第5章　数组和广义表

5.1 数组的定义

5.2 数组的顺序与实现

5.3 矩阵的压缩存储

5.4 广义表的定义

5.5 广义表的存储结构

5.1 数组的定义

- 数组
  - 数组是由n(n>1)个具有相同数据类型的数据元素a1，a2，…，an组成的有序序列，且该序列必须存储在一块地址连续的存储单元中。
  - 数组中的数据元素具有相同数据类型。
  - 数组是一种随机存取结构，给定一组下标，就可以直接访问与其对应的数据元素。
  - 数组中的数据元素个数是固定的，是一种定长的线性表。
  - 数组一般不作插入和删除操作，一旦建立了数组，则结构中的数据元素个数和数据元素之间的关系就不再发生变动。

5.1 数组的定义

- 数组是一组偶对(下标值，数据元素值)的集合。
  - 在数组中，对于一组有意义的下标，都存在一个与其对应的值。一维数组对应着一个下标值，二维数组对应着两个下标值，如此类推。
  - 数组中的每个数据元素都对应于一组下标（j_1，j_2，…，j_n ）

    其中：$0 \le j_i \le b_i-1$

    b_i称为第i维的长度(i=1,2,…,n)

5.1 数组的定义

- 数组的抽象数据类型定义

ADT Array{

数据对象：$j_i = 0,1,\ldots,b_i-1$ ， $i=1,2,\ldots,n$;

D = { $a_{j_1 j_2 \cdots j_n}$ | n(>0)称为数组的维数，b_i是数组第i维的长度，j_i是数组元素第i维的下标，$a_{j_1 j_2 \cdots j_n} \in$ ElemSet }

数据关系：R = {R1, R2, …, Rn}

Ri={ $<a_{j_1 j_2 \cdots j_i \cdots j_n}, a_{j_1 j_2 \cdots j_{i+1} \cdots j_n}>$ | $0 \leqq j_k \leqq b_k-1$ ， $1 \leqq k \leqq n$ 且$k \neq i$ ，$0 \leqq j_i \leqq b_i-2$， $a_{j_1 j_2 \cdots j_i \cdots j_n}, a_{j_1 j_2 \cdots j_{i+1} \cdots j_n} \in D$， $i=1,2,\ldots,n$ }

基本操作： 取值、赋值 ……

} ADT Array

5.1 数组的定义

- 直观的n维数组

以二维数组为例讨论。将二维数组看成是一个定长的线性表，其每个元素又是一个定长的线性表。

设二维数组$A=(a_{ij})m\times n$，则

$A=(\alpha_0, \alpha_2, \ldots, \alpha_p)$ (p=m-1或n-1)

其中，每个数据元素α_j是一个列向量(线性表)：

$\alpha_j=(a_{0j}, a_{1j}, \ldots, a_{m-1,j})$ $0\leqq j\leqq n-1$

或是一个行向量：

$\alpha_i=(a_{i0}, a_{i1}, \ldots, a_{i,n-1})$ $0\leqq i\leqq m-1$

5.1 数组的定义

- 直观的n维数组

$$A=\begin{pmatrix} a_{00} & a_{01} & \cdots & a_{0,n-1} \\ a_{10} & a_{11} & \cdots & a_{1,n-1} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m-1,0} & a_{m-1,1} & \cdots & a_{m-1,n-1} \end{pmatrix}$$

(a) 矩阵形式表示

$$A=\begin{pmatrix} [a_{00} & a_{01} & \cdots & a_{0,n-1}] \\ [a_{10} & a_{11} & \cdots & a_{1,n-1}] \\ \cdots & \cdots & \cdots & \cdots \\ [a_{m-1,0} & a_{m-1,1} & \cdots & a_{m-1,n-1}] \end{pmatrix}$$

(b) 行向量的一维数组

$$A=\begin{pmatrix} \begin{pmatrix} a_{00} \\ a_{10} \\ \vdots \\ a_{m-1,0} \end{pmatrix} & \begin{pmatrix} a_{01} \\ a_{11} \\ \vdots \\ a_{m-1,1} \end{pmatrix} & \vdots & \begin{pmatrix} a_{0,n-1} \\ a_{1,n-1} \\ \vdots \\ a_{m-1,n-1} \end{pmatrix} \end{pmatrix}$$

(c) 列向量的一维数组

5.2 数组的顺序表示和实现

一. 数组的顺序存储

- 数组一般不做插入和删除操作，也就是说，数组一旦建立，结构中的元素个数和元素间的关系就不再发生变化
  - 因此一般都是采用顺序存储的方法来表示数组。

- 问题：计算机的内存结构是一维(线性)地址结构，对于多维数组，将其存放(映射)到内存一维结构时，有个次序约定问题。即必须按某种次序将数组元素排成一列序列，然后将这个线性序列存放到内存中。

