大数据的矩阵计算基础——第13周

法律声明

【声明】本视频和幻灯片为炼数成金网络课程的教学资料，所有资料只能在课程内使用，不得在课程以外范围散播，违者将可能被追究法律和经济责任。

课程详情访问炼数成金培训网站

http://edu.dataguru.cn

关注炼数成金企业微信

■提供全面的数据价值资讯，涵盖商业智能与数据分析、大数据、企业信息化、数字化技术等，各种高性价比课程信息，赶紧掏出您的手机关注吧！

本周内容

- 稀疏矩阵
  - 基本概念
  - 压缩储存方法
  - 运算算法

数组的储存

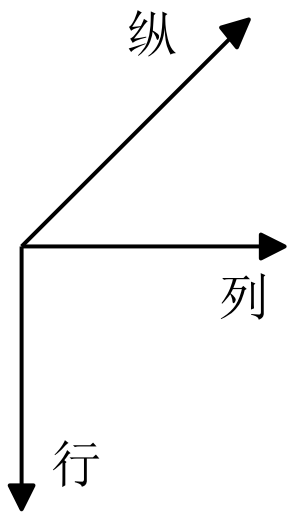
- 数组的顺序存储结构有两种：一种是按行序存储，如高级语言BASIC、 COBOL和PASCAL语言都是以行序为主；另一种是按列序存储，如高级语言中的FORTRAN语言就是以列序为主显然，二维数组$A_{m\times n}$以行为主的存储序列为：

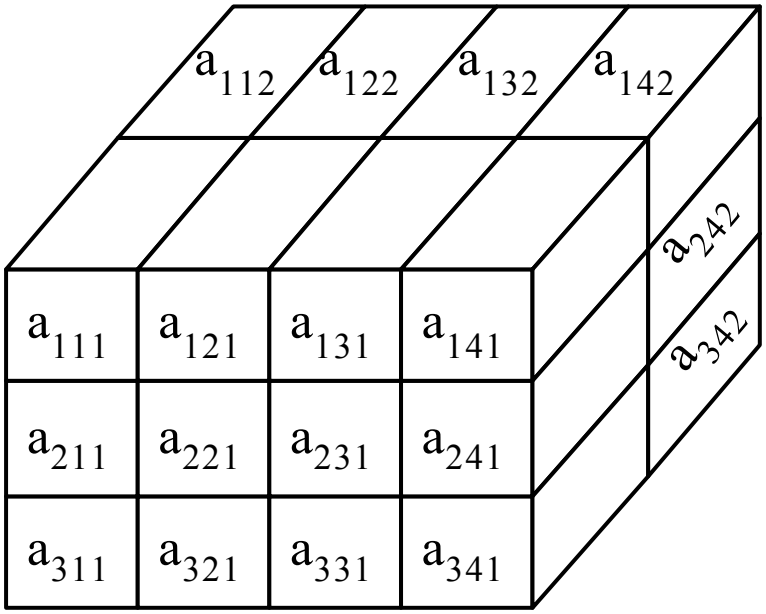
  $a_{11}, a_{12}, \ldots, a_{1n}, a_{21}, a_{22}, \ldots, a_{2n}, \ldots, a_{m1}, a_{m2}, \ldots, a_{mn}$

- 而以列为主的存储序列为：

  $a_{11}, a_{21}, \ldots, a_{m1}, a_{12}, a_{22}, \ldots, a_{m2}, \ldots, a_{1n}, a_{2n}, \ldots, a_{mn}$

数组的储存


纵
列
行


a112
a122
a132
a142
a242
a342
a111
a121
a131
a141
a211
a221
a231
a241
a311
a321
a331
a341

线性表

- 线性表是一种线性数据结构，其用途非常广泛，可用于信息检索、存储管理、模拟技术和通信等领域。

- 线性表：是n个(表长n≥0)同类型数据元素的有限序列。元素间是线性(前后件关系)逻辑关系。

- 特点：
  - 有且仅有一个根结点，它无前件；
  - 有且仅有一个终端结点，它无后件；
  - 除根结点外，每个结点只有一个前件（predecessor）；
  - 除尾结点外，每个结点只有一个后件（successor）。
  - n = 0的线性表为空表。

线性表的顺序储存结构

- 顺序存储结构(Sequential Mapping)：
- 用内存中一组地址连续的单元依次存放表中元素，每个元素的存储空间大小相同。

- 计算元素ai的地址：
  假设每个元素占k个字节，首元素的地址为ADR(a1)，则有:

