数据结构与程序设计（信息类）

Data Structure and Programming

北航软件学院 林广艳

第 4 章

广义表、矩阵与串

目 录

CONTENTS

第 4 章

广义表、矩阵与串

目 录

CONTENTS

（二）二维数组A[1..m,1..n]

$$A[1..m,1..n] = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ & \cdots & \cdots & & \\ & \cdots & \cdots & & \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{pmatrix}$$

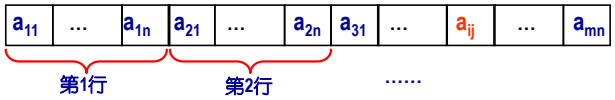
二维数组的存储

行序为主序分配方式

列序为主序分配方式

(1) 行序为主序分配方式

$$A[1..m,1..n] = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & \cdots & a_{2n} \\ & \cdots & \cdots & & & \\ & \cdots & \cdots & & & \\ a_{m1} & a_{m2} & a_{m3} & \cdots & \cdots & a_{mn} \end{pmatrix}$$



特点 前一行最后一个元素的存储位置与后一行的第一个元素的存储位置相邻。

对于二维数组

$$A[1..m,1..n] = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & \cdots & a_{3n} \\ \cdots & \cdots & & & & \\ \cdots & \cdots & & & a_{ij} & \\ \cdots & \cdots & & & & \\ a_{m1} & a_{m2} & a_{m3} & \cdots & \cdots & a_{mn} \end{pmatrix}$$

i-1行

j-1个元素

首地址加上被求元素前面的所有元素占用的单元数

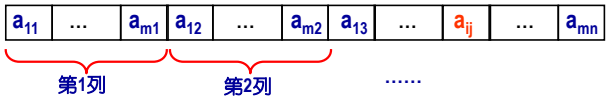
若已知每个元素占k个存储单元，并且第一个元素的存储地址LOC(a_{11})，则

$LOC(a_{ij}) = LOC(a_{11}) + (i-1)\times n\times k + (j-1)\times k$

$= LOC(a_{11}) + [\,(i-1)\times n+(j-1)\,]\times k$

(2) 列序为主序分配方式

$$A[1..m,1..n] = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & \cdots & a_{2n} \\ & \cdots & \cdots & & & \\ & \cdots & \cdots & & & \\ a_{m1} & a_{m2} & a_{m3} & \cdots & \cdots & a_{mn} \end{pmatrix}$$



特点 前一列最后一个元素的存储位置与后一列的第一个元素的存储位置相邻。

$$A[1..m,1..n] = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & \cdots & a_{3n} \\ \cdots & \cdots & & & & \\ \cdots & \cdots & & & a_{ij} & \\ \cdots & \cdots & & & & \\ a_{m1} & a_{m2} & a_{m3} & \cdots & \cdots & a_{mn} \end{pmatrix}$$

i-1个元素

j-1 列

若已知每个元素占k个存储单元，并且第一个元素的存储地址LOC(a_{11})，则

$LOC(a_{ij}) = LOC(a_{11}) + (j-1)\times m\times k + (i-1)\times k$

$= LOC(a_{11}) + [\,(j-1)\times m+(i-1)\,]\times k$

特殊矩阵的压缩存储

目的：节省存储空间

所谓 **压缩存储** 是指为多个值相同的元素，或者位置分布有规律的元素分配尽可能少的存储空间，而对0元素一般情况下不分配存储空间。

$$A=\begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ & \cdots & \cdots & & \\ & \cdots & \cdots & & \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{pmatrix} \Rightarrow A[0..m-1][0..n-1]$$

传统做法

(一)对称矩阵的压缩存储

一个n阶矩阵A的元素满足性质

$a_{ij} = a_{ji}$　　$1 \le i, j \le n$

则称矩阵A为n阶**对称矩阵**。

$$A=\begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ & \cdots & \cdots & & \\ & \cdots & \cdots & & \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{pmatrix}$$

传统做法

定义一个二维数组A[0..n-1][0..n-1]

$$A=\begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ & \cdots & \cdots & & \\ & \cdots & \cdots & & \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{pmatrix}$$

