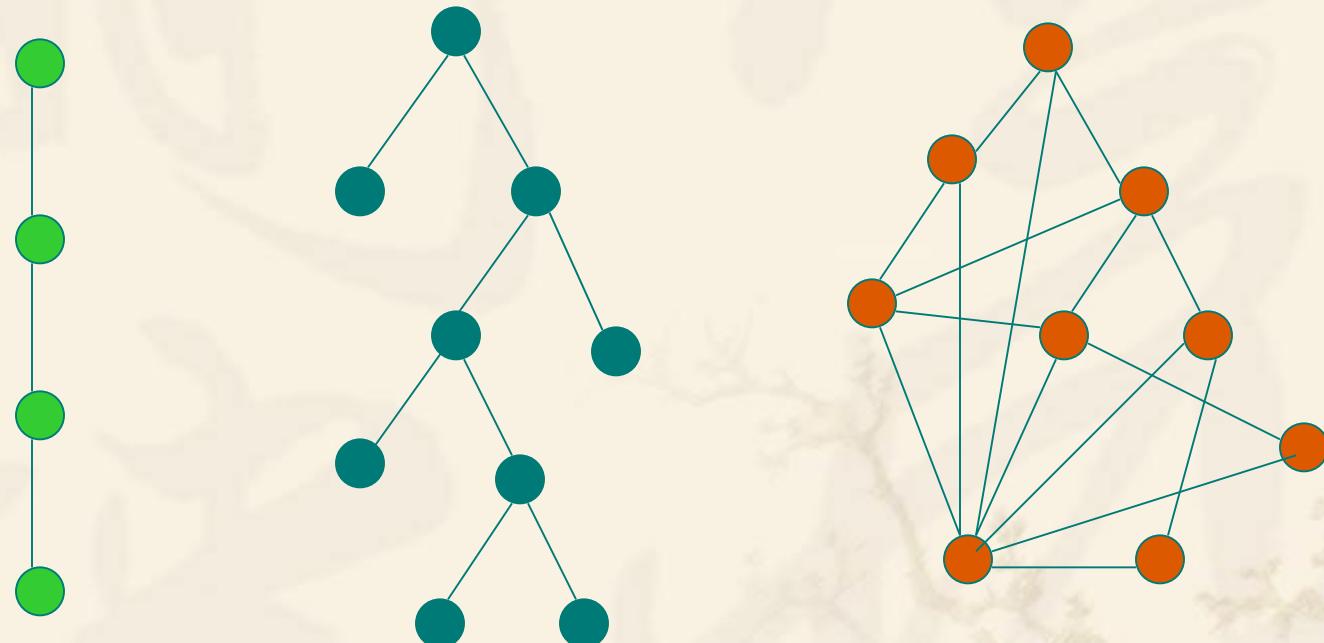


数据结构

华中科技大学
计算机学院



第五章 数组和广义表

引言：

线性表： $L = (a_1, a_2, \dots, a_n)$, a_i 是同类型的元素,
 $1 \leq i \leq n$

数组： $A = (a_1, a_2, \dots, a_n)$

若 a_i 是同类型的元素, A 是一维数组, $1 \leq i \leq n$

若 a_i 是同类型的定长线性表, A 是多维数组,
 $1 \leq i \leq n$

广义表： $L_S = (a_1, a_2, \dots, a_n)$

a_i 可以是同类型的元素或广义表, $1 \leq i \leq n$

5.1 数组的基本概念及其操作

数组是相同类型的数据的有限的、有序的组合。

5.1.1 数组的递归定义

1. 一维数组：

是一个定长线性表 (a_1, a_2, \dots, a_n) 。

其中： a_i 为数据元素， i 为下标/序号， $1 \leq i \leq n$
 (a_1, a_2, \dots, a_n) 又称为向量。

2. 二维数组是一个定长线性表 $(\alpha_1, \alpha_2, \dots, \alpha_m)$ ，
 其中： $\alpha_i = (a_{i1}, a_{i2}, \dots, a_{in})$ 为行向量， $1 \leq i \leq m$
 由 m 个行向量组成，记作：

$$A_{m*n} = \begin{pmatrix} (a_{11} & a_{12} & \dots & a_{1n}) \\ (a_{21} & a_{22} & \dots & a_{2n}) \\ \dots & \dots & \dots & \dots \\ (a_{m1} & a_{m2} & \dots & a_{mn}) \end{pmatrix}$$

即 $A_{m*n} = ((a_{11} a_{12} \dots a_{1n}), (a_{21} a_{22} \dots a_{2n}), \dots, (a_{m1} a_{m2} \dots a_{mn}))$
 或由 n 个列向量组成，记作：

$$A_{m*x} = \left(\begin{array}{ccc|c} \hat{a}_{11} & \hat{a}_{12} & & \hat{a}_{1n} \\ a_{21} & a_{22} & & a_{2n} \\ | & | & & | \\ a_{m1} & a_{m2} & & a_{mn} \end{array} \right)$$

3. 三维数组是一个定长线性表($\beta_1, \beta_2, \dots, \beta_p$)。

其中: $\beta_k = (\alpha_1, \alpha_2, \dots, \alpha_m)$ 为定长二维数组, $1 \leq k \leq p$

例 三维数组 $A[1..3, 1..4, 1..2]$, $p=3$, $m=4$, $n=2$

$$A_{3*4*2} = \begin{array}{c} \left. \begin{array}{cc} a_{111} & a_{112} \\ a_{121} & a_{122} \\ a_{131} & a_{132} \\ a_{141} & a_{142} \end{array} \right\} \text{第1页} \\ \left. \begin{array}{cc} a_{211} & a_{212} \\ a_{221} & a_{222} \\ a_{231} & a_{232} \\ a_{241} & a_{242} \end{array} \right\} \text{第2页} \\ \left. \begin{array}{cc} a_{311} & a_{312} \\ a_{321} & a_{322} \\ a_{331} & a_{332} \\ a_{341} & a_{342} \end{array} \right\} \text{第3页} \end{array}$$

5.1.2 数组的类型定义和变量说明：

例1 int a[10]; //10个整数的一维数组
char b[4][5]; //4行5列个字符的二维数组
float c[3][4][2]; //3*4*2个实数的三维数组

例2 #define m 4 //定义符号常量m
#define n 5 //定义符号常量n
int a[m]; //m个整数的一维数组
char b[m][n]; //m行n列个字符的二维数组