$$ADR(a_i) = ADR(a_1) + (i-1)\ k$$

- 顺序存储结构是一种随机存取结构。
- 在高级语言环境中，常用一维数组来存储线性表。

顺序表

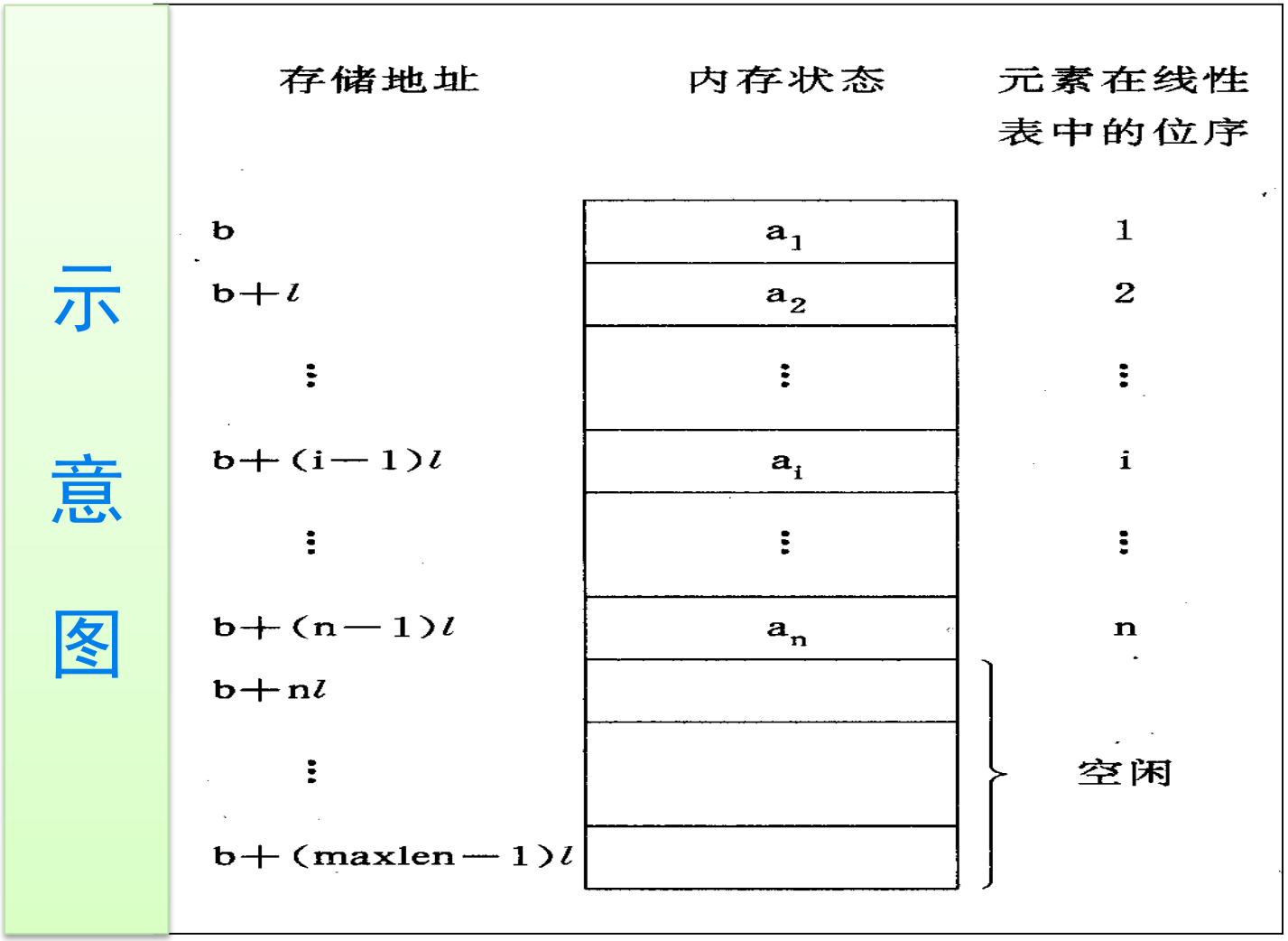
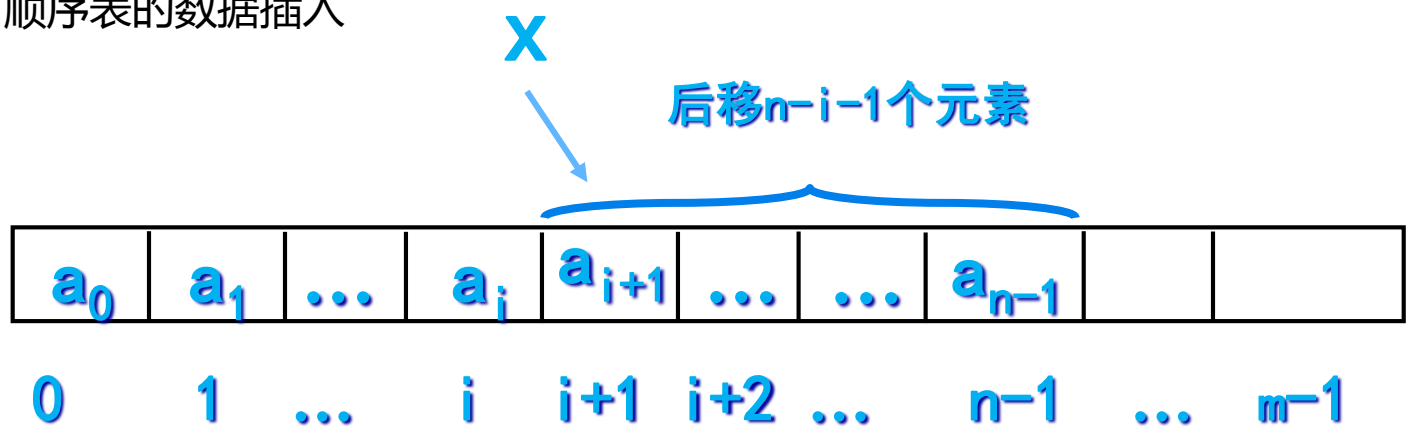


