

数据结构

Data Structure

2017年秋季学期
刘鹏远

数组与广义表

(2)

三元组顺序表

- 1、无法知道矩阵究竟多大
- 2、不知道有几个非零元素

因此：为可靠描述，再加一行“**总体**”信息：即**总行数**、**总列数**，再加上**非零元素总个数**

建立了矩阵与三元组顺序表间的一一对应关系

	i	j	value
0	6	6	8
1	1	2	12
2	1	3	9
3	3	1	-3
4	3	5	14
5	4	3	24
6	5	2	18
7	6	1	15
8	6	4	-7

三元组顺序表

在前面定义的基础上，

```
typedef struct
{
    int rn;      /* 行数 */
    int cn;      /* 列数 */
    int tn;      /* 非0元素个数 */
    Triple data[MAX_SIZE];
}
```

```
typedef struct
{
    int row;
    int col;
    elem_type value;
}Triple;
```

}triple_matrix; //前页是直接Triple triple[size+1]，其第0元素为总体信息；教材中将data[0]空置；PPT中未空置。
教材/PPT中，稀疏矩阵行列均定义为从1开始

rn行数 cn列数 tn元素个数		
7		
8		
9		
1	2	12
1	3	9
3	1	-3
3	8	4
4	3	24
4	6	2
5	2	18
6	7	-7
7	4	-6
↑ row	↑ col	↑ value

操作与应用：

```
int init(triple_matrix *);
```

```
int add(triple_matrix *, triple_matrix,
triple_matrix);
```

```
int sub(triple_matrix *, triple_matrix,
triple_matrix);
```

```
int mul(triple_matrix *, triple_matrix,
triple_matrix);
```

```
int transpose(triple_matrix *, triple_matrix);
```

```
int traverse(triple_matrix);
```

```
int init(triple_matrix *M){  
    printf("please input m,n,t\n");  
    scanf("%d,%d,%d", &M->cn, &M->rn, &M->tn); //边界略  
    for(i=0;i<M->tn;i++){  
        printf("please input i,j,e\n"); //边界处理略  
        //为减小相加等操作时间复杂度，一般按（行）主序输入  
        scanf("%d,%d,%d", &M->data[i].row, &M->data[i].col,  
            &M->data[i].value);  
    }  
    return 1;}  
}
```

```
int add(triple_matrix *Q, triple_matrix M, triple_matrix N){  
    int m=0, n=0, k=0;  
    if(M.cn!=N.cn || M.rn!=N.rn) return 0;//大小不同，不能加  
    Q->cn = M.cn;    Q->rn = M.rn;  
    while(m<M.tn && n<N.tn){  
        //扫描M,N中的元素  
        if(M.data[m].row<N.data[n].row) //行小的直接放入Q  
            Q->data[k++] = M.data[m++];  
        else if(M.data[m].row>N.data[n].row) //行小的直接放入Q  
            Q->data[k++] = N.data[n++];  
    }
```

else{//两者在同一行上

if(M.data[m].col<N.data[n].col)

Q->data[k++] = M.data[m++];

else if(M.data[m].col>N.data[n].col)

Q->data[k++] = N.data[n++];

else{

Q->data[k] = M.data[m];

Q->data[k].value = M.data[m++].value + N.data[n++].value;

if(Q->data[k].value) k++;//非零则k不动 }

}}


```
while(m<M.tn){  
    Q->data[k++]=M.data[m++];  
}
```

```
while(n<N.tn){  
    Q->data[k++]=N.data[m++];  
}
```

```
Q->tn=k; //总元素个数
```

```
if(k>MAX_SIZE) return 0;
```

```
return 1;
```

```
}
```

稀疏矩阵矩阵转置

要保证转置后的三元组表仍然保持行主序
需要什么方法？

方法一：直接交换`row,col`，然后排序，
快速排序时间复杂度为 $O(tn * \lg(tn))$

还有其他方法吗？

```
transpose_naive(triple_matrix *, triple_matrix)
```

The diagram illustrates the data structure for the first iteration. It consists of two 8x3 grids, **M.data** and **T.data**, and three variables: **M.mu**, **M.nu**, and **M.tu**.

M.data (purple border):

0	1	2	12
1	1	3	9
2	3	1	-3
3	3	6	14
4	4	3	24
5	5	2	18
6	6	1	15
7	6	4	-7

T.data (black border):

0			
1			
2			
3			
4			
5			
6			
7			

Variables:

- M.mu**: 6
- M.nu**: 7
- M.tu**: 8

	M. data
0	1 2 12
1	1 3 9
2	3 1 -3
3	3 6 14
4	4 3 24
5	5 2 18
6	6 1 15
7	6 4 -7

M. mu

6

M. nu

7

M. tu

8

	T. data
0	
1	
2	
3	
4	
5	
6	
7	

T. mu

7

T. nu

6

T. tu

8

【例】

	M. data		
0	1	2	12
1	1	3	9
2	3	1	-3
3	3	6	14
4	4	3	24
5	5	2	18
6	6	1	15
7	6	4	-7

M. mu
6

M. nu
7

M. tu
8

	T. data		
0			
1			
2			
3			
4			
5			
6			
7			

T. mu
7

T. nu
6

T. tu
8

【例】

	M. data		
0	1	2	12
1	1	3	9
2	3	1	-3
3	3	6	14
4	4	3	24
5	5	2	18
6	6	1	15
7	6	4	-7

M. mu

6

M. nu

7

M. tu

8

	T. data		
0			
1			
2			
3			
4			
5			
6			
7			

T. mu

7

T. nu

6

T. tu

8

【例】

	M. mu		M. data		T. data		T. mu
	6		0 1 2 12		0 1 3 -3		7
			1 1 3 9		1		
	M. nu		2 3 1 -3		2		T. nu
	7		3 3 6 14		3		6
			4 4 3 24		4		
	M. tu		5 5 2 18		5		T. tu
	8		6 6 1 15		6		8
			7 6 4 -7		7		

【例】

	M. mu		M. data
	6		0 1 2 12
			1 1 3 9
			2 3 1 -3
			3 3 6 14
			4 4 3 24
			5 5 2 18
			6 6 1 15
			7 6 4 -7

	T. data		T. mu
	0 1 3 -3		7
	1		
	2		
	3		
	4		
	5		
	6		
	7		

	T. nu
	6
	T. tu
	8

【例】

M. data			
0	1	2	12
1	1	3	9
2	3	1	-3
3	3	6	14
4	4	3	24
5	5	2	18
6	6	1	15
7	6	4	-7

M. mu

6

M. nu

7

M. tu

8

T. data			
0	1	3	-3
1			
2			
3			
4			
5			
6			
7			

T. mu

7

T. nu

6

T. tu

8

【例】

M. mu
6

M. nu
7

M. tu
8

	M. data		
0	1	2	12
1	1	3	9
2	3	1	-3
3	3	6	14
4	4	3	24
5	5	2	18
6	6	1	15
7	6	4	-7

	T. data		
0	1	3	-3
1			
2			
3			
4			
5			
6			
7			

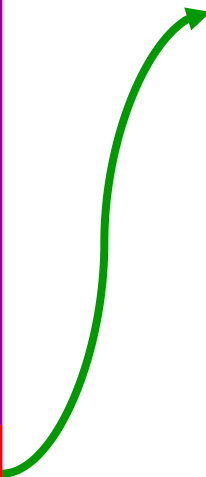
T. mu
7

T. nu
6

T. tu
8

【例】

M. mu			M. data			T. data			T. mu		
6			0	1	2	12	0	1	3	-3	
M. nu			1	1	3	9	1	1	6	15	
7			2	3	1	-3	2				
M. tu			3	3	6	14	3				
8			4	4	3	24	4				
			5	5	2	18	5				
			6	6	1	15	6				
			7	6	4	-7	7				



【例】

	M. data
0	1 2 12
1	1 3 9
2	3 1 -3
3	3 6 14
4	4 3 24
5	5 2 18
6	6 1 15
7	6 4 -7

M. mu
6

M. nu
7

M. tu
8

	T. data
0	1 3 -3
1	1 6 15
2	
3	
4	
5	
6	
7	

T. mu
7

T. nu
6

T. tu
8

【例】

	M. data				T. data			
	0	1	2	12	0	1	3	-3
	1	1	3	9	1	1	6	15
M. mu	2	3	1	-3	2	2	1	12
6	3	3	6	14	3			
	4	4	3	24	4			
M. nu	5	5	2	18	5			
7	6	6	1	15	6			
	7	6	4	-7	7			
M. tu								
8								

