第三章 数组

本章内容

3.1 数组的基本概念

一. 数组的定义

数组 是下标与值组成的偶对的有穷集合。

二. 数组的基本操作

对数组的操作是基于下标进行的

1. 给定一组下标，检索、存取或修改相应元素的值。
2. 检索满足条件的数组元素。
3. 对数组元素进行排序。

一般情况下,数组没有插入、删除运算。因此，数组的规模是固定的。称为静态结构

3.2 数组的存储结构

一. 一维数组A[1..n]

$$A[1..n] = (\ a_1,\ a_2,\ a_3,\ \ldots,\ a_{n-1},\ a_n\ )$$

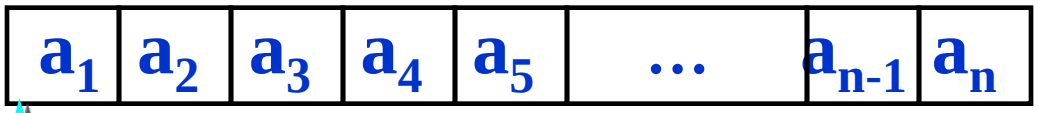
典型的线性结构

1. 一维数组的特点

除了第一个元素外，其他每个元素有且仅有一个直接前驱元素；

除了最后那个元素外，其他每个元素有且仅有一个直接后继元素。

2. 一维数组的存储



$LOC(a_1)$

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$LOC(a_1)$，则

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

二. 二维数组A[1..m, 1..n]