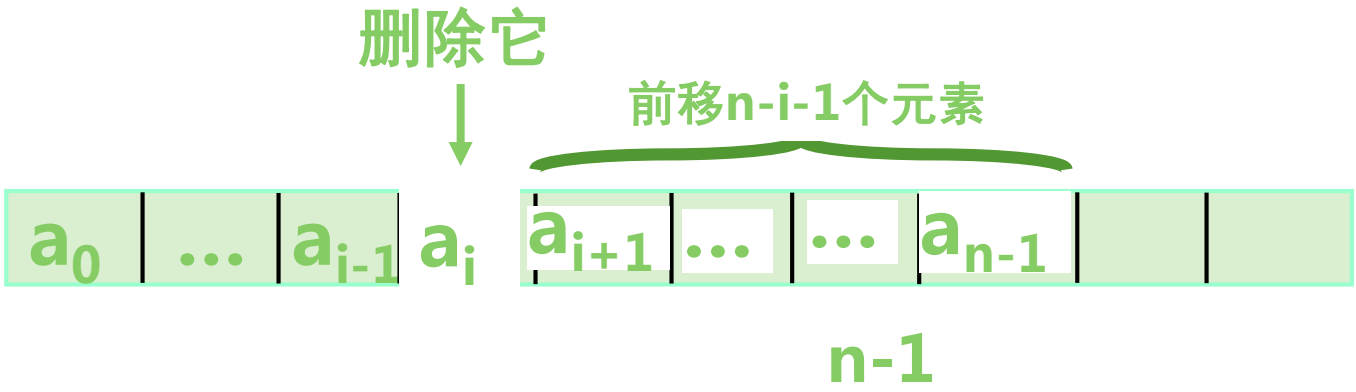
示意图
存储地址
内存状态
元素在线性表中的位序
b
b+l
b+(i-1)l
b+(n-1)l
b+nl
b+(maxlen-1)l
a_1
a_2
a_i
a_n
1
2
i
n
空闲

顺序表

- 顺序表的数据插入



- 顺序表的数据删除



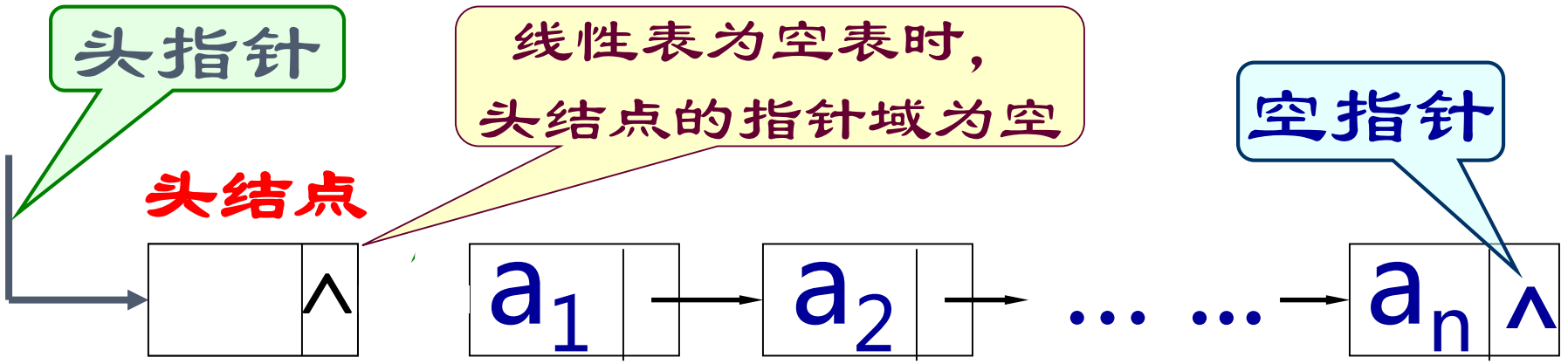
顺序表的优缺点

- 优点：
- (1) 方法简单，各种高级语言中都有数组，容易实现。
  (2) 不用为表示结点间的逻辑关系而增加额外的存储开销。
  (3) 顺序表具有按元素序号随机访问的特点。

- 缺点：
- (1)顺序存储插入与删除一个元素，必须移动大了的数据元素，以此对大的线性表，特别是在元素的插入和删除很频繁的情况下，采取顺序存储很是不方便，效率低；
- (2)顺序存储空间容易满，出现上溢，程序访问容易出问题，顺序存储结构下，存储空间不便扩充；
- (3)顺序存储空间的分配问题，分多了浪费，分少了空间不足上溢。