例3 #define m 4 //定义符号常量m
#define n 5 //定义符号常量n
typedef int ara[m]; //一维数组类型ara
typedef char arb[m][n]; //二维数组类型arb
ara a; //ara类型的变量a
arb b; //arb类型的变量b

 C语言中定义静态数组时，元素数目必须是常量

错例1 int m=4, n=5;
int a[m][n]; //m, n是变量

错例2 int p;
scanf ("%d", &p);
int c[p]; //p是变量

5.1.3 数组的操作

1. 生成一个数组： int a[7]; //生成静态一维数组

2. 赋值/修改

a[1]=15; (a[1])++;

3. 取元素的值：

a[0]=a[1]*2;

4. 销毁一个数组

a	32	16						
	0	1	2	3	4	5	6	

5.1.4 程序设计举例

例1 main()

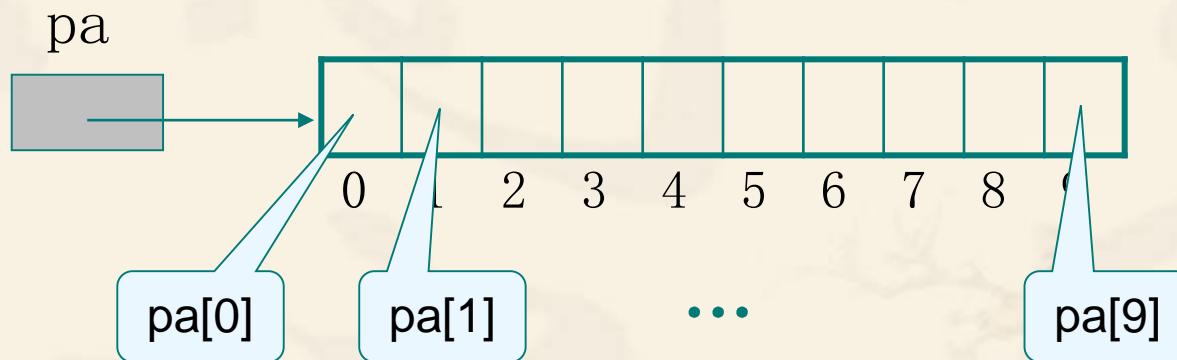
```
{ int i, a[10]; //生成一维数组a  
for (i=0; i<10; i++)  
    scanf ("%d", &a[i]); //输入元素  
for (i=0; i<10; i++)  
    printf ("%d ", a[i]*a[i]); //输出元素的平方
```

}

退出时
释放a

例2 生成动态的10个整数的一维数组

```
int *pa; //指针变量pa  
pa=(int *)malloc(10*sizeof(int)); //动态数组pa
```



```
main()
{
    int i, n, *pa;
    scanf("%d", &n);                                //动态输入n
    pa=(int *)malloc(n*sizeof(int)); //生成动态数组*pa
    for (i=0; i<n; i++)
        *(pa+i)=2*i;                                //指针法引用数组元素, 赋值
    for (i=0; i<n; i++)
        printf("%d, ", *(pa+i)); //输出数组元素0, 2, 4, 6, ...
    for (i=0; i<n; i++)
        scanf("%d", &pa[i]); //下标法引用数组元素, 输入
    for (i=0; i<n; i++)
        printf("%d, ", pa[i]); //输出数组元素
    free(pa); //释放(销毁)数组空间
}
```

5.2 数组的顺序表示和实现

5.2.1 顺序表示(顺序存储结构)

1. 以行序为主序的顺序存储方式

左边的下标后变化，右边的下标先变化

2. 以列序为主序的顺序存储方式

左边的下标先变化，右边的下标后变化

例1 二维数组 $a[1..3, 1..2]$, b是首地址, s是元素所占的单元数

逻辑结构		序号	内存	地址	序号	内存	地址
a11	a12	1	a11	b	1	a11	b
a21	a22	2	a12	b+s	2	a21	b+s
a31	a32	3	a21	b+2*s	3	a31	b+2*s
		4	a22	b+3*s	4	a12	b+3*s
		5	a31	b+4*s	5	a22	b+4*s
		6	a32	b+5*s	6	a32	b+5*s

以行序为主序

以列序为主序

例2 三维数组a[1..2, 1..3, 1..2]

a111 a112	a211 a212
a121 a122	a221 a222
a131 a132	a231 a232

第1页

逻辑结构

序号	内存地址
1	a111 b
2	a112 b+s
3	a121 b+2*s
4	a122 b+3*s
5	a131 b+4*s
6	a132 b+5*s
7	a211 b+6*s
8	a212
9	a221
10	a222
11	a231
12	a232 b+11*s

以行序为主序

序号	内存地址
1	a111 b
2	a211 b+s
3	a121 b+2*s
4	a221 b+3*s
5	a131 b+4*s
6	a231 b+5*s
7	a112 b+6*s
8	a212
9	a122
10	a222
11	a132
12	a232 b+11*s

以列序为主序

5. 2. 2数组的映象函数

数组元素的存储地址

例1 一维数组a[0..n-1]

a0	a1	a2	...	ai	...	a(n-1)	
----	----	----	-----	----	-----	--------	--

下标 0 1 2 i n-1
地址 b b+s b+2*s b+i*s b+(n-1)*s

设:b为首地址, s为每个元素所占的存储单元数
则:元素a[i]的存储地址:

$$\text{Loc}(i) = \text{Loc}(0) + i * s = b + i * s \quad 0 \leq i \leq n-1$$

例2 一维数组a[1..n]

a1	a2	a3	...	ai	...	an	
----	----	----	-----	----	-----	----	--

下标 1 2 3 i n
地址 b b+s b+2s b+(i-1)s b+(n-1)s

元素a[i]的存储地址

$$\text{Loc}(i) = \text{Loc}(1) + (i-1)*s = b + (i-1)*s \quad 1 \leq i \leq n$$

例3 二维数组 $a[1..m, 1..n]$, 假定无零行零列

$$A_{m \times n} = \left(\begin{array}{cccc} a_{11} & \dots & a_{1j} & \dots & a_{1n} \\ \dots & \dots & \dots & \dots & \dots \\ a_{i1} & \dots & a_{ij} & \dots & a_{in} \\ \dots & \dots & \dots & \dots & \dots \\ a_{m1} & \dots & a_{mj} & \dots & a_{mn} \end{array} \right) \quad \left. \begin{array}{l} \text{共 } i-1 \text{ 行} \\ \text{共 } j-1 \text{ 列} \end{array} \right\}$$