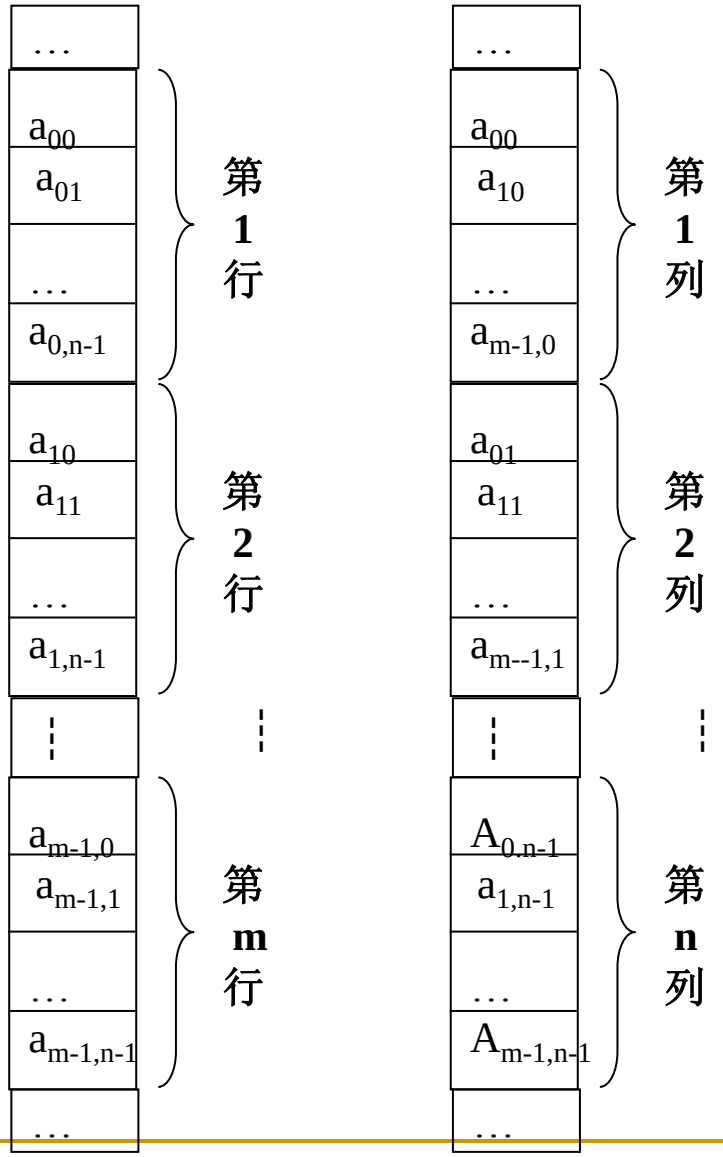
5.2 数组的顺序表示和实现

- 多维数组的两种顺序存储方式
  - 行优先顺序(Row Major Order)：将数组元素按行排列，第i+1个行向量紧接在第i个行向量后面。对二维数组，按行优先顺序存储的线性序列为：
    - $a_{00},a_{01},a_{02},\ldots,a_{0,n-1}$，$a_{10},a_{11},\ldots a_{1,n-1}$ ，……， $a_{m-1,0},a_{m-1,1},\ldots,a_{m-1,n-1}$
    - PASCAL、C是按行优先顺序存储的。
  - 列优先顺序(Column Major Order)：将数组元素按列向量排列，第j+1个列向量紧接在第j个列向量之后，对二维数组，按列优先顺序存储的线性序列为：
    - $a_{00},a_{10},\ldots,a_{m-1,0}$， $a_{01},a_{11},\ldots a_{m-1,1}$， ……，
    - Fortune语言是以列为优先顺序存储

5.2 数组的顺序表示和实现

$$A=\begin{pmatrix} a_{00} & a_{01} & \cdots & a_{0,n-1} \\ a_{10} & a_{11} & \cdots & a_{1,n-1} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m-1,0} & a_{m-1,1} & \cdots & a_{m-1,n-1} \end{pmatrix}$$

(a)　二维数组



(b)　行优先顺序存储　　**(c)　列优先顺序存储**

二维数组及其顺序存储图例形式

5.2 数组的顺序表示和实现

二. 数组的地址计算

- 以二维数组为例

  - 设有二维数组$A=(a_{ij})_{m\times n}$，若每个元素占用的存储单元数为d(个)，$LOC[a_{00}]$表示元素a_{00}的首地址（即数组的首地址（基地址）），以行优先存储，则元素$LOC[a_{ij}]$的存储地址为？

(1) 第0行中的每个元素对应的地址是：

$$LOC[a_{0j}] = LOC[a_{00}] + j\times d \qquad j=0,1,2,\ldots,n-1$$

(2) 第1行中的每个元素对应的地址是：

$$LOC[a_{1j}] = LOC[a_{00}] + (1\times n + j)\times d \qquad j=0,1,2,\ldots,n-1$$

…… ……

5.2 数组的顺序表示和实现

二. 数组的地址计算

- 以行优先存储

(3) 第**m-1**行中的每个元素对应的(首)地址是：

$$LOC[a_{m-1,j}] = LOC[a_{00}] + ((m-1)*n + j) \times d \qquad j=0,1,2,\ldots,n-1$$