$a_{ij} = a_{ji}$

LTA[0..n(n+1)/2-1]

a_{11}	a_{21}	a_{22}	…	a_{ij}	…	a_{nn}
k = 0	1	2	… …			n(n+1)/2-1

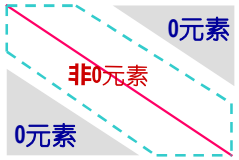
A中任意一元素a_{ij}与LTA[k]之间存在对应关系：

$$k=\begin{cases} i\times(i-1)/2+j-1 & \text{当} i \ge j \text{时} \\ j\times(j-1)/2+i-1 & \text{当} i < j \text{时} \end{cases}$$

(二)对角矩阵的压缩存储

若一个矩阵中，值非0的元素对称地集中在主对角线两旁的一个带状区域中(该区域之外的元素都为0元素)，称这样的矩阵为**对角矩阵**。

B =

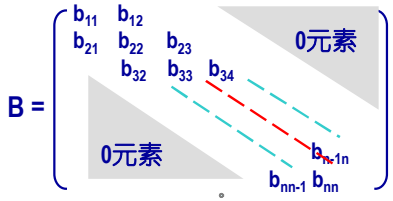


n×n

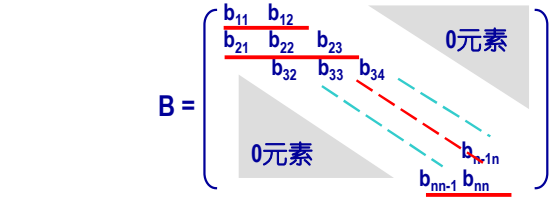
传统做法

定义一个二维数组B[0..n-1][0..n-1]

例. 三对角矩阵的压缩存储



有多少个非零元素？ *有3n−2个非零元素*



LTB[0..3n−3]

B中任一非零元素 b_{ij} 与LTB[K]之间存在对应关系

$k = 2 \times i + j - 3$

《计算机与数字工程》 2001年06期， 杨文茂 刘明杰

主对角线两边非对称分布的带状稀疏矩阵的压缩存储通用寻址公式

稀疏矩阵的三元组表表示

（一）什么是稀疏矩阵?

一个较大的矩阵中，零元素的个数相对于整个矩阵元素的总个数所占比例较大时，可以称该矩阵为一个**稀疏矩阵**。

$$A = \begin{pmatrix} 15 & 0 & 0 & 22 & 0 & -15 \\ 0 & 11 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & -6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 91 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 28 & 0 & 0 & 0 \end{pmatrix}$$

传统做法

定义一个二维数组B[0..5][0..5]

（二）稀疏矩阵的三元组表表示

三元组 (i, j, value)

$$\begin{pmatrix} 15 & 0 & 0 & 22 & 0 & -15 \\ 0 & 11 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & -6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 91 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 28 & 0 & 0 & 0 \end{pmatrix}$$

例如

(1,1,15) 表示第1行、第1列、值为15的元素;

(1,4,22) 表示第1行、第4列、值为22的元素;

(1,6,−15) 表示第1行、第6列、值为−15的元素;

(2,2,11) (2,3,3) (3,4,−6)

(5,1,91) (6,3,28)

一个特殊的三元组

（m，n，t）其中，m, n, t 分别表示稀疏矩阵的总的行数、总的列数与非零元素的总个数。

三元组表存储方法：

若一个m×n阶稀疏矩阵具有t个非零元素，则用t+1个三元组来存储，其中第一个三元组分别用来给出稀疏矩阵的总行数m、总列数n以及非零元素的总个数t；第二个三元组到第t+1个三元组按行序为主序的方式依次存储t个非零元素。

例

$$A=\begin{pmatrix}15&0&0&22&0&-15\\0&11&3&0&0&0\\0&0&0&-6&0&0\\0&0&0&0&0&0\\91&0&0&0&0&0\\0&0&28&0&0&0\end{pmatrix}$$

A[0..5][0..5]

