第 6 章 数组和广义表

教材中练习题及参考答案

1. 如何理解数组是线性表的推广。

答：数组可以看成是线性表在下述含义上的扩展：线性表中的数据元素本身也是一个线性表。在d（$d \geqslant 1$）维数组中的每个数据元素都受着d个关系的约束，在每个关系中，数据元素都有一个后继元素（除最后一个元素外）和一个前驱元素（除第一个元素外）。

因此，这 d 个关系中的任一关系，就其单个关系而言，仍是线性关系。例如，$m \times n$ 的二维数组的形式化定义如下：

$A=(D, R)$

其中：

$D = \{ a_{ij} \mid 0 \leqslant i \leqslant m-1, 0 \leqslant j \leqslant n-1 \}$ //数据元素的集合

$R = \{ \text{ROW}, \text{COL} \}$

$\text{ROW} = \{ <a_{i,j}, a_{i+1,j}> \mid 0 \leqslant i \leqslant m-2, 0 \leqslant j \leqslant n-1 \}$ //行关系

$\text{COL} = \{ <a_{i,j}, a_{i,j+1}> \mid 0 \leqslant i \leqslant m-1, 0 \leqslant j \leqslant n-2 \}$ //列关系

2. 有三维数组 a[0..7，0..8，0..9]采用按行序优先存储，数组的起始地址是 1000，每个元素占用 2 个字节，试给出下面结果：

（1）元素 $a_{1,6,8}$ 的起始地址。

（2）数组 a 所占用的存储空间。

答：（1）$\text{LOC}(a_{1,6,8})=\text{LOC}(a_{0,0,0})+[1 \times 9 \times 10+6 \times 10+8] \times 2=1000+316=1316$。

（2）数组 a 所占用存储空间$=8 \times 9 \times 10 \times 2=1440$ 字节。

3. 如果某个一维数组 A 的元素个数 n 很大，存在大量重复的元素，且所有元素值相同的元素紧挨在一起，请设计一种压缩存储方式使得存储空间更节省。

答：设数组的元素类型为 ElemType，采用一种结构体数组 B 来实现压缩存储，该结构体数组的元素类型如下：

```
struct
{    ElemType data;     //元素值
     int length;        //重复元素的个数
}
```

如数组 A[]={1，1，1，5，5，5，5，3，3，3，3，4，4，4，4，4，4}，共有 17 个元素，对应的压缩存储 B 为：{1，3}，{5，4}，{3，4}，{4，6}}。从中看出，如果重复元素越多，采用这种压缩存储方式越节省存储空间。

4. 一个 n 阶对称矩阵 A 采用压缩存储在一维数组 B 中，则 B 包含多少个元素？

答：通常 B 中包含 n 阶对称矩阵 A 的下三角和主对角线上的元素，其元素个数为 $1+2+\cdots+n=\frac{n(n+1)}{2}$。所以 B 包含 $\frac{n(n+1)}{2}$ 个元素。

5. 设 $n\times n$ 的上三角矩阵 $A[0..n-1, 0..n-1]$已压缩到一维数组 $B[0..m]$中，若按列为主序存储，则 $A[i][j]$对应的 B 中存储位置 k 为多少，给出推导过程。

答：对于上三角部分或者主对角中的元素 $A[i][j]$（$i\leqslant j$），按列为主序存储时，前面有 0～$j-1$ 共 j 列，第 0 列有 1 个元素，第 1 列有 2 个元素，…，第 $j-1$ 列有 j 个元素，所以这 j 列的元素个数=$1+2+\cdots+j=j(j+1)/2$；在第 j 列中，$A[i][j]$元素前有 $A[0..i-1, j]$共 i 个元素。所以 $A[i][j]$元素前有 $j(j+1)/2+i$ 个元素，而 B 的下标从 0 开始，所以 $A[i][j]$在 B 中的位置 $k=j(j+1)/2+i$。

6. 利用三元组存储任意稀疏数组 A 时，假设其中一个元素和一个整数占用的存储空间相同，问在什么条件下才能节省存储空间。

答：设稀疏矩阵 A 有 t 个非零元素，加上行数 rows、列数 cols 和非零元素个数 nums（也算一个三元组），那么三元组顺序表的存储空间总数为 $3(t+1)$，若用二维数组存储时占用存储空间总数为 $m\times n$，只有当 $3(t+1)<m\times n$ 即 $t<m\times n/3-1$ 时，采用三元组存储才能节省存储空间。