由此可知，二维数组中任一元素a_{ij}的地址是：

$$LOC[a_{ij}] = LOC[a_{00}] + (i \times n + j) \times d$$

$$i=0,1,2,\ldots,m-1 \qquad j=0,1,2,\ldots,n-1$$

5.2 数组的顺序表示和实现

二. 数组的地址计算

A_{mxn}以列序为主序存储

$$A_{mxn}=\begin{pmatrix}\begin{pmatrix}a_{00}\\a_{10}\\\vdots\\a_{m-1,0}\end{pmatrix}&\begin{pmatrix}a_{01}\\a_{11}\\\vdots\\a_{m-1,1}\end{pmatrix}&\begin{pmatrix}a_{02}\\a_{12}\\\vdots\\a_{m1,2}\end{pmatrix}&\cdots&\begin{pmatrix}a_{0,n-1}\\a_{1,n-1}\\\vdots\\a_{m-1,n-1}\end{pmatrix}\end{pmatrix}$$

a_{00}	a_{10}	…	$a_{m-1,0}$	a_{01}	a_{11}	…	$a_{m-1,1}$	…	$a_{0,n-1}$	…	$a_{m-1,n-1}$

- LOC(a_{00})是二维数组的起始存储地址，d为每个数据元素占用存储单元的长度(数目)，则

$$LOC(a_{ij}) = LOC(a_{00}) + (i + j \times m) \times d$$

其中，i=0,1,2,…,m-1　j=0,1,2,…,n-1

5.2 数组的顺序表示和实现

二. 数组的地址计算

- 推而广之，对于n维数组中任一元素$a_{j_1 j_2 \ldots j_n}$的地址是

$$LOC(a_{j_1 j_2 \ldots j_n}) = LOC(a_{00\ldots0}) + [(b_2 \times \ldots \times b_n) \times j_1 + (b_3 \times \ldots \times b_n) \times j_2 + \ldots + b_n \times j_{n-1} + j_n] \times d$$

例题

例：一个二维数组A，行下标的范围是1到6，列下标的范围是0到7，每个数组元素用相邻的6个字节存储，存储器按字节编址。那么，这个数组的体积是_____288_____个字节。

例题

例：已知二维数组$A_{m,m}$按行存储的元素地址公式是：

$Loc(a_{ij}) = Loc(a_{11}) + [(i-1) \times m + (j-1)] \times K$ ， 请问按列存储的公式相同吗？

答：尽管是方阵，但公式仍不同。应为：

$Loc(a_{ij}) = Loc(a_{11}) + [(j-1) \times m + (i-1)] \times K$

例题

〖考研题〗 ：设数组a[1…60，1…70]的基地址为2048，每个元素占2个存储单元，若以列序为主序顺序存储，则元素a[32, 58]的存储地址为_____8950_____。

答：请注意审题！

根据列优先公式 $Loc(a_{i,j})=Loc(a_{11})+[(j-1)\times m+(i-1)]\times K$

得：$LOC(a_{32,58})=2048+[(58-1)\times 60+(32-1)]\times 2=8950$

想一想：若数组是a[0…59，0…69]，结果是否仍为8950？

维界虽未变，但此时的a[32, 58]不再是原来的a[32, 58]

练习

一. 已知二维数组$A_{40\times50}$，数组左上角元素下标为[0,0]，每个元素使用4个字节空间，数组A的起始存储地址为10000。

- 若采用行序为主，求数组元素A[20, 31]在内存的地址；
- 若采用列序为主，求数组元素A[20, 31]在内存的地址。

5.3 矩阵的压缩存储

- 在科学与工程计算问题中，矩阵是一种常用的数学对象，在高级语言编程时，通常将一个矩阵描述为一个二维数组。这样，可以对其元素进行随机存取，进行各种矩阵运算。

- 对于高阶矩阵，若其中非零元素呈某种规律分布或者矩阵中有大量的零元素，为了节省存储空间，避免存储重复的非零元素或零元素。对这类矩阵进行压缩存储：
  - 多个相同的非零元素只分配一个存储空间；
  - 零元素不分配空间。

5.3 矩阵的压缩存储

一. 特殊矩阵

- 特殊矩阵是指非零元素或零元素的分布有一定规律的矩阵
  - 对称矩阵：矩阵中，对角线两边对应位置上元素的值相同（$a_{ij}=a_{ji}$）
  - 三角矩阵：矩阵中，对角线上（下）边元素值为常数（或者0），称下（上）三角矩阵
  - 如果只存储对称矩阵对角线上的值和对角线以上部分的值，则与上三角矩阵存储方法相同
  - 如果只存储对称矩阵对角线上的值和对角线以下部分的值，则与下三角矩阵存储方法相同

5.3 矩阵的压缩存储

一. 特殊矩阵

- 特殊矩阵是指非零元素或零元素的分布有一定规律的矩阵
- 对称矩阵，若一个n阶方阵$A=(a_{ij})_{n\times n}$中的元素满足性质：

$$a_{ij}=a_{ji} \quad 1\leqq i,j\leqq n且i\neq j$$

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

$$A=\begin{pmatrix} a_{11} & & & & \\ a_{21} & a_{22} & & & \\ a_{31} & a_{32} & a_{33} & & \\ \cdots & \cdots & \cdots & \cdots & \\ a_{n1} & a_{n2} & \cdots & & a_{nn} \end{pmatrix}$$