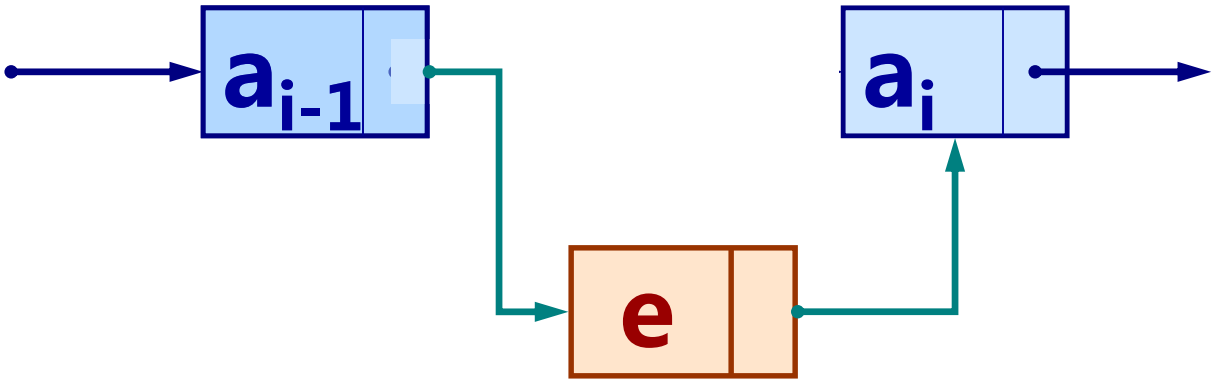
单链表

- 结点
- 数据域
- 指针

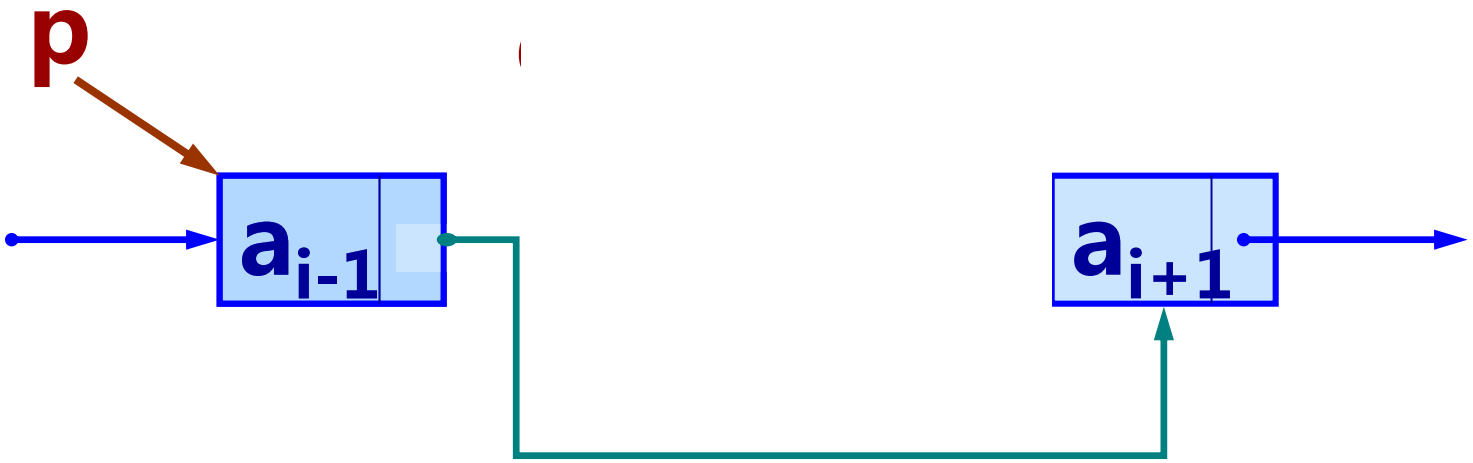


单链表

- 数据的插入



- 数据的删除



双链表

- 双向链表中每个结点除了有向后指针外，还有指向其前一个结点的指针，这样形成的链表中有两条不同方向的链，因此从某一结点均可向两个方向访问。
- 双联表的结点：

- 其中链域prior和next分别指向本结点的直接前趋和直接后继结点。

链表的优缺点

- 优点：

(1) 能有效利用存储空间；

(2) 用“指针”指示数据元素之间的后继关系，便于进行“插入”、“删除”等操作。

- 缺点：

(1)不能随机存取数据元素 ；

(2)它还丢失了一些顺序表有的长处，如线性表的“表长”（单链表中表长是一个隐含值，必须从前到后遍历整个表才能得到）和数据元素在线性表中的“位序”；

(3)不便于在表尾插入元素，需遍历整个表才能找到插入的位置。

稀疏矩阵

- 设矩阵A_{mn}中有s个非零元素，若s远远小于矩阵元素的总数(即s<<m×n)，则称A为稀疏矩阵（Sparse matrix）。

$$A_{6'7}=\begin{pmatrix}0&0&0&22&0&0&15\\0&11&0&0&0&17&0\\0&0&0&-6&0&0&0\\0&0&0&0&0&39&0\\91&0&0&0&0&0&0\\0&0&28&0&0&0&0\end{pmatrix}$$

稀疏矩阵的压缩储存

- 为了节省存储单元，可只存储非零元素
- 由于非零元素的分布一般是没有规律的，因此在存储非零元素的同时，还必须存储非零元素所在的位置（行与列），才能迅速确定一个非零元素是矩阵中的哪一个元素

- 三元组可以采用顺序表示方法，也可以采用链式表示方法，这样就产生了对稀疏矩阵的不同压缩存储方式：顺序存储和链式存储

三元组表

- 三元组表(i,j,value)

- 将表示稀疏矩阵的非零元素的三元组按行优先(或列优先)的顺序排列(跳过零元素)，并依次存放在向量中，这种稀疏矩阵的顺序存储结构称为三元组表。

$$A_{6\times 7}=\begin{pmatrix} 0 & 0 & 0 & 22 & 0 & 0 & 15 \\ 0 & 11 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 39 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 28 & 0 & 0 & 0 & 0 \end{pmatrix}$$