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

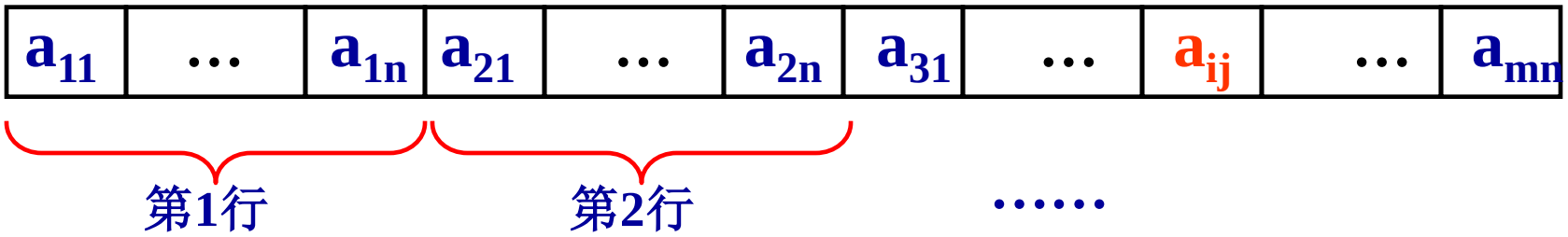
二维数组的存储

行序为主序分配方式

列序为主序分配方式

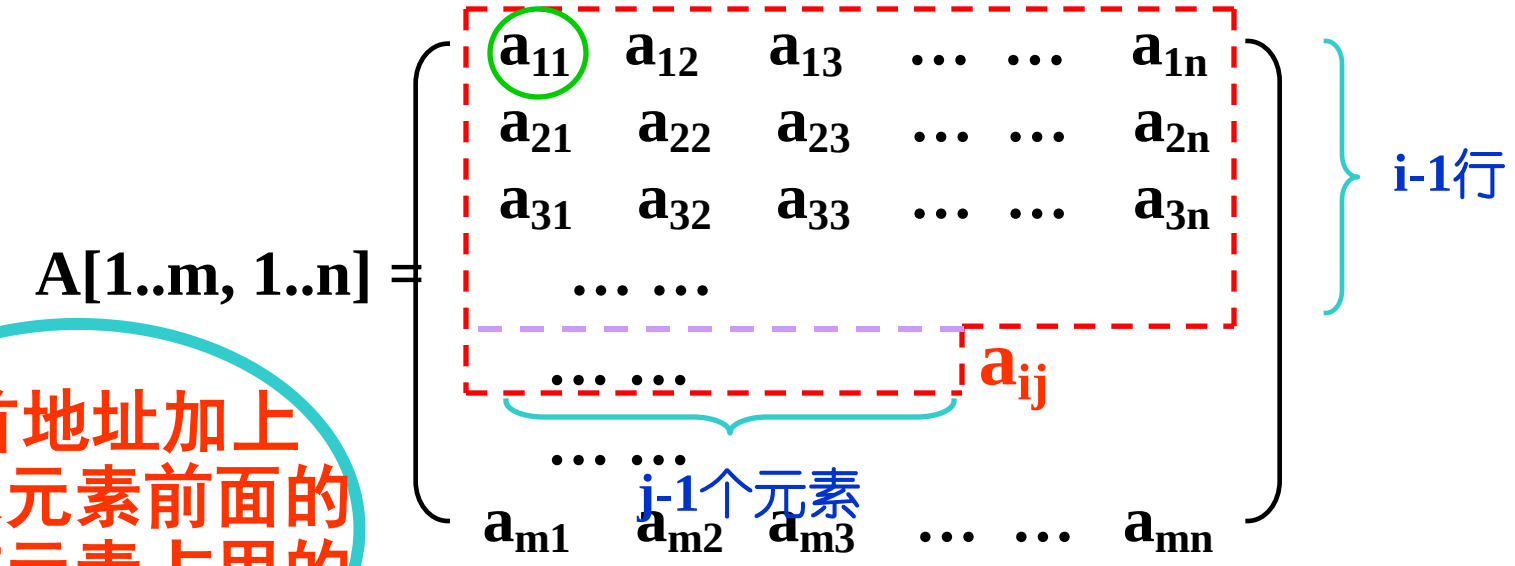
1. 行序为主序分配方式

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



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。



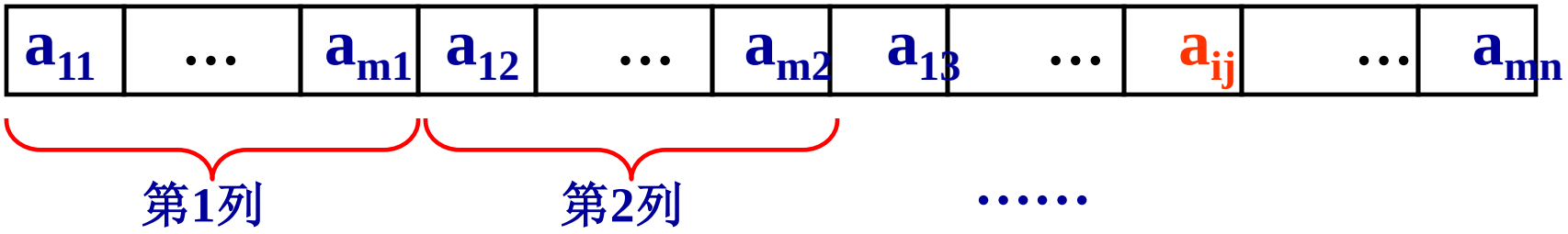
首地址加上被求元素前面的所有元素占用的单元数

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

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

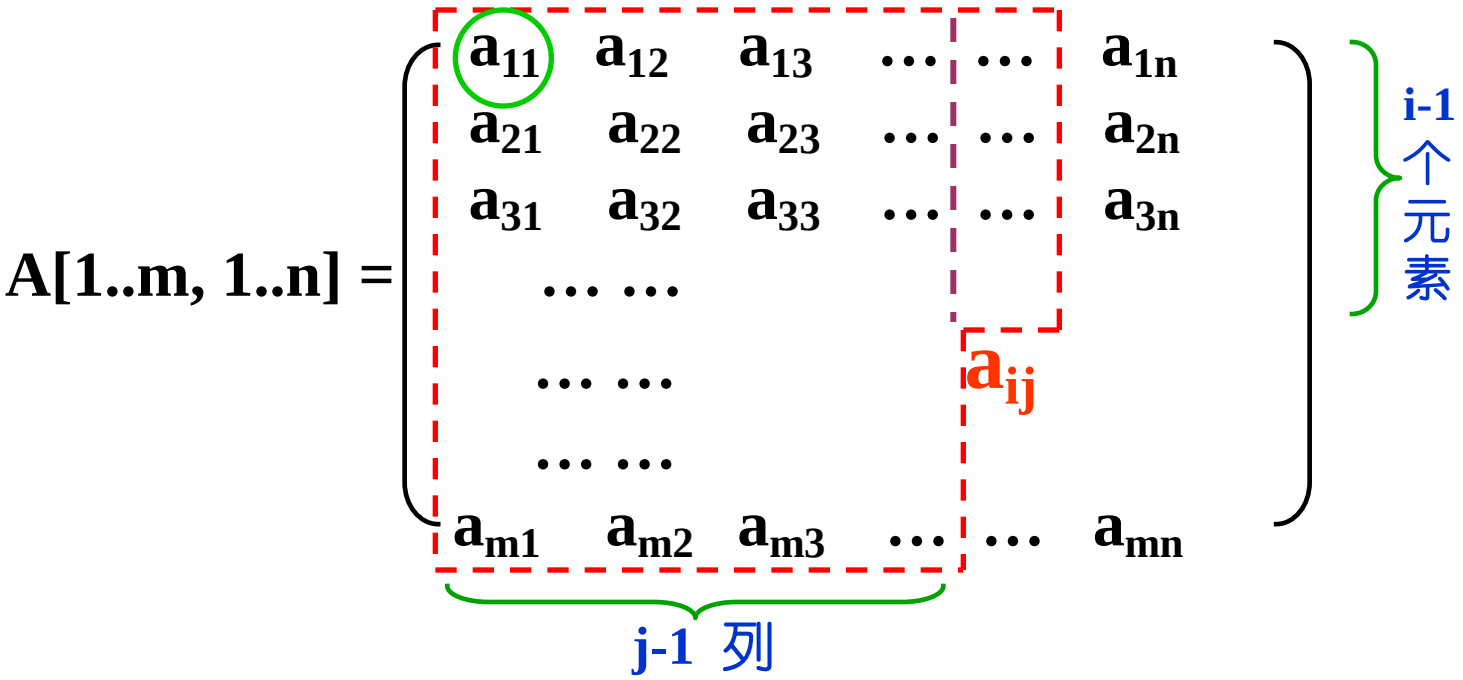
2. 列序为主序分配方式

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



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。



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

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

3.3 特殊矩阵的压缩存储

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。

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

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。

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

传统做法：

定义一个二维数组 A[0..n-1][0..n-1]