7. 用十字链表存储一个有 k 个非 0 元素的 $m\times n$ 的稀疏矩阵，则其总的节点数为多少？

答：该十字链表有一个十字链表表头节点，MAX(m，n)个行、列表头节点。另外，每个非0元素对应一个节点，即k个元素节点。所以共有MAX(m，n)+k+1个节点。

8. 求下列广义表运算的结果

（1）*head*[(x，y，z)]

（2）*tail*[((a，b)，(x，y))]

> **注意：**为了清楚起见，在括号层次较多时，将 *head* 和 *tail* 的参数用中括号表示。例如 *head*[G]、*tail*[G]分别表示求广义表 G 的表头和表尾。

答：（1）*head*[(x，y，z)]=x。

（2）*tail*[((a，b)，(x，y))]=((x，y))。

9. 设定二维整数数组 $B[0..m-1, 0..n-1]$的数据在行、列方向上都按从小到大的顺序排序，且整型变量 x 中的数据在 B 中存在。设计一个算法，找出一对满足 $B[i][j]=x$ 的 i、j 值。要求比较次数不超过 $m+n$。

解：从二维数组 B 的右上角的元素开始比较。每次比较有三种可能的结果：若相等，则比较结束；若 x 大于右上角元素，则可断定二维数组的最上面一行肯定没有与 x 相等的数据，下次比较时搜索范围可减少一行；若 x 小于右上角元素，则可断定二维数组的最右面一列肯定不包含与 x 相等的数据，下次比较时可把最右一列剔除出搜索范围。这样，每次比较可使搜索范围减少一行或一列，最多经过 $m+n$ 次比较就可找到要求的与 x 相等的元素。对应的程序如下：

```
#include <stdio.h>
#define M 3             //行数常量
#define N 4             //列数常量
void Find(int B[M][N],int x,int &i,int &j)
{   i=0;j=N-1;
    while (B[i][j]!=x)
        if(B[i][j]<x) i++;
        else j--;
}
int main()
{   int i,j,x=11;
    int B[M][N]={{1,2,3,4},{5,6,7,8},{9,10,11,12}};
    Find(B,x,i,j);
    printf("B[%d][%d]=%d\n",i,j,x);
    return 1;
}
```

10. 设计一个算法，计算一个三元组表表示的稀疏矩阵的对角线元素之和。

解：对于稀疏矩阵三元组表 *a*，从 *a.data*[0]开始查看，若其行号等于列号，表示是一个对角线上的元素，则进行累加，最后返回累加值。算法如下：

```
bool diagonal(TSMatrix a,ElemType &sum)
{   sum=0;
    if (a.rows!=a.cols)                 //行号不等于列号，返回 false
    {   printf("不是对角矩阵\n");
        return false;
    }
    for (int i=0;i<a.nums;i++)
        if (a.data[i].r==a.data[i].c)   //行号等于列号
            sum+=a.data[i].d;
    return true;
}
```

11. 设计一个算法 Same(*g*1，*g*2)，判断两个广义表 *g*1 和 *g*2 是否相同。

解：判断广义表是否相同过程是，若 *g*1 和 *g*2 均为 NULL，则返回 true；若 *g*1 和 *g*2 中一个为 NULL，另一不为 NULL，则返回 false；若 *g*1 和 *g*2 均不为 NULL，若同为原子且原子值不相等，则返回 false，若同为原子且原子值相等，则返回 Same(*g*1->*link*，*g*2->*link*)，若同为子表，则返回 Same(*g*1->*val.sublist*，*g*2->*val.sublist*) & Same(*g*1->*link*，*g*2->*link*)的结果，若一个为原子另一个为子表，则返回 false。对应的算法如下：

```
bool Same(GLNode *g1,GLNode *g2)
{   if (g1==NULL && g2==NULL)               //均为 NULL 的情况
        return true;                        //返回真
    else if (g1==NULL || g2==NULL)          //一个为 NULL,另一不为 NULL 的情况
        return false;                       //返回假
    else                                    //均不空的情况
    {   if (g1->tag==0 && g2->tag==0)       //均为原子的情况
        {   if (g1->val.data!=g2->val.data) //原子不相等
                return false;               //返回假
            return(Same(g1->link,g2->link)); //返回兄弟比较的结果
        }
```

```
            else if (g1->tag==1 && g2->tag==1)    //均为子表的情况
                return(Same(g1->val.sublist,g2->val.sublist)
                     & Same(g1->link,g2->link));
            else                                  //一个为原子,另一为子表的情况
                return false;                     //返回假
        }
    }
```