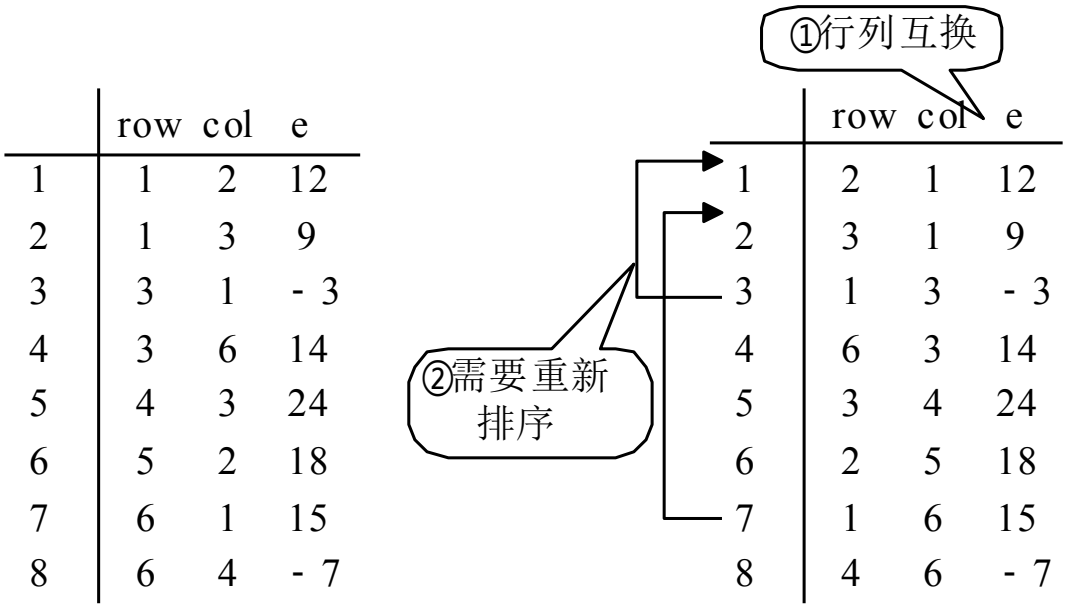
- 三元组表：((1,4,22), (1,7,15), (2,2,11), (3,4,-6), (4,6,39),(6,3,28))

顺序储存

- (1) 用一个二维数组A[0..m ,1..3] : integer
- (2) 存储方法：a[0,1]—存放非零元素个数，a[0,2]—总行数
- a[0,3]—存放总列数
- (3) 按行存放：每个非零元素所在行、列数以及值

稀疏矩阵的转置

- 方法1：简单地交换a->data中i和j中的内容，得到按列优先顺序存储倒b->data;再将b->data重排成按行优先顺序的三元组表。



	row	col	e
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

	row	col	e
1	2	1	12
2	3	1	9
3	1	3	- 3
4	6	3	14
5	3	4	24
6	2	5	18
7	1	6	15
8	4	6	- 7

稀疏矩阵的转置

- 方法2：由于A的列是B的行，因此，按a->data的列序转置，所得到的转置矩阵B的三元组表b->data必定是按行优先存放的。
- 动画演示:

http://student.zjzk.cn/course_ware/data_structure/web/flashhtml/sanyuanzu.htm

	row	col	e
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

(a) 三元组表 A

	row	col	e
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

(b) 三元组表 B

稀疏矩阵的乘法

- 以两个稀疏矩阵相乘为例
- A[6][7]：（（1，2，12），（1，3，9），（3，1，-3），（3，6，14），（4，3，24），（5，2，18），（6，1，15），（6，4，-7））
- B[7][5]：（（1，3，4），（2，1，3），（2，4，2），（3，1，-3），（3，2，1），（4，3，2），（4，2，7），（5，3，8），（6，4，3），（7，2，5））。

顺序储存的缺点

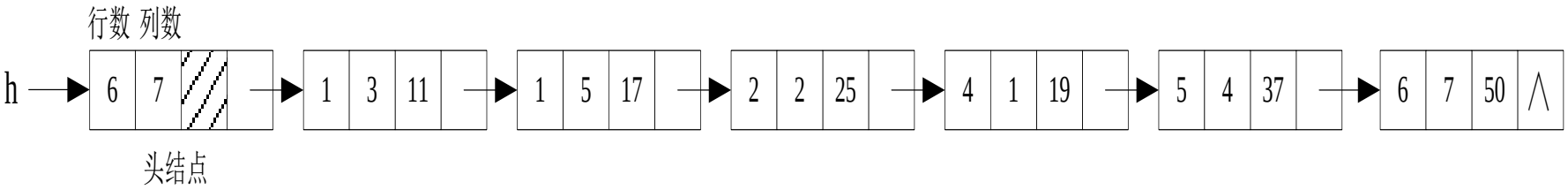
- 与用二维数组存储稀疏矩阵比较，用三元组表表示的稀疏矩阵不仅节约了空间，而且使得矩阵某些运算的运算时间比经典算法还少。但是在进行矩阵加法、减法和乘法等运算时，有时矩阵中的非零元素的位置和个数会发生很大的变化。如A=A+B， 将矩阵B加到矩阵A上，此时若还用三元组表表示法，势必会为了保持三元组表“以行序为主序”而大量移动元素。

稀疏矩阵的三元链表

- (1)三元组链表
- 用链表存储的三元组线性表。
- (2)行指针数组结构的三元组链表
- 把每行非零元素三元组组织成一个单链表，再设计一个指针类型的数组存储所有单链表的头指针。
- (3)三元组十字链表
- 把非零元素三元组按行和按列组织成单链表，这样稀疏矩阵中的每个非零元素三元组结点都将既勾链在行单链表上，又勾链在列单链表上，形成十字形状。

