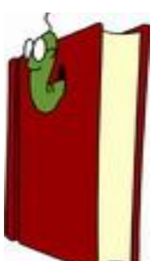
第四章 数组

- 数组的定义和特点
- 数组的顺序存储结构
- 矩阵的压缩存储



本章小结　　课后作业

数组

- 数组可以看作是一种特殊的线性表，即线性表数据元素本身又是一个线性表。
- 数组的定义和特点
  - 定义

$$A_{m\times n} = \begin{bmatrix} (a_{11} & a_{12} & \cdots & \cdots & a_{1n}) \\ (a_{21} & a_{22} & \cdots & \cdots & a_{2n}) \\ (\cdots & \cdots & \cdots & \cdots & \cdots) \\ (a_{m1} & a_{m2} & \cdots & \cdots & a_{mn}) \end{bmatrix}$$

  - 数组特点
    - ✓ 数组结构固定，数组行列数不可变
    - ✓ 数据元素同构
  - 数组运算
    - ✓ 给定一组下标，存取/修改相应的数据元素
    - ✓ 一般不进行插入/删除操作

Back

数组的顺序存储结构

可随机存取任一元素

次序约定——将多维数组以某种次序存储在一维内存中

以*行序*为主序: **BASIC、PASCAL、C**

0	a_{11}
1	a_{12}
	…
n-1	a_{1n}
n	a_{21}
	a_{22}
	…
	a_{2n}
	…
	a_{m1}
	a_{m2}
	…
m*n-1	a_{mn}

$$Loc(a_{ij})=Loc(a_{11})+[(i-1)n+(j-1)]*L$$

以*列序*为主序: **FORTRAN**

0	a_{11}
1	a_{21}
	…
m-1	a_{m1}
m	a_{12}
	a_{22}
	…
	a_{m2}
	…
	a_{1n}
	a_{2n}
	…
m*n-1	a_{mn}

$$Loc(a_{ij})=Loc(a_{11})+[(j-1)m+(i-1)]*L$$

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ & \cdots\cdots & & \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

Back

矩阵的压缩存储

- 在数值分析中，经常出现一些阶数很高的矩阵，同时在矩阵中有很多值相同的元素或零元素，可对这些矩阵进行**压缩存储**。
- 压缩存储的基本思想——
  - 值相同的元素分配一个存储空间；
  - 零元素不分配空间
- **压缩存储**后，若能得到元素a_{ij}的存储地址，则仍具有**随机存取**功能。

特殊矩阵
- 对称矩阵
- 三角矩阵
- 对角矩阵

值相同的元素或零元素分布有规律

稀疏矩阵

非零元少且分布无规律

特殊矩阵压缩存储后，是否还具有随机存取功能？

Back

矩阵的压缩存储

特殊矩阵

对称矩阵

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ & \cdots\cdots & & \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

$$a_{ij} = a_{ji}$$

按行序为主序

a_{11}
a_{21}
a_{22}
a_{31}
a_{32}
a_{33}
……
a_{n1}
……
a_{nn}

压缩存储元素个数：$1+2+\ldots+n=\dfrac{n(n+1)}{2}$

$$Loc(a_{ij})=Loc(a_{11})+\left[\frac{i(i-1)}{2}+(j-1)\right]*L\text{（}i\geq j\text{）}$$

$$Loc(a_{ij})=Loc(a_{11})+\left[\left(\frac{j(j-1)}{2}+(i-1)\right]*L\text{（}i<j\text{）}$$

压缩存储

矩阵的压缩存储

特殊矩阵

三角矩阵

$$\begin{bmatrix} a_{11} & 0 & 0 & \dots & 0 \\ a_{21} & a_{22} & 0 & \dots & 0 \\ & \dots\dots & & & 0 \\ a_{n1} & a_{n2} & \dots & & a_{nn} \end{bmatrix}$$