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

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

a_{11}	a_{21}	a_{22}	…	a_{ij}	…	a_{nn}

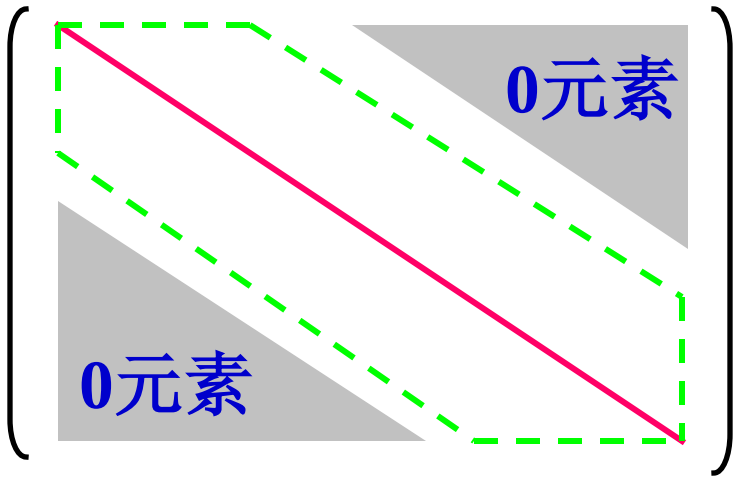
k = 0　1　2　… …　n*(n+1)/2−1

A中任意一元素a_{ij}与LTA[k] 之间存在对应关系：

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

二. 对角矩阵的压缩存储

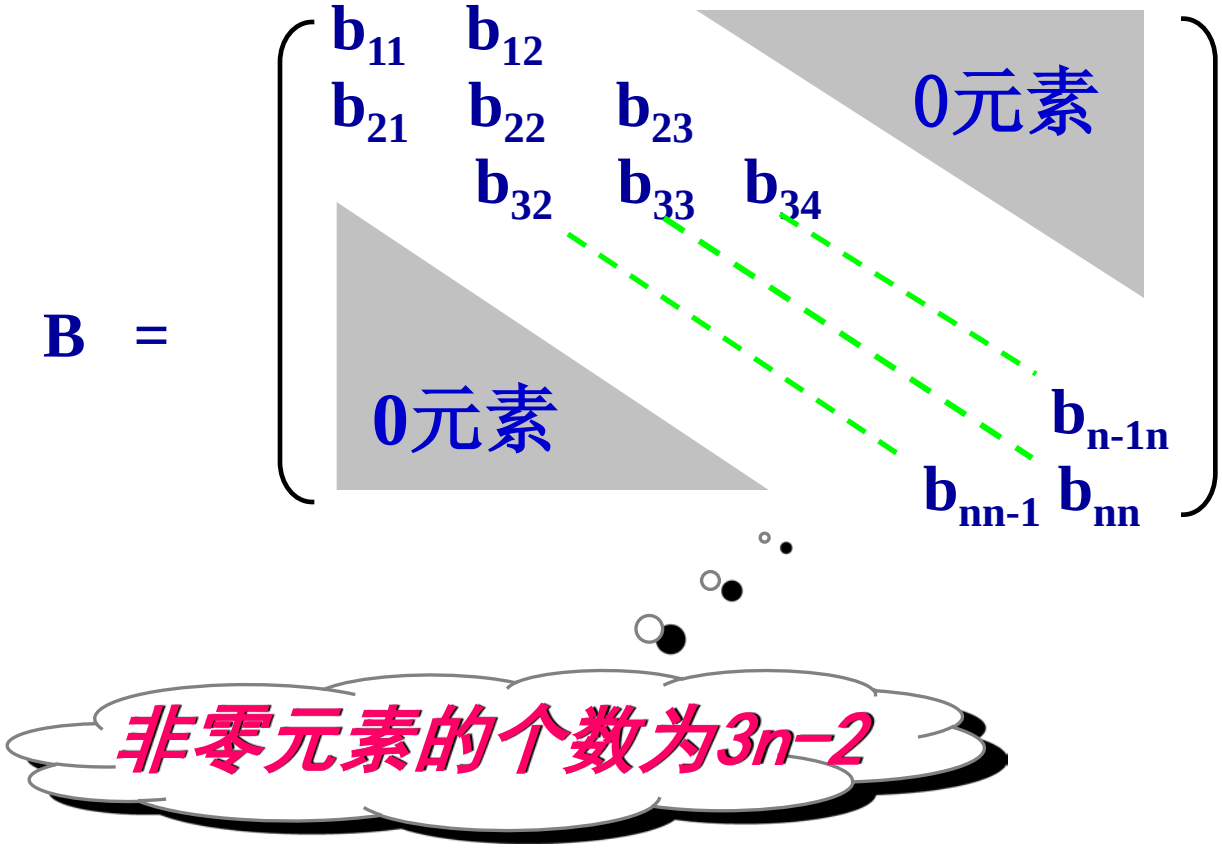
若一个矩阵中，值非0的元素对称地集中在主对角线两旁的一个带状区域中(该区域之外的元素都为0元素)，称这样的矩阵为**对角矩阵**。

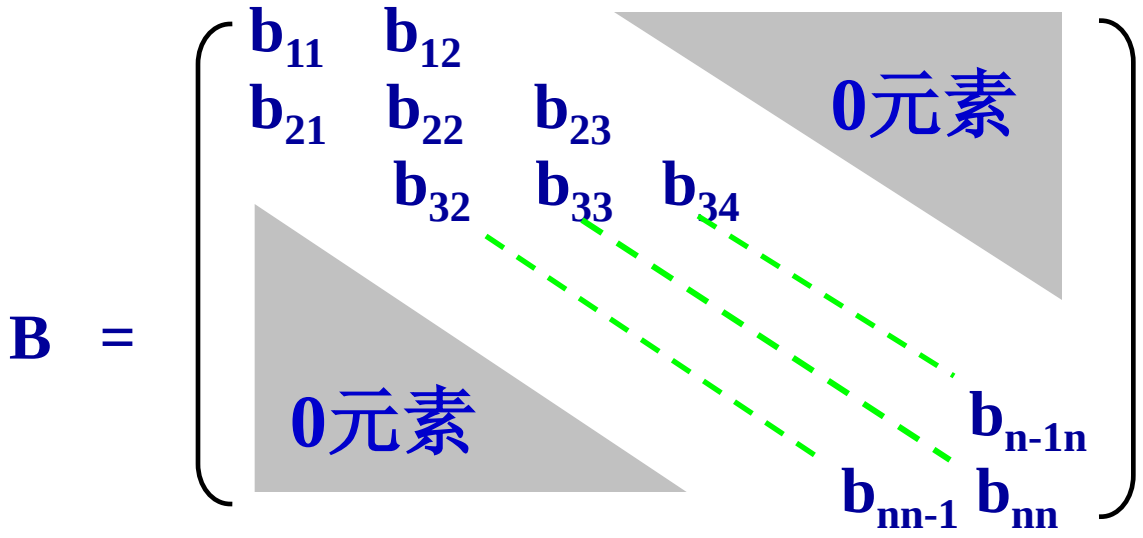


传统做法：

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





LTB[0:3n–3]