(1) 以行序为主序, $a[i][j]$ 的地址为

$$\begin{aligned} \text{Loc}(i, j) &= \text{Loc}(1, 1) + (n * (i-1) + j-1) * s \\ &= b + (n * (i-1) + j-1) * s \quad 1 \leq i \leq m, 1 \leq j \leq n \end{aligned}$$

其中: b为首地址, s为每个元素所占的存储单元数

n:列数 m:行数

例3 二维数组 $a[1..m, 1..n]$, 假定无零行零列

$$A_{m \times n} = \left(\begin{array}{cccccc} a_{11} & a_{12} & \dots & a_{1j} & \dots & a_{1n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{i1} & a_{i2} & \dots & a_{ij} & \dots & a_{in} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mj} & \dots & a_{mn} \end{array} \right) \quad \left. \right\} \text{共 } i-1 \text{ 行}$$

$\underbrace{\qquad\qquad\qquad}_{\text{共 } j-1 \text{ 列}}$

(2) 以列序为主序, $a[i][j]$ 的地址为

$$\begin{aligned} \text{Loc}(i, j) &= \text{Loc}(1, 1) + (m * (j-1) + i-1) * s \\ &= b + (m * (j-1) + i-1) * s \quad 1 \leq i \leq m, 1 \leq j \leq n \end{aligned}$$

其中: b 为首地址, s 为每个元素所占的存储单元数

n : 列数 m : 行数

例4 二维数组a[0..m-1, 0..n-1] (有零行零列)

$$A_{mxn} = \left(\begin{array}{cccc} a_{00} & \dots & a_{0j} & \dots & a_{0n-1} \\ a_{10} & \dots & a_{1j} & \dots & a_{1n-1} \\ \dots & \dots & \dots & \dots & \dots \\ a_{i0} & \dots & a_{ij} & \dots & a_{in-1} \\ \dots & \dots & \dots & \dots & \dots \\ a_{m-10} & \dots & a_{m-1j} & \dots & a_{m-1n-1} \end{array} \right) \quad \left. \begin{array}{l} \text{共 } i \text{ 行} \\ \text{共 } j \text{ 列} \end{array} \right\}$$

(1) 以行序为主序, $a[i][j]$ 的地址为

$$\begin{aligned} \text{Loc}(i, j) &= \text{Loc}(0, 0) + (n*i + j)*s \\ &= b + (n*i + j)*s \quad 0 \leq i \leq m-1, 0 \leq j \leq n-1 \end{aligned}$$

其中: b为首地址, s为每个元素所占的存储单元数

n: 列数 m: 行数

例4 二维数组a[0..m-1, 0..n-1] (有零行零列)

$$A_{mxn} = \left(\begin{array}{cccccc} a_{00} & a_{01} & \dots & a_{0j} & \dots & a_{0n-1} \\ a_{10} & a_{11} & \dots & a_{1j} & \dots & a_{1n-1} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{i0} & a_{i1} & \dots & a_{ij} & \dots & a_{in-1} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{m-10} & a_{m-11} & \dots & a_{m-1j} & \dots & a_{m-1n-1} \end{array} \right) \quad \text{共 } i \text{ 行}$$

共 j 列

(2) 以列序为主序, $a[i][j]$ 的地址为

$$\begin{aligned} \text{Loc}(i, j) &= \text{Loc}(0, 0) + (m*j+i)*s \\ &= b + (m*j+i)*s \quad 0 \leq i \leq m-1, 0 \leq j \leq n-1 \end{aligned}$$

其中: b为首地址, s为每个元素所占的存储单元数

n: 列数 m: 行数

例5 三维数组 $a[1..p, 1..m, 1..n]$, 假定无0页0行0列

1) 以行序为主序, $a[k][i][j]$ 的地址为

$$\begin{aligned} \text{Loc}(k, i, j) &= \text{Loc}(1, 1, 1) + (m*n*(k-1) + n(i-1) + j-1)*s \\ &= b + (m*n*(k-1) + n(i-1) + j-1)*s \\ &\quad 1 \leq k \leq p, \quad 1 \leq i \leq m, \quad 1 \leq j \leq n \end{aligned}$$

其中:

b为首地址, s为每个元素所占的存储单元数

p---页数 n---列数 m---行数

(2) 以列序为主序, $a[k][i][j]$ 的地址为

$$\begin{aligned} \text{Loc}(k, i, j) &= \text{Loc}(1, 1, 1) + (p*m*(j-1) + p*(i-1) + k-1)*s \\ &= b + (p*m*(j-1) + p*(i-1) + k-1)*s \\ &\quad 1 \leq k \leq p, \quad 1 \leq i \leq m, \quad 1 \leq j \leq n \end{aligned}$$

其中:

b为首地址, s为每个元素所占的存储单元数

p---页数 n---列数 m---行数

例5 三维数组 $a[0..p-1, 0..m-1, 0..n-1]$,

(1) 以行序为主序, $a[k][i][j]$ 的地址为

$$\text{Loc}(k, i, j) = \text{Loc}(0, 0, 0) + (m*n*k + n*i + j)*s$$

$$= b + (m*n*k + n*i + j)*s$$

$$0 \leq k \leq p-1, \quad 0 \leq i \leq m-1, \quad 0 \leq j \leq n-1$$

其中:

b为首地址, s为每个元素所占的存储单元数

p---页数 n--列数 m-行数

(2) 以列序为主序, $a[k][i][j]$ 的地址为

$$\text{Loc}(k, i, j) = \text{Loc}(0, 0, 0) + (p*m*j + p*i + k)*s$$

$$= b + (p*m*j + p*i + k)*s$$

$$0 \leq k \leq p-1, \quad 0 \leq i \leq m-1, \quad 0 \leq j \leq n-1$$

其中:

b为首地址, s为每个元素所占的存储单元数

p---页数 n--列数 m-行数

5.3 矩阵的压缩存储

5.3.1 特殊矩阵的压缩存储

1. n阶对称矩阵

$$A_{n \times n} = \begin{cases} & \text{上三角} \\ \left[\begin{array}{cccc} a_{11} & & a_{1n} & \\ & a_{ji} & & \\ & a_{ij} & & \\ a_{n1} & & a_{nn} & \end{array} \right] & \\ & \text{下三角} \end{cases}$$

