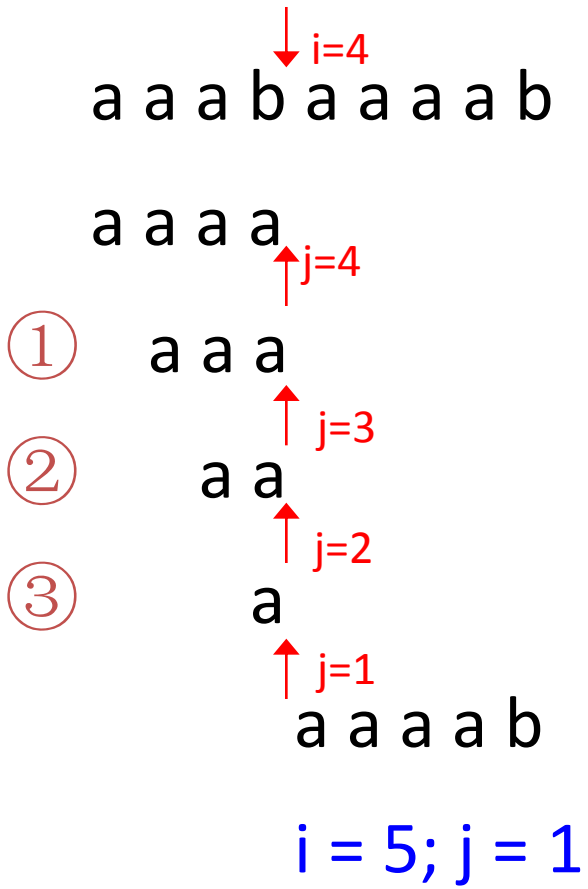
- KMP算法的时间复杂度

设主串s的长度为n,模式串t长度为m,在KMP算法中求next数组的时间复杂度为O(m),在后面的匹配中因主串s的下标不减即不回溯,比较次数可记为n,所以KMP算法总的时间复杂度为O(n+m)。

- **next函数的改进**



j	1 2 3 4 5
模式	a a a a b
next[j]	0 1 2 3 4
nextval[j]	0 0 0 0 4

next[j] = k，而$p_j=p_k$，则主串中s_i和p_j不等时，不需再和p_k进行比较，而直接和$p_{next[k]}$进行比较。

j	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
模式串	**a**	**b**	**c**	**a**	**a**	**b**	**b**	**c**	**a**	**b**	**c**	**a**	**a**	**b**	**d**	**a**	**b**
next[j]	**0**	**1**	**1**	**1**	**2**	**2**	**3**	**1**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**1**	**2**
nextval[j]	0	1	1	0	2	1	3	1	0	1	1	0	2	1	7	0	1

- **void get_nextval(SString T, int &nextval[])**

```
{
    i= 1; nextval[1] = 0; j = 0;
    while( i<T[0]){
        if(j==0 || T[i] == T[j]){
            ++i; ++j;
            if(T[i] != T[j]) nextval[i] = j;
            else  nextval[i] = nextval[j];
        }
        else  j = nextval[j];
    }
}
```

next[i] = j;

4.4 数组

本节所讨论的数组与高级语言中的数组区别：

- 高级语言中的数组是顺序结构；
- 而本章的数组既可以是顺序的，也可以是链式结构，用户可根据需要选择。

数组的抽象数据类型

ADT Array {

数据对象:

$$j_i = 0, \cdots b_i - 1, \mathrm{i} = 1,2,\cdots,\mathrm{n}$$

$$D = \{\mathrm{a}_{\mathrm{j}\ \mathrm{j}_2\cdots\mathrm{j}_\mathrm{n}} \mid \mathrm{a}_{\mathrm{j}_1\mathrm{j}_2\cdots\mathrm{j}_\mathrm{n}} \in ElemSet$$