	T. data			
	0	1	3	-3
	1	1	6	15
	2	2	1	12
	3			
	4			
	5			
	6			
	7			

T. mu	7
T. nu	6
T. tu	8

【例】

	M. data
0	1 2 12
1	1 3 9
2	3 1 -3
3	3 6 14
4	4 3 24
5	5 2 18
6	6 1 15
7	6 4 -7

M. mu
6

M. nu
7

M. tu
8

	T. data
0	1 3 -3
1	1 6 15
2	2 1 12
3	
4	
5	
6	
7	

T. mu
7

T. nu
6

T. tu
8

【例】

	M. data
0	12
1	9
2	-3
3	14
4	24
5	18
6	15
7	-7

M. mu
6

M. nu
7

M. tu
8

	T. data
0	-3
1	15
2	12
3	
4	
5	
6	
7	

T. mu
7

T. nu
6

T. tu
8

【例】

	M. data
0	12
1	9
2	-3
3	14
4	24
5	18
6	15
7	-7

M. mu
6

M. nu
7

M. tu
8

	T. data
0	-3
1	15
2	12
3	
4	
5	
6	
7	

T. mu
7

T. nu
6

T. tu
8

【例】

	M. data		
0	1	2	12
1	1	3	9
2	3	1	-3
3	3	6	14
4	4	3	24
5	5	2	18
6	6	1	15
7	6	4	-7

M. mu
6

M. nu
7

M. tu
8

	T. data		
0	1	3	-3
1	1	6	15
2	2	1	12
3			
4			
5			
6			
7			

T. mu
7

T. nu
6

T. tu
8

【例】

M. data			
0	1	2	12
1	1	3	9
2	3	1	-3
3	3	6	14
4	4	3	24
5	5	2	18
6	6	1	15
7	6	4	-7

M. mu
6

M. nu
7

M. tu
8

T. data			
0	1	3	-3
1	1	6	15
2	2	1	12
3			
4			
5			
6			
7			

T. mu
7

T. nu
6

T. tu
8

【例】

M. mu

6

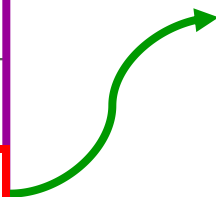
M. nu

7

M. tu

8

0	1	2	12
1	1	3	9
2	3	1	-3
3	3	6	14
4	4	3	24
5	5	2	18
6	6	1	15
7	6	4	-7



0	1	3	-3
1	1	6	15
2	2	1	12
3	2	5	18
4			
5			
6			
7			

T. mu

7

T. nu

6

T. tu

8

【例】

	M. data		
0	1	2	12
1	1	3	9
2	3	1	-3
3	3	6	14
4	4	3	24
5	5	2	18
6	6	1	15
7	6	4	-7

M. mu

6

M. nu

7

M. tu

8

	T. data		
0	1	3	-3
1	1	6	15
2	2	1	12
3	2	5	18
4			
5			
6			
7			

T. mu

7

T. nu

6

T. tu

8

【例】

	M. data		
0	1	2	12
1	1	3	9
2	3	1	-3
3	3	6	14
4	4	3	24
5	5	2	18
6	6	1	15
7	6	4	-7

M. mu
6

M. nu
7

M. tu
8

	T. data		
0	1	3	-3
1	1	6	15
2	2	1	12
3	2	5	18
4			
5			
6			
7			

T. mu
7

T. nu
6

T. tu
8

【例】

	M. data
0	1 2 12
1	1 3 9
2	3 1 -3
3	3 6 14
4	4 3 24
5	5 2 18
6	6 1 15
7	6 4 -7

M. mu
6

M. nu
7

M. tu
8

	T. data
0	1 3 -3
1	1 6 15
2	2 1 12
3	2 5 18
4	
5	
6	
7	

T. mu
7

T. nu
6

T. tu
8

【例】

	M. data
0	12
1	9
2	-3
3	14
4	24
5	18
6	15
7	-7

M. mu
6

M. nu
7

M. tu
8

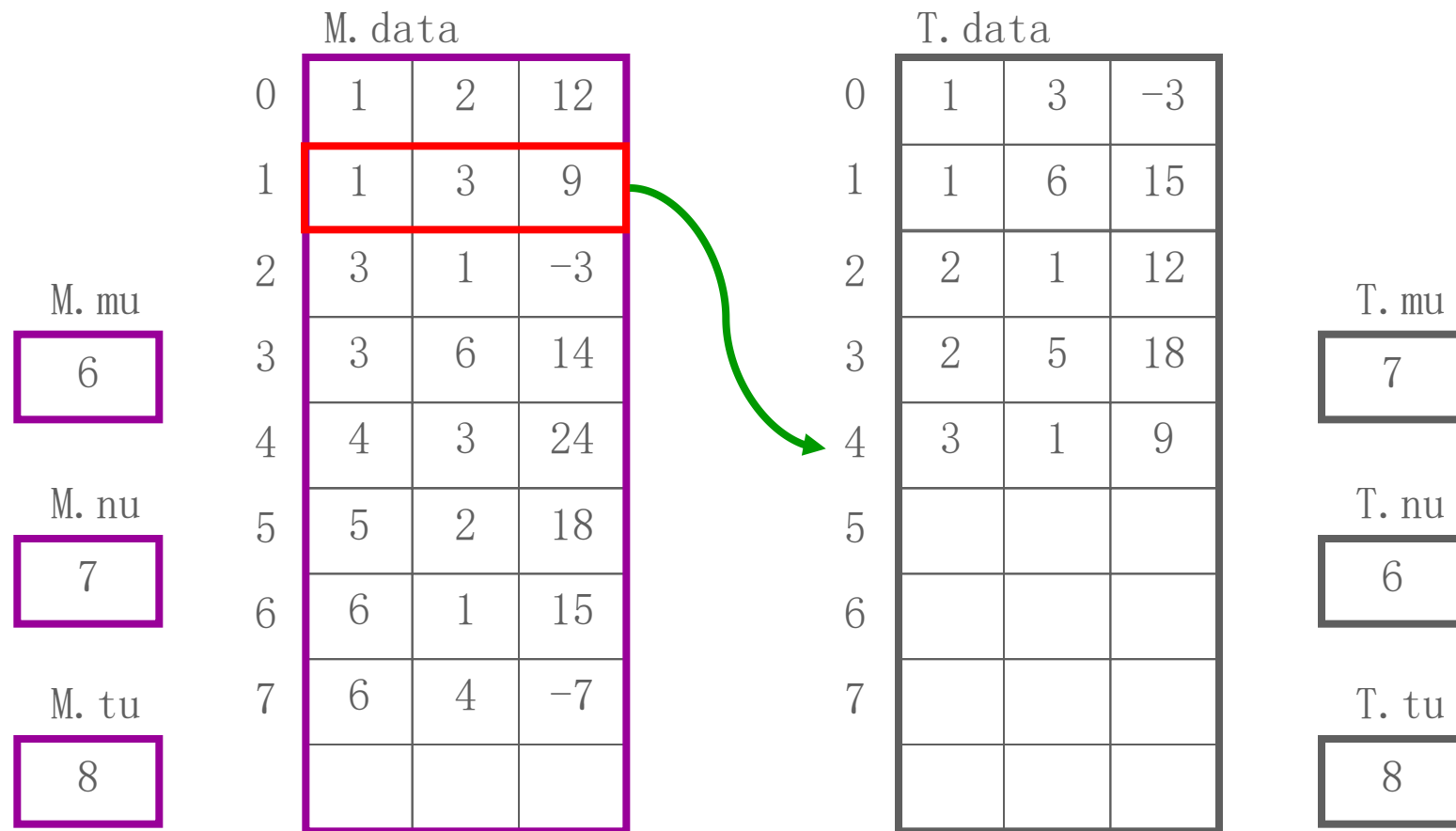
	T. data
0	-3
1	15
2	12
3	18
4	
5	
6	
7	

T. mu
7

T. nu
6

T. tu
8

【例】



【例】

	M. data		
0	1	2	12
1	1	3	9
2	3	1	-3
3	3	6	14
4	4	3	24
5	5	2	18
6	6	1	15
7	6	4	-7

M. mu

6

M. nu

7

M. tu

8

	T. data		
0	1	3	-3
1	1	6	15
2	2	1	12
3	2	5	18
4	3	1	9
5			
6			
7			

T. mu

7

T. nu

6

T. tu

8

【例】

	M. data		
0	1	2	12
1	1	3	9
2	3	1	-3
3	3	6	14
4	4	3	24
5	5	2	18
6	6	1	15
7	6	4	-7

M. mu
6

M. nu
7

M. tu
8

	T. data		
0	1	3	-3
1	1	6	15
2	2	1	12
3	2	5	18
4	3	1	9
5			
6			
7			

T. mu
7

T. nu
6

T. tu
8

【例】

	M. data		
0	1	2	12
1	1	3	9
2	3	1	-3
3	3	6	14
4	4	3	24
5	5	2	18
6	6	1	15
7	6	4	-7

M. mu
6

M. nu
7

M. tu
8

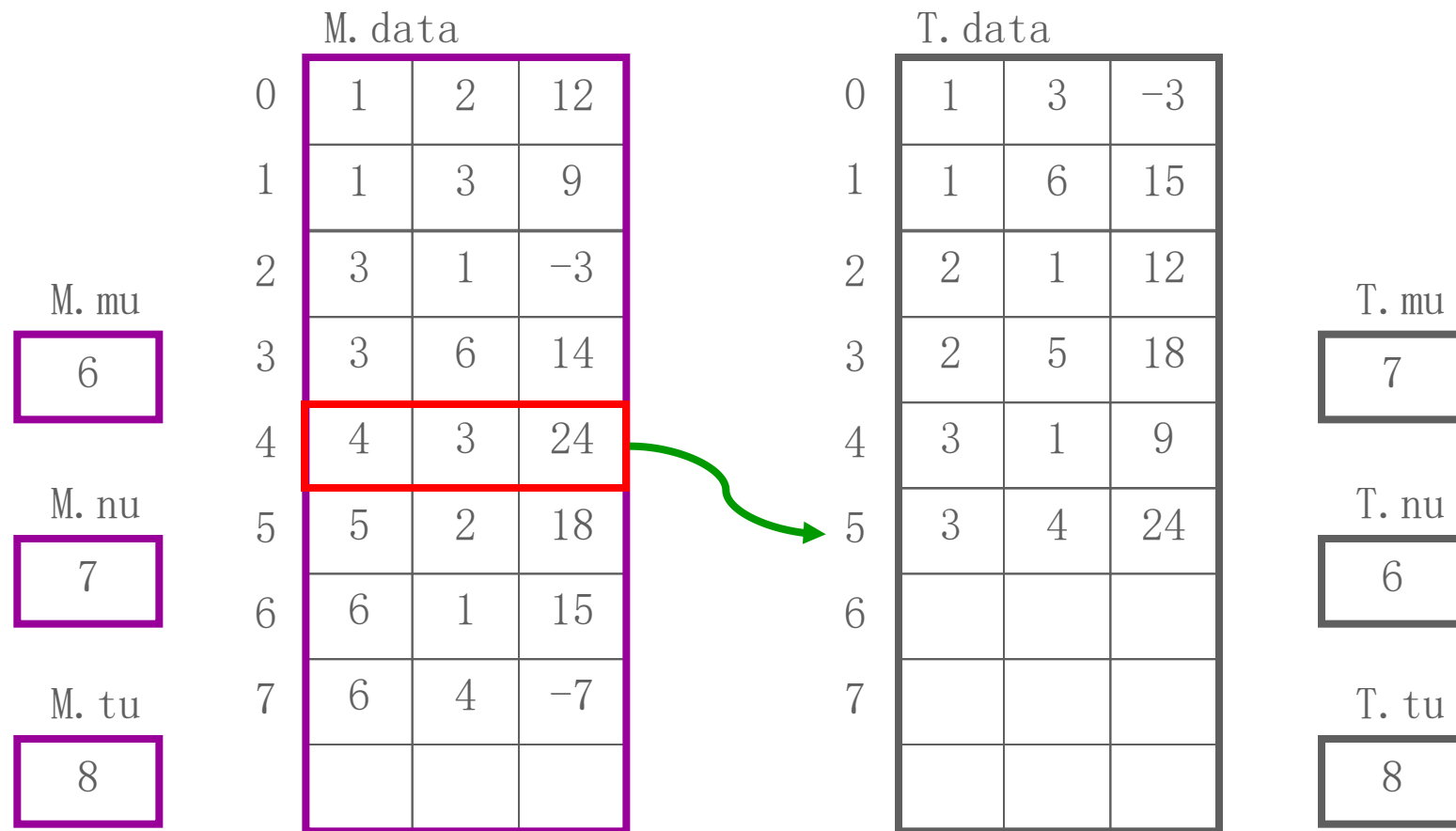
	T. data		
0	1	3	-3
1	1	6	15
2	2	1	12
3	2	5	18
4	3	1	9
5			
6			
7			