上三角

下三角

$a_{ij} = a_{ji}$
 $1 \leq i, j \leq n$

假定以行序为主，顺序存储下三角元素到SA[1.. maxlen]

a ₁₁	a ₂₁	a ₂₂	a ₃₁	...	a _{i1}	...	a _{ij}	...	a _{ii}	...	a _{n1}	...	a _{nn}
-----------------	-----------------	-----------------	-----------------	-----	-----------------	-----	-----------------	-----	-----------------	-----	-----------------	-----	-----------------

k=1 2 3 4 ... i(i-1)/2+j ... n(n+1)/2

如何求a_{ij}在SA中的位置，即序号k？

(1) 设a_{ij}在下三角， $i \geq j$

∴ 第1~i-1行共有元素

$$1+2+3+\dots+(i-1)=i(i-1)/2 \text{ (个)}$$

a_{i1}~a_{ij}共有j个元素

∴ a_{ij}的序号为：

$$k=i(i-1)/2+j$$

(2) 设 a_{ij} 在上三角, $i < j$

\because 上三角的 $a_{ij} =$ 下三角的 a_{ji}

下三角的 a_{ji} 的序号为

$$k = j(j-1)/2 + i \quad i < j$$

\therefore 上三角的 a_{ij} 的序号为

$$k = j(j-1)/2 + i \quad i < j$$

由(1)和(2), 任意 a_{ij} 在 SA 中的序号, 为

$$k(i, j) = \begin{cases} i(i-1)/2 + j & i \geq j \\ j(j-1)/2 + i & i < j \end{cases}$$

称为在 SA 中的映象函数, 下标转换公式

2. 三对角矩阵

$$A_{nxn} = \begin{pmatrix} a_{11} & a_{12} & & & & \\ a_{21} & a_{22} & a_{23} & & & \\ & a_{32} & a_{33} & a_{34} & & \\ & \dots & & a_{ij} & \dots & \\ & & & a_{n-1,n-1} & a_{n-1,n} & \\ & & & & a_{nn-1} & a_{nn} \end{pmatrix}$$

全0 全0

假定以行序为主，顺序存储非0元素到SA[1..maxleng]：

a ₁₁	a ₁₂	a ₂₁	a ₂₂	a ₂₃	a ₃₂	...	a _{ij}	...	a _{nn-1}	a _{nn}
-----------------	-----------------	-----------------	-----------------	-----------------	-----------------	-----	-----------------	-----	-------------------	-----------------

k=1 2 3 4 5 6 ... ? ... 3n-2

任意 $a_{ij} \neq 0$, 在 SA 中的序号：

$$k = (3 * (i-1) - 1) + (j-i+2) = 2(i-1) + j$$

5.3.2 稀疏矩阵的压缩存储

1. 三元组表

例 稀疏矩阵M及其转置矩阵T

$$M = \begin{pmatrix} 0 & 13 & 9 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -3 & 0 & 0 & 0 & 0 & 14 & 0 \\ 0 & 0 & 24 & 0 & 0 & 0 & 0 \\ 0 & 18 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -7 & 0 & 0 & 0 \end{pmatrix}$$

$$T = \begin{pmatrix} 0 & 0 & -3 & 0 & 0 & 0 & 0 \\ 13 & 0 & 0 & 0 & 18 & 0 & 0 \\ 9 & 0 & 0 & 24 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 14 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

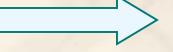
i	j	v=a[i][j]
1	2	13
1	3	9
3	1	-3
3	6	14
4	3	24
5	2	18
6	4	-7

M的三元组表



i	j	v=a[i][j]
1	3	-3
2	1	13
2	5	18
3	1	9
3	4	24
4	6	-7
6	3	14

T的三元组表



问题: 如何由矩阵M求矩阵T, 即由表A求表B?

M的三元表存储结构

1	2	13
1	3	9
3	1	-3
3	6	14
4	3	24
5	2	18
6	4	-7

行数(mu) : 6

列数(nu) : 7

非零元(tu) : 7

用C语言定义三元组表

```
#define MAXSIZE 100
typedef struct {
    int i, j; // 非零元行、列下标
    ElemtType e;
} Triple; // 定义三元组
```

```
typedef struct {
    Triple data[MAXSIZE+1];
    int mu, nu, tu;
} TSMatrix; // 定义三元组表
```

```
TSMatrix M;
```

M的三元表存储结构

1	2	13
1	3	9
3	1	-3
3	6	14
4	3	24
5	2	18
6	4	-7

行数(mu) : 6

列数(nu) : 7

非零元(tu) : 7

T的三元表存储结构

2	1	13
3	1	9
1	3	-3
6	3	14
3	4	24
2	5	18
4	6	-7

行数(mu) : 7

列数(nu) : 6

非零元(nu) : 7

行、列互换

不符合以行为主的存放

M的三元表存储结构

1	2	13
1	3	9
3	1	-3
3	6	14
4	3	24
5	2	18
6	4	-7
/\	/\	/\

行数(mu): 6

列数(nu): 7

非零元(tu): 7

```
T. mu=M. nu; T. nu=M. Mu; T. tu=M. tu;  
if (T. tu) {  
    q=1; /*指示向T写时的位置*/  
    for (col=1; col<=M. nu; ++col)  
        for (p=1; p<=M. tu; ++p) /*扫描M三元表*/  
            if (M. data[p]. j==col) /*当前行*/  
                {T. data[q]. i=M. data[p]. j;  
                 T. data[q]. j=M. data[p]. i;  
                 T. data[q]. e=M. data[p]. e;  
                 q++; }  
    }  
return OK;
```

