线 性 表 扩 展 2

学习目标

- 了解广义表
- 了解多维数组，熟知特殊矩阵存储
- 了解稀疏矩阵存储

广 义 表

广义表

广义表是线性表的推广

- 对于线性表而言， n个元素都是基本的单元素；
- 广义表中，这些元素不仅可以是单元素也可以是另一个广义表。

A=()

B=(e)

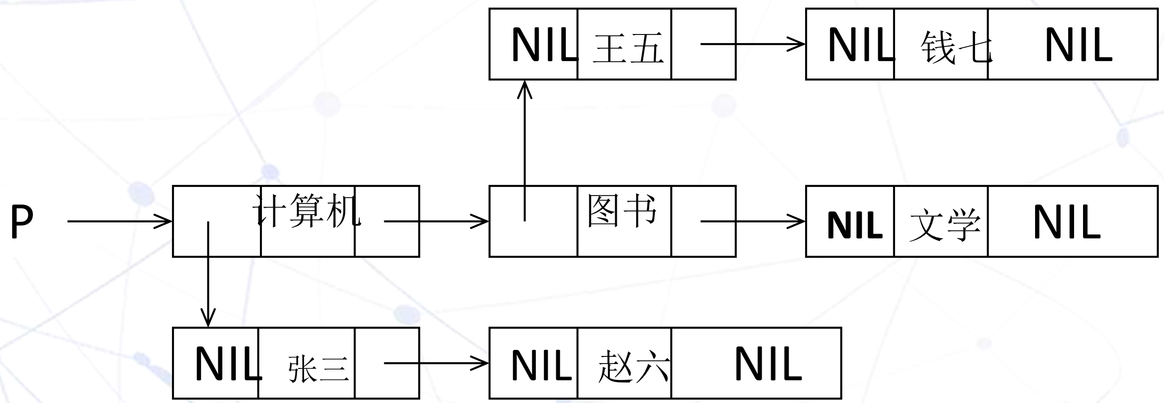
C=(b,c,d)

D1=(a,(b,c,d))

D2=(a,C)

(张三，计算机)
(王五，图书)
(李四，文学)
(赵六，计算机)
(钱七，图书)

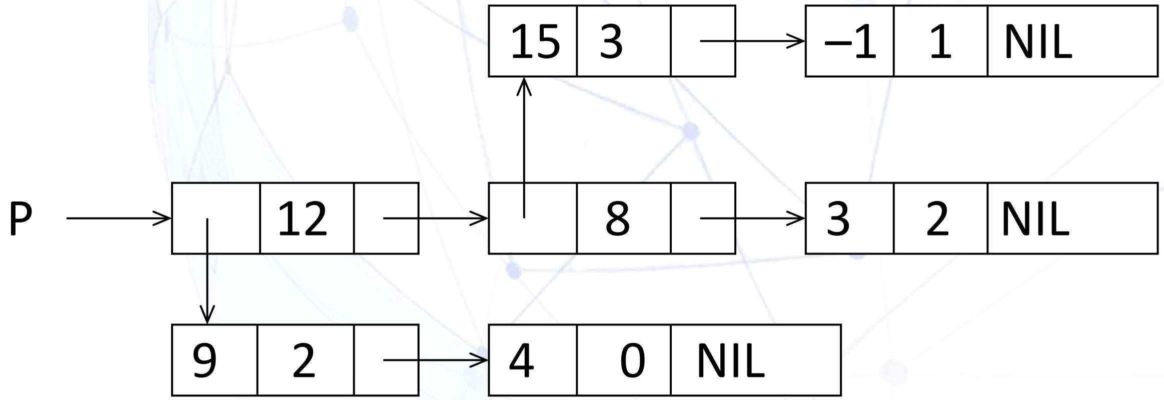
staff=（
（计算机，（张三，赵六）），
（图书，（王五，钱七）），
（文学，（李四））
）



广义表

例如：二元多项式的表示

$$P(x,y)=9x^{12}y^{2}+4x^{12}+15x^{8}y^{3}-x^{8}y+3x^{2} \longrightarrow P(x,y)=(9y^{2}+4)x^{12}+(15y^{3}-y)x^{8}+3x^{2}$$