传统做法

三元组表

LTMB=

0	6	6	8
1	1	1	15
2	1	4	22
3	1	6	-15
4	2	2	11
5	2	3	3
6	3	4	-6
7	5	1	91
8	6	3	28

LTMB[0..8][0..2]

三元组表法

```
#define maxSize   12500
```

非零元素结构：

```
typedef struct{
      int    i, j;
    ElemType  e;
}Triple;
```

```
typedef  struct{
   int    mu,nu,tu; //矩阵的行数、列数和非零元个数
   Triple   data[maxSize]; //非零元三元组表
}TSMatrix ;
```

LTMB=

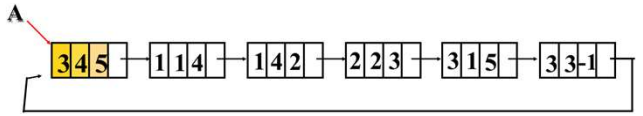
6	6	8
1	1	15
1	4	22
1	6	-15
2	2	11
2	3	3
3	4	-6
5	1	91
6	3	28

稀疏矩阵的　　链表表示

例如，如下一个稀疏矩阵：

$$A=\begin{pmatrix}4&0&0&2\\0&3&0&0\\5&0&-1&0\end{pmatrix}$$

若以行序为主序依次将所有非零元素链接起来，则可以得到如下所示的一个带头结点的循环链表：



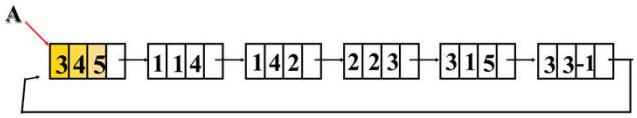
稀疏矩阵的 十字 链表表示

链结点类型：

```
typedef struct SPMnode{
    int   row,col;
    ElemType   value;
    struct  SPMnode   *link;
}SPMnode;
```

表头结点类型：

```
typedef struct SPMatrix{
   int m,n,t;
   SPMnode   *link;
}SPMatrix;
```

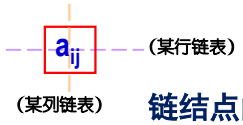


缺点 要确定一个元素比较麻烦，导致相关操作效率低

$$A=\begin{pmatrix}4&0&0&2\\0&3&0&0\\5&0&-1&0\end{pmatrix}$$

十字链表

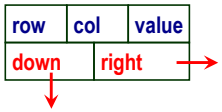
为稀疏矩阵的每一行设置一个单独的循环链表，同样为每一列设置一个单独的循环链表。矩阵中每一个非零元素同时包含在两个循环链表中，即包含在它所在的行链表与所在的列链表中，即两个链表的交汇处。



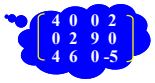
链结点的结构会长成什么样？

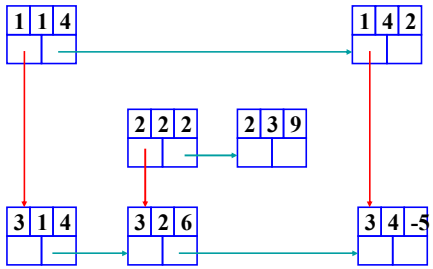
对于一个m×n的稀疏矩阵，分别建立m个行的循环链表与n个列的循环链表，每个非零元素用一个链结点存储。

链结点的构造为：



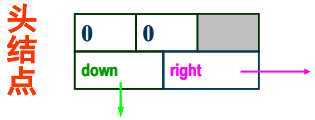
其中，row, col, value 分别表示某个非零元素所在的行号、列号和元素的值；down 和 right 分别为向下与向右指针，分别用来链接同一列中的与同一行中的所有非零元素对应的链结点。