数据关系:

$$R_1 = \{< a_{j_1\cdots j_i\cdots j_n}, a_{j_1\cdots j_i+1\cdots j_n} > |$$

$$0 \le \mathrm{j_k} \le b_k - 1,\ \ 1 \le \mathrm{k} \le \mathrm{n},\ \text{且}\mathrm{k} \ne \mathrm{i},$$

$$0 \le \mathrm{j_i} \le b_k - 2,$$

$$a_{j_1\cdots j_i\cdots j_n}, a_{j_1\cdots j_i+1\cdots j_n} \in D, i = 2,\cdots,n\}$$

基本操作:

(1) InitArray (&A,n,bound1, …boundn)
//构造数组A

(2) DestroyArray (&A) // 销毁数组A

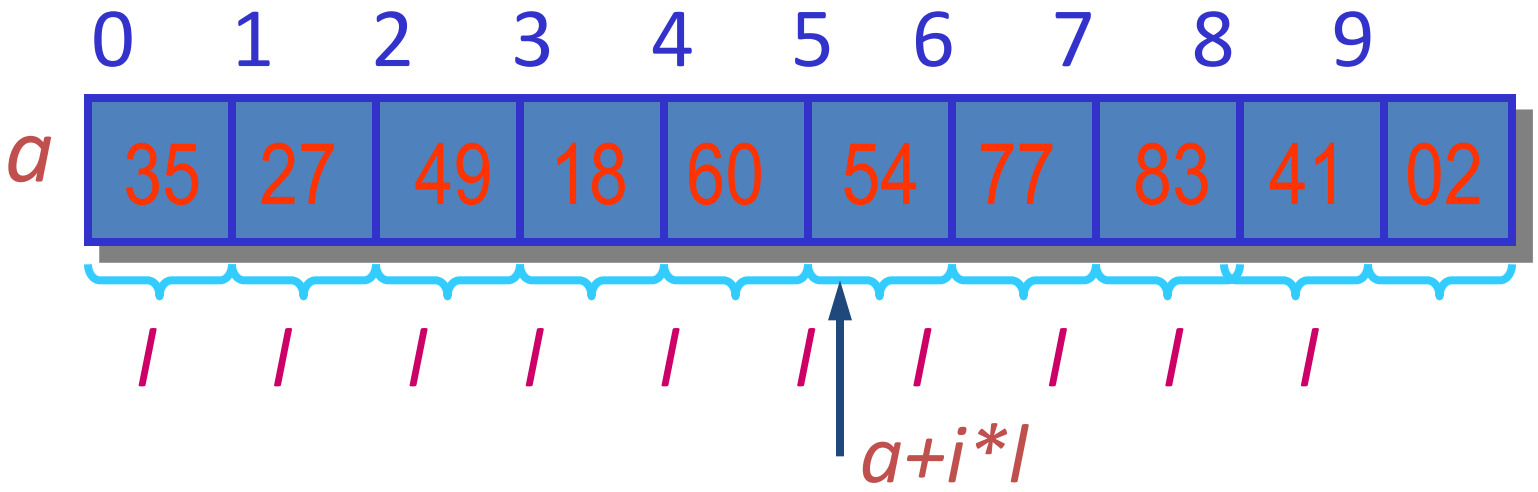
(3) Value(A,&e,index1,…,indexn) //取数组元素值

(4) Assign (A,&e,index1,…,indexn) //给数组元素赋值

}ADT Array

一维数组

$$LOC(i) = \begin{cases} a, & i = 0 \\ LOC(i-1)+l = a+i*l, & i > 0 \end{cases}$$



$$LOC(i) = LOC(i-1)+l = a+i*l$$

二维数组

$A=(\alpha_1,\alpha_2,\cdots,\alpha_p)$ (p=m或n)