T. mu
7

T. nu
6

T. tu
8

【例】



【例】

	M. mu		M. data
	6	0	1 2 12
		1	1 3 9
		2	3 1 -3
		3	3 6 14
		4	4 3 24
	M. nu	5	5 2 18
	7	6	6 1 15
		7	6 4 -7
	M. tu		
	8		

	T. data		T. mu
0	1 3 -3		7
1	1 6 15		
2	2 1 12		
3	2 5 18		
4	3 1 9		
5	3 4 24		T. nu
6			6
7			
			T. tu
			8

【例】

	M. data		
0	1	2	12
1	1	3	9
2	3	1	-3
3	3	6	14
4	4	3	24
5	5	2	18
6	6	1	15
7	6	4	-7

M. mu

6

M. nu

7

M. tu

8

	T. data		
0	1	3	-3
1	1	6	15
2	2	1	12
3	2	5	18
4	3	1	9
5	3	4	24
6			
7			

T. mu

7

T. nu

6

T. tu

8

【例】

	M. data
0	1 2 12
1	1 3 9
2	3 1 -3
3	3 6 14
4	4 3 24
5	5 2 18
6	6 1 15
7	6 4 -7

M. mu
6

M. nu
7

M. tu
8

	T. data
0	1 3 -3
1	1 6 15
2	2 1 12
3	2 5 18
4	3 1 9
5	3 4 24
6	
7	

T. mu
7

T. nu
6

T. tu
8

【例】

	M. data
0	1 2 12
1	1 3 9
2	3 1 -3
3	3 6 14
4	4 3 24
5	5 2 18
6	6 1 15
7	6 4 -7

M. mu
6

M. nu
7

M. tu
8

	T. data
0	1 3 -3
1	1 6 15
2	2 1 12
3	2 5 18
4	3 1 9
5	3 4 24
6	
7	

T. mu
7

T. nu
6

T. tu
8

【例】

	M. data
0	1 2 12
1	1 3 9
2	3 1 -3
3	3 6 14
4	4 3 24
5	5 2 18
6	6 1 15
7	6 4 -7

M. mu
6

M. nu
7

M. tu
8

	T. data
0	1 3 -3
1	1 6 15
2	2 1 12
3	2 5 18
4	3 1 9
5	3 4 24
6	
7	

T. mu
7

T. nu
6

T. tu
8

【例】

	M. data		
0	1	2	12
1	1	3	9
2	3	1	-3
3	3	6	14
4	4	3	24
5	5	2	18
6	6	1	15
7	6	4	-7

M. mu
6

M. nu
7

M. tu
8

	T. data		
0	1	3	-3
1	1	6	15
2	2	1	12
3	2	5	18
4	3	1	9
5	3	4	24
6			
7			

T. mu
7

T. nu
6

T. tu
8

【例】

	M. mu		M. data
	6		0 1 2 12
			1 1 3 9
			2 3 1 -3
			3 3 6 14
			4 4 3 24
			5 5 2 18
			6 6 1 15
			7 6 4 -7

	T. mu		T. data
	7		0 1 3 -3
			1 1 6 15
			2 2 1 12
			3 2 5 18
			4 3 1 9
			5 3 4 24
			6
			7

	T. mu
	7
	T. nu
	6
	T. tu
	8

【例】

	M. data		
0	1	2	12
1	1	3	9
2	3	1	-3
3	3	6	14
4	4	3	24
5	5	2	18
6	6	1	15
7	6	4	-7

M. mu
6

M. nu
7

M. tu
8

	T. data		
0	1	3	-3
1	1	6	15
2	2	1	12
3	2	5	18
4	3	1	9
5	3	4	24
6			
7			

T. mu
7

T. nu
6

T. tu
8

【例】

	M. data
0	1 2 12
1	1 3 9
2	3 1 -3
3	3 6 14
4	4 3 24
5	5 2 18
6	6 1 15
7	6 4 -7

M. mu
6

M. nu
7

M. tu
8

	T. data
0	1 3 -3
1	1 6 15
2	2 1 12
3	2 5 18
4	3 1 9
5	3 4 24
6	
7	

T. mu
7

T. nu
6

T. tu
8

【例】

	M. data		
0	1	2	12
1	1	3	9
2	3	1	-3
3	3	6	14
4	4	3	24
5	5	2	18
6	6	1	15
7	6	4	-7

M. mu

6

M. nu

7

M. tu

8

	T. data		
0	1	3	-3
1	1	6	15
2	2	1	12
3	2	5	18
4	3	1	9
5	3	4	24
6			
7			

T. mu

7

T. nu

6

T. tu

8

【例】

M. mu
6

M. nu
7

M. tu
8

	M. data		
0	1	2	12
1	1	3	9
2	3	1	-3
3	3	6	14
4	4	3	24
5	5	2	18
6	6	1	15
7	6	4	-7

	T. data		
0	1	3	-3
1	1	6	15
2	2	1	12
3	2	5	18
4	3	1	9
5	3	4	24
6			
7			

T. mu
7

T. nu
6

T. tu
8

【例】

M. mu

6

M. nu

7

M. tu

8

	M. data		
0	1	2	12
1	1	3	9
2	3	1	-3
3	3	6	14
4	4	3	24
5	5	2	18
6	6	1	15
7	6	4	-7

	T. data		
0	1	3	-3
1	1	6	15
2	2	1	12
3	2	5	18
4	3	1	9
5	3	4	24
6	4	6	-7
7			

T. mu

7

T. nu

6

T. tu

8

【例】

	M. data
0	1 2 12
1	1 3 9
2	3 1 -3
3	3 6 14
4	4 3 24
5	5 2 18
6	6 1 15
7	6 4 -7

M. mu
6
M. nu
7
M. tu
8

	T. data
0	1 3 -3
1	1 6 15
2	2 1 12
3	2 5 18
4	3 1 9
5	3 4 24
6	4 6 -7
7	

T. mu
7
T. nu
6
T. tu
8

【例】

	M. data		
0	1	2	12
1	1	3	9
2	3	1	-3
3	3	6	14
4	4	3	24
5	5	2	18
6	6	1	15
7	6	4	-7

M. mu
6

M. nu
7

M. tu
8

	T. data		
0	1	3	-3
1	1	6	15
2	2	1	12
3	2	5	18
4	3	1	9
5	3	4	24
6	4	6	-7
7			

T. mu
7

T. nu
6

T. tu
8

【例】

	M. data		
0	1	2	12
1	1	3	9
2	3	1	-3
3	3	6	14
4	4	3	24
5	5	2	18
6	6	1	15
7	6	4	-7

M. mu

6

M. nu

7

M. tu

8

	T. data		
0	1	3	-3
1	1	6	15
2	2	1	12
3	2	5	18
4	3	1	9
5	3	4	24
6	4	6	-7
7			

T. mu

7

T. nu

6

T. tu

8

【例】

	M. data
0	1 2 12
1	1 3 9
2	3 1 -3
3	3 6 14
4	4 3 24
5	5 2 18
6	6 1 15
7	6 4 -7

M. mu
6

M. nu
7

M. tu
8

	T. data
0	1 3 -3
1	1 6 15
2	2 1 12
3	2 5 18
4	3 1 9
5	3 4 24
6	4 6 -7
7	

T. mu
7

T. nu
6

T. tu
8

【例】

	M. data		
0	1	2	12
1	1	3	9
2	3	1	-3
3	3	6	14
4	4	3	24
5	5	2	18
6	6	1	15
7	6	4	-7

M. mu
6

M. nu
7

M. tu
8

	T. data		
0	1	3	-3
1	1	6	15
2	2	1	12
3	2	5	18
4	3	1	9
5	3	4	24
6	4	6	-7
7			

T. mu
7

T. nu
6

T. tu
8

【例】

	M. data		
0	1	2	12
1	1	3	9
2	3	1	-3
3	3	6	14
4	4	3	24
5	5	2	18
6	6	1	15
7	6	4	-7

M. mu

6

M. nu

7

M. tu

8

	T. data		
0	1	3	-3
1	1	6	15
2	2	1	12
3	2	5	18
4	3	1	9
5	3	4	24
6	4	6	-7
7			

T. mu

7

T. nu

6

T. tu

8

【例】

	M. data		
0	1	2	12
1	1	3	9
2	3	1	-3
3	3	6	14
4	4	3	24
5	5	2	18
6	6	1	15
7	6	4	-7

M. mu

6

M. nu

7

M. tu

8

	T. data		
0	1	3	-3
1	1	6	15
2	2	1	12
3	2	5	18
4	3	1	9
5	3	4	24
6	4	6	-7
7			

T. mu

7

T. nu

6

T. tu

8

【例】

	M. data		
0	1	2	12
1	1	3	9
2	3	1	-3
3	3	6	14
4	4	3	24
5	5	2	18
6	6	1	15
7	6	4	-7

M. mu
6

M. nu
7

M. tu
8

	T. data		
0	1	3	-3
1	1	6	15
2	2	1	12
3	2	5	18
4	3	1	9
5	3	4	24
6	4	6	-7
7			

T. mu
7

T. nu
6

T. tu
8

【例】

	M. data
0	1 2 12
1	1 3 9
2	3 1 -3
3	3 6 14
4	4 3 24
5	5 2 18
6	6 1 15
7	6 4 -7

M. mu
6

M. nu
7

M. tu
8

	T. data
0	1 3 -3
1	1 6 15
2	2 1 12
3	2 5 18
4	3 1 9
5	3 4 24
6	4 6 -7
7	

T. mu
7

T. nu
6

T. tu
8

【例】

	M. data
0	1 2 12
1	1 3 9
2	3 1 -3
3	3 6 14
4	4 3 24
5	5 2 18
6	6 1 15
7	6 4 -7

M. mu
6

M. nu
7

M. tu
8

	T. data
0	1 3 -3
1	1 6 15
2	2 1 12
3	2 5 18
4	3 1 9
5	3 4 24
6	4 6 -7
7	

T. mu
7

T. nu
6

T. tu
8

【例】

	M. data
0	12
1	9
2	-3