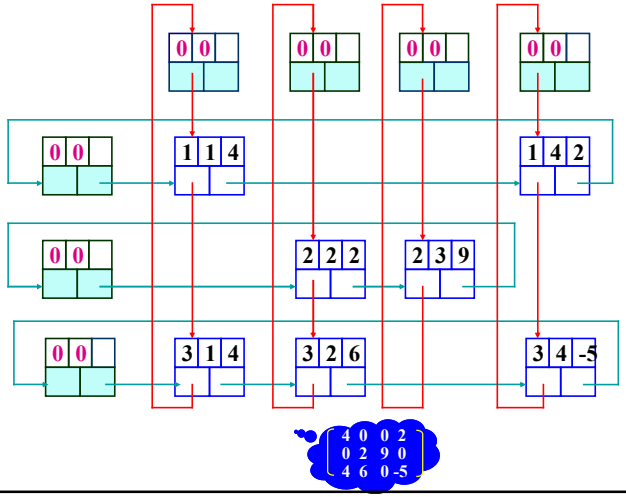


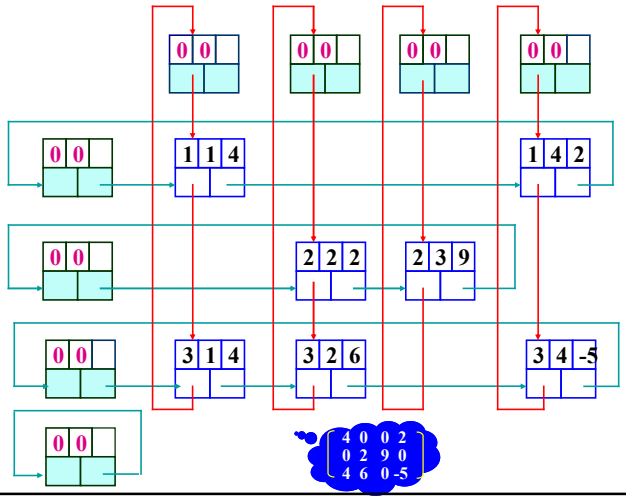


对m个行链表，分别设置m个行链表**表头结点**。 表头结点的构造与链表中其他链结点一样， 只是令row与col 的值分别为0，right域指向相应行链表的第一个结链点. 同理, 对n个列链表，分别设置n个列链表表头结点指向相应列链表的第一个结链点。

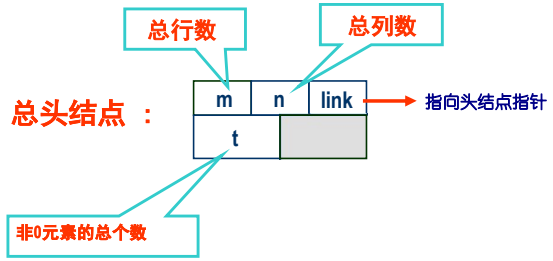
一共设置**MAX(m, n)**个行列头结点.







再设置一个**总头结点**(如下所示)，通过Value(即下图的Link)域把所有头结点也链接成一个循环链表。

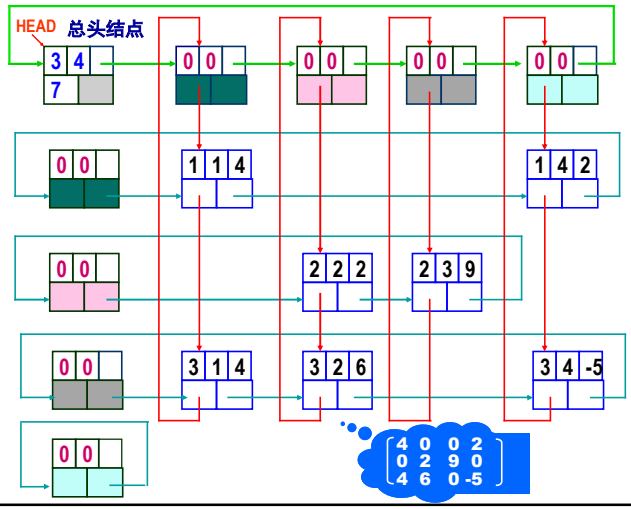


例

对于如下稀疏矩阵B，

$$B = \begin{pmatrix} 4 & 0 & 0 & 2 \\ 0 & 2 & 9 & 0 \\ 4 & 6 & 0 & -5 \end{pmatrix}$$