算法1:
时间复杂度:
 $O(nu*tu)$

col	1	2	3	4	5	6	7
num[col]	1	2	2	1	0	1	0
cptot[col]	1	2	4	6	7	7	8

算法2：
时间复杂度：
 $O(nu+tu)$

$cptot[1]=1;$

$cptot[col]=cptot[col-1]+num[col-1] \quad 2 \leq col \leq a.$ nu

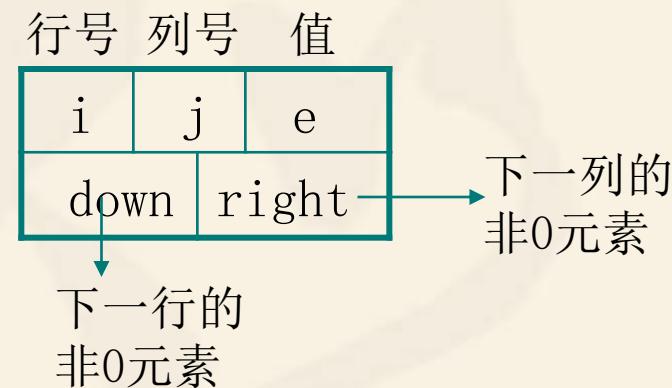
```

for(p=1;p<=M. tu;++p) /*扫描M三元表*/
{col=M. data[p]. j; /*确定当前元素列号*/
 q=cptot[col]; /*确定当前元素M. data[p]
 在T的当前存放位置*/
 T. data[q]. j=M. data[p]. i; T. data[q]. i=M. data[p]. j;
 T. data[q]. e=M. data[p]. e;
 ++cptot[col]; /*指示M中相同列号元素的下一空位置
 */
}

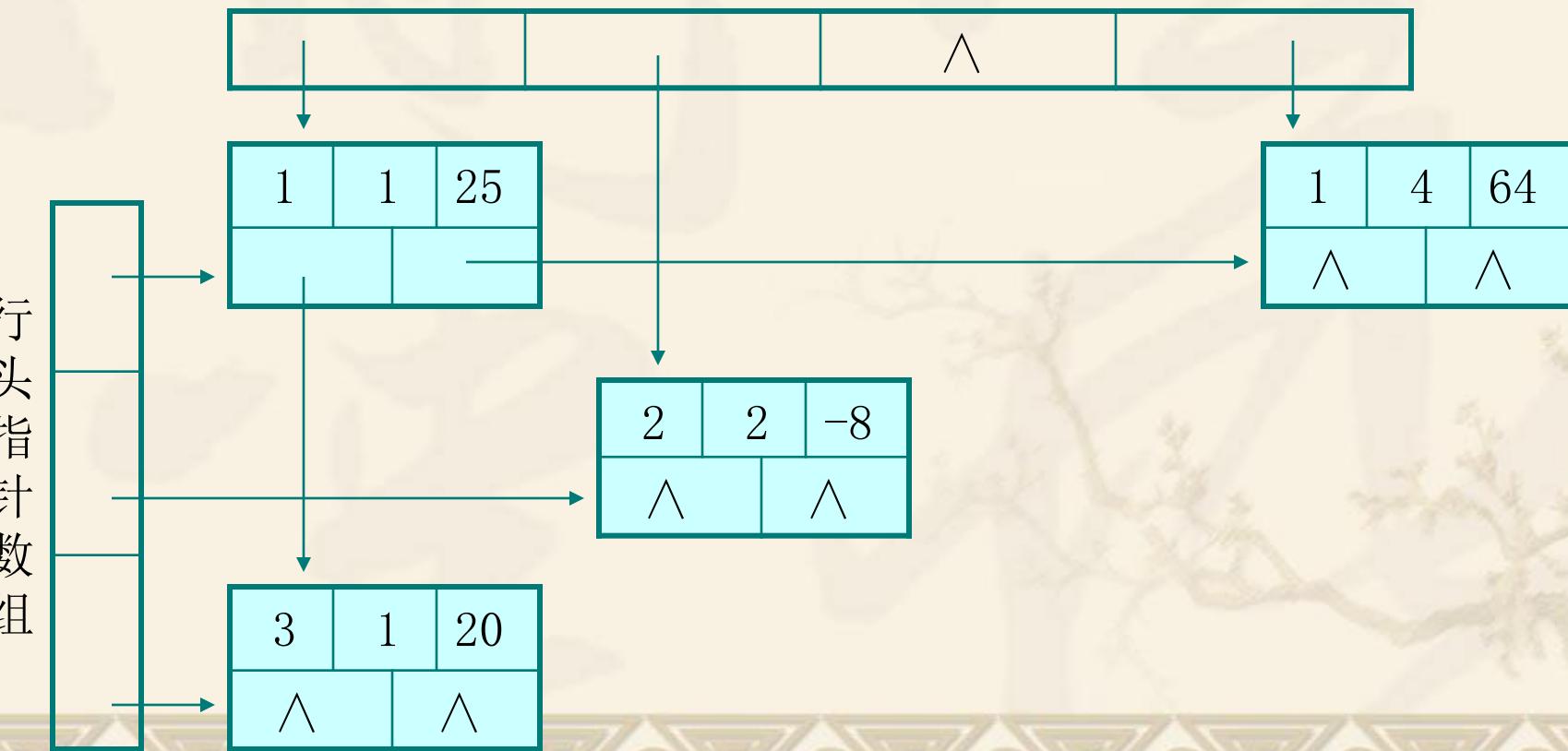
```

2. 十字链接表 例 稀疏矩阵

$$M = \begin{pmatrix} 25 & 0 & 0 & 64 \\ 0 & -8 & 0 & 0 \\ 20 & 0 & 0 & 0 \end{pmatrix}$$



列头指针数组



5.4 广义表(generalized list), 列表(lists)

5.4.1 广义表的定义和术语

$n(n \geq 0)$ 个数据元素或广义表的一个有限序列叫做广义表。

记作: $LS = (e_1, e_2, \dots, e_n)$ 。 n 为 LS 的长度。

其中: LS ——广义表名

e_i ——单元素、原子, 约定用小写, $1 \leq i \leq n$

e_i ——广义表, 约定用大写, $1 \leq i \leq n$

(1) 空表 $LS = ()$, $n=0$

(2) 非空表 $LS = (e_1, e_2, \dots, e_n)$ $n > 0$

其中:

e_1 —— LS 的表头/首部, 记作: $\text{Head}(LS) = e_1$

(e_2, \dots, e_n) —— LS 的表尾/尾部, 记作:

$\text{Tail}(LS) = (e_2, \dots, e_n)$

例：

(1) $A = ()$ // 空表

(2) $B = (e)$

$\text{Head}(B) = e$

$\text{Tail}(B) = ()$

(3) $C = (a, b, c)$

$\text{Head}(C) = a$

$\text{Tail}(C) = (b, c)$

$\text{Head}(\text{Tail}(C)) = b$

$\text{Tail}(\text{Tail}(C)) = (c)$

(4) $D = (a, (b, c))$

$\text{Head}(D) = a$

$\text{Tail}(D) = ((b, c))$

$D2 = ((a, b), c)$

$\text{Head}(D2) = (a, b)$

$\text{Tail}(D2) = (c)$

(5) $E = ((a, b), c, (d, e))$

$\text{Head}(E) = (a, b)$

$\text{Tail}(E) = (c, (d, e))$

$\text{Head}(\text{Tail}(E)) = c$

$\text{Tail}(\text{Tail}(E)) = ((d, e))$

(6) $F = (A, B, C, d) = ((), (e), (a, b, c), d)$

$\text{Head}(F) = ()$

$\text{Tail}(F) = ((e), (a, b, c), d)$

(7) $G = (a, G) \quad // \text{递归广义表}$

$= (a, (a, G)) = (a, (a, (a, G)))$

$= (a, (a, (a, (a, \dots G))))$

$\text{Head}(G) = a$

$\text{Tail}(G) = (G) = ((a, G))$

(8) $H = ((), ((), ()))$

$\text{Head}(H) = ()$

$\text{Tail}(H) = (((), ())))$

5.4.2 广义表的图型表示——树型结构

约定 □——单元素/原子

○——列表，若有表名，附表名

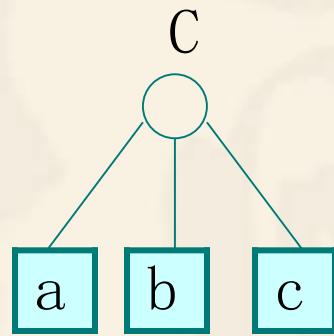
例 (1) $A = ()$



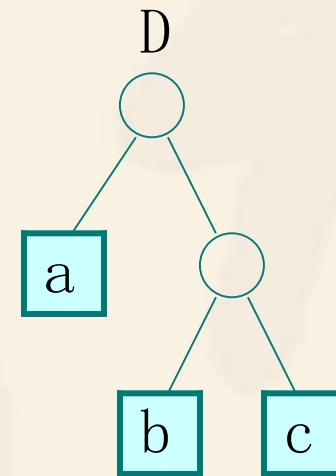
(2) $B = (a)$



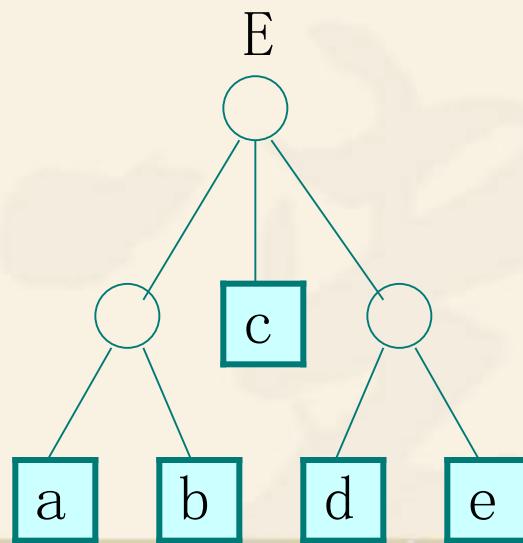
(3) $C = (a, b, c)$



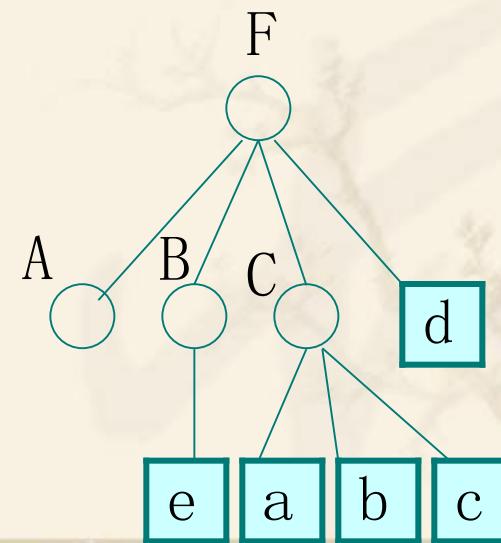
(4) $D = (a, (b, c))$



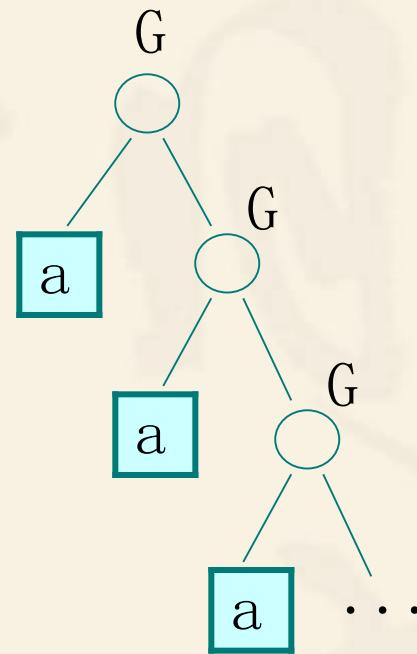
(5) $E = ((a, b), c, (d, e))$



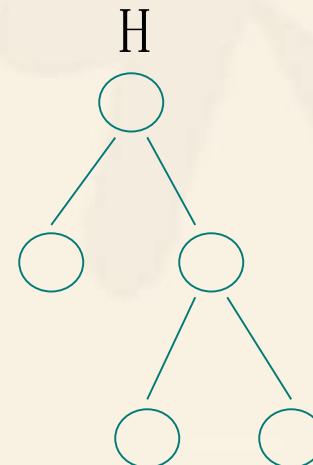
(6) $F = (A, B, C, d) = (((), (e), (a, b, c)), d)$