压缩存储元素个数： $1+2+\dots+n=\dfrac{n(n+1)}{2}$

按行序为主序

a_{11}
a_{21}
a_{22}
a_{31}
a_{32}
a_{33}
……
a_{n1}
……
a_{nn}

$$Loc(a_{ij})=Loc(a_{11})+\left[\frac{i(i-1)}{2}+(j-1)\right]*L \quad （i\geq j）$$

$$a_{ij}=0 \qquad （i<j）$$

压缩存储

矩阵的压缩存储

特殊矩阵

对角矩阵

$$\begin{bmatrix} a_{11} & a_{12} & 0 & 0 & \cdots & 0 \\ a_{21} & a_{22} & a_{23} & 0 & \cdots & 0 \\ 0 & a_{32} & a_{33} & a_{34} & 0 \quad \cdots & 0 \\ & & \cdots\cdots & & & \\ 0 & 0 \ \cdots & a_{n-1,n-2} & a_{n-1,n-1} & & a_{n-1,n} \\ 0 & 0 & \cdots & a_{n,n-1} & & a_{nn} \end{bmatrix}$$

按行序为主序

a_{11}
a_{12}
a_{21}
a_{22}
a_{23}
a_{32}
a_{33}
a_{34}
……
$a_{n\ n-1}$
a_{nn}

压缩存储元素个数：$(n-2)\times 3+4=3n-2$

$Loc(a_{ij})=Loc(a_{11})+[2(i-1)+(j-1)]*L$ （$|i-j|\le 1$）

$a_{ij}=0$　　其它

压缩存储

矩阵的压缩存储

稀疏矩阵

稀疏矩阵的定义

稀疏矩阵的顺序压缩存储

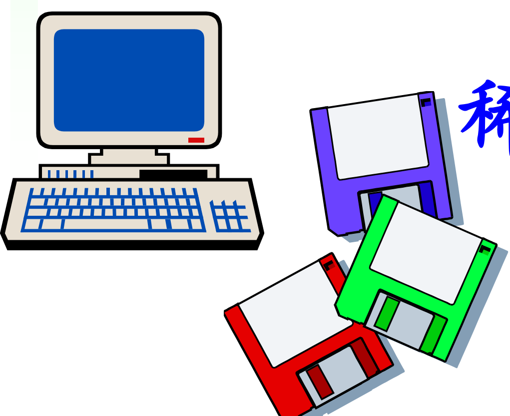
顺序压缩存储后，稀疏矩阵是否还能随机存取？

- 三元组表
- 行逻辑链接的顺序表
- 伪地址表示法

顺序压缩存储后，稀疏矩阵**失去了随机存取功能**

稀疏矩阵的转置

- 按原矩阵M的列序转置
- 快速转置



稀疏矩阵的链式压缩存储

- 带行指针向量的单链表表示
- 十字链表

Back

矩阵的压缩存储

稀疏矩阵

- 定义：非零元较零元少，且分布没有一定规律的矩阵。

  假设 **m** 行 **n** 列的矩阵含 **t** 个非零元素 $\delta = \dfrac{t}{m \times n}$

  通常认为 $\delta \le 0.05$ 的矩阵为稀疏矩阵

  稀疏因子

- 压缩存储原则：只存储每个非零元的行、列下标及其值和矩阵的行列维数

M由{**(1,2,12), (1,3,9), (3,1,-3), (3,6,14), (4,3,24),(5,2,18), (6,1,15), (6,4,-7)** }和矩阵维数（**6,7**）唯一确定

$$M = \begin{bmatrix} 0 & 12 & 9 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -3 & 0 & 0 & 0 & 0 & 14 & 0 \\ 0 & 0 & 24 & 0 & 0 & 0 & 0 \\ 0 & 18 & 0 & 0 & 0 & 0 & 0 \\ 15 & 0 & 0 & -7 & 0 & 0 & 0 \end{bmatrix}_{6\times 7}$$

稀疏矩阵

矩阵的压缩存储

稀疏矩阵的顺序压缩存储

三元组表

i	j	e

三元组结构

```
typedef struct
{    int  i, j;             //非零元的行、列下标
     ElemType  e;           //非零元的值
 } Triple;
```

稀疏矩阵结构

```
#define  MAXSIZE  100    //非零元最大个数
typedef struct
{    Triple  data[MAXSIZE + 1];
                            //三元组表，data[0]未用
     int  mu, nu, tu;  //矩阵行、列数、非零元个数
} TSMatrix;
```

矩阵的压缩存储

稀疏矩阵的顺序压缩存储

三元组表

$$M = \begin{bmatrix} 0 & 12 & 9 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -3 & 0 & 0 & 0 & 0 & 14 & 0 \\ 0 & 0 & 24 & 0 & 0 & 0 & 0 \\ 0 & 18 & 0 & 0 & 0 & 0 & 0 \\ 15 & 0 & 0 & -7 & 0 & 0 & 0 \end{bmatrix}_{6\times 7}$$