$\alpha_i=(a_{i1},a_{i2},\cdots,a_{in})\quad 1\le i\le m$

$\alpha_j=(a_{1j},a_{2j},\cdots,a_{mj})\quad 1\le j\le n$

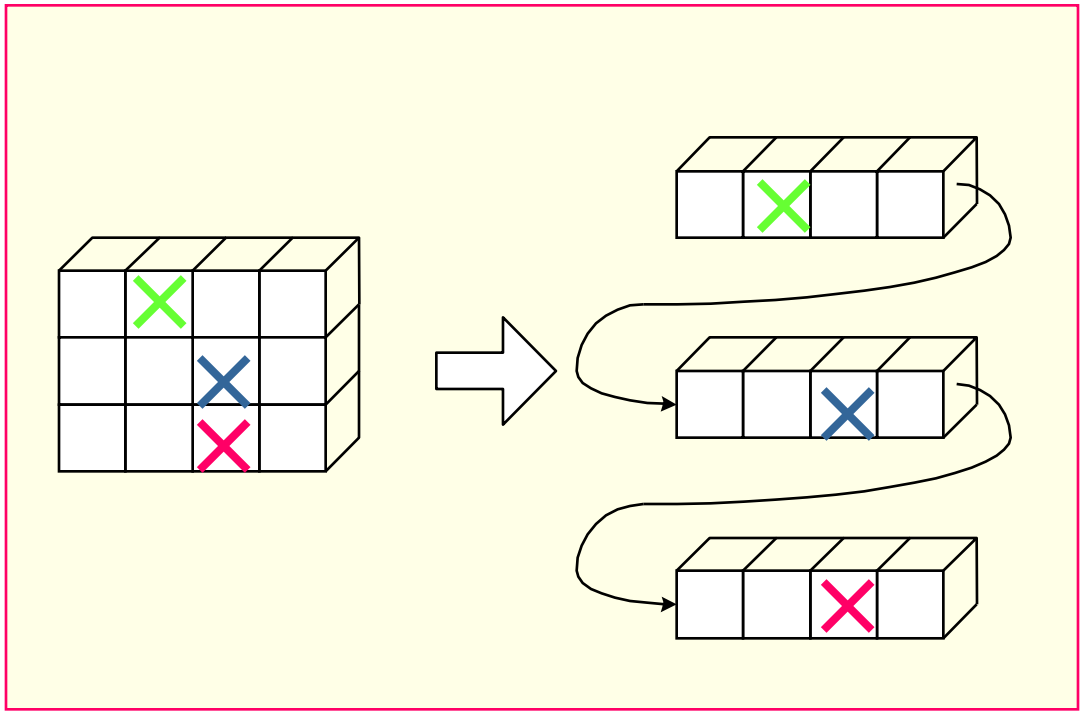
$$A_{m\times n}=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