广义表的性质

由广义表的定义，可以得到广义表有以下五个性质：

1)次序性。在广义表中，各表元素在表中以线性序列排序，每个元素至多有一个前驱和一个后继。次序不能交换。

2)有长度。广义表中元素个数一定，不能是无限的。

3)有深度。广义表是多层次结构。表元素可以是原子，也可以是子表。表中括号的重数即为广义表的**深度**。

4)可递归。广义表本身可以是自己的子表。

5)可共享。广义表可以为其它广义表共享。

广义表的操作

广义表的表头和表尾：

- **表头**：任何非空广义表的第一个元素称为表头。
- **表尾**：除去表头后剩余的元素组成的表称为表尾。

广义表的操作主要是取头和取尾操作。

任何一个非空广义表其表头可能是原子，也可能是列表，而其表尾必定为列表。

A=()

B=(e，f)

C=(a，(b，c))

D=(B，A，C)

E=(a，E)

GetHead(B)＝e；GetTail(B)=(f)。

GetHead(D)＝B；GetTail(D)＝(A，C)。

多 维 数 组

学习目标

（多维）数组的定义和特点

（多维）数组的抽象数据类型定义

多维数组的存储

特殊矩阵的存储

稀疏矩阵的存储

数组的逻辑结构

- 数组的定义
- 数组的特点
- 数组的抽象数据类型定义

八皇后问题

【问题】八皇后问题是数学家高斯于1850年提出的。问题是：在8×8的棋盘上摆放八个皇后，使其不能互相攻击，即任意两个皇后都不能处于同一行、同一列或同一斜线上。。

【想法——数据表示】如何表示棋盘？如何获得每个皇后的位置信息进而判断是否互相攻击？

	Q						
			Q				
					Q		
							Q
				Q			
Q							
						Q	
		Q					

用二维数组保存

很多问题的表现形式是矩阵，很多科学问题的数据模型是矩阵

数组的定义

数组：由一组类型相同的数据元素构成的有序集合，每个数据元素称为一个数组元素（简称为元素），每个元素受 $n(n\geq 1)$个线性关系的约束，每个元素在 n 个线性关系中的序号i_1、i_2、…、i_n称为该元素的下标，并称该数组为 n 维数组。

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

数组的特点

- 数组有什么特点呢?

（1）元素本身可以具有某种结构，属于同一数据类型；

$$\begin{array}{cc} & \begin{array}{cccc} A_1 & A_2 & & A_n \end{array} \\ A= & \left(\begin{array}{cccc} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{array}\right) \end{array}$$

$A=(A_1,\ A_2,\ \ldots,\ A_n)$

其中：

$A_j=(a_{1j},\ a_{2j},\ \ldots,\ a_{mj})\ (1\leq j\leq n)$

$A=(A_1,\ A_2,\ \ldots,\ A_m)$

其中：

$A_i=(a_{i1},\ a_{i2},\ \ldots,\ a_{in})\ (1\leq i\leq m)$

二维数组是数据元素为线性表的线性表

数组的特点

数组有什么特点呢?

（1）元素本身可以具有某种结构，属于同一数据类型；

（2）数组是一个具有固定格式和数量的数据集合。

$$
\begin{array}{cc}
 & \begin{array}{cccc} \phantom{a_{11}} & x & & \phantom{a_{1n}} \\ & \downarrow & & \end{array} \\
A= & \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}
\end{array}
$$

在数组上一般不能执行插入或删除某个数组元素的操作

数组的特点

数组有什么基本操作呢？

（1）存取：给定一组下标，读出对应的数组元素

（2）修改：给定一组下标，存储或修改与其相对应的数组元素

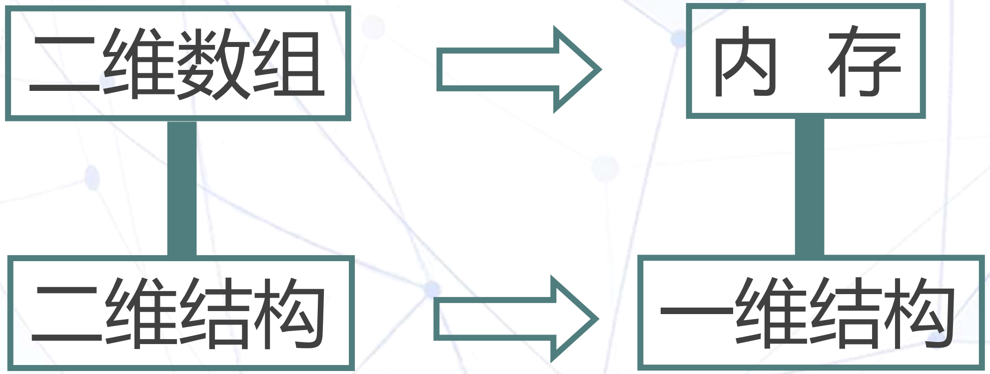
寻址

ADT Matrix
DataModel
  相同类型的数据元素的有序集合，每个元素受n（$n \geq 1$）个线性关系的约束
Operation
  InitMatrix：数组的初始化
  DestroyMatrix：数组的销毁
  GetMatrix：读操作，读取这组下标对应的数组元素
  SetMatrix：写操作，存储或修改这组下标对应的数组元素
endADT

多维数组的存储

如何存储（多维）数组呢?