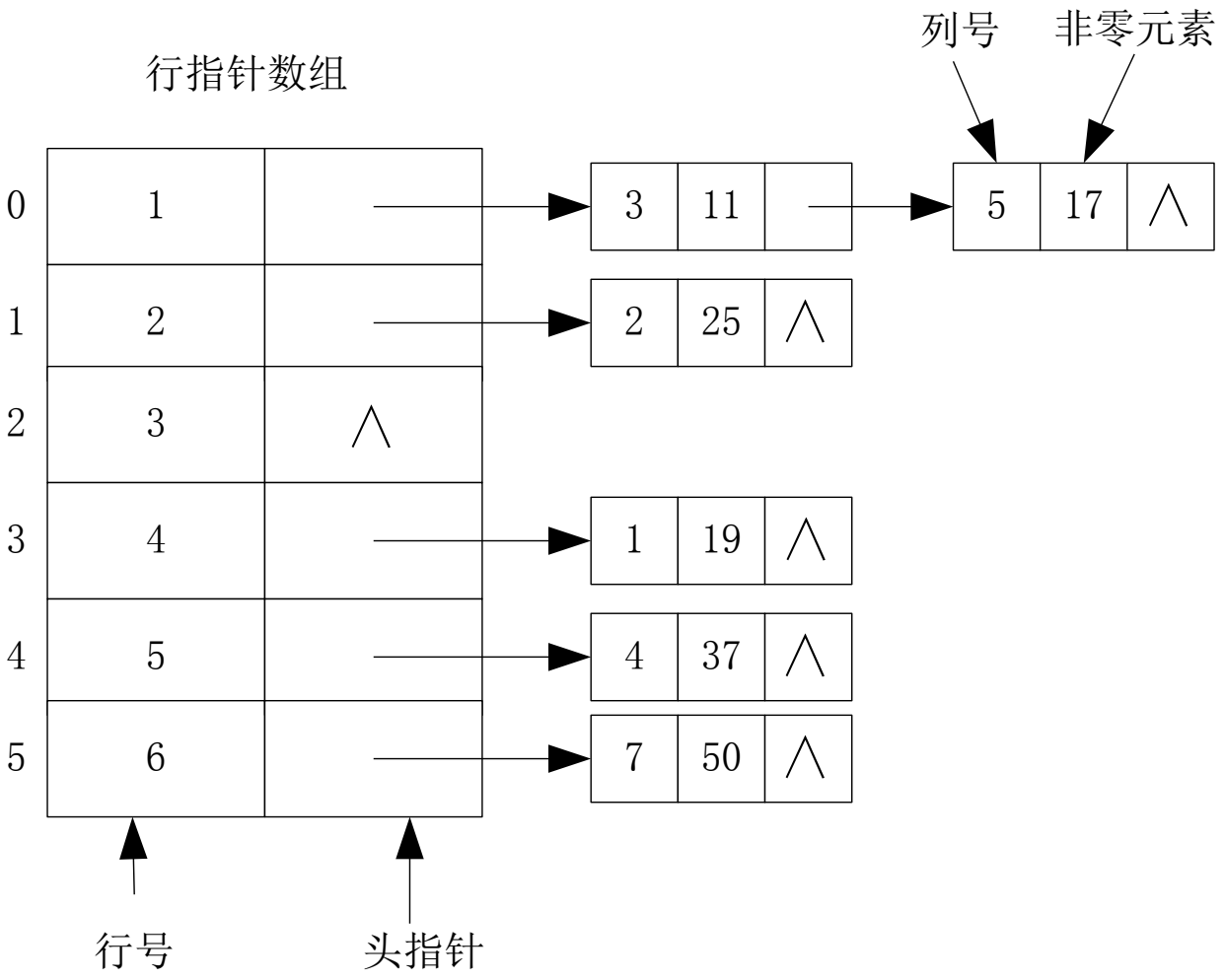
三元链表

- 三元组链表中每个结点的数据域由稀疏矩阵非零元的行号、列号和元素值组成。稀疏矩阵三元组线性表的带头结点的三元组链表结构如图所示，其中头结点的行号域存储了稀疏矩阵的行数，列号域存储了稀疏矩阵的列数。



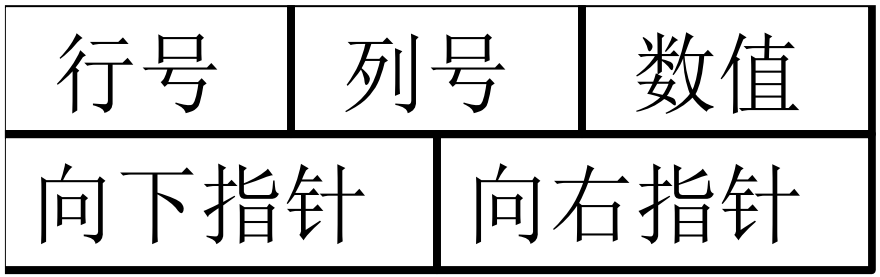
- 这种三元组链表的缺点是实现矩阵运算操作算法的时间复杂度高，因为算法中若要访问某行某列中的一个元素时，必须从头指针进入后逐个结点查找。为降低矩阵运算操作算法的时间复杂度，提出了行指针数组结构的三元组链表，其结构如图所示：