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

数组的顺序存储

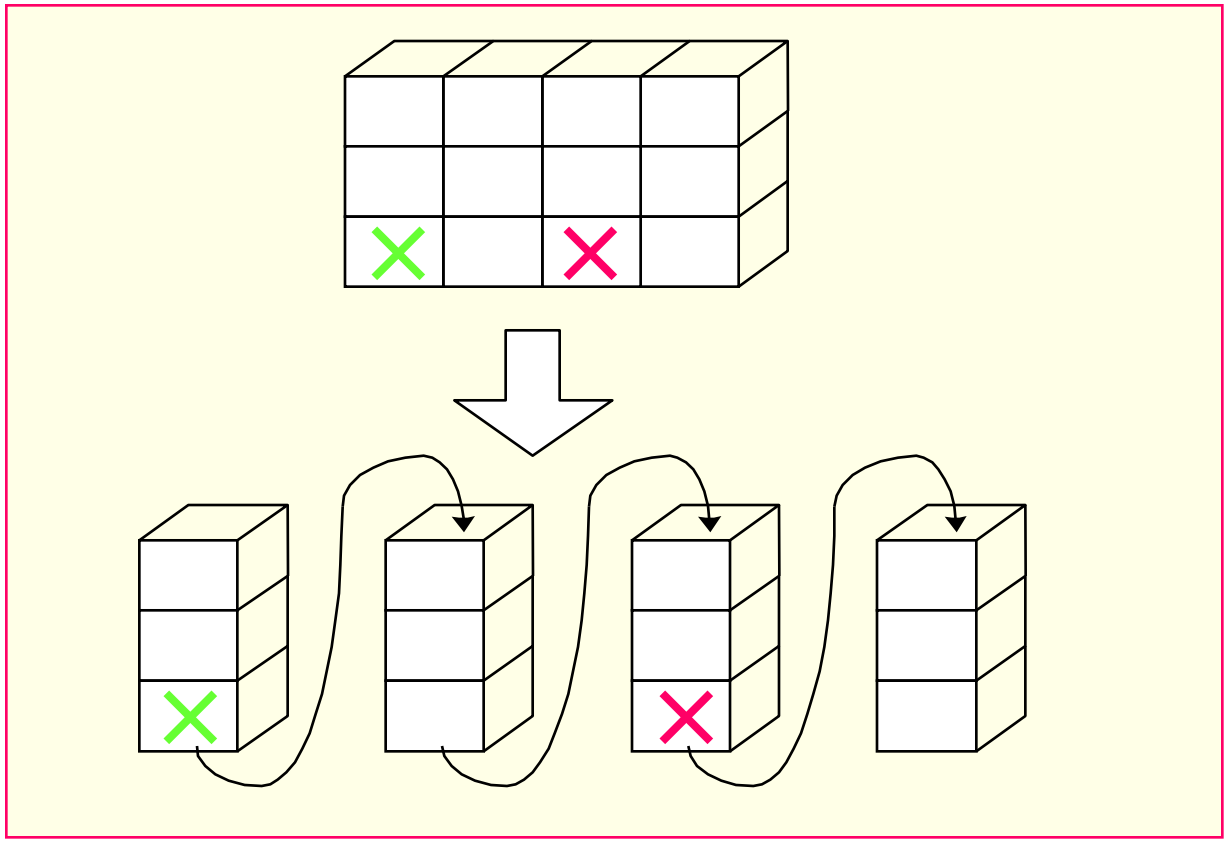
- 以行序为主序

  C, PASCAL



- 以列序为主序

FORTRAN



二维数组的行序优先表示

a[n][m]

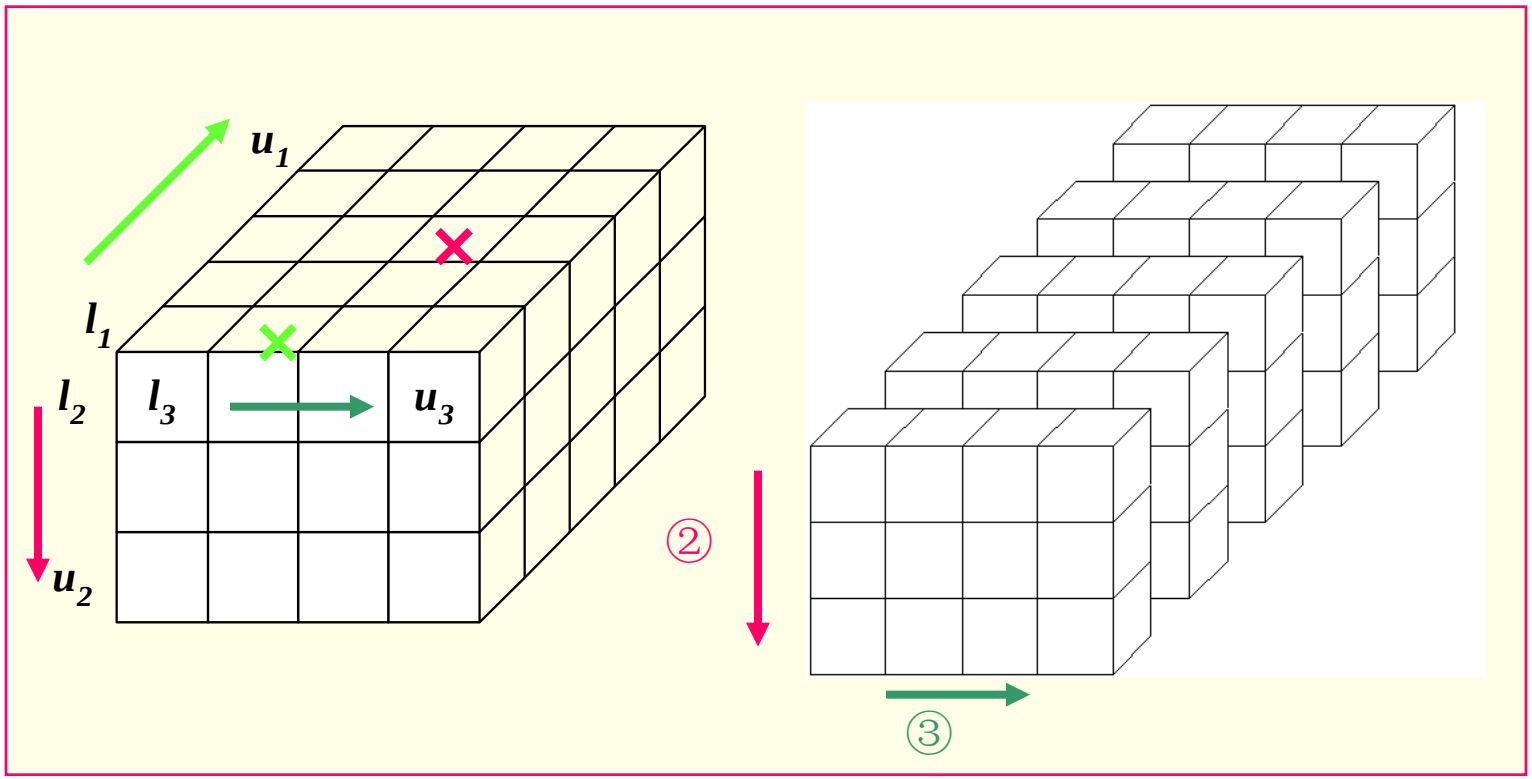
$$\boldsymbol{a}=\begin{pmatrix} a[0][0] & a[0][1] & \cdots & a[0][m-1] \\ a[1][0] & a[1][1] & \cdots & a[1][m-1] \\ a[2][0] & a[2][1] & \cdots & a[2][m-1] \\ \vdots & \vdots & \ddots & \vdots \\ a[n-1][0] & a[n-1][1] & \cdots & a[n-1][m-1] \end{pmatrix}$$

