型可以为简单类型，也可以为复杂类型。
- 许多实际应用问题所涉的基本操作有很大相似性，不应为每个具体应用单独编写一个程序。
- 从具体应用中**抽象出共性的逻辑结构和基本操作**（**抽象数据类型**），然后实现其存储结构和基本操作。

2.3 线性表的类型定义



线性表的重要基本操作

1. 初始化
2. 取值
3. 查找
4. 插入
5. 删除

2.4 线性表的顺序表示和实现



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：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。

简言之，逻辑上相邻，物理上也相邻

顺序存储方法：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。

顺序存储

存储地址	存储内容
L_0	元素1
L_0+m	元素2

$L_0+(i-1)*m$	元素i

$L_0+(n-1)*m$	元素n

$$\text{Loc}(\text{元素}i) = L_0 + (i-1)*m$$

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;
```

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
```

补充：C语言的动态分配函数（<stdlib.h>）

malloc(m**):** 开辟**m**字节长度的地址空间
，并返回这段空间的首地址

sizeof(x**):** 计算变量**x**的长度

free(p**):** 释放指针**p**所指变量的存储空间
，即彻底删除一个变量

补充：C++的动态存储分配

new 类型名T (初值列表)

功能：

```
int *p1 = new int;  
或 int *p1 = new int(10);
```

申请用于存放T类型对象的内存空间，并依初值列表赋以初值

结果值：

成功：T类型的指针，指向新分配的内存

失败：0 (NULL)

delete 指针P

```
delete p1;
```

功能：

释放指针P所指向的内存。P必须是new操作的返回值

- 函数调用时传送给形参表的实参必须与形参在类型、个数、顺序上保持一致
- 参数传递有两种方式
 - **传值方式** (参数为整型、实型、字符型等)
 - **传地址**
 - 参数为指针变量
 - 参数为引用类型
 - 参数为数组名

传值方式

把实参的值传送给函数局部工作区相应的副本中，函数使用这个副本执行必要的功能。
函数修改的是副本的值，实参的值不变

```
#include <iostream.h>

void swap(float m,float n)
{float temp;

temp=m;

m=n;

n=temp;

}
```

```
void main()
{float a,b;

cin>>a>>b;

swap(a,b);

cout<<a<<endl<<b<<endl;

}
```

传地址方式 - - 指针变量作参数

形参变化影响实参

```
#include <iostream.h>

void swap(float *m,float *n)
{float t;

t=*m;

*m=*n;

*n=t;

}
```

```
void main()
{float a,b,*p1,*p2;

cin>>a>>b;

p1=&a; p2=&b;
swap(p1, p2);

cout<<a<<endl<<b<<endl;

}
```

传地址方式 - - 指针变量作参数

形参变化不影响实参？？

```
#include <iostream.h>

void swap(float *m,float *n)
{float *t;

 t=m;

 m=n;

 n=t;

}
```

```
void main()
{float a,b,*p1,*p2;

 cin>>a>>b;

 p1=&a; p2=&b;
 swap(p1, p2);

 cout<<a<<endl<<b<<endl;

}
```

传地址方式 - - 引用类型作参数

什么是引用???

引用：它用来给一个对象提供一个替代的名字。

```
#include<iostream.h>
void main(){
    int i=5;
    int &j=i;
    i=7;
    cout<<"i="<<i<<" j="<<j;
}
```

- ✓ **j**是一个引用类型，代表**i**的一个替代名
- ✓ **i**值改变时，**j**值也跟着改变，所以会输出
i=7 j=7

传地址方式 - - 引用类型作参数

```
#include <iostream.h>

void swap(float & m, float & n)
{float temp;
 temp=m;
 m=n;
 n=temp;
}
```

```
void main()
{float a,b;
 cin>>a>>b;
 swap(a,b);
 cout<<a<<endl<<b<<endl;
}
```

引用类型作形参的三点说明

- (1) 传递引用给函数与传递指针的效果是一样的，**形参变化实参也发生变化**。
- (2) 引用类型作形参，在内存中并没有产生实参的副本，它**直接对实参操作**；而一般变量作参数，形参与实参就占用不同的存储单元，所以形参变量的值是实参变量的副本。因此，当**参数传递的数据量较大**时，用引用比用一般变量传递参数的时间和空间效率都好。

引用类型作形参的三点说明

(3) 指针参数虽然也能达到与使用引用的效果，但在被调函数中需要重复使用“***指针变量名**”的形式进行运算，这很容易产生错误且程序的阅读性较差；另一方面，在主调函数的调用点处，必须用**变量的地址作为实参**。



Back

传地址方式 - - 数组名作参数

传递的是数组的首地址

对形参数组所做的任何改变都将反映到实参数组中

```
#include<iostream.h>
```

```
void sub(char);
```

```
void main ( void )
```

```
{ char a[10]="hello";
```

```
    sub(a);
```

```
    cout<<a<<endl;
```

```
}
```

```
void sub(char b[ ])
```

```
{ b[ ]="world";}
```

用数组作函数的参数，求10个整数的最大数

```
#include <iostream.h>
```

```
#define N 10
```

```
int max(int a[]);
```

```
void main ( ) {
```

```
    int a[10];
```

```
    int i,m;
```

```
    for(i=0;i<N;i++)
```

```
        cin>>a[i];
```

```
    m=max(a);
```

```
    cout<<"the max number is:"<<m;
```

```
}
```

```
int max(int b[]){
```

```
    int i,n;
```

```
    n=b[0];
```

```
    for(i=1;i<N;i++)
```

```
        if(n<b[i]) n=b[i];
```

```
    return n;
```

```
}
```

练习：用数组作为函数的参数，将数组中n个整数按相反的顺序存放，要求输入和输出在主函数中完成

```
#include <iostream.h>

#define N 10

void sub(int b[ ]){
    int i,j,temp,m;
    m=N/2;
    for(i=0;i<m;i++){
        j=N-1-i;
        temp=b[i];
        b[i]= b[j];
        b[j]=temp; }
    return ;}
```

```
void main ( ) {
    int a[10],i;
    for(i=0;i<N;i++)
        cin>>a[i];
    sub(a);
    for(i=0;i<N;i++)
        cout<<a[i];
}
```

Back