对称矩阵示例

5.3 矩阵的压缩存储

一. 特殊矩阵

- 对称矩阵中的元素关于主对角线对称，因此，让每一对对称元素a_{ij}和$a_{ji}(i\neq j)$分配一个存储空间，则n^2个元素压缩存储到$n(n+1)/2$个存储空间，能节约近一半的存储空间。

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

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

5.3 矩阵的压缩存储

一. 特殊矩阵

- 假设按“行优先顺序”存储下三角(包括对角线)中的元素

  - 设用一维数组(向量)M[$\frac{n(n+1)}{2}$]存储n阶对称矩阵, 为了便于访问，必须找出矩阵A中的元素的下标值（i,j）和向量M[k]的下标值k之间的对应关系。

k	0	1	2	3	…	n(n-1)/2	…	$\frac{n\times(n+1)}{2}$ -1
M	a_{11}	a_{21}	a_{22}	a_{31}	a_{32}	a_{33}	…	a_{n1}	a_{n2}	…	a_{nn}

对称矩阵的压缩存储示例

5.3 矩阵的压缩存储

一. 特殊矩阵

- 根据上述的下标对应关系，对于矩阵中的任意元素a_{ij}，均可在一维数组M中唯一确定其位置k；反之，对所有

  k=0,1,2, …,n(n+1)/2-1，都能确定M[k]中的元素在矩阵A中的位置(i,j)。

  称M[n(n+1)/2]为n阶对称矩阵A的压缩存储。

5.3 矩阵的压缩存储

一. 特殊矩阵

- K与下标i、j的关系
  - 若i >= j：a_{ij}在下三角中，直接保存在M中。a_{ij}之前的i-1行共有元素个数：

    1+2+…+(i-1) = i×(i-1)/2

    而在第i行上，a_{ij}之前恰有j-1个元素，因此，元素a_{ij}保存在向量M中时的下标值k之间的对应关系是　　k = i×(i-1)/2 + j-1
  - 若i<j：则a_{ij}是在上三角中。因为$a_{ij}=a_{ji}$，在向量M中保存的是a_{ji}。依上述分析可得：　　k = j×(j-1)/2 + i-1　　i<j
  - 所以，若以行序为主序存储矩阵的下三角（包括对角线）中的元素，则

$$K=\begin{cases} i\times(i-1)/2+(j-1) & i\geqq j \\ j\times(j-1)/2+(i-1) & i<j \end{cases} \qquad 1\leqq i,j\leqq n \qquad k=0,1,\ldots,n\times(n+1)/2-1$$

5.3 矩阵的压缩存储

一. 特殊矩阵

- 三角矩阵： 以主对角线划分，三角矩阵有上三角和下三角两种。
  - 上三角矩阵的下三角（不包括主对角线）中的元素均为常数c(一般为0)。
  - 下三角矩阵正好相反，它的主对角线上方均为常数，

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ c & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \\ c & c & \cdots & a_{nn} \end{pmatrix}$$

(a) 上三角矩阵示例

$$\begin{pmatrix} a_{11} & c & \cdots & c \\ a_{21} & a_{22} & \cdots & c \\ \cdots & \cdots & \cdots & \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}$$

(b) 下三角矩阵示例

5.3 矩阵的压缩存储

一. 特殊矩阵

- 三角矩阵中的重复元素c可共享一个存储空间，其余的元素正好有$\frac{n\times(n+1)}{2}$个，因此，三角矩阵可压缩存储到向量M[0，…，$\frac{n\times(n+1)}{2}$]中，其中c存放在向量的最后1个分量中。

5.3 矩阵的压缩存储

下三角矩阵

$$\begin{bmatrix} a_{11} & c & c & \cdots & c \\ a_{21} & a_{22} & c & \cdots & c \\ a_{31} & a_{32} & a_{33} & \cdots & c \\ \cdots & \cdots & \cdots & \cdots & c \\ a_{n,1} & a_{n,2} & a_{n,3} & \cdots & a_{n,n} \end{bmatrix}$$

	0	1	2	3	4	5	6	7	8		n(n+1)/2-1
M	a_{11}	a_{21}	a_{22}	a_{31}	a_{32}	a_{33}	a_{41}	a_{42}	a_{43}	……	a_{nn}	c

若 $i \geq j$，矩阵元素 a_{ij} 在数组M中的存放位置为

$$\underbrace{1 + 2 + \cdots + (i-1)}_{\text{前}i\text{-}1\text{行元素总数}} + \underbrace{j-1}_{\text{第}i\text{行第}j\text{个元素前的元素个数}} = i * (i - 1) / 2 + j-1$$

5.3 矩阵的压缩存储

一. 特殊矩阵

- 下三角矩阵元素a_{ij}保存在向量M中时的下标值k与（i,j）之间的对应关系是：

$$K=\begin{cases}\dfrac{i\times(i-1)}{2}+(j-1) & \text{当i>=j时}\\ \dfrac{n\times(n+1)}{2} & \text{当i<j时}\end{cases}$$