设数组开始存放位置 LOC(0, 0) = *a*

LOC (*j*, *k*) = *a* + *j* * *m* + *k*

三维数组

按页/行/列存放，页优先的顺序存储



三维数组

- a[m1][m2] [m3] 各维元素个数为 m_1, m_2, m_3
- 下标为 i_1, i_2, i_3的数组元素的存储位置：

$$LOC(i_1, i_2, i_3) = a + i_1 * m_2 * m_3 + i_2 * m_3 + i_3$$

- $i_1 * m_2 * m_3$：前i_1页总元素个数
- $i_2 * m_3$：第i_1页的前i_2行总元素个数
- i_3：第 i_2 行前 i_3 列元素个数

n维数组

- 各维元素个数为 $m_1, m_2, m_3, \ldots, m_n$
- 下标为 $i_1, i_2, i_3, \ldots, i_n$ 的数组元素的存储位置：

$$LOC(i_1, i_2, \cdots, i_n) = \mathrm{a} + i_1 * m_2 * m_3 * \cdots * m_n +$$

$$+ i_2 * m_3 * m_4 * \cdots * m_n + \cdots\cdots + i_{n-1} * m_n + i_n$$

$$= \mathrm{a} + \left(\sum_{j=1}^{n-1} i_j * \prod_{k=j+1}^{n} m_k \right) + i_n$$

n维数组

$$LOC[j_1, j_2, \cdots, j_n] = LOC[0,0,\cdots,0] + \left(\sum_{i=1}^{n} c_i j_i\right) \mathrm{L}$$

$$c_n = L, c_{i-1} = b_i \times c_i, \ 1 < \mathrm{i} \leq \mathrm{n}$$

练习

设有一个二维数组$A[m][n]$按行优先顺序存储，假设$A[0][0]$存放位置在$644_{(10)}$，$A[2][2]$存放位置在$676_{(10)}$，每个元素占一个空间，问$A[3][3]_{(10)}$存放在什么位置？脚注$_{(10)}$表示用10进制表示。

设数组元素A[i][j]存放在起始地址为Loc (i, j) 的存储单元中

∵ Loc (2, 2) = Loc (0, 0) + 2 * n + 2 = 644 + 2 * n + 2 = 676.
∴ n = (676 - 2 - 644) / 2 = 15
∴ Loc (3, 3) = Loc (0, 0) + 3 * 15 + 3 = 644 + 45 + 3 = 692.