3	14
4	24
5	18
6	15
7	-7
M. mu	6
M. nu	7
M. tu	8

	T. data
0	-3
1	15
2	12
3	18
4	9
5	24
6	-7
7	
T. mu	7
T. nu	6
T. tu	8

【例】

	M. data
0	1 2 12
1	1 3 9
2	3 1 -3
3	3 6 14
4	4 3 24
5	5 2 18
6	6 1 15
7	6 4 -7

M. mu
6

M. nu
7

M. tu
8

	T. data
0	1 3 -3
1	1 6 15
2	2 1 12
3	2 5 18
4	3 1 9
5	3 4 24
6	4 6 -7
7	

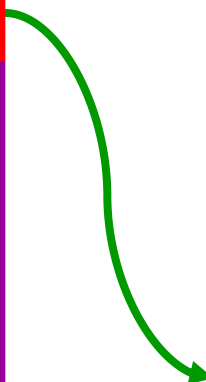
T. mu
7

T. nu
6

T. tu
8

【例】

	M. data				T. data				
	0	1	2	12	0	1	3	-3	
	1	1	3	9	1	1	6	15	
	2	3	1	-3	2	2	1	12	
M. mu	3	3	6	14	3	2	5	18	T. mu
6	4	4	3	24	4	3	1	9	7
	5	5	2	18	5	3	4	24	
M. nu	6	6	1	15	6	4	6	-7	T. nu
7	7	6	4	-7	7	6	3	14	6
M. tu									T. tu
8									8



【例】

	M. mu		M. data
	6	0	1 2 12
		1	1 3 9
		2	3 1 -3
		3	3 6 14
		4	4 3 24
	M. nu	5	5 2 18
	7	6	6 1 15
		7	6 4 -7
	M. tu		
	8		

	T. data		T. mu
0	1 3 -3		7
1	1 6 15		
2	2 1 12		
3	2 5 18		
4	3 1 9		
5	3 4 24		T. nu
6	4 6 -7		6
7	6 3 14		
			T. tu
			8

【例】

	M. data
0	12
1	9
2	-3
3	14
4	24
5	18
6	15
7	-7

M. mu
6

M. nu
7

M. tu
8

	T. data
0	-3
1	15
2	12
3	18
4	9
5	24
6	-7
7	14

T. mu
7

T. nu
6

T. tu
8

【例】

	M. data		
0	1	2	12
1	1	3	9
2	3	1	-3
3	3	6	14
4	4	3	24
5	5	2	18
6	6	1	15
7	6	4	-7

M. mu

6

M. nu

7

M. tu

8

	T. data		
0	1	3	-3
1	1	6	15
2	2	1	12
3	2	5	18
4	3	1	9
5	3	4	24
6	4	6	-7
7	6	3	14

T. mu

7

T. nu

6

T. tu

8

【例】

	M. data
0	1 2 12
1	1 3 9
2	3 1 -3
3	3 6 14
4	4 3 24
5	5 2 18
6	6 1 15
7	6 4 -7

M. mu
6

M. nu
7

M. tu
8

	T. data
0	1 3 -3
1	1 6 15
2	2 1 12
3	2 5 18
4	3 1 9
5	3 4 24
6	4 6 -7
7	6 3 14

T. mu
7

T. nu
6

T. tu
8

【例】

	M. data
0	1 2 12
1	1 3 9
2	3 1 -3
3	3 6 14
4	4 3 24
5	5 2 18
6	6 1 15
7	6 4 -7

M. mu
6

M. nu
7

M. tu
8

	T. data
0	1 3 -3
1	1 6 15
2	2 1 12
3	2 5 18
4	3 1 9
5	3 4 24
6	4 6 -7
7	6 3 14

T. mu
7

T. nu
6

T. tu
8

【例】

	M. data
0	1 2 12
1	1 3 9
2	3 1 -3
3	3 6 14
4	4 3 24
5	5 2 18
6	6 1 15
7	6 4 -7
M. mu	6
M. nu	7
M. tu	8

	T. data
0	1 3 -3
1	1 6 15
2	2 1 12
3	2 5 18
4	3 1 9
5	3 4 24
6	4 6 -7
7	6 3 14
T. mu	7
T. nu	6
T. tu	8

【例】

	M. data		
0	1	2	12
1	1	3	9
2	3	1	-3
3	3	6	14
4	4	3	24
5	5	2	18
6	6	1	15
7	6	4	-7

M. mu

6

M. nu

7

M. tu

8

	T. data		
0	1	3	-3
1	1	6	15
2	2	1	12
3	2	5	18
4	3	1	9
5	3	4	24
6	4	6	-7
7	6	3	14

T. mu

7

T. nu

6

T. tu

8

【例】

	M. data		
0	1	2	12
1	1	3	9
2	3	1	-3
3	3	6	14
4	4	3	24
5	5	2	18
6	6	1	15
7	6	4	-7

M. mu
6

M. nu
7

M. tu
8

	T. data		
0	1	3	-3
1	1	6	15
2	2	1	12
3	2	5	18
4	3	1	9
5	3	4	24
6	4	6	-7
7	6	3	14

T. mu
7

T. nu
6

T. tu
8

【例】

	M. data
0	12
1	9
2	-3
3	14
4	24
5	18
6	15
7	-7

M. mu
6

M. nu
7

M. tu
8

	T. data
0	-3
1	15
2	12
3	18
4	9
5	24
6	-7
7	14

T. mu
7

T. nu
6

T. tu
8

【例】

	M. data		
0	1	2	12
1	1	3	9
2	3	1	-3
3	3	6	14
4	4	3	24
5	5	2	18
6	6	1	15
7	6	4	-7

M. mu

6

M. nu

7

M. tu

8

	T. data		
0	1	3	-3
1	1	6	15
2	2	1	12
3	2	5	18
4	3	1	9
5	3	4	24
6	4	6	-7
7	6	3	14

T. mu

7

T. nu

6

T. tu

8

【例】

M. mu
6

M. nu
7

M. tu
8

	M. data		
0	1	2	12
1	1	3	9
2	3	1	-3
3	3	6	14
4	4	3	24
5	5	2	18
6	6	1	15
7	6	4	-7

	T. data		
0	1	3	-3
1	1	6	15
2	2	1	12
3	2	5	18
4	3	1	9
5	3	4	24
6	4	6	-7
7	6	3	14

T. mu
7

T. nu
6

T. tu
8

【例】

	M. data		
0	1	2	12
1	1	3	9
2	3	1	-3
3	3	6	14
4	4	3	24
5	5	2	18
6	6	1	15
7	6	4	-7

M. mu
6

M. nu
7

M. tu
8

	T. data		
0	1	3	-3
1	1	6	15
2	2	1	12
3	2	5	18
4	3	1	9
5	3	4	24
6	4	6	-7
7	6	3	14

T. mu
7

T. nu
6

T. tu
8

```
int transpose_naive(triple_matrix *Q, triple_matrix M){  
    int t,j,k=0;  
    for(j=0; j<M.cn; j++){  
        for(t=0;t<M.tn;t++){  
            if(M.data[t].col == j){//找到j列元素  
                Q->data[k].col = M.data[t].row;  
                Q->data[k].row = M.data[t].col;  
                Q->data[k++].value = M.data[t].value;  
            }  
        }  
    }  
    return 1;}  
}
```

【算法分析】

对M中每一列的处理均需扫描整个三元组表，时间复杂度为 $O(nu \cdot tu)$

当 tu 与 $\mu \cdot nu$ 等数量级时，时间复杂度达 $O(nu^2 \cdot \mu)$

高于非压缩矩阵转置算法的 $O(\mu \cdot nu)$

因此我称之为naive方法，适用于 $t \ll \mu \cdot nu$ 情况

【改进分析】

思考如何改进？慢在哪里？

对每一列均要扫描整个三元组表，为什么？

不知道转置后三元组表中的每列元素的信息。

具体需要什么信息？

每列每个元素在转置后的三元组表的位置

由于：转置后的三元组表也保持行主序，因此对转置前的矩阵中同一列元素：在转置后的三元组表中，应按照行号从小到大排列

如能知道：

1、每列有多少个元素

2、每列第一个元素转置后在三元组表中的位置

即可实现对每个元素转置后位置的定位！！！！

OK，需要两个数组num/cpot，保存1, 2的信息。

信息从哪里来？

通过遍历一次三元组表得到

先看看动画过程

注：num中保存M中各列
(即T中各行)的非零元数

【例】

M. mu

6

M. nu

7

M. tu

8

M. data

num

T. data

T. mu

7

T. nu

6

T. tu

8

注：num中保存M中各列
(即T中各行)的非零元数

【例】

M. mu

6

M. nu

7

M. tu

8

M. data

num

0	1	2	12
1	1	3	9
2	3	1	-3
3	3	6	14
4	4	3	24
5	5	2	18
6	6	1	15
7	6	4	-7
8			

T. data

T. mu

7

T. nu

6

T. tu

8

注：num中保存M中各列
(即T中各行)的非零元数

【例】

M. mu

6

M. nu

7

M. tu

8

M. data

num

T. data

T. mu

7

T. nu

6

T. tu

8

注：num中保存M中各列
(即T中各行)的非零元数

【例】

M. mu

6

M. nu

7

M. tu

8

M. data

num

T. data

T. mu

7

T. nu

6

T. tu

8

注：num中保存M中各列
(即T中各行)的非零元数

【例】

M. mu

6

M. nu

7

M. tu

8

M. data

num

T. data

T. mu

7

T. nu

6

T. tu

8

注：num中保存M中各列
(即T中各行)的非零元数

【例】

		M. data				num	
M. mu	0	1	2	12		0	1
	1	1	3	9		1	1
	2	3	1	-3		2	1
	3	3	6	14		3	0
	4	4	3	24		4	0
	5	5	2	18		5	0
	6	6	1	15		6	0
	7	6	4	-7			