三元链表

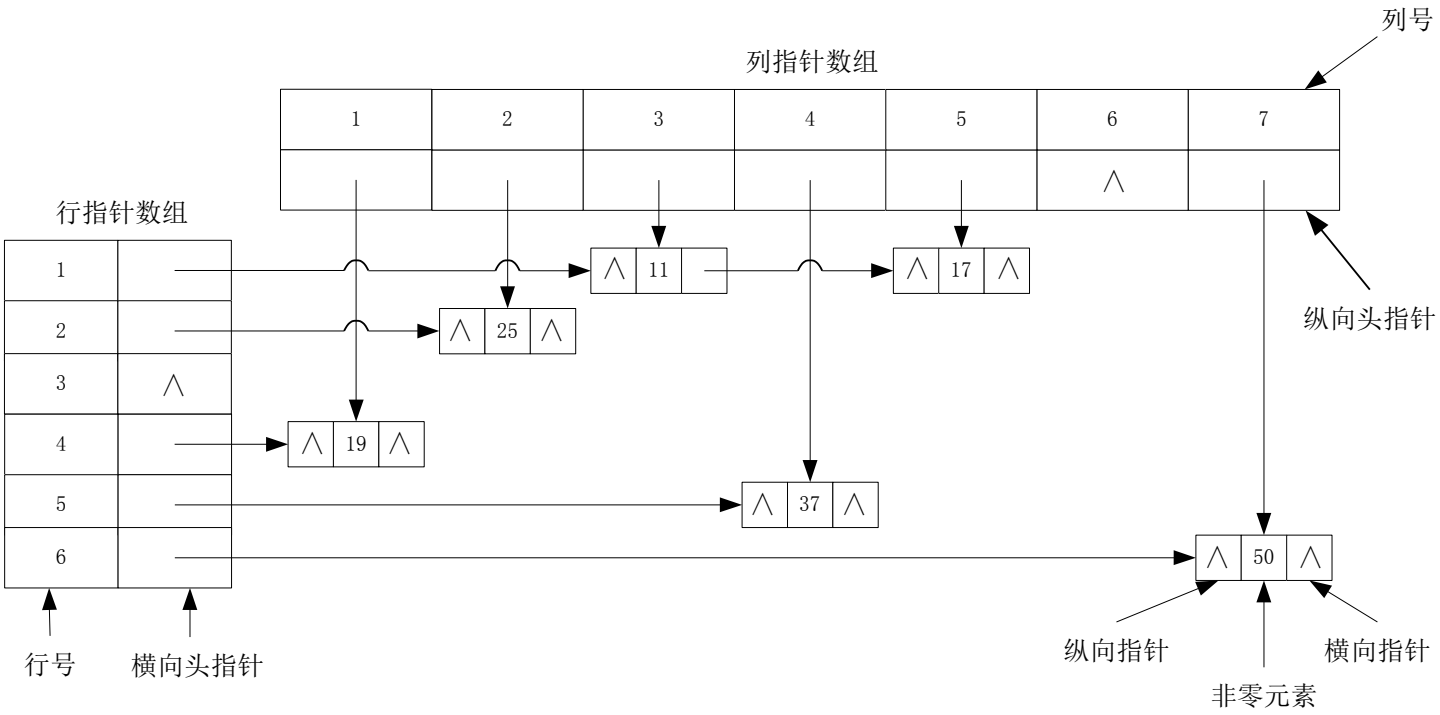


三元链表

- 各单链表均不带头结点。由于每个单链表中的行号域数值均相同，所以单链表中省略了三元组的行号域，而把行号统一放在了指针数组的行号域中。
- 行指针数组结构的三元组链表对于从某行进入后找某列元素的操作比较容易实现，但对于从某列进入后找某行元素的操作就不容易实现，为此又提出了三元组十字链表结构，结构如下：



十字链表


列号
列指针数组
1
2
3
4
5
6
7
行指针数组
1
2
3
4
5
6
11
17
25
19
37
50
纵向头指针
行号
横向头指针
纵向指针
横向指针
非零元素