练习

设有二维数组A[10,20]，其每个元素占两个字节，A[0][0]存储地址为100，若按行优先顺序存储，则元素A[6,6]的存储地址为_____，352 优先顺序存储，元素A[6,6]的存储地址为_____。 232

(6*20+6)*2+100=352

(6*10+6)*2+100=232

特殊矩阵的压缩存储

1. 什么是压缩存储？

若多个数据元素的值都相同，则只分配一个元素值的存储空间，且零元素不占存储空间。

2. 什么样的矩阵能够压缩？

一些特殊矩阵，如：对称矩阵，对角矩阵，三角矩阵，稀疏矩阵等。

3. 什么叫稀疏矩阵？

矩阵中非零元素的个数较少（一般小于5%）

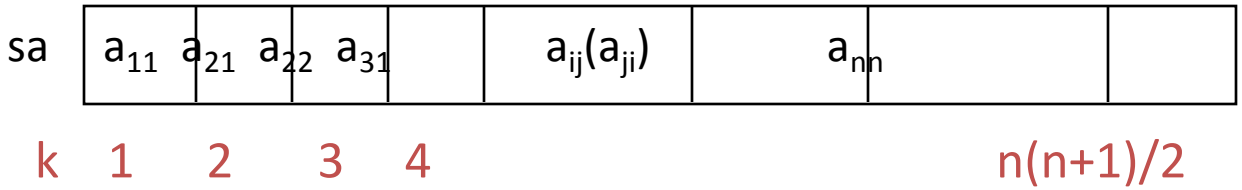
数组下标(i,j)　确定　存储地址

1. 对称矩阵

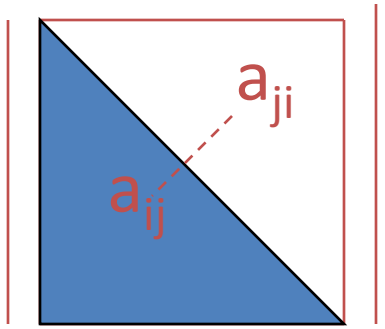
[特点] 在n×n的矩阵a中，满足如下性质：

$$a_{ij}=a_{ji}\ \ (1 \le i, j \le n)$$

[存储方法] 只存储下(或者上)三角(包括主对角线)的数据元素。共占用n(n+1)/2个元素空间。



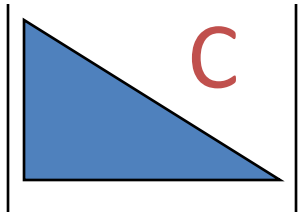
$$k=\begin{cases} i(i-1)/2+j & \text{当} i \ge j \\ j(j-1)/2+i & \text{当} i < j \end{cases}$$



2. 三角矩阵

[特点] 对角线以下(或者以上)的数据元素(不包括对角线)全部为常数c。

上三角矩阵



下三角矩阵

[存储方法] 重复元素c共享一个元素存储空间，共占用n(n+1)/2+1个元素空间: sa[1.. n(n+1)/2+1]

上三角矩阵

$$k=\begin{cases}(i-1)\times(2n-i+2)/2+j-i+1 & i\le j\\ n(n+1)/2+1 & i>j\end{cases}$$