$1 \leqq i, j \leqq n$

练习

一. 在一个10阶对称矩阵A中，若采用行优先压缩存储矩阵上三角元素，每个元素使用8个字节，数组内存首址为100。

- 若a[1,1]是数组首元素，求矩阵元素A[6, 7]的内存地址；
- 若a[0,0]是数组首元素，求矩阵元素A[2, 3]的内存地址。

元素个数

上三角矩阵

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ c & a_{22} & a_{23} & \cdots & a_{2n} \\ c & c & a_{33} & \cdots & a_{3n} \\ \cdots & \cdots & \cdots & a_{ii} & a_{in} \\ c & c & c & \cdots & a_{n,n} \end{bmatrix}$$

n
n-1
n-2
…
n-i+1
…
1

	0	1	2	。	。	。	(n-1)	n	(n+1)	。	。	。	n(n+1)/2-1
M	a_{11}	a_{12}	a_{13}	…		$a_{1\,n}$	a_{22}	a_{23}	…		…	a_{nn}	c

若 $i \le j$，矩阵元素 a_{ij} 在数组M中的存放位置为

$$n + (n-1) + (n-2) + \cdots + (n-(i-1)+1) \quad + \quad j-i$$

前i-1行元素总数

第i行第j个元素前的元素个数

5.3 矩阵的压缩存储

一. 特殊矩阵

- 矩阵中，除了主对角线和主对角线上或下方若干条对角线上的元素之外，其余元素皆为零。即所有的非零元素集中在以主对角线为了中心的带状区域中。

$$A=\begin{pmatrix} a_{11} & a_{12} & 0 & \cdots & 0 \\ a_{21} & a_{22} & a_{23} & 0 & \cdots & 0 \\ 0 & a_{32} & a_{33} & a_{34} & 0 & \cdots & 0 \\ & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & 0 & a_{n-1\,n-2} & a_{n-1\,n-1} & a_{n-1\,n} \\ 0 & \cdots & 0 & 0 & a_{n\,n-1} & a_{n\,n} \end{pmatrix}$$

三对角矩阵示例

5.3 矩阵的压缩存储

一. 特殊矩阵

- 综上所述，各种特殊矩阵其非零元素的分布都是有规律的，因此总能找到一种方法将它们压缩存储到一个向量中，并且一般都能找到矩阵中的元素与该向量的对应关系，通过这个关系，仍能对矩阵的元素进行随机存取。

练习

一. 已知矩阵A[4][5]，矩阵元素为整型，每个元素使用4个字节空间，现用一维数组B存储该矩阵，数组B的内存首址为10000，求矩阵元素A[2, 3]在内存的地址（矩阵左上角元素下标为[1,1]）

二. 已知对称矩阵M[5][5]，矩阵元素为整型，每个元素使用4个字节空间，现用一维数组B存储该矩阵，数组B的内存首址为10000，求矩阵元素M[5, 4]在内存的地址（矩阵左上角元素下标为[1,1]）

5.3 矩阵的压缩存储

二. 稀疏矩阵

- 对于稀疏矩阵，目前还没有一个确切的定义。
  - 设矩阵A是一个n×m的矩阵中有s个非零元素，
  - 设 δ=s/(n×m)，称δ为稀疏因子，如果某一矩阵的稀疏因子δ满足δ≦0.05时称为稀疏矩阵

$$A=\begin{pmatrix} 0 & 12 & 9 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -3 & 0 & 0 & 0 & 0 & 14 & 0 \\ 0 & 0 & 24 & 0 & 0 & 0 & 0 \\ 0 & 18 & 0 & 0 & 0 & 0 & 0 \\ 15 & 0 & 0 & -7 & 0 & 0 & 0 \end{pmatrix}$$

稀疏矩阵示例

5.3 矩阵的压缩存储

二. 稀疏矩阵

- 对于稀疏矩阵，采用压缩存储方法时，只存储非0元素
  - 存储非0元素的行下标值、列下标值、元素值
  - 因此一个三元组(i, j, a_{ij})唯一确定稀疏矩阵的一个非零元素。

如上图稀疏矩阵A的三元组线性表为：

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

5.3 矩阵的压缩存储

二. 稀疏矩阵

- 稀疏矩阵的类型定义

```
typedef struct {
        int i, j;                   //非零元素的下标
        ElementType  e;             //非零元素的值
}Triple;
typedef struct {
        Triple data[MAXSIZE];       //非零元素的三元组表
        int mu, nu, tu;             //矩阵的行数、列数和非零元素的个数
}TSMatrix;
```

