定义

数组是一种特殊的线性表，即线性表中数据元素本身也是一个线性表
特点：

- 数组结构固定
- 数组元素同构
- 数组运算——给定一组下标，对数据元素存取

以行序为主序存储：$Loc(a_{ij}) = Loc(a_{11}) + [(i-1)n + (j-1)] \times l$ （一行n个元素，l是类型的大小）
以列序为主序存储：$Loc(a_{ij}) = Loc(a_{11}) + [(j-1)m + (i-1)] \times l$（一列m个元素，l是类型的大小）
i, j从1开始

矩阵的压缩存储

对称矩阵

$a_{ij} = a_{ji}$
假设只存下半部分，分别为$a_{11}, a_{21}, a_{22}, a_{31}, \cdots$
一共有

$$1 + 2 + \cdots + n = \frac{n(n+1)}{2}$$

那么索引为0到$\frac{n(n+1)}{2} - 1$

$$Loc(a_{ij}) = Loc(a_{11}) + [\begin{cases} \frac{i(i-1)}{2} + j - 1 & i \geq j \\ \frac{j(j-1)}{2} + i - 1 & i < j \end{cases}] \times l$$

- $\frac{i(i-1)}{2}$：在i行之前的元素，$0 + 1 + \cdots + i - 1$
- $j - 1$：在i行，a_{ij}之前的元素个数
- $j > i$时：交换i, j
- i, j从1开始
  三角矩阵与之类似

对角矩阵

对角线两侧一排有元素

$$Loc(a_{ij}) = Loc(a_{11}) + [2(i-1) + (j-1)] \times l$$

FxMarkBrown's Blog

- 前$i-1$行一共有2（第1行）$+3(i-2)$（2到$i-1$行）$=3i-4$
- 第i行，非零元素从$i-1$列开始，到$i+1$列结束，所以在a_{ij}前还有$j-(i-1)$个元素
- 整理得到结果

稀疏矩阵

一个稀疏矩阵可以由$\{(i,j,e),\cdots\}$和两个维数唯一确定
顺序存储结构：三元组顺序表

```C++
#define M 20
typedef struct {
    int i,j;
    ElemType e;
} Triple;

typedef struct {
    Triple data[M+1]; // 索引0不用，从1开始
    int mu, nu, tu; // 两个维度和三元组顺序表内的元素（也就是非零元素）的个数
} TSMatrix;
```

对于三元组表

索引	i	j	e
0			
1	$\cdots$	$\cdots$	$\cdots$

转置

先扫一遍待转置矩阵的j，将j为1的放入转置矩阵的前几行，之后扫j为2的,.....

FxMarkBrown's Blog

```C++
Status TransposeSMatirx(TSMatrix M, TSMatrix T) {
    T.mu = M.nu; T.nu = M.mu; T.tu = M.tu;
    if(T.tu) {
        q = 1;
        for(col = 1; col <= M.nu; col++)
            for(p = 1; p<= M.tu; p++)
                if(M.data[p].j == col) {
                    T.data[q].i = M.data[p].j;
                    T.data[q].j = M.data[p].i;
                    T.data[q].e = M.data[p].e;
                    ++q;
                }
    }

    return OK;
}
```

复杂度$O(nu \times tu)$
在矩阵内元素稠密时退化为普通转置

快速转置

考虑直接将的三元表内列号为j的元素放在T的三元表内行号为j的地方，依次排列
所以需要获取T中不同行中非零元素的个数和它们开始的索引，也就是中不同列中非零元素的个数和它们开始的索引

FxMarkBrown's Blog

```C++
Status FastTransposeSMatrix(TSMatrix M, TSMatrix &T) {
    // 采用三元组顺序表存储表示，求稀疏矩阵M的转置矩阵T
    T.mu = M.nu; T.nu = M.mu; T.tu = M.tu;
    if(T.tu){
        for(col = 1; col <= M.nu; ++col) num[col] = 0;
        for(t = 1; t <= M.tu; ++t) ++num[M.data[t].j]; // 求M中该列非零元的个数，也就是T中该行非零元的个数
        cpot[1] = 1;
        for(col = 2;col <= M.nu;++col) cpot[col] = cpot[col - 1] + num[col - 1]; // 求M中非零列第一个元素在三元表中的位置，也就是T中该行第一个元素在三元表中的位置
        for(p = 1; p< M.tu; ++p) {
            col = M.data[p].j; q = cpot[col]; // 获得行
            T.data[q].i = M.data[p].j; T.data[q].j = M.data[p].i;
            T.data[q].e = M.data[p].e;
            ++cpot[col]; // 调整到下一个位置
        }
    }
```

FxMarkBrown's Blog