下三角矩阵

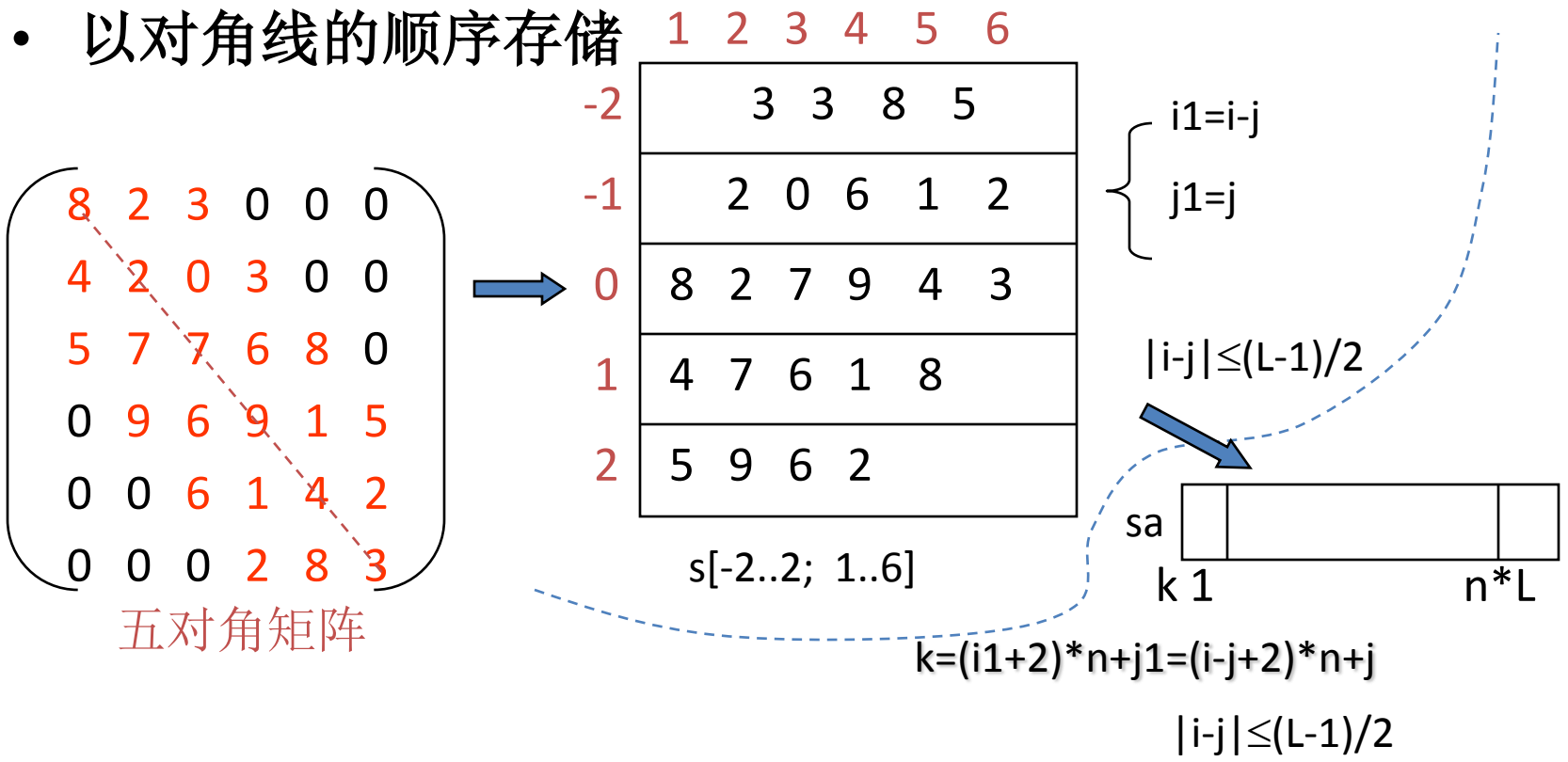
$$k=\begin{cases}i\times(i-1)/2+j & i\ge j\\ n(n+1)/2+1 & i<j\end{cases}$$

3. 对角矩阵（带状矩阵）

[特点] 在n×n的方阵中，非零元素集中在主对角线及其两侧共L(奇数)条对角线的带状区域内 — L对角矩阵。

[存储方法]

