本节主题：

稀疏矩阵的三元组表示

稀疏矩阵

- 定义
  - 一个阶数较大的矩阵中的非零元素个数s相对于矩阵元素的总个数 t 很小时，即s<<t时，称该矩阵为**稀疏矩阵**。
- 例
  - 一个100×100的矩阵，其中只有100个非零元素

稀疏矩阵的压缩存储方法

- 策略
  - 只存储非零元素
- 约束
  - 稀疏矩阵中非零元素的分布没有任何规律
- 方案
  - 存储非零元素
  - 同时存储该非零元素所对应的行下标和列下标
  - 稀疏矩阵中的每一个非零元素需由一个三元组(i，j，a_{ij})唯一确定，稀疏矩阵中的所有非零元素构成**三元组线性表**。

$$A_{6\times7}=\begin{bmatrix}0&0&1&0&0&0&0\\0&2&0&0&0&0&0\\3&0&0&0&0&0&0\\0&0&0&5&0&0&0\\0&0&0&0&6&0&0\\0&0&0&0&0&7&4\end{bmatrix}$$

i	j	a_{ij}
0	2	1
1	1	2
2	0	3
3	3	5
4	4	6
5	5	7
5	6	4

定义存储结构

```
#define MaxSize  100
typedef struct
{
    int r;        //行号
    int c;        //列号
    ElemType d;   //元素值
} TupNode;        //三元组定义
typedef struct
{
    int rows;     //行数值
    int cols;     //列数值
    int nums;     //非零元素个数
    TupNode data[MaxSize];
} TSMatrix;       //三元组顺序表定义
```

6
7
7

$$A_{6\times7}=\begin{bmatrix}0&0&1&0&0&0&0\\0&2&0&0&0&0&0\\3&0&0&0&0&0&0\\0&0&0&5&0&0&0\\0&0&0&0&6&0&0\\0&0&0&0&0&7&4\end{bmatrix}$$

0	2	1	[0]
1	1	2	[1]
2	0	3	[2]
3	3	5	[3]
4	4	6	[4]
5	5	7	[5]
5	6	4	[6]
			[...]

- 约定：data域中表示的非零元素通常以行序为主序顺序排列——下标按行有序的存储结构。
- 目标：简化大多数矩阵运算算法。

从二维矩阵创建其三元组表示

约定：data域以行序为主序顺序排列

- 算法：以行序方式扫描二维矩阵A，将其非零的元素加入到三元组t

```
void CreatMat(TSMatrix &t, ElemType A[M][N])
{
    int i,j;
    t.rows=M;
    t.cols=N;
    t.nums=0;
    for (i=0; i<M; i++)
    {
        for (j=0; j<N; j++)
            if (A[i][j]!=0) //只存非零值
            {
                t.data[t.nums].r=i;
                t.data[t.nums].c=j;
                t.data[t.nums].d=A[i][j];
                t.nums++;
            }
    }
}
```

t

6
7
7

$A_{6\times7} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 7 & 4 \end{bmatrix}$

0	2	1	[0]
1	1	2	[1]
2	0	3	[2]
3	3	5	[3]
4	4	6	[4]
5	5	7	[5]
5	6	4	[6]
			[...]

将指定位置的元素值赋给变量：执行x=A[i][j]

t

6
7
7

$$A_{6\times7}=\begin{bmatrix}0&0&1&0&0&0&0\\0&2&0&0&0&0&0\\3&0&0&0&0&0&0\\0&0&0&5&0&0&0\\0&0&0&0&6&0&0\\0&0&0&0&0&7&4\end{bmatrix}$$

0	2	1	[0]
1	1	2	[1]
2	0	3	[2]
3	3	5	[3]
4	4	6	[4]
5	5	7	[5]
5	6	4	[6]
			[...]

$a_{ij}\rightarrow x$

```
bool Assign(TSMatrix t,ElemType &x,int i,int j)
{
   int k=0;
   if (i>=t.rows || j>=t.cols)
      return false;        //失败时返回false
   while (k<t.nums && i>t.data[k].r)
      k++;                 //查找行
   while (k<t.nums && i==t.data[k].r && j>t.data[k].c)
      k++;                 //查找列
   if (t.data[k].r==i && t.data[k].c==j)
      x=t.data[k].d;
   else
      x=0;                 //没有找到，是零元素
   return true;            //成功时返回true
}
```

三元组元素赋值：执行A[i][j]=x 运算

①将一个非0元素修改为非0值，如A[5][6]=8

$$A_{6\times7}=\begin{bmatrix}0&0&1&0&0&0&0\\0&2&0&0&0&0&0\\3&0&0&0&0&0&0\\0&0&0&5&0&0&0\\0&0&0&0&6&0&0\\0&0&0&0&0&7&\mathbf{8}\end{bmatrix}$$

0	2	1	[0]
1	1	2	[1]
2	0	3	[2]
3	3	5	[3]
4	4	6	[4]
5	5	7	[5]
5	6	**8**	[6]
			[...]