5.3 矩阵的压缩存储

三. 三元组顺序表

- 矩阵的转置

6			mu行数
7			nu列数
8			tu元素个数
1	2	12	
1	3	9	
3	1	-3	
3	6	14	
4	3	24	
5	2	18	
6	1	15	
6	4	-7	

row col value

(a) 原矩阵的三元组表

7			mu行数
6			nu列数
8			tu元素个数
1	3	-3	
1	6	15	
2	1	12	
2	5	18	
3	1	9	
3	4	24	
4	6	-7	
6	3	14	

row col value

(b)转置矩阵的三元组表

稀疏矩阵及其转置矩阵的三元组顺序表

5.3 矩阵的压缩存储

三. 三元组顺序表

- 稀疏矩阵的转置

设矩阵列数为m，对矩阵三元组表扫描m次；第k次扫描，找寻所有列号为k的项，将其行号变列号、列号变行号，顺次存于转置矩阵三元组表中。

$$\begin{pmatrix} 0 & 0 & -3 & 0 & 0 & 15 \\ 12 & 0 & 0 & 0 & 18 & 0 \\ 9 & 0 & 0 & 24 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 14 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

行	列	值
0	2	-3
0	5	15
1	0	12
1	4	18
2	0	9
2	3	24
5	2	14

5.3 矩阵的压缩存储

三. 三元组顺序表

- 稀疏矩阵的转置

设矩阵列数为m，对矩阵三元组表扫描m次；第k次扫描，找寻所有列号为k的项，将其行号变列号、列号变行号，顺次存于转置矩阵三元组表中。

行	列	值
0	1	12
0	2	9
2	0	−3
2	5	14
3	2	24
4	1	18
5	0	15

5.4 广义表的定义

一. 概念

- 广义表是线性表的推广和扩充，在人工智能领域中应用十分广泛。
  - 在第2章中，我们把线性表定义为n(n≧0)个元素a_1，a_2，…，a_n的有穷序列，该序列中的所有元素具有相同的数据类型且只能是原子项(Atom)。
  - 所谓原子项可以是一个数或一个结构，是指结构上不可再分的。若放松对元素的这种限制，容许它们具有其自身结构，就产生了广义表的概念。

5.4 广义表的定义

一. 概念

- 广义表(Lists，又称为列表)：是由n(n ≧0)个元素组成的有穷序列，记作： LS=(a_1，a_2，…，a_n)
  - 其中a_i或者是原子项，或者是一个广义表。LS是广义表的名字，n为它的长度。若a_i是广义表，则称为LS的子表。
  - 习惯上：原子用小写字母，子表用大写字母

5.4 广义表的定义

一. 概念

- 广义表的术语：

  有广义表LS=(a_1，a_2，…，a_n)

  - 若广义表**LS**非空，则**a_1**(表中第一个元素)称为表头；
  - 其余元素组成的子表称为表尾：**(a_2，a_3，…，a_n)**
  - 表的长度：广义表中所包含的元素(包括原子和子表)的个数
  - 表的深度：广义表中括号的重数，是**LS**中各**a_i**（i=1,…,n）的深度的最大值加**1**。*空表的深度为**1**，原子的深度为**0***。

5.4 广义表的定义

一. 概念

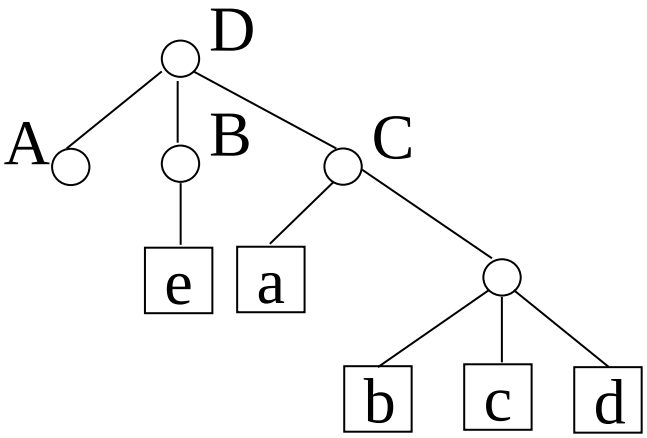
- 广义表的性质
  - 广义表的元素可以是原子，也可以是子表，子表的元素又可以是子表， ….即广义表是一个多层次的结构。
  - 广义表可以被其它广义表所共享，也可以共享其它广义表。广义表共享其它广义表时通过表名引用。
  - 广义表本身可以是一个递归表。
  - 根据对表头、表尾的定义，任何一个非空广义表的表头可以是原子，也可以是子表， 而表尾必定是广义表。

5.4 广义表的定义

一. 概念

- 广义表深度和长度

广 义 表	表长n	表深h
A=()	0	1
B=(e)	1	1
C=(a,(b,c,d))	2	2
D=(A,B,C)	3	3
E=(a,E)	2	∞
F=(())	1	2