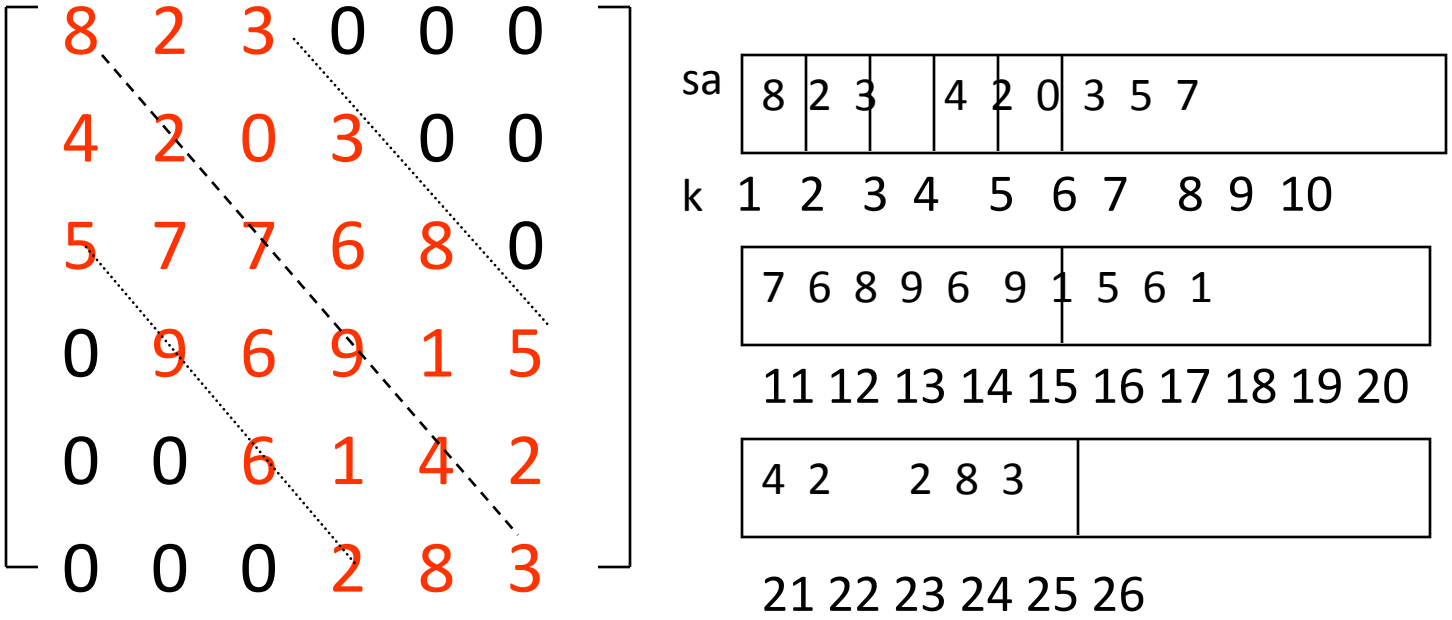
- **以对角线的顺序存储**



- **只存储带状区内的元素**

  除首行和末行，按每行 L个元素，共(n-2)L+(L+1)个元素。sa[1..(n-1)L+1]

  k=(i-1)L+1+(j-i)　　|i-j|≤(L-1)/2



稀疏矩阵

[特点] 大多数元素为零。

[常用存储方法] 只记录每一非零元素(i,j,a_{ij})

节省空间，但丧失随机存取功能

- 顺序存储：三元组表
- 链式存储：十字(正交)链表

$$\begin{bmatrix} 15 & 0 & 0 & 22 & 0 & -15 \\ 0 & 11 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & -6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 91 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 28 & 0 & 0 & 0 \end{bmatrix}_{6\times6}$$

4.5 广义表

- 广义表（列表）： $n\ (\geq 0)$个表元素组成的有限序列，

  记作$LS = (a_0, a_1, a_2, \ldots, a_{n-1})$

  LS是表名，a_i是表元素，它可以是表 (称为子表)，可以是数据元素(称为原子)。

  - n为表的长度。$n = 0$ 的广义表为空表。

广义表与线性表的区别？

- 线性表的成分都是结构上不可分的单元素
- 广义表的成分可以是单元素，也可以是有结构的表
- 线性表是一种特殊的广义表
- 广义表不一定是线性表，也不一定是线性结构

广义表的基本运算

- （1）求表头GetHead(L)：非空广义表的第一个元素，可以是一个单元素，也可以是一个子表
- （2）求表尾GetTail(L)：非空广义表除去表头元素以外其它元素所构成的表。表尾一定是一个表