十字链表

- 表结点
- 行头结点和列头结点
- 总表头结点

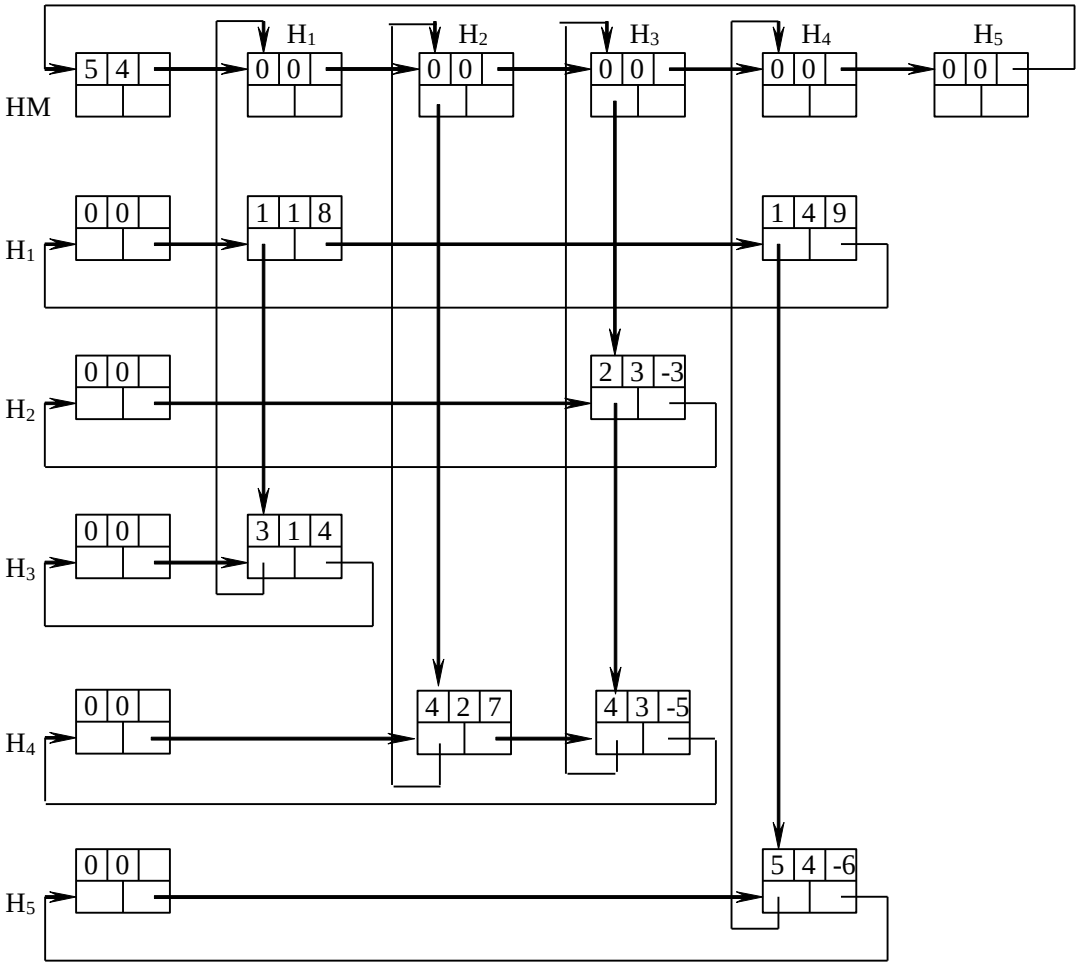
十字链表结点结构

十字链表

- 稀疏矩阵M的十字链表

$$M = \begin{bmatrix} 8 & 0 & 0 & 9 \\ 0 & 0 & -3 & 0 \\ 4 & 0 & 0 & 0 \\ 0 & 7 & -5 & 0 \\ 0 & 0 & 0 & -6 \end{bmatrix}$$

十字链表



储存方式比较

- 三元组顺序表又称有序的双下标法，其特点：非零元在表中按行序有序存储，因此便于进行依行顺序处理的矩阵运算。但是，若需按行号存取某一行的非零元素，则需从头开始进行查找。（时间复杂度较高）
- 行逻辑链接的顺序表：便于随机存取任意一行的非零元。
- 十字链表：当矩阵的非零元个数和位置在操作过程中变化较大时，就不适宜采用顺序存储结构来表示三元组的线性表。

稀疏矩阵的加法

- 基于两个十字链表储存的稀疏矩阵的加法

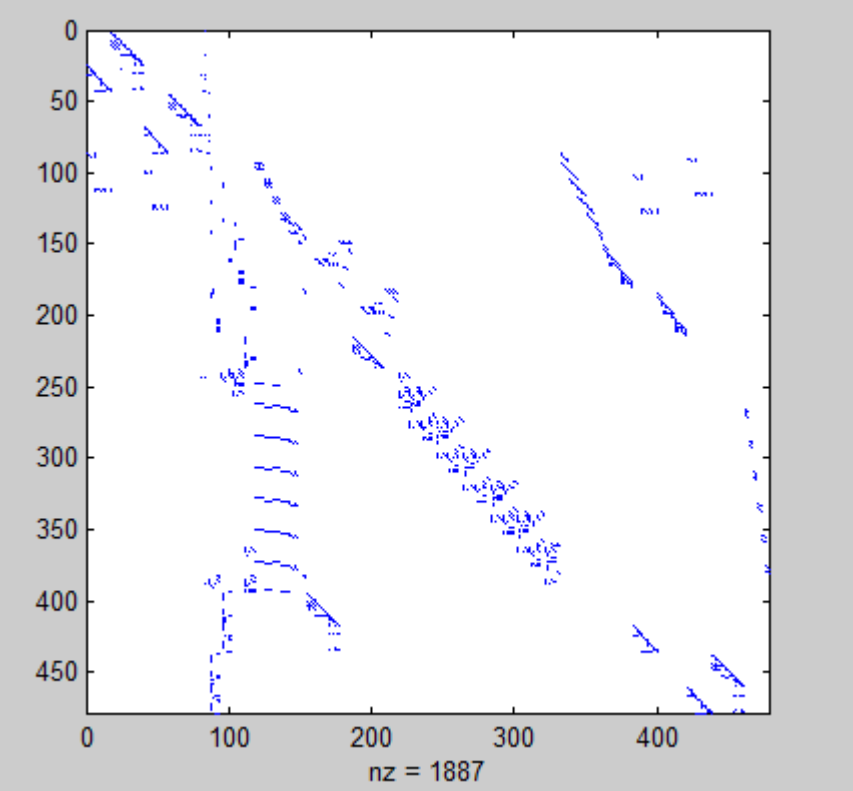
MATLAB中的稀疏矩阵

- 完全矩阵
- 稀疏矩阵

- 稀疏矩阵的构建
- sparse
- spdiags
- spconvert

零元素的观察与图示

- 观察零元素的分布情况
- spy



炼数成金逆向收费式网络课程

- Dataguru（炼数成金）是专业数据分析网站，提供教育，媒体，内容，社区，出版，数据分析业务等服务。我们的课程采用新兴的互联网教育形式，独创地发展了逆向收费式网络培训课程模式。既继承传统教育重学习氛围，重竞争压力的特点，同时又发挥互联网的威力打破时空限制，把天南地北志同道合的朋友组织在一起交流学习，使到原先孤立的学习个体组合成有组织的探索力量。并且把原先动辄成千上万的学习成本，直线下降至百元范围，造福大众。我们的目标是：低成本传播高价值知识，构架中国第一的网上知识流转阵地。
- 关于逆向收费式网络的详情，请看我们的培训网站 http://edu.dataguru.cn

Thanks

FAQ时间