广义表的图形表示

5.4 广义表的定义

一. 概念

- 注意

例如：

A = ( )
无任何元素的空表
长度为0
深度为1

B = (( ))=(A)
有一个元素的广义表，此元素为空表
长度为1
深度为2

5.4 广义表的定义

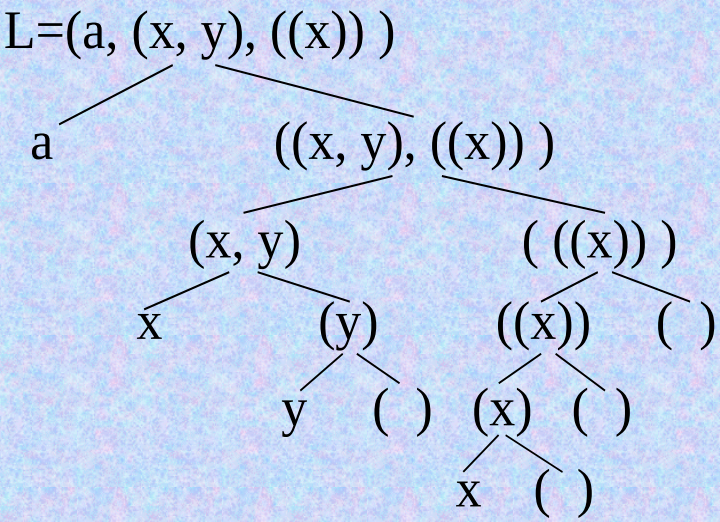
一. 概念

- 任何一个非空广义表
  - **表头**：表的第一个元素
  - **表尾**：表头以外其它表元素组成的**表**

  LS = (α1, α2, …, αn)

  均可分解为表头和表尾两部分
- 基本运算：
  - 取表头 **Head (LS) = α1**
  - 取表尾 **Tail (LS) = (α2, …, αn)**

例：



5.4 广义表的定义

一. 概念

- 例子

D = (E, F)，E = (a, (b, c))，F = (d, (e))

Head(D) = E=(a, (b, c))　　**Tail(D) = (F)= ((d, (e)))**

Head(E) = a　　**Tail(E) = ((b, c))**

Head((b, c)) =（b,c）　　**Tail((b, c)) = （）**

Head(b, c) = （b）　　**Tail(b, c) = （c）**

Head((c)) = （c）　　**Tail((c)) = （）**

5.4 广义表的定义

二. 存储结构

- 由于广义表中的数据元素具有不同的结构，通常用链式存储结构表示，每个数据元素用一个结点表示。因此，广义表中就有两类结点：
  - 一类是表结点，用来表示广义表项，由标志域，表头指针域，表尾指针域组成;
  - 另一类是原子结点，用来表示原子项，由标志域，原子的值域组成
  - 只要广义表非空，都是由表头和表尾组成。即一个确定的表头和表尾就唯一确定一个广义表。