T. data			
			0
			1
			2
			3
			4
			5
			6
			7

T. mu
7

T. nu
6

T. tu
8

注： num中保存M中各列
(即T中各行)的非零元数

【例】

		M. data					num
M. mu	0	1	2	12	0	1	
	1	1	3	9	1	1	
	2	3	1	-3	2	1	
	3	3	6	14	3	0	
	4	4	3	24	4	0	
	5	5	2	18	5	0	
	6	6	1	15	6	0	
	7	6	4	-7			
M. nu							
M. tu							

		T. data					
	0				0		
	1				1		
	2				2		
	3				3		
	4				4		
	5				5		
	6				6		
	7				7		
T. mu							
T. nu							
T. tu							

注：num中保存M中各列
(即T中各行)的非零元数

【例】

M. data				num		
M. mu	0	1	2	12	0	1
6	1	1	3	9	1	1
M. nu	2	3	1	-3	2	1
7	3	3	6	14	3	0
M. tu	4	4	3	24	4	0
8	5	5	2	18	5	1
	6	6	1	15	6	0
	7	6	4	-7		

T. data				T. mu	
	0				7
	1				
	2				
	3				
	4				6
	5				
	6				
	7				8

注：num中保存M中各列
(即T中各行)的非零元数

【例】

M. mu

6

M. nu

7

M. tu

8

M. data

num

T. data

T. mu

7

T. nu

6

T. tu

8

注：num中保存M中各列
(即T中各行)的非零元数

【例】

M. mu

6

M. nu

7

M. tu

8

M. data

num

T. data

T. mu

7

T. nu

6

T. tu

8

注： num中保存M中各列
(即T中各行)的非零元数

【例】

M. data				num		
M. mu 6 M. nu 7 M. tu 8	0	1	2	12	0	1
	1	1	3	9	1	1
	2	3	1	-3	2	2
	3	3	6	14	3	0
	4	4	3	24	4	0
	5	5	2	18	5	1
	6	6	1	15	6	0
	7	6	4	-7		

T. data				0	T. mu 7 T. nu 6 T. tu 8
				1	
				1	
				2	
				0	
				0	
				1	
				0	

注：num中保存M中各列
(即T中各行)的非零元数

【例】

M. data			num		
0	1	2	12	0	1
1	1	3	9	1	2
2	3	1	-3	2	2
3	3	6	14	3	0
4	4	3	24	4	0
5	5	2	18	5	1
6	6	1	15	6	0
7	6	4	-7		

M. mu

6

M. nu

7

M. tu

8

0 1 | 2 | 12 |

1 1 | 3 | 9 |

2 3 | 1 | -3 |

3 3 | 6 | 14 |

4 4 | 3 | 24 |

5 5 | 2 | 18 |

6 6 | 1 | 15 |

7 6 | 4 | -7 |

| | |

0 1 |

1 2 |

2 2 |

3 0 |

4 0 |

5 1 |

6 0 |

T. data			
			0
			1
			2
			3
			4
			5
			6
			7

T. mu
7

T. nu
6

T. tu
8

注：num中保存M中各列
(即T中各行)的非零元数

【例】

M. mu

6

M. nu

7

M. tu

8

M. data

num

T. data

T. mu

7

T. nu

6

T. tu

8

注：num中保存M中各列
(即T中各行)的非零元数

【例】

		M. data				num	
M. mu	0	1	2	12		0	2
	1	1	3	9		1	2
	2	3	1	-3		2	2
	3	3	6	14		3	0
	4	4	3	24		4	0
	5	5	2	18		5	1
	6	6	1	15		6	0
	7	6	4	-7			
M. nu							
M. tu							

T. data			
			0
			1
			2
			3
			4
			5
			6
			7

T. mu
7

T. nu
6

T. tu
8

注：num中保存M中各列
(即T中各行)的非零元数

【例】

M. data				num		
M. mu 6 M. nu 7 M. tu 8	0	1	2	12	0	2
	1	1	3	9	1	2
	2	3	1	-3	2	2
	3	3	6	14	3	0
	4	4	3	24	4	0
	5	5	2	18	5	1
	6	6	1	15	6	0
	7	6	4	-7		

T. data				T. mu 7 T. nu 6 T. tu 8	
				0	
				1	
				2	
				3	
				4	
				5	
				6	
				7	

注：num中保存M中各列
(即T中各行)的非零元数

【例】

M. mu

6

M. nu

7

M. tu

8

M. data			num		
0	1	2	12	0	2
1	1	3	9	1	2
2	3	1	-3	2	2
3	3	6	14	3	1
4	4	3	24	4	0
5	5	2	18	5	1
6	6	1	15	6	0
7	6	4	-7		

M. mu

6

M. nu

7

M. tu

8

T. data

T. mu

7

T. nu

6

T. tu

8

注：num中保存M中各列
(即T中各行)的非零元数

【例】

M. mu

6

M. nu

7

M. tu

8

M. data

num

T. data

T. mu

7

T. nu

6

T. tu

8

注：cpot中保存T中各行
非零元当前存储位置。

【例】

M. data				num	cpot	T. data		
0	1	2	12	0	2	0		0
1	1	3	9	1	2	1		1
2	3	1	-3	2	2	2		2
3	3	6	14	3	1	3		3
4	4	3	24	4	0	4		4
5	5	2	18	5	1	5		5
6	6	1	15	6	0	6		6
7	6	4	-7					7