行列下标　　非零元值

TSMatrix ma;

	i	j	e
0	6	7	8
1	1	2	12
2	1	3	9
3	3	1	-3
4	3	6	14
5	4	3	24
6	5	2	18
7	6	1	15
8	6	4	-7

ma.data

data[0]号单元未用
或存放矩阵行列维数和非零元个数

矩阵行数：　**ma.mu=6**
矩阵列数：　**ma.nu=7**
非零元个数：　**ma.tu=8**

矩阵的压缩存储

稀疏矩阵的顺序压缩存储

三元组表

行列下标　　非零元值

TSMatrix ma;

$$M = \begin{bmatrix} 0 & 12 & 9 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -3 & 0 & 0 & 0 & 0 & 14 & 0 \\ 0 & 0 & 24 & 0 & 0 & 0 & 0 \\ 0 & 18 & 0 & 0 & 0 & 0 & 0 \\ 15 & 0 & 0 & -7 & 0 & 0 & 0 \end{bmatrix}_{6\times7}$$

	i	j	e
0	6	7	8
1	1	2	12
2	1	3	9
3	3	1	-3
4	3	6	14
5	4	3	24
6	5	2	18
7	6	1	15
8	6	4	-7

ma.data

三元组表又称有序的双下标法，特点：

- 非零元在表中按行序有序存储
- 便于进行依行顺序处理的矩阵运算
- 若需存取某非零元，需从头查找。

如何存取M[i][j]? 如取M[4][3]的值

三元组表法中，M[i][j]的存储位置与其下标无关，而取决于之前的非零元个数，因此需要从头查找。

矩阵的压缩存储

稀疏矩阵的顺序压缩存储

带行逻辑链接的顺序表——带行链接信息的三元组表

- 增加一个数组**rpos[MaxRC+1]**（**MaxRC**为行数）
- **rpos[i]**表示**第i行第1个非零元在三元组表中的下标**，有：

$$\begin{cases} \text{rpos[1]=1} \\ \text{rpos[i]=rpos[i-1]+第i-1行非零元个数} \quad (i \geq 2) \end{cases}$$

第1行第1个非零元在三元组中的下标为1

rpos[0]不用或存矩阵行数

$$M = \begin{bmatrix} 0 & 12 & 9 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -3 & 0 & 0 & 0 & 0 & 14 & 0 \\ 0 & 0 & 24 & 0 & 0 & 0 & 0 \\ 0 & 18 & 0 & 0 & 0 & 0 & 0 \\ 15 & 0 & 0 & -7 & 0 & 0 & 0 \end{bmatrix}_{6\times 7}$$

M.rops

0	6
1	1
2	3
3	3
4	5
5	6
6	7

ma.data

0	6	7	8
1	1	2	12
2	1	3	9
3	3	1	-3
4	3	6	14
5	4	3	24
6	5	2	18
7	6	1	15
8	6	4	-7

矩阵的压缩存储

稀疏矩阵的顺序压缩存储

带行逻辑链接的顺序表——带行链接信息的三元组表

- 增加一个数组rpos[MaxRC+1]（MaxRC为行数）
- rpos[i]表示第i行第1个非零元在三元组表中的下标，有：

第1行第1个非零元在三元组中的下标为1

$$
\begin{cases}
rpos[1]=1 \\
rpos[i]=rpos[i-1]+\text{第}i-1\text{行非零元个数} & (i\geq 2)
\end{cases}
$$

如何存取 M[i][j]?

如取M[3][6]的值

查找M[i][j]，从第i行第1个非零元开始，找至第i+1行第一个非零元之前。

rpos[0]不用或存矩阵行数

M.rops

0	6
1	1
2	3
3	3
4	5
5	6
6	7

0	6	7	8
1	1	2	12
2	1	3	9
3	3	1	-3
4	3	6	14
5	4	3	24
6	5	2	18
7	6	1	15
8	6	4	-7

ma.data

矩阵的压缩存储

稀疏矩阵的顺序压缩存储

伪地址表示法

伪地址：行优先存储时，包括零元素在内的元素相对位置

伪地址能够表示元素在矩阵中的位置

$$
M = \begin{bmatrix}
0 & \overset{2}{12} & \overset{3}{9} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\overset{15}{-3} & 0 & 0 & 0 & 0 & \overset{20}{14} & 0 \\
0 & 0 & \overset{24}{24} & 0 & 0 & 0 & 0 \\
0 & \overset{30}{18} & 0 & 0 & 0 & 0 & 0 \\
\overset{36}{15} & 0 & 0 & \overset{39}{-7} & 0 & 0 & 0
\end{bmatrix}_{6\times 7}
$$