十字链表表示如下：



思考题

要编写一款可供第三方独立使用的矩阵运算软件，应该从哪下手？需要提供哪几类文件？

(1) 筛选拟支持的矩阵运算的种类：如有相同行数和列数的矩阵间的加法、减法；符合矩阵乘法规则要求的矩阵间的乘法；方阵间的除法；方阵的求逆；矩阵的求转置矩阵等基本功能。

对称正定矩阵分解与行列式求值；矩阵的三角分解；

实数矩阵的奇异值分解等高级功能。

(2) 采用适当的数据结构实现上述多种运算程序。

(3) 提供.h文件，.c或.cpp或.lib或.dll文件，软件工程要求的各类文档，至少应有用户手册或帮助文件。

数组的应用举例

(一) 一元n阶多项式的数组表示

一个标准的一元n阶多项式的各项若按降幂排列，可以表示为如下形式：

$A_n(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0 \quad (a_n \neq 0)$

例如

一个标准的6阶多项式

$A(x)=2x^6-4x^5+10x^4-7x^3+6x^2-4x+1$

$B(x)=2x^6-4x^5+6x^2+1$

$C(x)=x^{2000}-5$

非标准多项式

方法一

定义一个一维数组A[0..n+1]来存储多项式，其中，
A[0] 用来存放多项式的阶数n;
A[1]~A[n+1] 依次用来存放多项式的n+1项的系数。

对于多项式 $A(x)=10x^6-8x^5+3x^2-1$, 有

A[0..7]

6	10	−8	0	0	3	0	−1
0	1	2	3	4	5	6	7

对于多项式 $B(x)=x^{2000}-5$, 有

B[0..2001]

2000	1	0	0	… …	0	−5

1999项为0

方法二

定义一个一维数组A[0..2m]来存储多项式，其中，
A[0] 存放系数非零项的总项数m;
A[1]~A[2m] 依次存放系数非零项各项的系数与指数偶对（一共m个这样的偶对）。

对于多项式 $A(x)=10x^6-8x^5+3x^2-1$, 有

A[0..8]

4	10	6	−8	5	3	2	−1	0

第1项 第2项 第3项 第4项

对于多项式 $C(x)=x^{2000}-5$, 有

C[0..4]

2	1	2000	−5	0

第1项 第2项

对于那些**标准的**或者**基本标准的**多项式，宜采用方法1。

缺项很少

例

$A(x)=10x^6-8x^5+2x^4-7x^3+3x^2+x-1$

$B(x)=10x^6-2x^4-7x^3+3x^2-1$

对于那些**缺项很多的**多项式，宜采用方法2。

例

$C(x)=x^{2000}-5$

（二）n 阶“魔方”（n为任意奇数）

游戏

将1~9不重复地填在3行3列的9个方格中，分别使得每一行、每一列、两个对角线上的元素之和都等于15。

3阶魔方

6	1	8
7	5	3
2	9	4

一个3 阶魔方

15	8	1	24	17
16	14	7	5	23
22	20	13	6	4
3	21	19	12	10
9	2	25	18	11

……

一个5 阶魔方

规律

1. 将用做“魔方”的二维数组的所有元素清0；
2. 第一个数填在第一行居中的位置上(i=0，j=n/2)；
3. 以后每填一个数后，将位置移到当前位置(i,j)的左上角，即做动作i=i−1，j=j−1；
4. 根据不同情况对位置进行修正：
   （1）若位置(i,j)上已经填数，或者i，j同时小于0，将位置修改为i=i+2，j=j+1；
   （2）若i小于0，但j不小于0，修改i为n−1；
   （3）若j小于0，但i不小于0，修改j为n−1。

重复(循环)

以n=5阶魔方为例

分别以i和j表示行与列的位置

(i, j)	(i, j)	(i, j)	(i, j)	(i, j)	
(i, j)	15	8	1	24	17
(i, j)	16	14	7	5	23
(i, j)	22	20	13	6	4
(i, j)	3	21	19	12	10
(i, j)	9	2	25	18	11