(7) $G = (a, G)$



(8) $H = ((), ((), ()))$



5.4.3 广义表的操作

1. 求长度: $\text{Leng}(\text{LS})$

$A = ()$

$\text{Leng}(A) = 0$

$G = (a, G)$

$\text{Leng}(G) = 2$

$H = ((), ((), ()))$

$\text{Leng}(H) = 2$

$F = (A, B, C, d)$

$\text{Leng}(F) = 4$

2. 求表头: $\text{Head}(\text{LS})$

$G = (a, G)$

$\text{Head}(G) = a$

$E = ((a, b), c, (d, e))$

$\text{Head}(E) = (a, b)$

3. 求表尾: Tail(LS)

$$G = (a, G)$$

$$\text{Tail}(G) = (G) = ((a, G))$$

$$E = ((a, b), c, (d, e))$$

$$\text{Tail}(E) = (c, (d, e))$$

4. 求第*i*个元素: GetElem(LS, i)=ei $1 \leq i \leq n$

$$I = ((a, b), c, (), (d))$$

$$\text{GetElem}(I, 1) = (a, b)$$

$$\text{Get}(I, 2) = c$$

$$\text{GetElem}(E, 3) = ()$$

$$\text{Get}(I, 4) = (d)$$

5. 求深度: Depth(LS) --- LS所含括号的层数

(1) A=() Depth(A)=1

(2) E=((a, b), c, (d, e)) Depth(E)=2

(3) H=((), (()), ()) Depth(H)=3

6. 插入: InsertFirst(LS, e) --- e插入LS的第一个位置

设 A=()

执行: InsertFirst(A, a);

得: A=(a)

执行: InsertFirst(A, (b, (c)));

得: A=((b, (c)), a)

执行: InsertFirst(A, ());

得: A=((), (b, (c)), a)

7. 其它:

5.5 广义表的存储结构

广义表的元素具有不同结构，一般用链式存储结构。

原子结点：

tag=0	atom(元素)
-------	----------

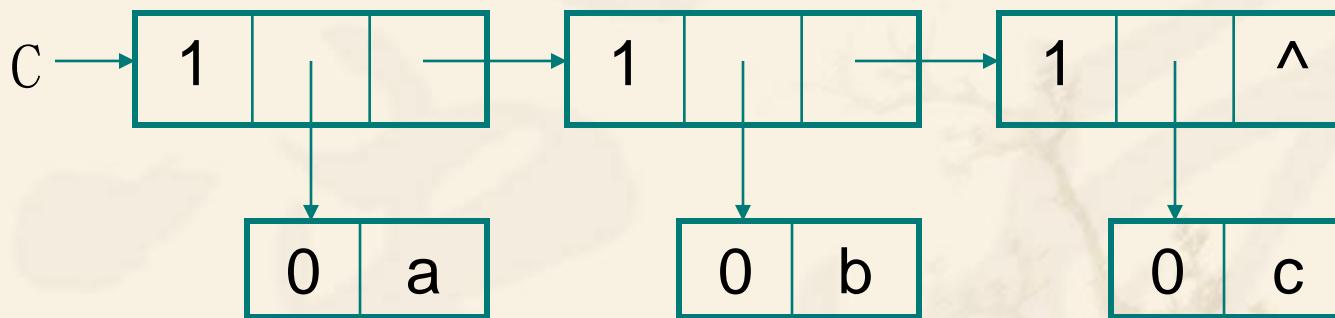
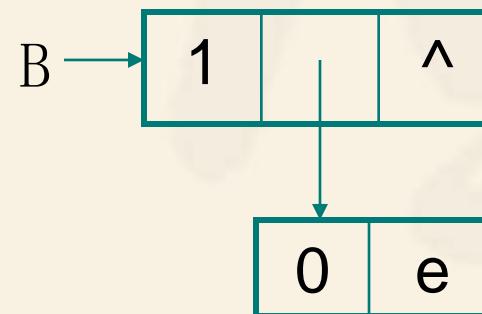
列表结点：

tag=1	hp(表头)	tp(表尾)
-------	--------	--------

```
typedef struct GLNode {  
    ElemTag tag;  
    union { AtomType atom;  
            struct { struct GLNode     *hp, *tp; } ptr;  
        }  
} *GList;
```

- (1) A=()
- (2) B=(e)
- (3) C=(a, b, c)

A=NULL

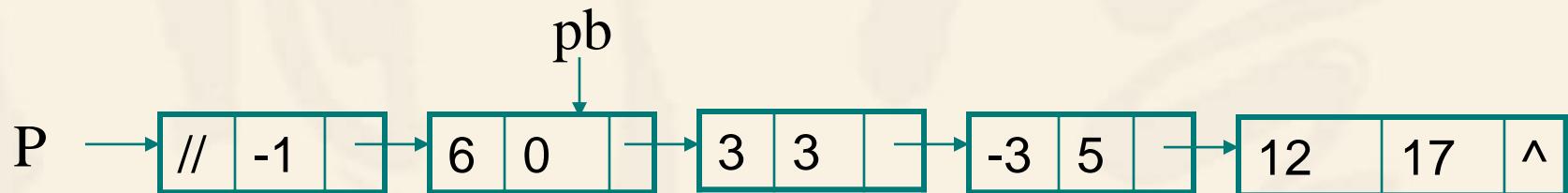


5.6 m元多项式的表示

一元多项式：

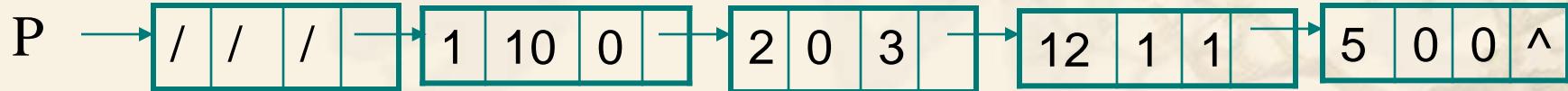
$$P(x) = 6 + 3x^3 - 3x^5 + 12x^{17}$$

coef	expn	next
------	------	------



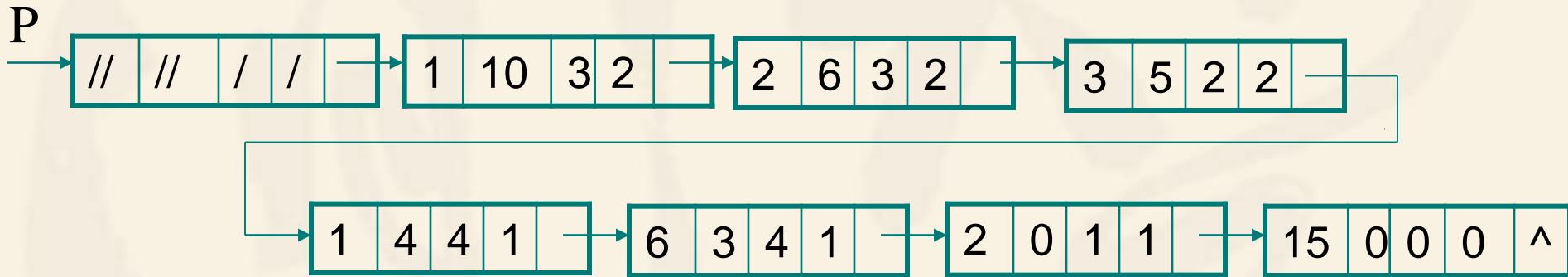
二元多项式：

$$P(x,y) = x^{10} + 2y^3 + 12xy + 5$$



三元多项式：

$$P(x,y,z) = x^{10}y^3z^2 + 2x^6y^3z^2 + 3x^5y^2z^2 + x^4y^4z + 6x^3y^4z + 2yz + 15$$