M. mu

6

M. nu

7

M. tu

8

T. mu

7

T. nu

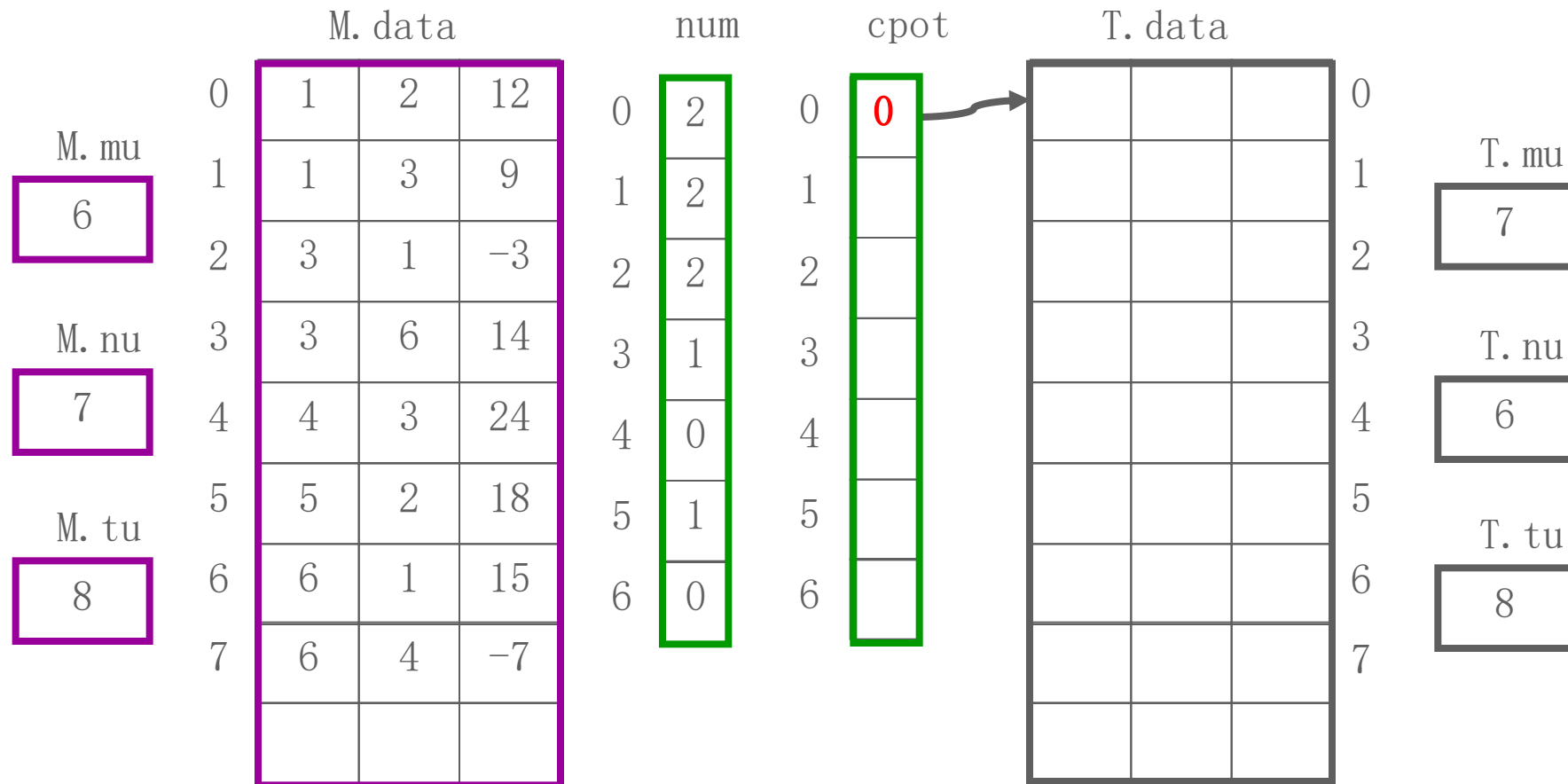
6

T. tu

8

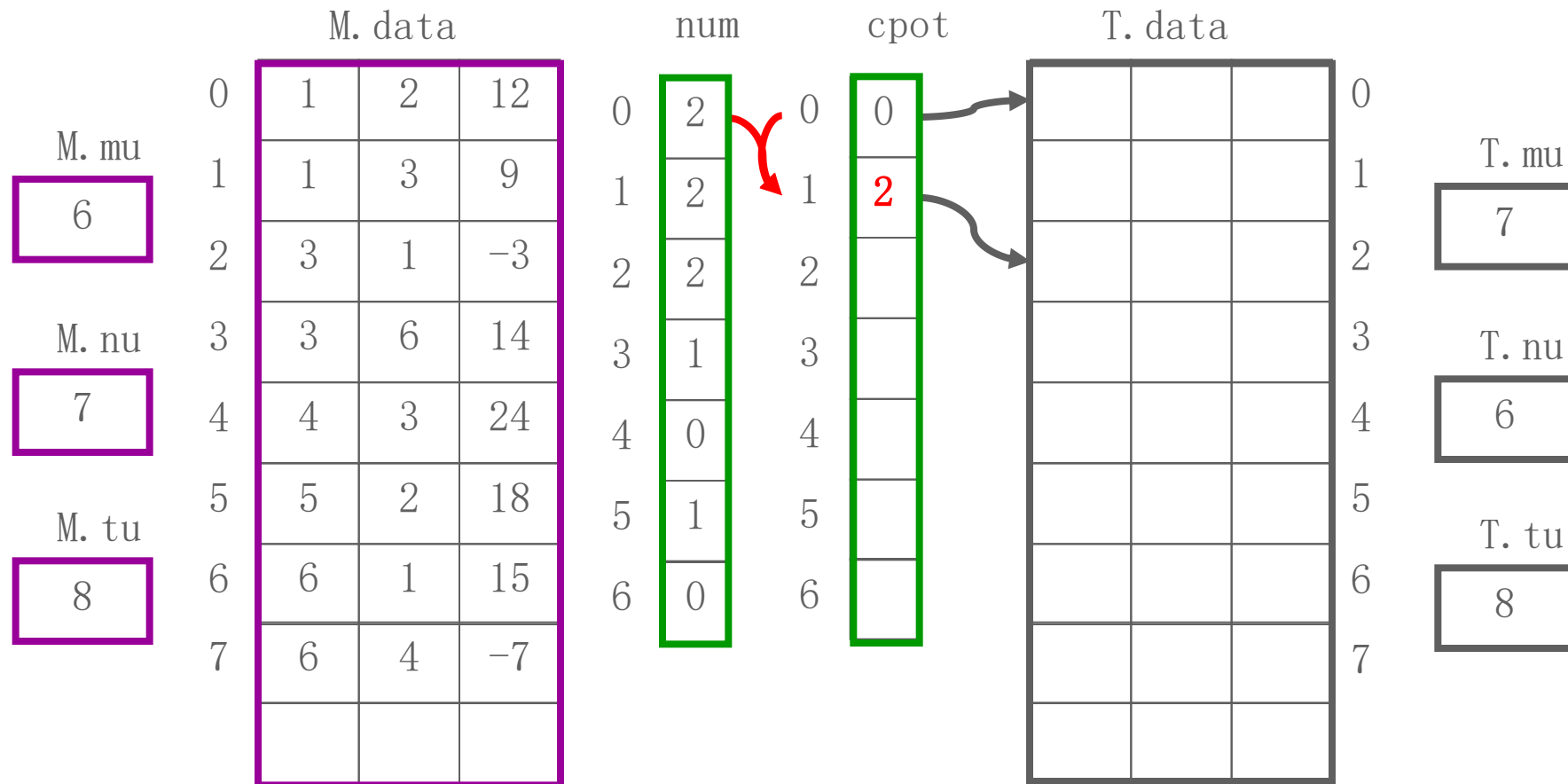
注：cpot中保存T中各行
非零元当前存储位置。

【例】



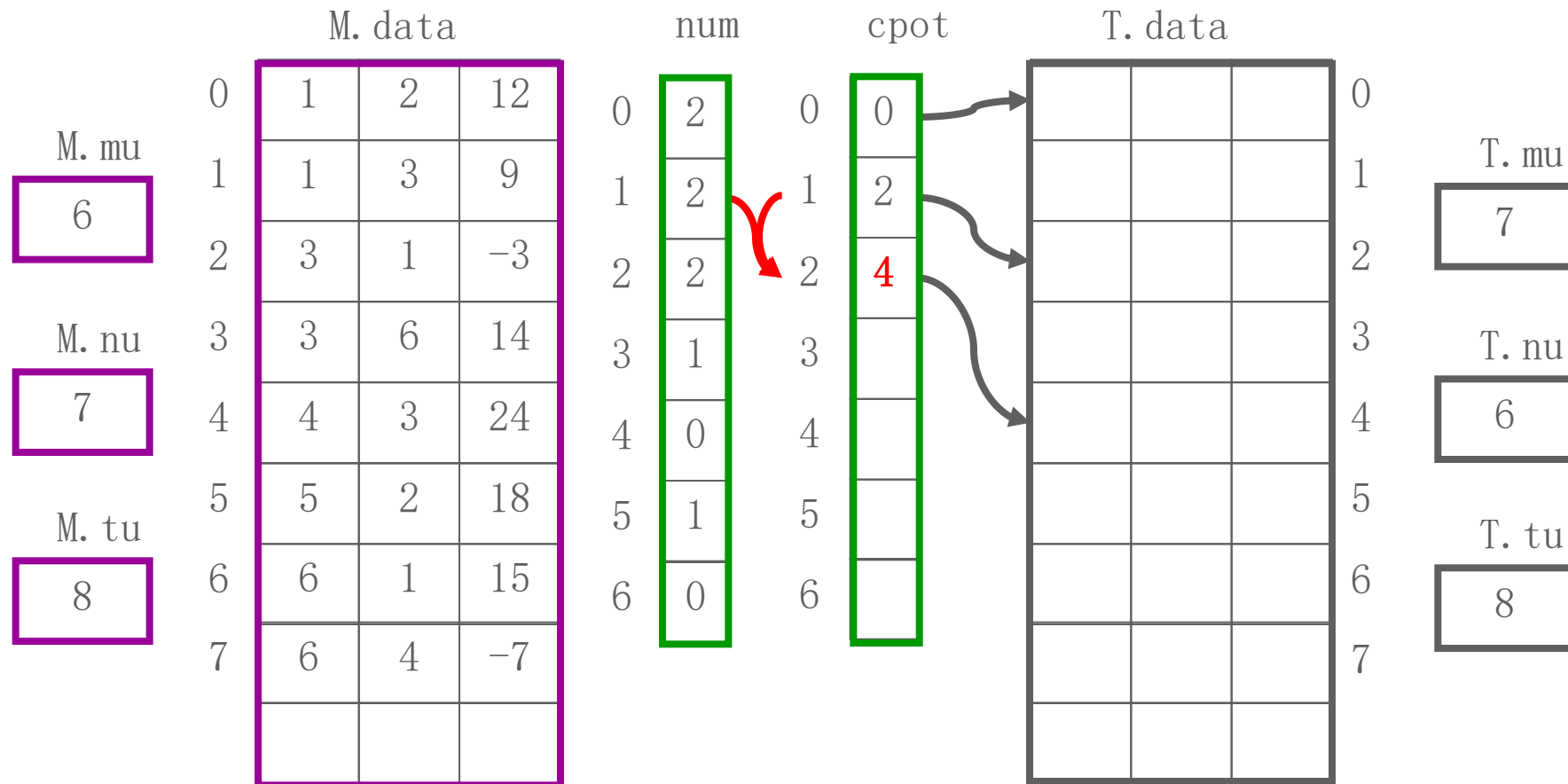
注：cpot中保存T中各行
非零元当前存储位置。

【例】



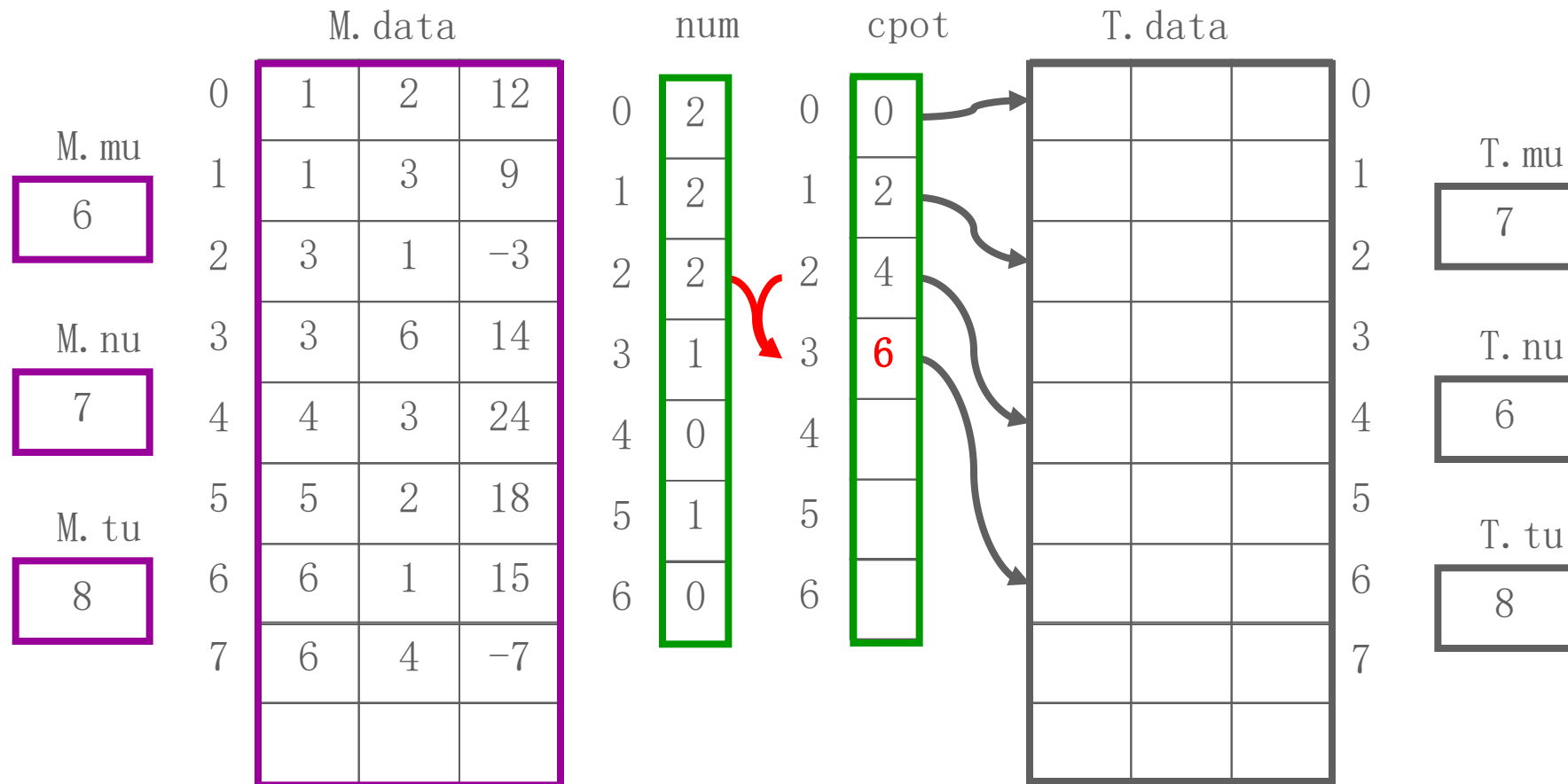
注：cpot中保存T中各行
非零元当前存储位置。

【例】



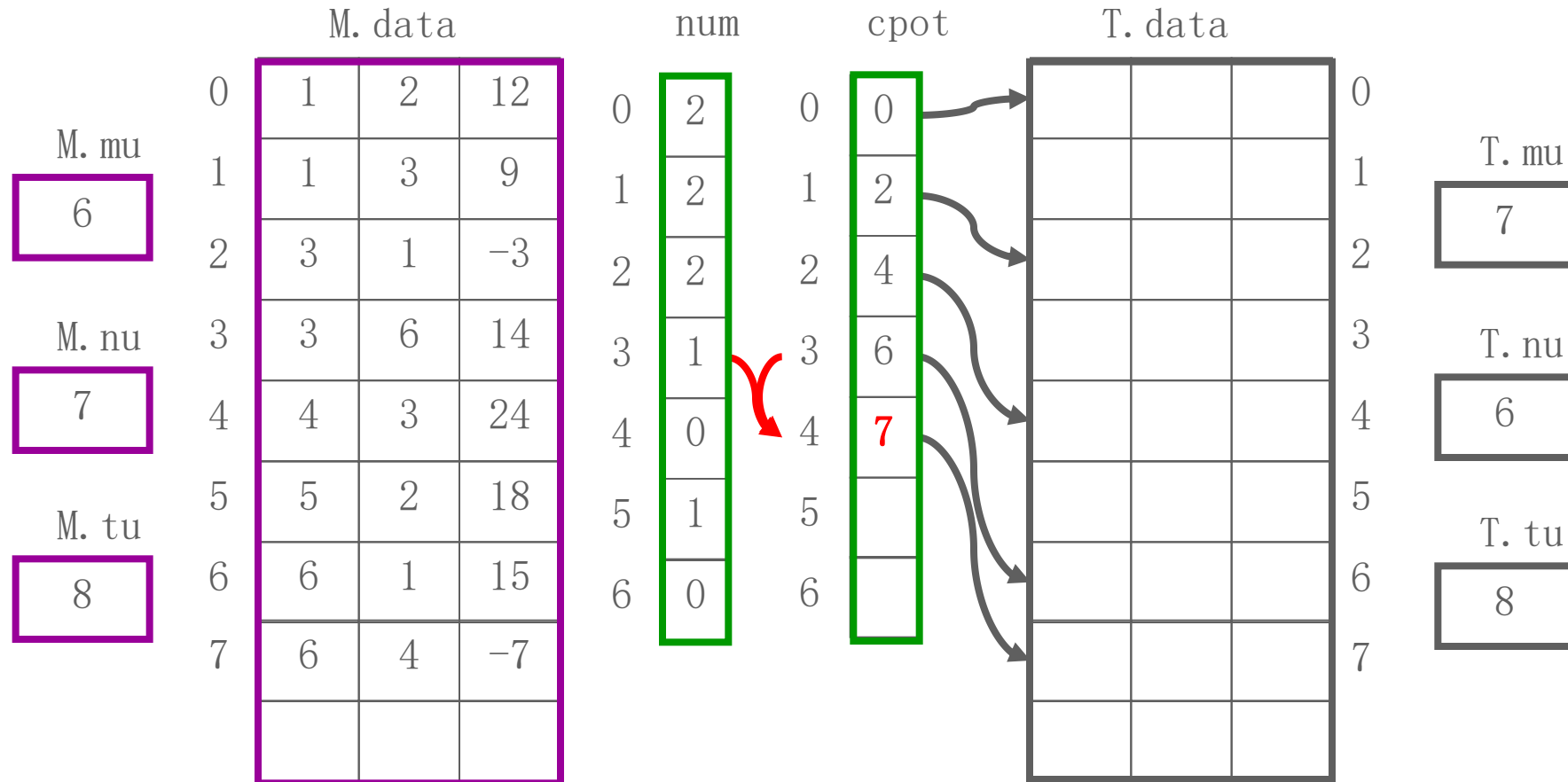
注：cpot中保存T中各行
非零元当前存储位置。

【例】



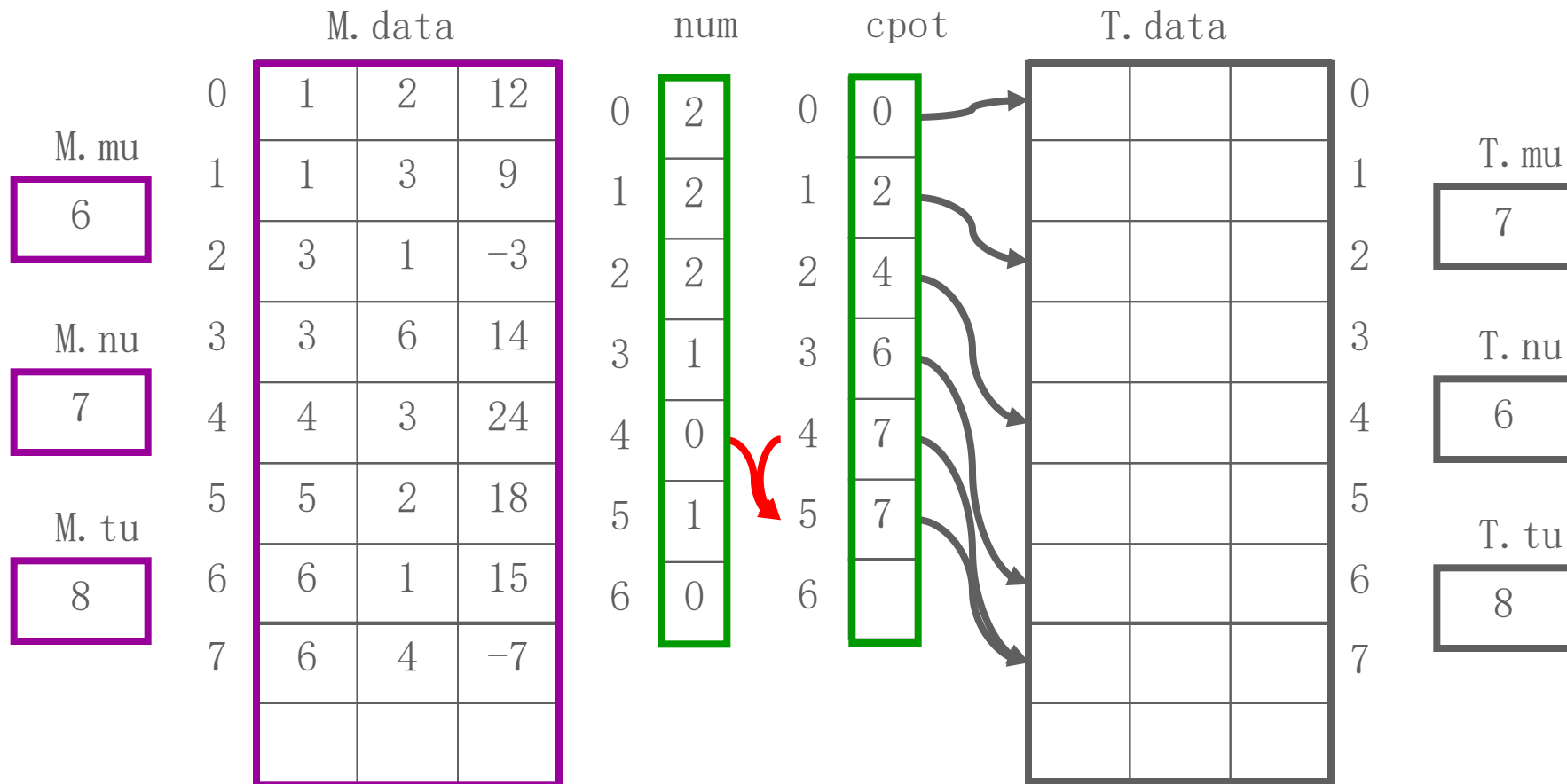
注：cpot中保存T中各行
非零元当前存储位置。

【例】



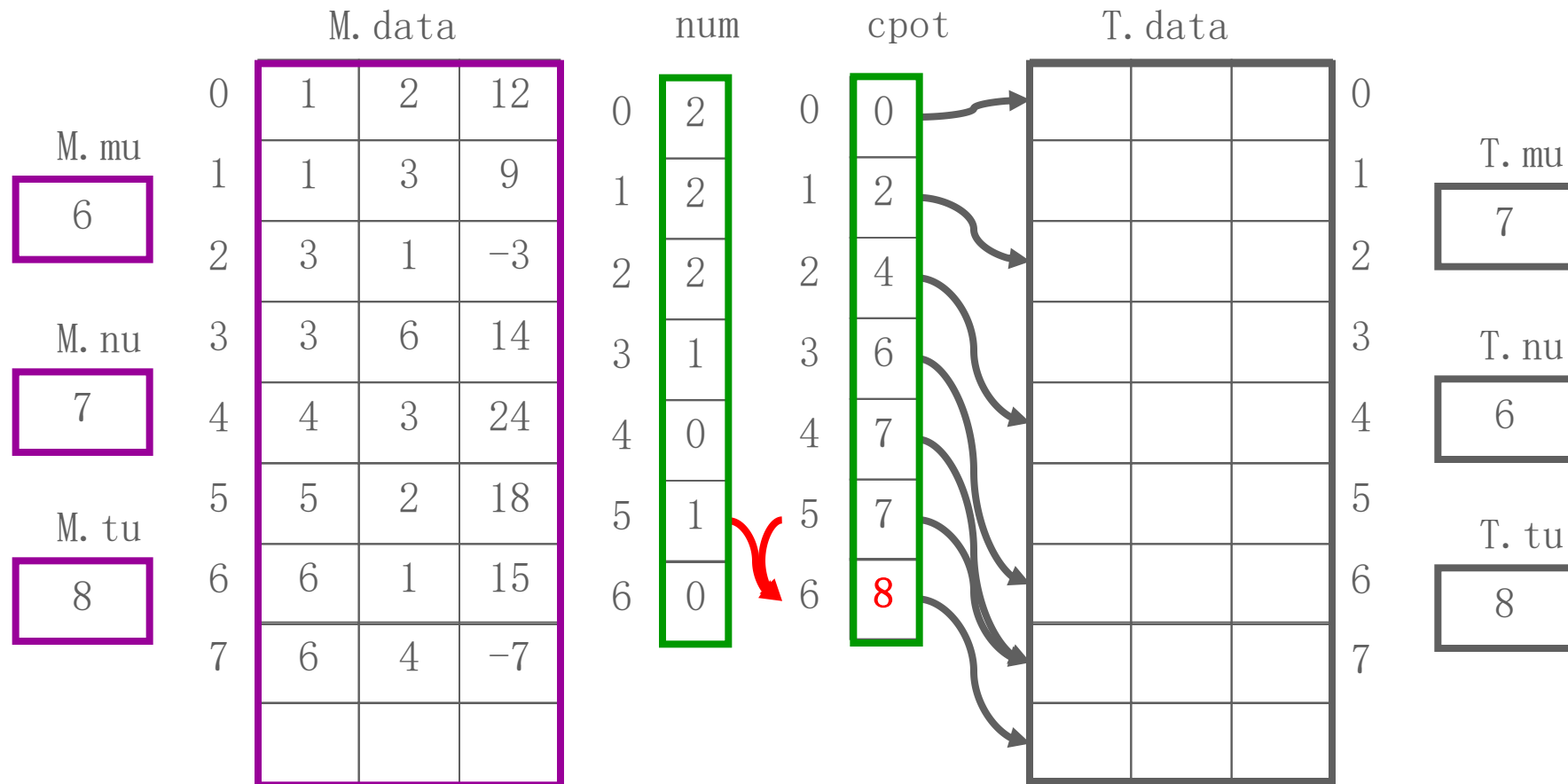
注：cpot中保存T中各行
非零元当前存储位置。

【例】



注：cpot中保存T中各行
非零元当前存储位置。

【例】



注：cpot中保存T中各行非零元当前存储位置。

【例】

The diagram illustrates the data layout for two models, 'M' and 'T'. Each model has three input features and one output feature. The 'M' model's data is shown as a 3x3 grid, and the 'T' model's data is shown as a 3x3 grid. The 'M' model's output is a 3x3 grid, and the 'T' model's output is a 3x3 grid.

	M. data	num	cpot	T. data
0	1	2	12	
1	1	3	9	
2	3	1	-3	
3	3	6	14	
4	4	3	24	
5	5	2	18	
6	6	1	15	
7	6	4	-7	

注：cpot中保存T中各行
非零元当前存储位置。

【例】

M. data			num	cpot	T. data		
0	1	2	12	0	2	0	
1	1	3	9	1	2	2	
2	3	1	-3	2	2	4	
3	3	6	14	3	1	6	
4	4	3	24	4	0	7	
5	5	2	18	5	1	7	
6	6	1	15	6	0	8	
7	6	4	-7				

M. mu

6

M. nu

7

M. tu

8

T. mu

7

T. nu

6

T. tu

8

注：cpot中保存T中各行
非零元当前存储位置。

【例】

M. data			num	cpot	T. data		
0	1	2	12	0	2	0	0
1	1	3	9	1	2	1	2
2	3	1	-3	2	2	4	
3	3	6	14	3	1	6	
4	4	3	24	4	0	7	
5	5	2	18	5	1	7	
6	6	1	15	6	0	8	
7	6	4	-7				

M. mu

6

M. nu

7

M. tu

8

T. mu

7

T. nu

6

T. tu

8

注：cpot中保存T中各行
非零元当前存储位置。

【例】

M. data				num		cpot		T. data			
0	1	2	12	0	2	0	0				0
1	1	3	9	1	2	1	3				1