伪地址 非零元值 矩阵行列维数

	addr	e
0	6	7
1	2	12
2	3	9
3	15	-3
4	20	14
5	24	24
6	30	18
7	36	15
8	39	-7

M.data

0	a_{11}
1	a_{12}
	…
n-1	a_{1n}
n	a_{21}
	a_{22}
	…
	a_{2n}
	…
	a_{m1}
	a_{m2}
	…
m*n-1	a_{mn}

矩阵的压缩存储

稀疏矩阵的顺序压缩存储

伪地址表示法

伪地址：行优先存储时，包括零元素在内的元素相对位置

如何存取M[i][j]?

如取M[3][6]的值

伪地址=2*7+6=20

$$M = \begin{bmatrix} 0 & 12 & 9 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -3 & 0 & 0 & 0 & 0 & 14 & 0 \\ 0 & 0 & 24 & 0 & 0 & 0 & 0 \\ 0 & 18 & 0 & 0 & 0 & 0 & 0 \\ 15 & 0 & 0 & -7 & 0 & 0 & 0 \end{bmatrix}_{6\times7}$$

伪地址　非零元值

	addr	e
0	6	7
	2	12
	3	9
	15	-3
	20	14
5	24	24
6	30	18
7	36	15
8	39	-7

M.data

矩阵行列维数

M[i][j]的存储位置与前面的非零元个数有关，需要从头查找。

稀疏矩阵

0	a_{11}
1	a_{12}
	…
n-1	a_{1n}
n	a_{21}
	a_{22}
	…
	a_{2n}
	…
	a_{m1}
	a_{m2}
	…
m*n-1	a_{mn}

矩阵的压缩存储

稀疏矩阵的转置

问题：已知稀疏矩阵M的三元组表，求其转置矩阵T的三元组表

问题分析：一般矩阵M的转置

$$M_{mu\times nu}=\begin{bmatrix} a_{11} & a_{12} & \cdots & \cdots & a_{1nu} \\ a_{21} & a_{22} & \cdots & \cdots & a_{2nu} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{mu1} & a_{mu2} & \cdots & \cdots & a_{munu} \end{bmatrix} \Rightarrow T_{nu\times mu}=\begin{bmatrix} a'_{11} & a'_{12} & \cdots & \cdots & a'_{1mu} \\ a'_{21} & a'_{22} & \cdots & \cdots & a'_{2mu} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a'_{nu1} & a'_{nu2} & \cdots & \cdots & a'_{numu} \end{bmatrix}$$

```
for(col=0;col<nu;col++)
    for(row=0;row<mu;row++)
        T[col][row]=M[row][col];
```

$$\boxed{T(n)=O(mu\times nu)}$$

矩阵的压缩存储

稀疏矩阵的转置

问题：已知稀疏矩阵M的三元组表，求其转置矩阵T的三元组表

问题分析：压缩存储后矩阵M的转置

M.data

	i	j	e
0	6	7	8
1	1	2	12
2	1	3	9
3	3	1	-3
4	3	6	14
5	4	3	24
6	5	2	18
7	6	1	15
8	6	4	-7

?

T.data

	i	j	e
0	7	6	8
1	1	3	-3
2	1	6	15
3	2	1	12
4	2	5	18
5	3	1	9
6	3	4	24
7	4	6	-7
8	6	3	14

矩阵的压缩存储

稀疏矩阵的转置

求解思路：

1. 将矩阵行、列数互换，非零元个数不变
2. 将每个三元组中的i和j相互调换，非零元值不变
3. 重新排序，使T.data中元素以T的行(M的列)为主序

M.data

	i	j	e
0	6	7	8
1	1	2	12
2	1	3	9
3	3	1	-3
4	3	6	14
5	4	3	24
6	5	2	18
7	6	1	15
8	6	4	-7