0 1 4

1 ~ 25

初始 i=0, j=⌊n/2⌋

i--; j--;

i+=2; j++;

i	j
0	2
4	1
3	0
4	4
3	3
4	2
3	1
2	0
4	4
3	3
2	2

i	j
1	1
0	0
1	4
2	3
1	2
0	1
4	4
3	3
2	2
1	1
0	0
4	4
3	3
2	2

一个5阶魔方

算法

```
void  magic(int a[], int n ) {
   int  i, j, num;
    for(i=0; i<n; i++)
        for(j=0; j<n; j++)
            a[i][j]=0;              /* 魔方清0 */
    i=0;
    j=n/2;                          /* 确定i与j的初始位置 */
    for(num=1; num<=n*n; num++){
        if(i<0 && j<0 || a[i][j]!=0){
            i+=2;
            j++;
        }
        a[i--][j--]=num;     /* 填数，并且左上移一个位置 */
        if(i<0 && j>=0)
            i=n-1;                  /* 修正i的位置 */
        if(j<0 && i>=0)
            j=n-1;                  /* 修正j的位置 */
    }
}
```

延伸阅读*

建议有兴趣的同学自己看看有关基于稀疏矩阵的矩阵加法和乘法等运算。

数组

- **数组的基本概念**
  - 数组的定义
  - 数组的基本操作
- **数组的存储方法**
  - 一维数组的存储
  - 二维数组的存储
    - 行序为主序、列序为主序方式(地址计算公式)
- **特殊矩阵的压缩存储**
  - 对称矩阵、(三)对角矩阵的压缩存储
  - 稀疏矩阵的压缩存储
    - 三元组表表示、十字链表表示
- **数组的应用举例**

本节内容不作为重点要求

第 4 章

广义表、矩阵与串

目录

CONTENTS

本节内容

广义表的基本概念

广义表的存储结构

多元多项式的广义表表示

广义表的概念

一、广义表的定义

一个长度为n≥0 的广义表是一个数据结构

$LS = (a_1, a_2, \ldots\ldots, a_{n-1}, a_n)$

其中，LS为广义表的名字，a_i为表中元素；a_i可以是原子元素，也可以是一个子表。n为表的长度，长度为0的表称为空表。

若a_i为不可再分割的具体信息，则称a_i为**原子元素**；
若a_i为一个子表，则称a_i为**表元素**。这里，用小写字母表示原子元素，用大写字母表示表元素。

$(a_1, a_2, a_3, \ldots\ldots, a_{n-1}, a_n)$

二、广义表的例子

A=( ) ——长度为0的空表。
B=(a) ——长度为1,且只有一个原子元素的广义表。
C=(a, (b,c)) ——长度为2的广义表。
D=(A, B, c) } ——长度为 3 的广义表。
E=(A, B, C) }
F=(a, F) ——长度为2 的递归的广义表。
……

F=(a, F)=(a, (a, (a, ...)))

(1)广义表是多层结构的
(2)广义表可为其他广义表所共享
(3)广义表可以是嵌套的

广义表的深度
----括号嵌套的重数。

二、广义表的例子

广义表的深度 ----括号嵌套的重数

A1 = (A) = (())是空表吗? 长度是 1 ,深度是 2

练习

A = ((a)) B = (a,(b,c,d),e,())
C = (x,((y),B,A))

问: C的长度 C的深度

广义表的存储结构

顺序存储? 还是链式存储?

LS = (a_1, a_2, a_3, ……, a_{n-1}, a_n)