2	3	1	-3	2	2	2	4	2	1	12	2
3	3	6	14	3	1	3	6				3
4	4	3	24	4	0	4	7				4
5	5	2	18	5	1	5	7				5
6	6	1	15	6	0	6	8				6
7	6	4	-7								7

M. mu

6

M. nu

7

M. tu

8

T. mu

7

T. nu

6

T. tu

8

注：cpot中保存T中各行
非零元当前存储位置。

【例】

M. data				num		cpot		T. data			
0	1	2	12	0	2	0	0				0
1	1	3	9	1	2	1	3				1
2	3	1	-3	2	2	2	4	2	1	12	2
3	3	6	14	3	1	3	6				3
4	4	3	24	4	0	4	7	3	1	9	4
5	5	2	18	5	1	5	7				5
6	6	1	15	6	0	6	8				6
7	6	4	-7								7

M. mu

6

M. nu

7

M. tu

8

T. mu

7

T. nu

6

T. tu

8

注：cpot中保存T中各行非零元当前存储位置。

【例】

M. data

0	1	2	12
1	1	3	9
2	3	1	-3
3	3	6	14
4	4	3	24
5	5	2	18
6	6	1	15
7	6	4	-7

num

0	2
1	2
2	2
3	1
4	0
5	1
6	0

cpot

0	0
1	3
2	5
3	6
4	7
5	7
6	8

T. data

2	1	12
3	1	9

M. mu

6

M. nu

7

M. tu

8

T. mu

7

T. nu

6

T. tu

8

注：cpot中保存T中各行非零元当前存储位置。

【例】

The diagram illustrates the layout of data for two models, 'M' and 'T'. The 'M' model has three input variables: M.mu (6), M.nu (7), and M.tu (8). The 'T' model has three input variables: T.mu (7), T.nu (6), and T.tu (8). The 'M' model's data is shown in a 3x8 grid, and the 'T' model's data is shown in a 3x8 grid. The 'M' model's data is highlighted with a red border, and the 'T' model's data is highlighted with a green border.

	M. data			num	cpot	T. data		
0	1	2	12	0	2	0		0
1	1	3	9	1	2	3		1
2	3	1	-3	2	2	5	2	2
3	3	6	14	3	1	6		3
4	4	3	24	4	0	7	3	4
5	5	2	18	5	1	7		5
6	6	1	15	6	0	8		6
7	6	4	-7					7

注：cpot中保存T中各行
非零元当前存储位置。

【例】

M. data				num	cpot		T. data		
M. mu 6	0	1	2	12	0	2	0	0	0
	1	1	3	9	1	2	1	3	1
	2	3	1	-3	2	2	2	5	2
	3	3	6	14	3	1	3	6	3
	4	4	3	24	4	0	4	7	4
	5	5	2	18	5	1	5	7	5
	6	6	1	15	6	0	6	8	6