T.data

	i	j	e
0	7	6	8
1	1	3	-3
2	1	6	15
3	2	1	12
4	2	5	18
5	3	1	9
6	3	4	24
7	4	6	-7
8	6	3	14

矩阵的压缩存储

稀疏矩阵的转置

方法一：按M的列序转置

1. 按矩阵T中三元组表T.data的顺序，依次在矩阵M的三元组表M.data中找到相应三元组进行转置
2. 为找到M.data中第i列所有非零元素，需对M.data扫描一遍
3. 由于M.data以M行序为主序，所以得到的恰是T.data中应有的顺序

M.data

	i	j	e
0	6	7	8
1	1	2	12
2	1	3	9
3	3	1	-3
4	3	6	14
5	4	3	24
6	5	2	18
7	6	1	15
8	6	4	-7

T.data

	i	j	e
0	7	6	8
1	1	3	-3
2	1	6	15
3	2	1	12
4	2	5	18
5	3	1	9
6	3	4	24
7	4	6	-7
8	6	3	14

矩阵的压缩存储

查找每一列非零元都需要对三元组表扫描一次，因此，算法效率较低。

稀疏矩阵的转置

方法一：按M的列序转置

① 按矩阵T中三元组表T.data的顺序，依次在矩阵M的三元组表M.data中找到相应三元组进行转置

② 为找到M.data中第i列所有非零元素，需对M.data扫描一遍

③ 由于M.data以M行序为主序，所以得到的恰是T.data中应有的顺序

col=[illegible]

M.data

	i	j	e
0	6	7	8
1	1	2	12
2	1	3	9
3	3	1	-3
4	3	6	14
5	4	3	24
6	5	2	18
7	6	1	15
8	6	4	-7

T.data

	i	j	e
0	7	6	8
1	1	3	-3
2	1	6	15
3	2	1	12
4	2	5	18
5	3	1	9
6	3	4	24
7	4	6	-7
8	6	3	14

算法结束

稀疏矩阵的转置 ——按M的列序转置

```
Status TransposeSMatrix(TSMatrix  M, TSMatrix &T)
{  int  col, p, k;
   T.mu=M.nu;    T.nu=M.mu;   T.tu=M.tu;
   if(T.tu)
  {    k=1;
      for( col=1;col<=M.nu;col++)
         for( p=1;p<=M.tu;p++)
            if( M.data[p].j==col )
            {   T.data[k]. i =M.data[p].j;
                T.data[k]. j=M.data[p].i;
                T.data[k]. e=M.data[p].e;
                k++;            }
      return  OK;
   }
   return  ERROR;
}
```

设置矩阵信息

有非零元，转置

k为T.data表下标

查找M每一列的非零元

扫描M的所有非零元

$T(n)=O(M.nu \times M.tu)$

若M.tu与M.mu×M.nu同数量级

则 $T(n)=O(M.mu \times M.nu^2)$

矩阵的压缩存储

稀疏矩阵的转置

方法二：快速转置

☺ 按**M.data**中三元组次序转置，结果放入**T.data**中<u>恰当位置</u>

	i	j	e
0	6	7	8
1	1	2	12
2	1	3	9
3	3	1	-3
4	3	6	14
5	4	3	24
6	5	2	18
7	6	1	15
8	6	4	-7

M.data

	i	j	e
0	7	6	8
1	1	3	-3
2	1	6	15
3	2	1	12
4	2	5	18
5	3	1	9
6	3	4	24
7	4	6	-7
8	6	3	14

T.data

矩阵的压缩存储

稀疏矩阵的转置

方法二：快速转置

- 按M.data中三元组次序转置，结果放入T.data中恰当位置
- 实现：设置2个数组

num[col]：矩阵M中第col列中非零元个数

cpot[col]：矩阵M中第col列第1个非零元在T.data中位置

$$\begin{cases} cpot[1]=1; \\ cpot[col]=cpot[col-1]+num[col-1]; \quad (2 \le col \le M.nu) \end{cases}$$