②将一个0元素修改为非0值，如A[3][5]=8

$$A_{6\times7}=\begin{bmatrix}0&0&1&0&0&0&0\\0&2&0&0&0&0&0\\3&0&0&0&0&0&0\\0&0&0&5&0&\mathbf{8}&0\\0&0&0&0&6&0&0\\0&0&0&0&0&7&4\end{bmatrix}$$

0	2	1	[0]
1	1	2	[1]
2	0	3	[2]
3	3	5	[3]
3	5	**8**	[4]
4	4	6	[5]
5	5	7	[6]
5	6	4	[7]
			[...]

三元组元素赋值算法

```
bool Value(TSMatrix &t,ElemType x,int i,int j)
{
   int k=0,k1;
   if (i>=t.rows || j>=t.cols)
       return false;   //失败时返回false
   while (k<t.nums && i>t.data[k].r)
        k++;              //查找行
   while (k<t.nums && i==t.data[k].r && j>t.data[k].c)
       k++;  //查找列
   if (t.data[k].r==i && t.data[k].c==j)
      t.data[k].d=x;  //存在时直接改
   else
   {
       //不存在时要插入
   }
   return true;  //成功时返回true
}
```

即完成顺序表的插入操作

```
for (k1=t.nums-1; k1>=k; k1--)
{
   t.data[k1+1].r=t.data[k1].r;
   t.data[k1+1].c=t.data[k1].c;
   t.data[k1+1].d=t.data[k1].d;
}
t.data[k].r=i;
t.data[k].c=j;
t.data[k].d=x;
t.nums++;
```

输出三元组

- 从头到尾扫描三元组t，依次输出元素值。

```
void DispMat(TSMatrix t)
{
    int i;
    if (t.nums<=0) return;
    printf("\t%d\t%d\t%d\n",t.rows,t.cols,t.nums);
    printf("  -----------------\n");
    for (i=0; i<t.nums; i++)
        printf("\t%d\t%d\t%d\n",t.data[i].r,t.data[i].c, t.data[i].d);
}
```

矩阵转置

- 对于一个m×n的矩阵$A_{m\times n}$，其转置矩阵是一个n×m的矩阵$B_{n\times m}$，满足$b_{ij}=a_{ji}$，其中$0\le i\le m-1, 0\le j\le n-1$。

$$A_{6\times 7}=\begin{bmatrix}0&0&1&0&0&0&0\\0&2&0&0&0&0&0\\3&0&0&0&0&0&0\\0&0&0&5&0&0&0\\0&0&0&0&6&0&0\\0&0&0&0&0&7&4\end{bmatrix}$$

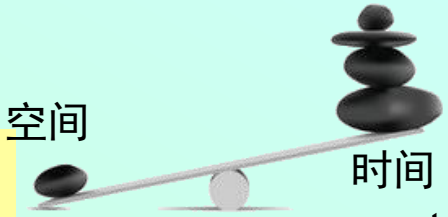
$$B_{7\times 6}=\begin{bmatrix}0&0&3&0&0&0\\0&2&0&0&0&0\\1&0&0&0&0&0\\0&0&0&5&0&0\\0&0&0&0&6&0\\0&0&0&0&0&7\\0&0&0&0&0&4\end{bmatrix}$$

r	c	d
0	2	1
1	1	2
2	0	3
3	3	5
4	4	6
5	5	7
5	6	4

r	c	d
0	2	3
1	1	2
2	0	1
3	3	5
4	4	6
5	5	7
6	5	4

矩阵转置算法

```
void TranTat(TSMatrix t,TSMatrix &tb)
{
   int p,q=0,v;
   tb.rows=t.cols;
   tb.cols=t.rows;
   tb.nums=t.nums;
   if (t.nums!=0)  //当存在非零元素时...
   {
      for (v=0; v<t.cols; v++)
         for (p=0; p<t.nums; p++)
            if (t.data[p].c==v)
            {
               tb.data[q].r=t.data[p].c;
               tb.data[q].c=t.data[p].r;
               tb.data[q].d=t.data[p].d;
               q++;
            }
   }
}
```



t

6
7
7

$$A_{6\times7}=\begin{bmatrix}0&0&1&0&0&0&0\\0&2&0&0&0&0&0\\3&0&0&0&0&0&0\\0&0&0&5&0&0&0\\0&0&0&0&6&0&0\\0&0&0&0&0&7&4\end{bmatrix}$$

- 时间复杂度：O(t.cols*t.nums)，
- 最坏情况：当稀疏矩阵中的非零元素个数t.nums和m*n同数量级时，$O(m*n^2)$。
- 对比：将二维数组存储在一个m行n列矩阵中时，转置算法的时间复杂度为O(m*n)。

0	2	1	[0]
1	1	2	[1]
2	0	3	[2]
3	3	5	[3]
4	4	6	[4]
5	5	7	[5]
5	6	4	[6]
			[...]

思考：矩阵加法？矩阵乘法？

6
7
7

$$A_{6\times7}=\begin{bmatrix}0&0&1&0&0&0&0\\0&2&0&0&0&0&0\\3&0&0&0&0&0&0\\0&0&0&5&0&0&0\\0&0&0&0&6&0&0\\0&0&0&0&0&7&4\end{bmatrix}$$

0	2	1	[0]
1	1	2	[1]
2	0	3	[2]
3	3	5	[3]
4	4	6	[4]
5	5	7	[5]
5	6	4	[6]
			[...]