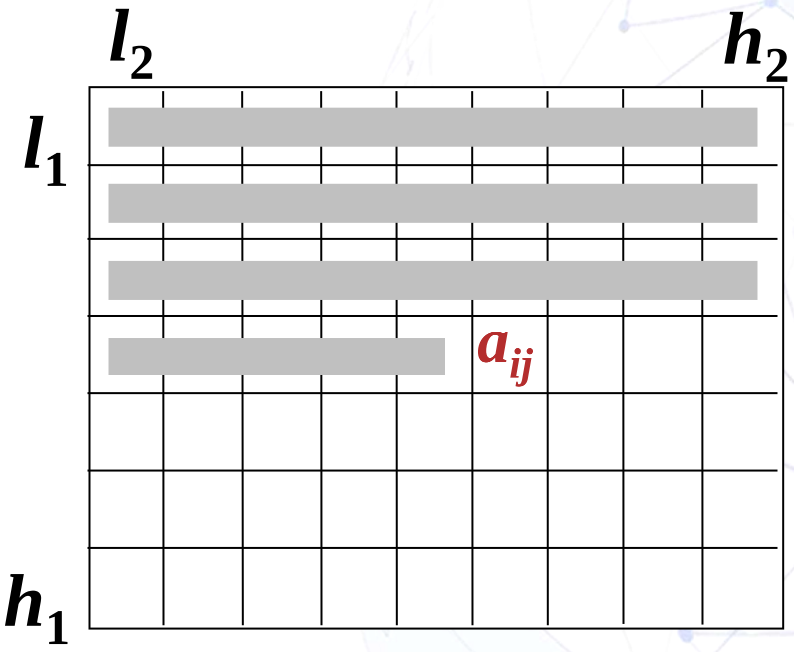
数组没有插入和删除操作，所以，不用预留空间，适合采用顺序存储



- 按行优先：先存储行号较小的元素，行号相同者先存储列号较小的元素
- 按列优先：先存储列号较小的元素，列号相同者先存储行号较小的元素

数组的存储结构

按行优先：先存储行号较小的元素，行号相同者先存储列号较小的元素



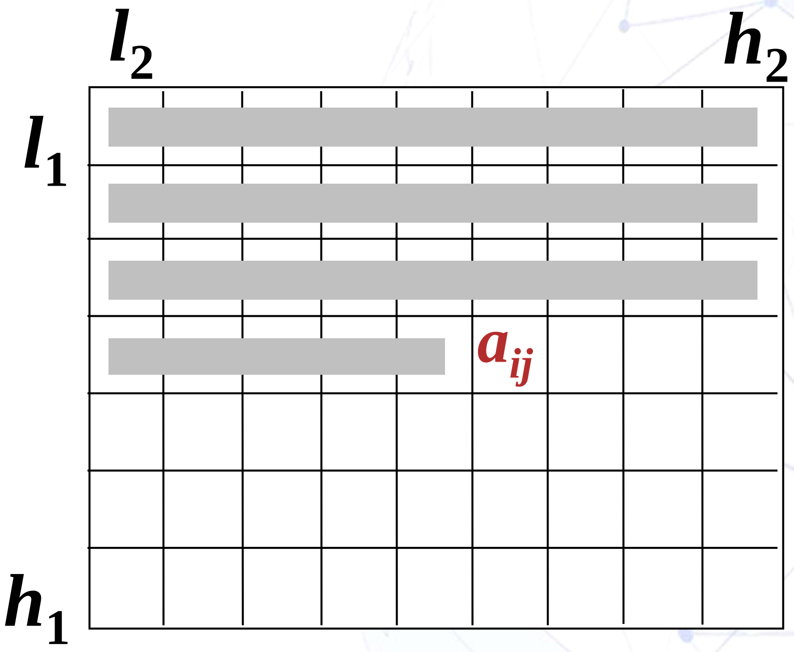
如何得到元素a_{ij}的存储地址？

第l_1行 第l_1+1行

$a_{l_1 l_2}$	$\cdots$	$a_{l_1 h_2}$	$a_{(l_1+1)l_2}$	$\cdots$	$a_{(l_1+1)h_2}$	$\cdots$	a_{ij}	$\cdots$	$a_{h_1 h_2}$