缺点：

- 若m元多项式无论各项的变元数多少，都按m个指数分配单元，造成空间浪费，若按实际分配，则操作困难；
- m值不同，结点大小不一致，存储管理不变

三元多项式变形：

$$P(x,y,z) = ((x^{10} + 2x^6)y^3 + 3x^5y^2)z^2 + ((x^4 + 6x^3)y^4 + 2y)z + 15$$

$$A(x,y) = (x^{10} + 2x^6)y^3 + 3x^5y^2$$

$$B(x,y) = (x^4 + 6x^3)y^4 + 2y$$

$$P=z((A,2), (B,1) , (15,0))$$

$$C(x) = x^{10} + 2x^6$$

$$D(x) = 3x^5$$

$$E(x) = x^4 + 6x^3$$

$$F(x) = 2$$

$$A=y((C,3), (D,2))$$

$$B=y((E,4) , (F,1))$$

$$C=x((1,10), (2,6))$$

$$D=x((3,5))$$

$$E=x((1,4), (6,3))$$

$$F=x((2,0))$$

原子结点:

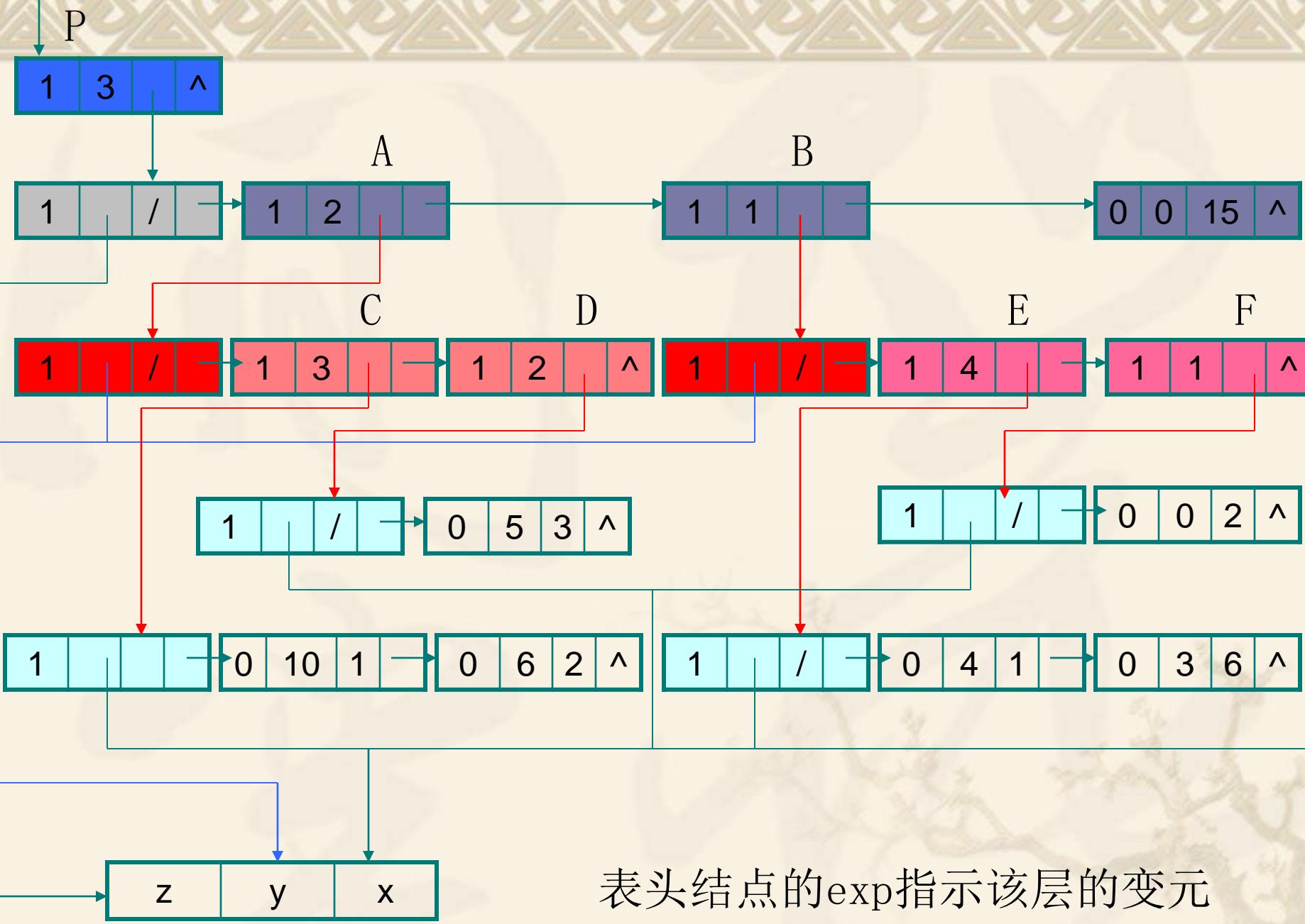
tag=0	exp	coef	tp
-------	-----	------	----

列表结点:

tag=1	exp	hp	tp
-------	-----	----	----

其中: exp为指数域; coef为系数域, tp指向同层下一结点

```
typedef struct MPNode {  
    ElemTag tag;  
    int      exp;           // 指数域  
    union { float coef;     // 系数域  
            struct MPNode *hp; // 表结点的指针域  
        } ;  
    struct MPNode *tp;       // 指向同层下一结点  
} *MPList;
```



课后作业

《数据结构题集》

5. 1