b_{11}	b_{12}	b_{21}	b_{22}	…	b_{ij}	…	b_{nn}
k = 0	1	2	3	…	…		3n–3

B中任一非零元素 b_{ij} 与LTB[K] 之间存在对应关系：

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

3.4 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 A[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

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

三元组表

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

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

传统做法

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

三元组法

思考题

1．有人说，数组除了插入和删除操作以外，主要操作还有存取、修改、检索和排序，你认为如何？

2. 对于一个具有t个非零元素的 m×n 阶矩阵，若采用三元组表方法存储，则当t 满足什么条件，这样做才有意义？

答案

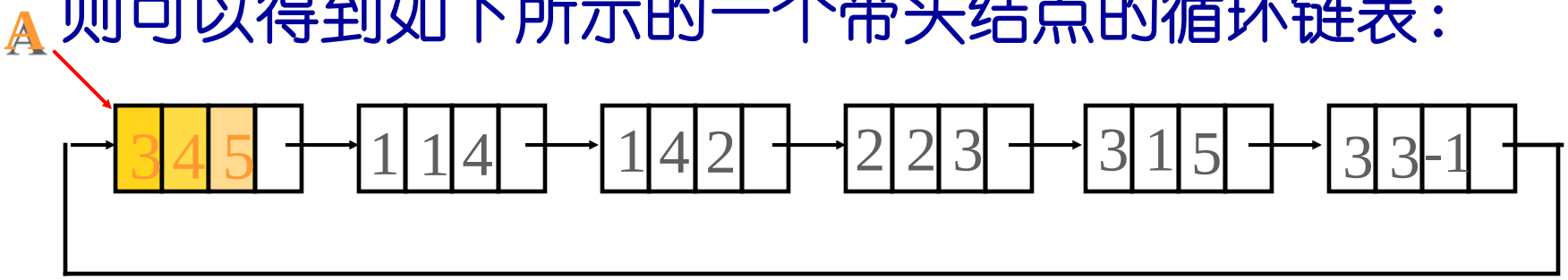
$$t < \frac{m \times n}{3} - 1$$

3.5 稀疏矩阵的十字链表表示

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：

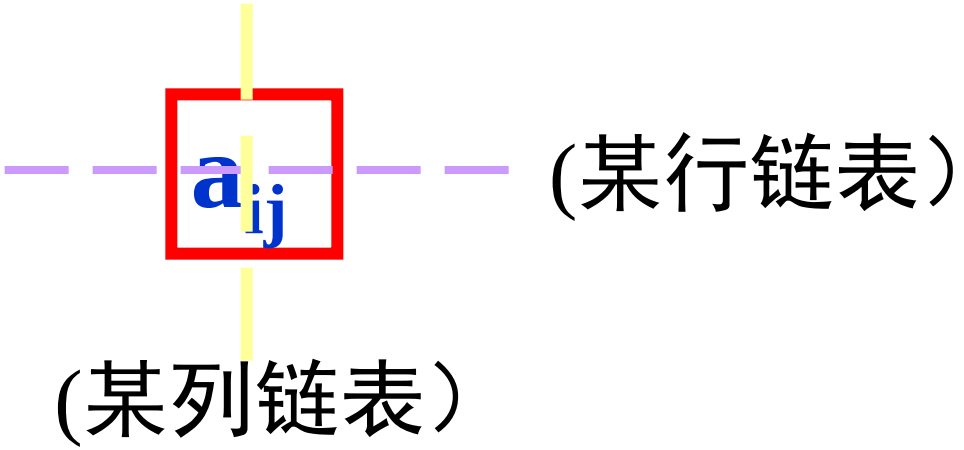


缺点

要确定一个元素比较麻烦，导致相关操作效率低

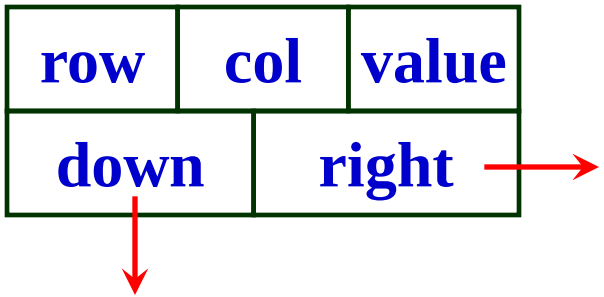
十字链表

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。



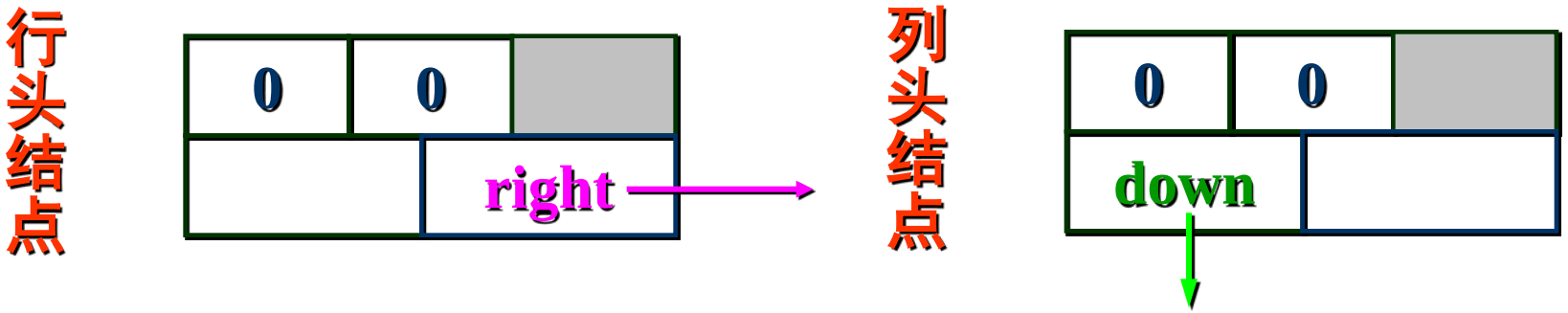
对于一个m×n的稀疏矩阵，分别建立m个行的循环链表与n个列的循环链表，每个非零元素用一个链结点存储。

链结点的构造为：

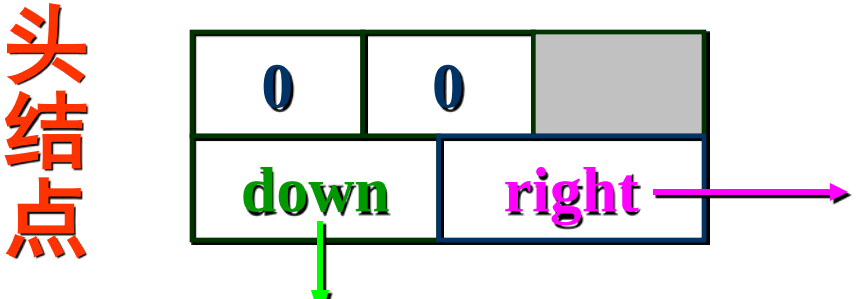


其中，**row, col, value** 分别表示某个非零元素所在的行号、列号和元素的值；**down** 和 **right** 分别为向下与向右指针，分别用来链接同一列中的与同一行中的所有非零元素对应的链结点。