数组的存储结构

$$\mathrm{Loc}(a_{ij})=\mathrm{Loc}(a_{l_1l_2})+((i-l_1)\times(h_2-l_2+1)+(j-l_2))\times c$$



a_{ij}前面的元素个数
=整行数×每行元素个数+本行中
a_{ij}前面的元素个数
$=(i-l_1)\times(h_2-l_2+1)+(j-l_2)$

按列优先与此类似，请自行给出

$\mathrm{Loc}(a_{l_1l_2}) \qquad \overbrace{\qquad\qquad\qquad\qquad\qquad\qquad}^{(i-l_1)\times(h_2-l_2+1)+(j-l_2)\text{个元素}} \qquad \mathrm{Loc}(a_{ij})$

$a_{l_1l_2}$	$\cdots$	$a_{l_1h_2}$	$a_{(l_1+1)l_2}$	$\cdots$	$a_{(l_1+1)h_2}$	$\cdots$	a_{ij}	$\cdots$	$a_{h_1h_2}$

单选题 5分

设置

1. 数组是一种复杂的数据结构，数组元素之间的关系既不是线性的，也不是树形的。

A 正确

B 错误

提交

单选题 5分

设置

2. 数组是一个具有固定格式和数量的数据结构。

A 正确

B 错误

提交

单选题 5分

设置

3. 数组包括插入、删除、查找、修改、求元素个数、判空等基本操作。

A 正确

B 错误

提交

单选题 5分

设置

4. 数组按行优先和按列优先的存储思想是相同的。

A 正确

B 错误

提交

单选题 5分

设置

5. 设**二维数组**r[4][5]的起始地址是d，则按行优先存取时，元素r[3][4]存储地址是（　）。

A d+7

B d+12

C d+19

D d+20

(3-0)*5=15

(4-0)=4

15+4=19

d+19

提交

矩阵

- 对称矩阵的压缩存储
- 三角矩阵的压缩存储
- 对角矩阵的压缩存储
- 稀疏矩阵的压缩存储

科学计算中的矩阵

在数学中，矩阵（Matrix）是一个按照长方阵列排列的复数或实数集合。

矩阵的运算是数值分析领域的重要问题。数值分析的主要分支致力于开发矩阵计算的有效算法，这是一个已持续几个世纪以来的课题，是一个不断扩大的研究领域。

计算机中的矩阵

特殊矩阵

- 什么是特殊矩阵?
- 特殊矩阵：矩阵中很多值相同的元素并且它们的分布有一定的规律
- 特殊矩阵如何压缩存储?

  为值相同的元素分配一个存储空间
- 特殊矩阵压缩存储后有什么要求吗?

  保证随机存取，即在$O(1)$时间内寻址

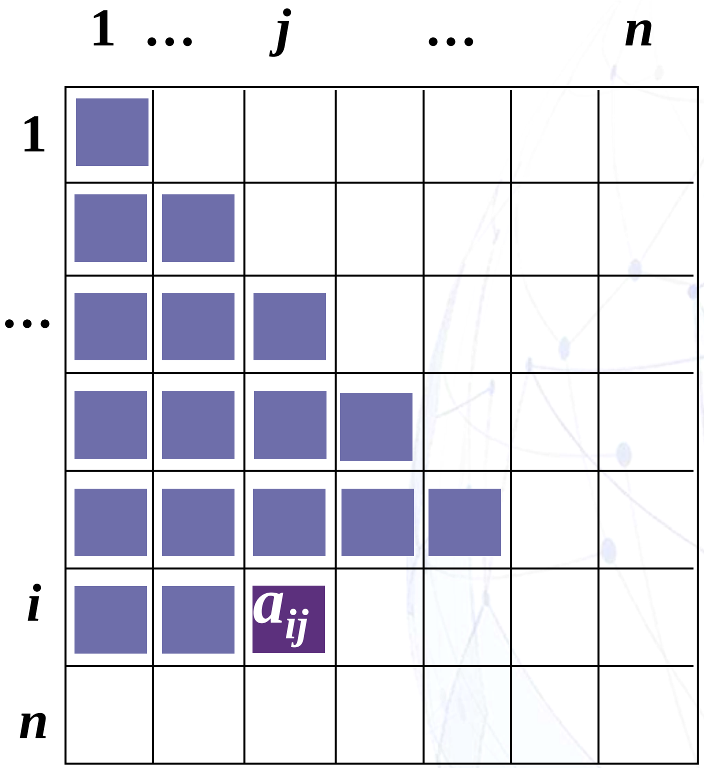
对称矩阵

$$A=\begin{pmatrix} 3 & 6 & 4 & 7 & 8 \\ 6 & 2 & 8 & 4 & 2 \\ 4 & 8 & 1 & 6 & 9 \\ 7 & 4 & 6 & 0 & 5 \\ 8 & 2 & 9 & 5 & 7 \end{pmatrix}$$