M.data

1	2	12
1	3	9
3	1	-3
3	6	14
4	3	24
5	2	18
6	1	15
6	4	-7

col	1	2	3	4	5	6	7
num[col]	2	2	2	1	0	1	0
cpot[col]	1	3	5	7	8	8	9

算法演示

稀疏矩阵的转置 ——快速转置

```
void fast_transpos ( TSMatrix M, TSMatrix &T )
{  int col,p,k , num[N],cpot[N];
   T.mu = M.nu;  T.nu = M.mu;  T.tu = M.tu;
   if(T.tu)
   {  for(col=1;col<=M.nu;col++ ) num[col]=0;
      for(p=1;p<=M.tu;p++) num[ M.data[p].j ]++;
      cpot[0]=0; cpot[1]=1;
      for(col=2;col<=M.nu; col++  )
           cpot[col]=cpot[col-1]+num[col-1];
      for( p=1; p<=M.tu; p++  )
      {  col=M.data[p].j;  k=cpot[col];
         T.data[k].i=M.data[p].j;  T.data[k].j=M.data[p].i;
         T.data[k].e=M.data[p].e; cpot[col]++;
      }
   }
}
```

设置矩阵信息

num[]清零

设置num[]

设置cpot[]

转置

T(n)=O(M.nu+M.tu)若M.tu与M.mu×M.nu同数量级

则 T(n)=O(M.mu×M.nu)

矩阵的压缩存储

稀疏矩阵的链式压缩存储

引入稀疏矩阵链式存储的原因：

- 用三元组表存储稀疏矩阵，在单纯的存储和做类似转置之类的运算时可以节约存储空间，且运算速度较快；
- 但当进行矩阵相加等运算时，稀疏矩阵的非零元位置和个数都会发生变化。使用三元组表必然会引起数组元素的大量移动。

矩阵的压缩存储

稀疏矩阵的链式压缩存储

带行指针向量的单链表表示

- ✓ 每行非零元用一个单链表存储
- ✓ 单链表结点存放非零元的所在列及值
- ✓ 用行指针数组存储各行单链表的头指针

$$M = \begin{bmatrix} 3 & 0 & 0 & 0 & 7 \\ 0 & 0 & -1 & 0 & 0 \\ -1 & -2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \end{bmatrix}$$

RowList M;

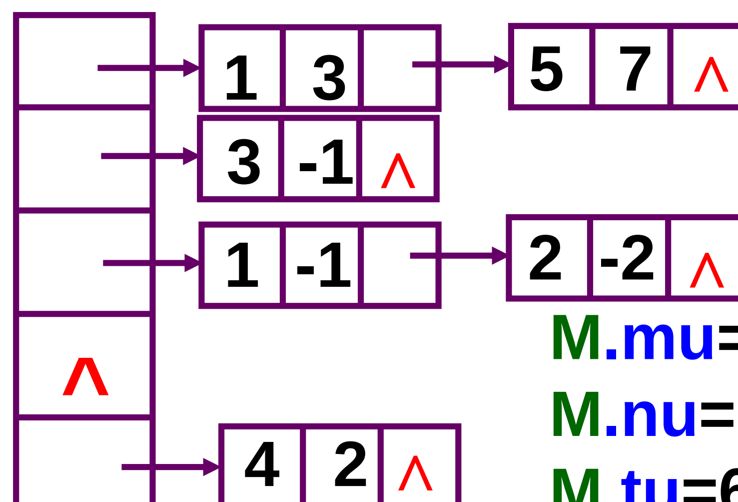
单链表结点

```
typedef struct RLNode
{   int   j;
    ElemType  e;
    struct  RLNode *right;
}RLNode,* RLink ;
```

数组结构

```
typedef struct
{   RLink  rhead[M];
    int mu, nu, tu;
}RowList;
```

M.rhead



M.mu=5

M.nu=5

M.tu=6

适合于按行进行操作的问题

矩阵的压缩存储

稀疏矩阵的链式压缩存储

row	col	val
down	right	

十字链表

✓ 非零元结点包含5个域（非零元的所在行、列、值，及指向同行、同列下一个非零元的指针）

结点结构

```
typedef struct OLNode
{    int  row, col;                  //非零元所在行、列
     ElemType  val;                  //非零元的值
     struct OLNode  *right, *down;
                                     //同行、同列的下一个非零元的指针
}OLNode,* OLink;
```

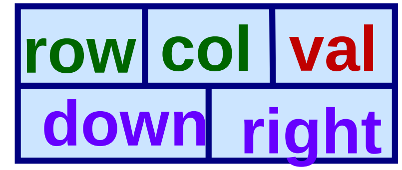
十字链表结构

```
typedef struct
{   OLink  rhead[M],chead[N]; //行、列指针数组
     int  mu, nu, tu;              //行、列数及非零元个数
}CrossList;
```