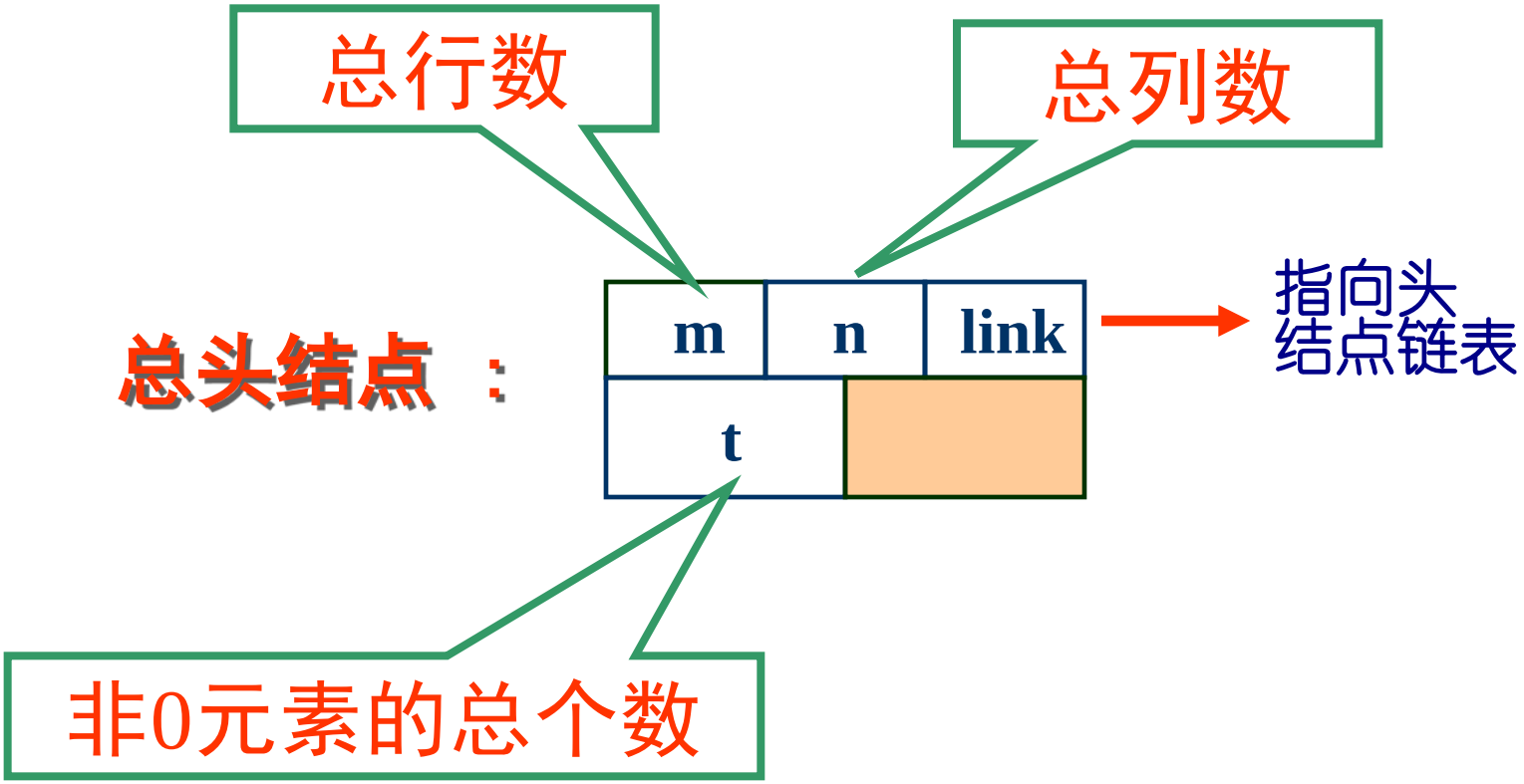
对m个行链表，分别设置m个行链表表头结点。 表头结点的构造与链表中其他链结点一样，只是令row与col 的值分别为0， right域指向相应行链表的第一个链结点。同理，对n个列链表，分别设置n个列链表表头结点。其中，令row和col的值分别为0，down域指向相应列链表的第一个链结点。



一共设置MAX(m, n)个头结点.