对称矩阵特点：$a_{ij}=a_{ji}$

如何压缩存储对称矩阵呢？ ⇨ 只存储下三角部分的元素

对称矩阵



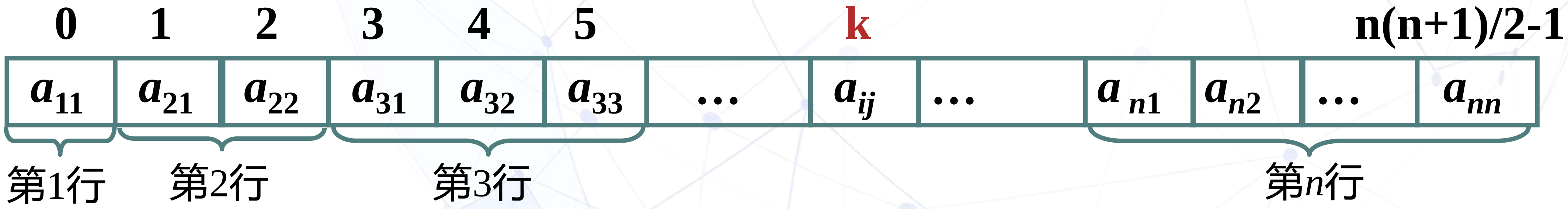
a_{ij} 在一维数组中的序号

$= i\times(i-1)/2+j$

$\because$ 一维数组下标从 0 开始

$\therefore a_{ij}$ 在一维数组中的下标

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



对称矩阵

对称矩阵压缩存储后的寻址方法

对于下三角中的元素a_{ij}（$i \geq j$）：$k = i \times (i-1)/2 + j - 1$

对于上三角中的元素a_{ij}（$i < j$），因为$a_{ij} = a_{ji}$，则 $k = j \times (j-1)/2 + i - 1$

0	1	2	3	4	5		k					n(n+1)/2-1
a_{11}	a_{21}	a_{22}	a_{31}	a_{32}	a_{33}	…	a_{ij}	…	a_{n1}	a_{n2}	…	a_{nn}
第1行	第2行		第3行						第n行			

单选题 5分

设置

1. 对称矩阵只存储下三角部分的元素，节省了存储空间但失去了随机存取功能。

A 正确

B 错误

提交

三角矩阵

$$\begin{pmatrix} 3 & c & c & c & c \\ 6 & 2 & c & c & c \\ 4 & 8 & 1 & c & c \\ 7 & 4 & 6 & 0 & c \\ 8 & 2 & 9 & 5 & 7 \end{pmatrix}$$

(a)　下三角矩阵

$$\begin{pmatrix} 3 & 4 & 8 & 1 & 0 \\ c & 2 & 9 & 4 & 6 \\ c & c & 1 & 5 & 7 \\ c & c & c & 0 & 8 \\ c & c & c & c & 7 \end{pmatrix}$$

(b)　上三角矩阵

如何压缩存储三角矩阵呢? ⇨

- 下（上）三角部分的元素
- 相同的常数只存储一个

三角矩阵

下三角矩阵的压缩存储

0	1	2	3	4	5		k						n(n+1)/2
a_{11}	a_{21}	a_{22}	a_{31}	a_{32}	a_{33}	…	a_{ij}	…	a_{n1}	a_{n2}	…	a_{nn}	c
第1行	第2行		第3行						第n行				

下三角矩阵压缩存储后的寻址方法

对于下三角中的元素a_{ij}（$i \geq j$）： $k = i \times (i-1)/2 + j - 1$

对于上三角中的元素a_{ij}（$i < j$）： $k = n \times (n+1)/2$

上三角矩阵的压缩存储请仿此给出

对角矩阵

对角矩阵：所有非零元素都集中在以主对角线为中心的带状区域中，所有其他元素都为零

$$A = \begin{pmatrix} a_{11} & a_{12} & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 & 0 \\ 0 & a_{32} & a_{33} & a_{34} & 0 \\ 0 & 0 & a_{43} & a_{44} & a_{45} \\ 0 & 0 & 0 & a_{54} & a_{55} \end{pmatrix}$$

如何压缩存储对角矩阵呢?