广义表一般采用链式存储结构，链结点的构造可以为

flag	info	link

其中，flag为标志位，令

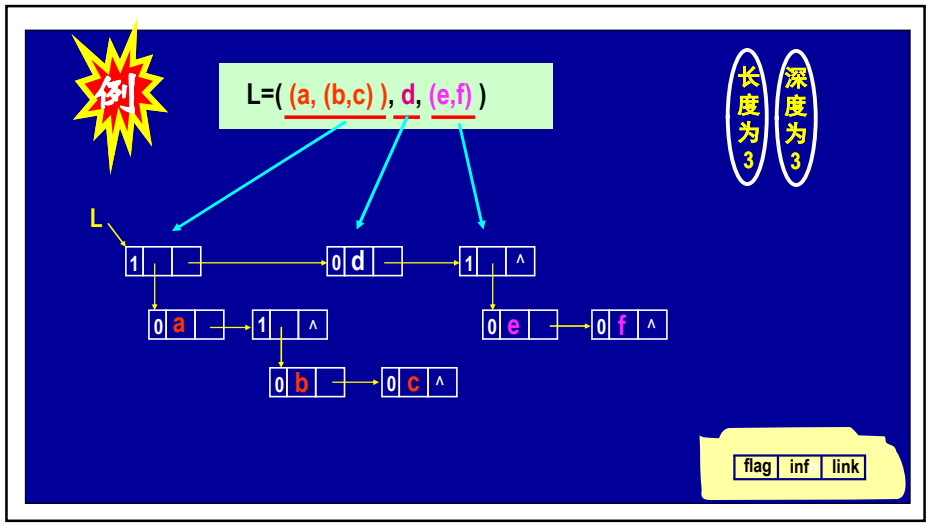
$$flag = \begin{cases} 1 & \text{表示本结点为表结点} \\ 0 & \text{表示本结点为原子结点} \end{cases}$$

当flag=0时，info域存放相应原子元素的信息；
当flag=1时，info域存放子表第一个元素对应的链结点的地址；
link域存放元素同一层的下一个元素所在链结点的 地址，当本元素为所在层的最后一个元素时，link域为NULL。

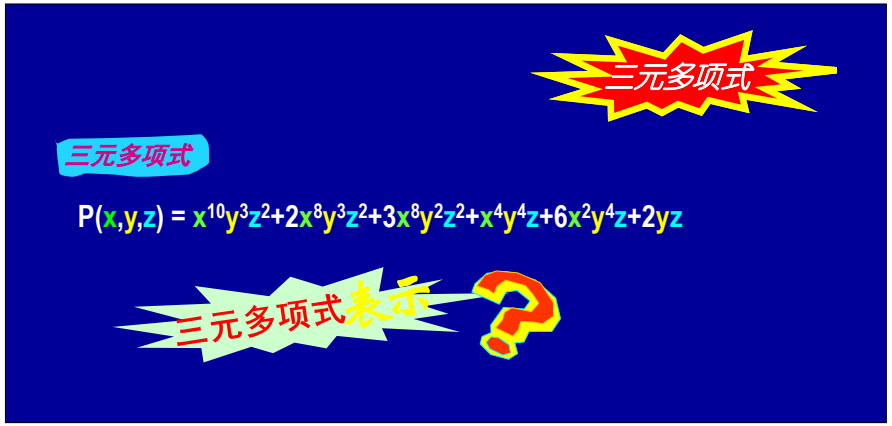
类型定义

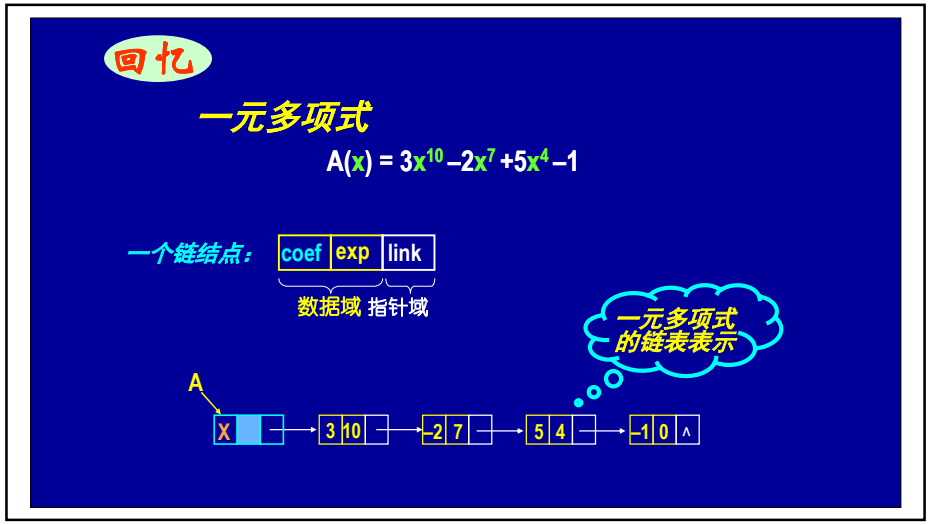
```
typedef  struct node{
    int  flag;
    union{
        DataType data;
        struct node *pointer;
    } info;
    struct node *link;
}BSNode, *BSLinkList;
```