标志tag=0	原子的值

(a) 原子结点

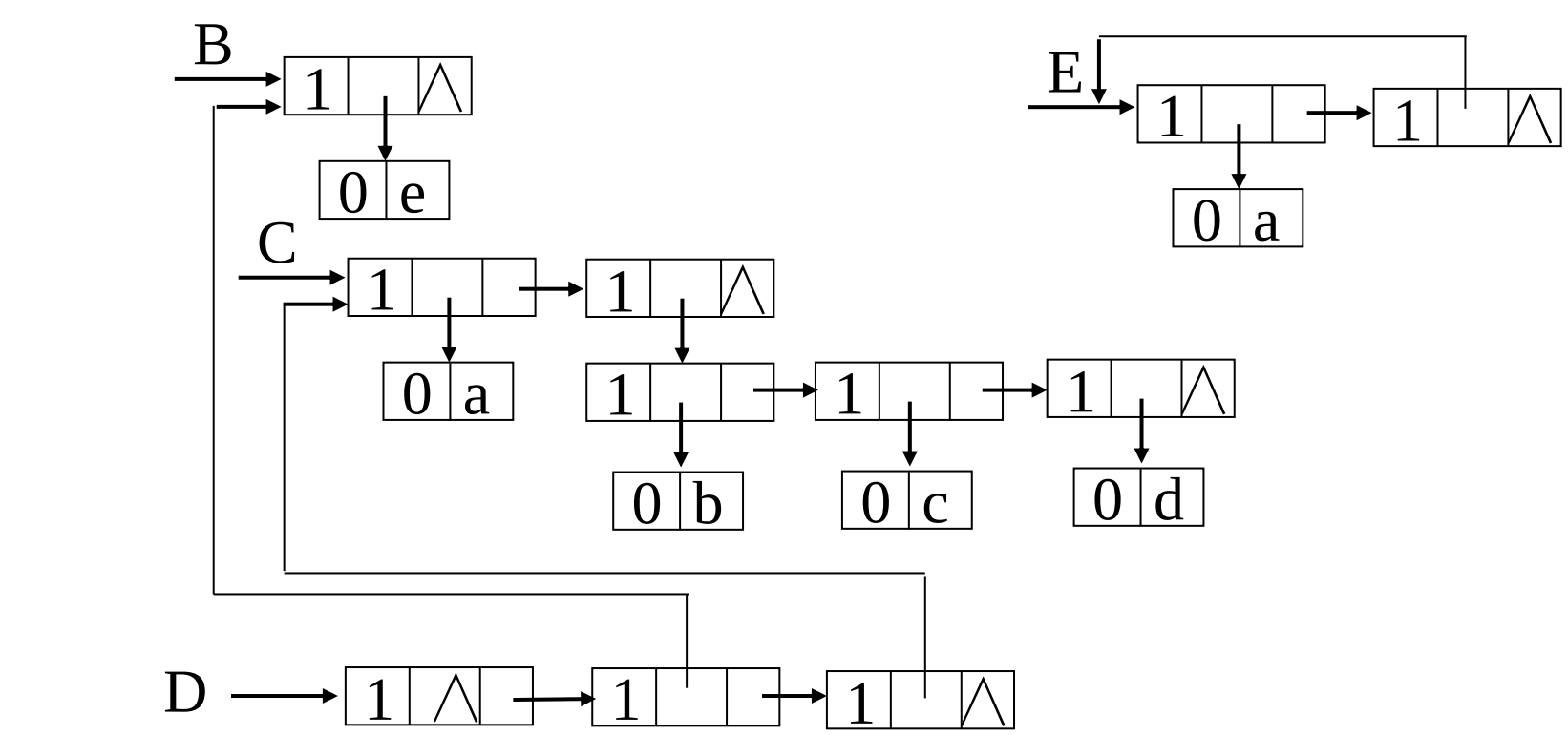
标志tag=1	表头指针hp	表尾指针tp

(b) 表结点

5.4 广义表的定义

二. 存储结构

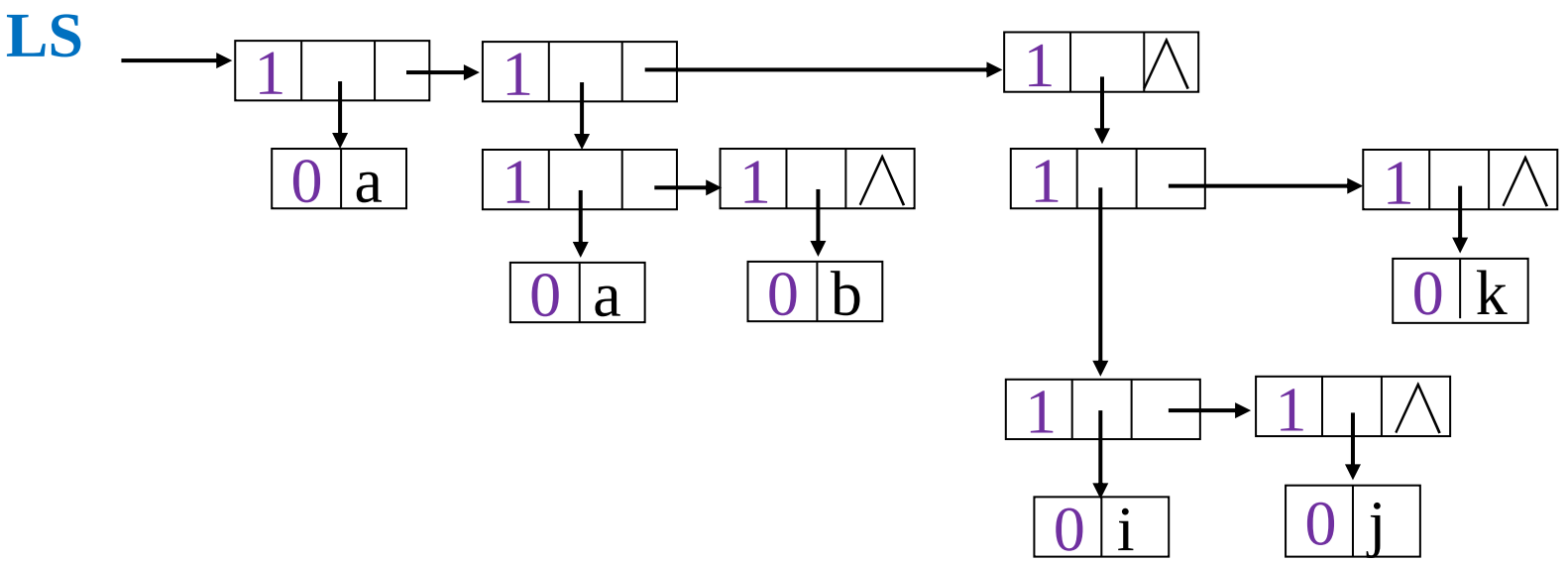
- 广义表的存储结构示例
  - 例子**A=()，B=(e)，C=(a, (b, c, d))，D=(A, B, C)，E=(a, E)**



广义表的存储结构示意图

练习

一. 一个广义表是(a, (a, b), ((i, j), k))，请画出该广义表的链式存储结构。



第5章总结

- 数组是一组偶对的集合，必然带： 下标值，数据值
- 数组的顺序存储
  - 行优先顺序、列优先顺序
  - 二维数组位置ij 和一维数组位置k之间的地址转换
- 特殊矩阵：非零元素或零元素的分布有一定规律的矩阵
  - 对称矩阵、三角矩阵、K对角矩阵
  - 掌握对称矩阵用一维数组存储时，矩阵下标ij和数组位置k的转换
- 稀疏矩阵：非0元素占矩阵元素总数比例较少的矩阵
  - 三元组顺序表的表示方法
- 广义表是线性表的推广和扩充，是由n个元素组成的有穷序列
  - 基本概念：表头、表尾、长度、深度
  - 链式存储结构表示