⇩

只存储非零元素

对角矩阵

$$A = \begin{pmatrix} \boxed{a_{11}} & \boxed{a_{12}} & 0 & 0 & 0 \\ \boxed{a_{21}} & \boxed{a_{22}} & \boxed{a_{23}} & 0 & 0 \\ 0 & \boxed{a_{32}} & \boxed{a_{33}} & \boxed{a_{34}} & 0 \\ 0 & 0 & \boxed{a_{43}} & \boxed{a_{44}} & \boxed{a_{45}} \\ 0 & 0 & 0 & \boxed{a_{54}} & \boxed{a_{55}} \end{pmatrix}$$

元素 a_{ij} 在一维数组中的序号

$=2+3(i-2)+(j-i+2)$

$=2i+j-2$

∵一维数组下标从 0 开始

∴元素 a_{ij} 在一维数组中的下标

$=2i+j-3$

0	1	2	3	4	5	6	7	8	9	10	11	12
a_{11}	a_{12}	a_{21}	a_{22}	a_{23}	a_{32}	a_{33}	a_{34}	a_{43}	a_{44}	a_{45}	a_{54}	a_{55}

$\underbrace{a_{11}\ a_{12}}_{\text{第1行}}\ \underbrace{a_{21}\ a_{22}\ a_{23}}_{\text{第2行}}\ \underbrace{a_{32}\ a_{33}\ a_{34}}_{\text{第3行}}$

单选题 5分

设置

2. 对于n阶5对角矩阵采用压缩存储，压缩比约为（　）。

A 1/2

B 5/n

C $5/n^2$

D 1/5

提交

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & 0 & 0 \\ a_{21} & a_{22} & a_{23} & a_{24} & 0 \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\ 0 & a_{42} & a_{43} & a_{44} & a_{45} \\ 0 & 0 & a_{53} & a_{54} & a_{55} \end{pmatrix}$$

5对角矩阵，从3，4，5…..，4， 3

(3+4)*2+(n-4)*5=14+5n-20=5n-6

(5n-6)/n*n 约为5/n

稀疏矩阵

- 稀疏矩阵的压缩存储——三元组顺序表
- 稀疏矩阵的压缩存储——十字链表

稀疏矩阵

什么是稀疏矩阵?

稀疏矩阵：矩阵中有很多零元素，并且分布没有规律

稀疏矩阵如何压缩存储?

只存储非零元素，零元素不分配存储空间

$$A = \begin{pmatrix} 3 & 0 & 0 & 7 \\ 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 8 \end{pmatrix}$$

稠密度=5/20=0.25

如何只存储非零元素?

三元组：（行号，列号，非零元素值）

稀疏矩阵

- 三元组表：将稀疏矩阵的非零元素对应的三元组所构成的集合，按行优先的顺序排列成一个线性表

- 如何存储三元组表？

```
template <typename DataType>
struct element
{
    int row, col;
    DataType item
};
```

((1, 1, 3), (1, 4, 7), (2, 3, 1), (3, 1, 2), (5, 4, 8))

$$
\boldsymbol{A} = \begin{pmatrix} \mathbf{3} & \mathbf{0} & \mathbf{0} & \mathbf{7} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{2} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{8} \end{pmatrix}
$$

- 三元组：（行号，列号，非零元素值）

三元组顺序表

- 三元组顺序表：采用顺序存储结构存储三元组表
- 三元组顺序表需要预留存储单元吗?

((1, 1, 3), (1, 4, 7), (2, 3, 1), (3, 1, 2), (5, 4, 8))

稀疏矩阵的修改操作

⇓

三元组表的插入/删除操作

$$A = \begin{pmatrix} 3 & 0 & 0 & 7 \\ 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 8 \end{pmatrix}$$

三元组顺序表

	row	col	item
0	1	1	3
1	1	4	7
2	2	3	1
3	3	1	2
4	5	4	8
	空闲	空闲	空闲
MaxTerm−1			
	5（非零元个数）		
	5（矩阵的行数）		
	6（矩阵的列数）		

是否对应惟一的稀疏矩阵?