flag	info	link



5.3 多元多项式的广义表表示





方法2

链结点的构造设计为

若该项的系数为关于其他变量的多项式时，此域存放指向该多项式的指针。

其中，coef 表示多项式的某一项的**系数**，
exp 表示多项式的某一项的指数，
link 为链接多项式中同一层各链结点的指针。

例 三元多项式

$$P(x,y,z) = x^{10}y^3z^2+2x^8y^3z^2+3x^8y^2z^2+x^4y^4z+6x^2y^4z+2yz$$
$$= ((x^{10}+2x^8)y^3+3x^8y^2)z^2+((x^4+6x^2)y^4+2y)z$$

$P(z) = Az^2+Bz$

其中：

$A(x,y) = (x^{10}+2x^8)y^3+3x^8y^2$　　$B(x,y) = (x^4+6x^2)y^4+2y$

$A(y) = Cy^3+Dy^2$　　$B(y) = Ey^4+Fy$

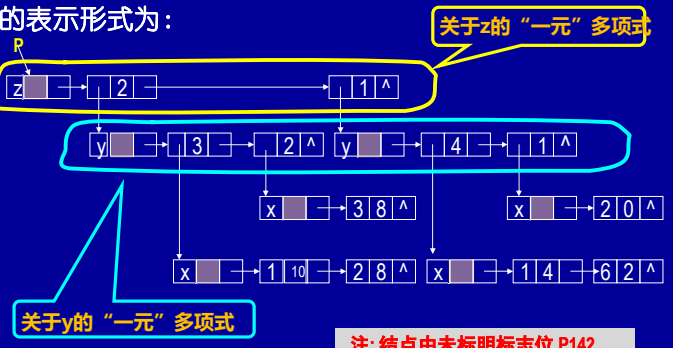
$C(x) = x^{10}+2x^8$　　$E(x) = x^4+6x^2$

$D(x) = 3x^8$　　$F(x) = 2x0$

一元多项式

$$P(x,y,z)=x^{10}y^3z^2+2x^8y^3z^2+3x^8y^2z^2+x^4y^4z+6x^2y^4z+2yz$$
$$= ((x^{10}+2x^8)y^3+3x^8y^2)z^2+((x^4+6x^2)y^4+2y)z$$

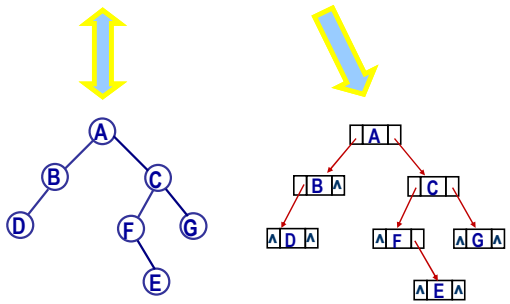
广义表的表示形式为：



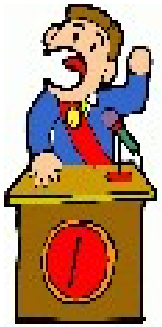
注：结点中未标明标志位，P142

后续章节中的二叉树的广义表表示法

A(B(D), C(F(, E), G)) @



本节内容小结



- 广义表
  - 广义表的基本概念
    - 广义表的定义
    - 基本的名词概念
  - 广义表的存储方法
    - 链表表示法
  - 三元多项式的广义表表示

本节内容不作为重点