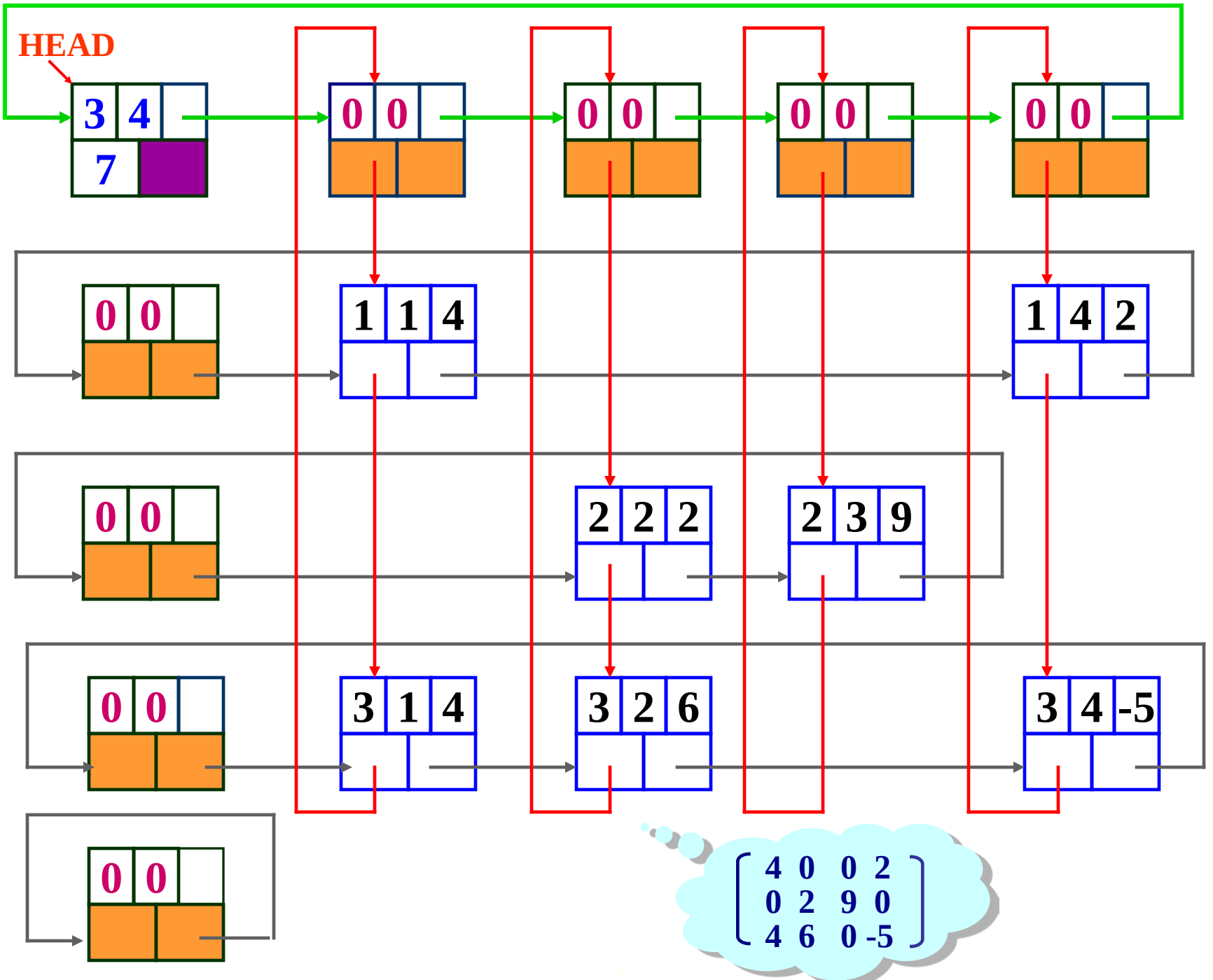
另外，再设置一个总的头结点(如下所示)，通过link域把所有表头结点也链接成一个循环链表。



对于如下稀疏矩阵**B**,

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

十字链表表示如下：


HEAD
3 4
7
0 0
0 0
0 0
0 0
0 0
0 0
1 1 4
1 4 2
0 0
2 2 2
2 3 9
0 0
3 1 4
3 2 6
3 4 -5
0 0
4 0 0 2
0 2 9 0
4 6 0 -5

3.5 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:

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

一个标准的6阶多项式

例如

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

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

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

非标准多项式

方法一

定义一个一维数组A[0..n+1]来表示多项式。其中，
A[0]用来存放多项式的阶数n；
A[1]到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]来表示多项式。其中，
A[0]存放系数非零项的总项数m；
A[1]到~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

对于多项式 $C(x)=x^{2000}-5$, 有

C[0..4]

2	1	2000	-5	0

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。


```
void  MAGIC( int A[][n], int n )
{
    int  i, j, num;
    for ( i=0; i<n; i++ )
        for ( j=0;  j<n; j++ )
            A[i] [j]=0;                          /* 魔方清零 */
    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;
        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 */
    }
}
```

本章内容小结

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

本章内容不作为重点要求