$$A=\begin{array}{cc}\begin{pmatrix}3&0&0&7\\0&0&1&0\\2&0&0&0\\0&0&0&0\\0&0&0&8\end{pmatrix} & \begin{matrix}0\\0\\0\\0\\0\end{matrix}\\ \begin{matrix}0&0&0&0\end{matrix} & 0\end{array}$$

不唯一，所以需要限定行数和列数

三元组顺序表

	row	col	item
0	1	1	3
1	1	4	7
2	2	3	1
3	3	1	2
4	5	4	8
	空闲	空闲	空闲
MaxTerm−1			
	5（非零元个数）		
	5（矩阵的行数）		
	6（矩阵的列数）		

是否对应惟一的稀疏矩阵?

```
const int MaxTerm = 100;
struct SparseMatrix
{
      element data[MaxTerm];
      int mu, nu, tu;
};
```

稀疏矩阵

$$\begin{pmatrix} 0 & 0 & 0 & 22 & 0 & 0 & 15 \\ 0 & 11 & 0 & 0 & 0 & 17 & 0 \\ 0 & 0 & 0 & -6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 39 & 0 \\ 91 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 28 & 0 & 0 & 0 & 0 \end{pmatrix}$$

数组存储 →

从0编号

	行 row	列 col	值 item
[0]	0	3	22
[1]	0	6	15
[2]	1	1	11
[3]	1	5	17
[4]	2	3	−6
[5]	3	5	39
[6]	4	0	91
[7]	5	2	28

稀疏矩阵

$$\begin{pmatrix} 0 & 0 & 0 & 22 & 0 & 0 & 15 \\ 0 & 11 & 0 & 0 & 0 & 17 & 0 \\ 0 & 0 & 0 & -6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 39 & 0 \\ 91 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 28 & 0 & 0 & 0 & 0 \end{pmatrix}$$

	行 row	列 col	值 item
[0]	0	3	22
[1]	0	6	15
[2]	1	1	11
[3]	1	5	17
[4]	2	3	−6
[5]	3	5	39
[6]	4	0	91
[7]	5	2	28

它的转置 →

转置矩阵

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 91 & 0 \\ 0 & 11 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 28 \\ 22 & 0 & -6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 17 & 0 & 39 & 0 & 0 \\ 15 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

	行 row	列 col	值 item
[0]	0	4	91
[1]	1	1	11
[2]	2	5	28
[3]	3	0	22
[4]	3	2	−6
[5]	5	1	17
[6]	5	3	39
[7]	6	0	15

用三元组表表示的稀疏矩阵及其转置

原矩阵三元组表

	行 row	列 col	值 item
[0]	0	3	22
[1]	0	6	15
[2]	1	1	11
[3]	1	5	17
[4]	2	3	−6
[5]	3	5	39
[6]	4	0	91
[7]	5	2	28

转置矩阵三元组表

	行 row	列 col	值 item
[0]	0	4	91
[1]	1	1	11
[2]	2	5	28
[3]	3	0	22
[4]	3	2	−6
[5]	5	1	17
[6]	5	3	39
[7]	6	0	1

矩阵的快速转置操作

附设两个数组num和cpot, num[col] 存放矩阵A中第col列的非零元素的个数，cpot[col]存放A中第col列的第一个非零元素在其转置B中的位置

```
cpot[0] = 0;
for (i=1; i<cols; i++)
  cpot[i] = cpot[i-1] + num[i-1];
```

	行 row	列 col	值 item
[0]	0	3	22
[1]	0	6	15
[2]	1	1	11
[3]	1	5	17
[4]	2	3	−6
[5]	3	5	39
[6]	4	0	91
[7]	5	2	28

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

	row	col	item
[0]	0	4	91
[1]	1	1	11
[2]	2	5	28
[3]	3	0	22
[4]	3	2	−6
[5]	5	1	17
[6]	5	3	39
[7]	6	0	