矩阵的压缩存储

稀疏矩阵的链式压缩存储

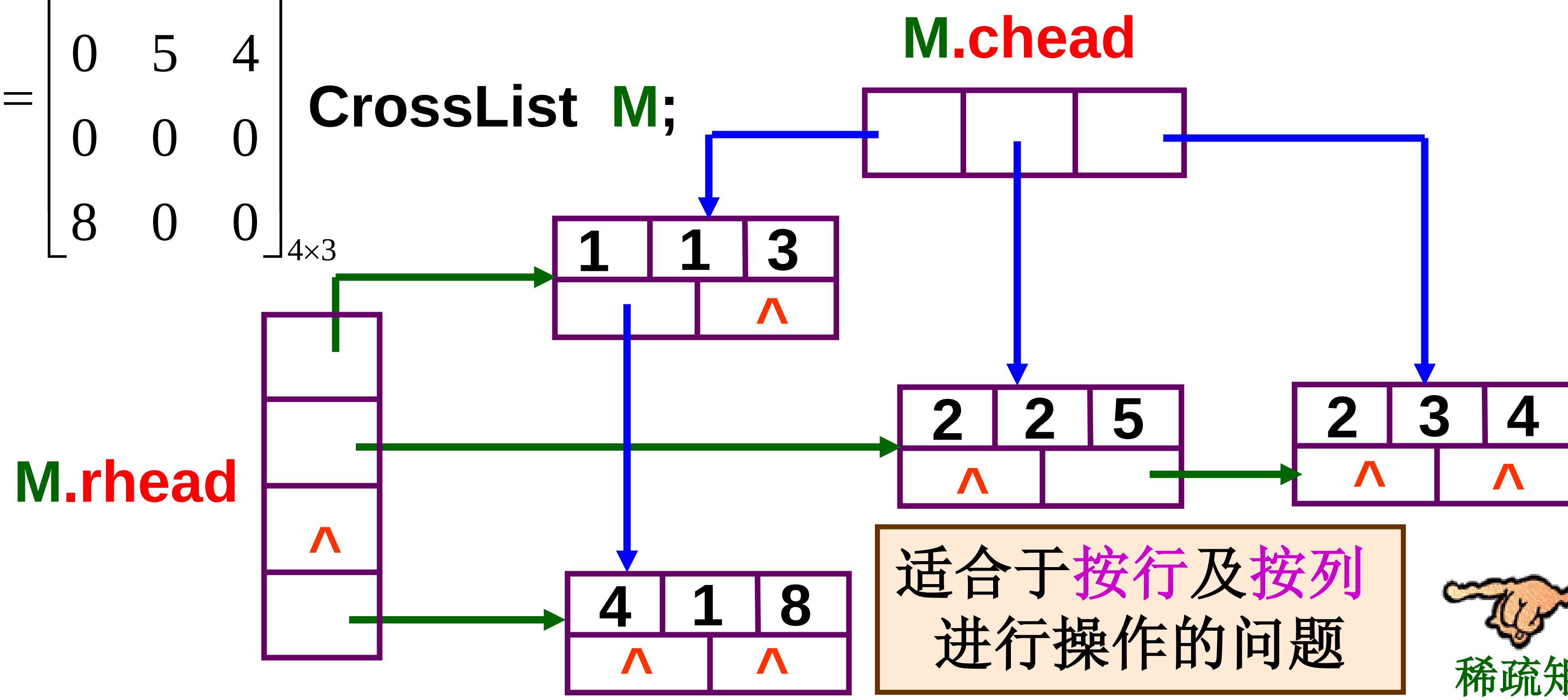
十字链表



✓ 非零元结点包含5个域（非零元的所在行、列、值，及指向同行、同列下一个非零元的指针）

$$M=\begin{bmatrix}3&0&0\\0&5&4\\0&0&0\\8&0&0\end{bmatrix}_{4\times3}$$

CrossList M;



适合于按行及按列进行操作的问题

稀疏矩阵

本　章　小　结

- 数组的定义、顺序存储方式
- 掌握特殊矩阵(对称阵、三角阵、对角阵)的压缩存储方式
- 掌握稀疏矩阵的顺序、链式压缩存储方式，了解转置算法的实现

Back