十字链表

三元组顺序表不适合什么情况？

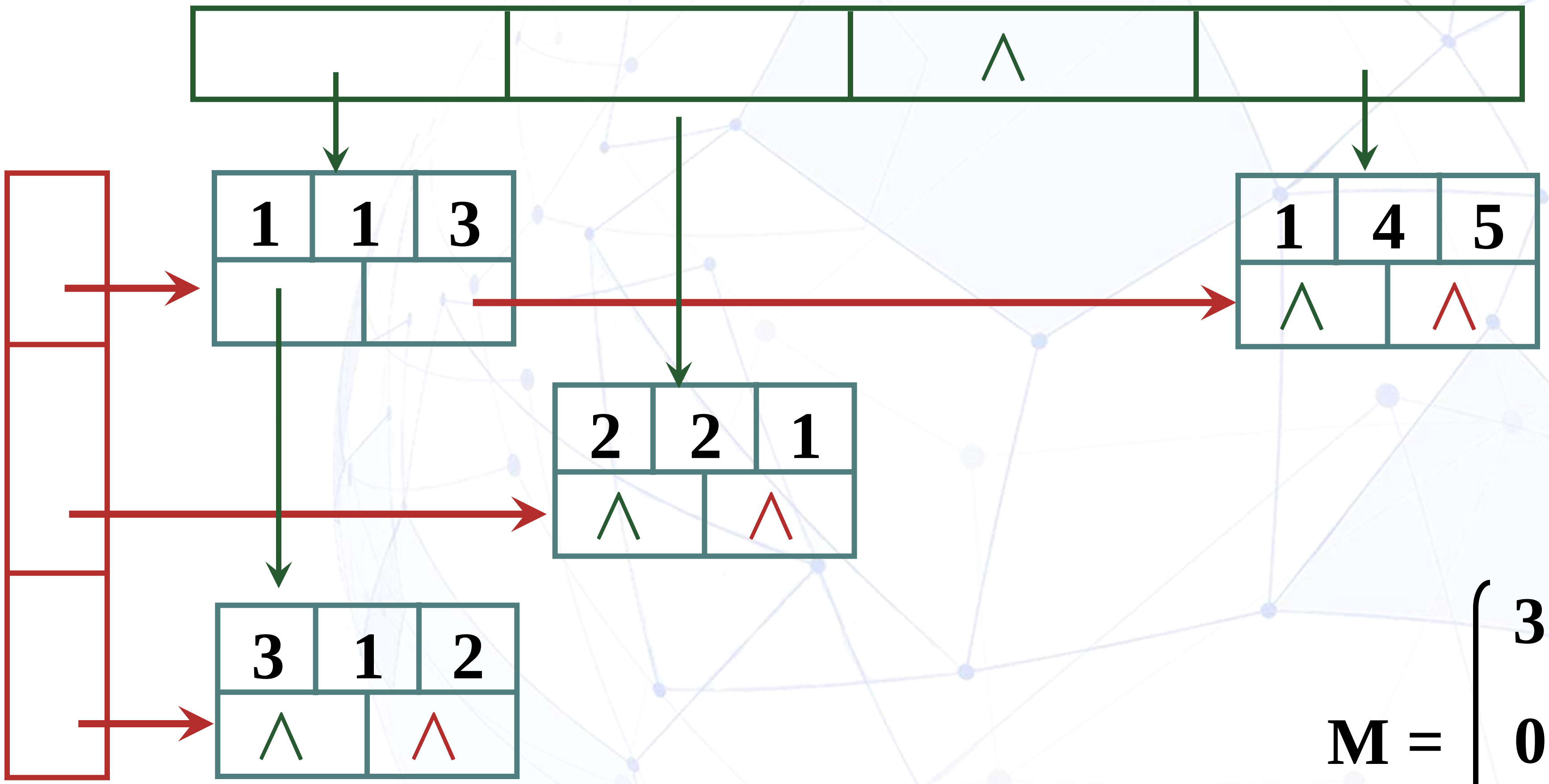
稀疏矩阵的加法、乘法等操作，非零元素的个数及位置都会发生变化，则在三元组顺序表中就要进行插入和删除操作，顺序存储就十分不便

十字链表：采用链接存储结构存储三元组表

<table>
<tr><td>row</td><td>col</td><td>item</td></tr>
<tr><td colspan="2">down</td><td>right</td></tr>
</table>

```
struct OrthNode
{
    Element data;
    OrthNode *right, *down;
};
```

十字链表



$$M = \begin{pmatrix} 3 & 0 & 0 & 5 \\ 0 & 1 & 0 & 0 \\ 2 & 0 & 0 & 0 \end{pmatrix}$$

单选题 5分

设置

3. 只有特殊矩阵和稀疏矩阵可以进行压缩存储。

A 正确

B 错误

提交

单选题 5分

设置

4. 采用三元组表存储稀疏矩阵，有时并不能节省存储空间。

A 正确

B 错误

提交

单选题 5分

设置

5. 采用十字链表存储稀疏矩阵，十字链表的结点个数就是非零元素的个数。

A 正确

B 错误

提交

单选题 5分

设置

6. 稀疏矩阵压缩存储后，必会失去随机存取功能。

A 正确

B 错误

提交