	7	6	4	-7					7
M. nu 7									
M. tu 8									

注：cpot中保存T中各行非零元当前存储位置。

【例】

M. data

0	1	2	12
1	1	3	9
2	3	1	-3
3	3	6	14
4	4	3	24
5	5	2	18
6	6	1	15
7	6	4	-7

M. mu

6

M. nu

7

M. tu

8

num

0	2
1	2
2	2
3	1
4	0
5	1
6	0

cspot

0	0
1	3
2	5
3	6
4	7
5	7
6	8

T. data

1	3	-3	0
			1
2	1	12	2
			3
3	1	9	4
			5
			6
			7

T. mu

7

T. nu

6

T. tu

8

注：cpot中保存T中各行非零元当前存储位置。

【例】

[illegible]

注：cpot中保存T中各行
非零元当前存储位置。

【例】

M. data			num	cpot	T. data		
0	1	2	12	0	2	0	1
1	1	3	9	1	2	1	3
2	3	1	-3	2	2	5	2
3	3	6	14	3	1	6	3
4	4	3	24	4	7	4	7
5	5	2	18	5	7	5	7
6	6	1	15	6	8	6	8
7	6	4	-7			7	

M. mu
6

M. nu
7

M. tu
8

T. mu
7

T. nu
6

T. tu
8

注：cpot中保存T中各行非零元当前存储位置。

【例】

Diagram illustrating the layout of a 3D tensor M (3x4x3) and its corresponding 2D slices $M.\mu$, $M.\nu$, and $M.tu$.

The tensor M is represented as a 3D grid with dimensions 3 (rows), 4 (columns), and 3 (depth). The values in the tensor are:

Row	Col	Depth	Value
0	0	0	1
0	0	1	2
0	0	2	12
0	1	0	1
0	1	1	3
0	1	2	9
0	2	0	3
0	2	1	1
0	2	2	-3
1	0	0	3
1	0	1	6
1	0	2	14
1	1	0	4
1	1	1	3
1	1	2	24
1	2	0	5
1	2	1	2
1	2	2	18
2	0	0	6
2	0	1	1
2	0	2	15
2	1	0	6
2	1	1	4
2	1	2	-7
2	2	0	0
2	2	1	0
2	2	2	0

The slices $M.\mu$, $M.\nu$, and $M.tu$ are 2D grids of size 3x4, representing the first, second, and third dimensions of M , respectively. The values in the slices are:

$M.\mu$ (first dimension):

Row	Col	Value
0	0	1
0	1	2
0	2	12
1	0	1
1	1	3
1	2	9
2	0	3
2	1	1
2	2	-3

$M.\nu$ (second dimension):

Row	Col	Value
0	0	3
0	1	6
0	2	14
1	0	4
1	1	3
1	2	24
2	0	5
2	1	2
2	2	18

$M.tu$ (third dimension):

Row	Col	Value
0	0	6
0	1	1
0	2	15
1	0	6
1	1	4
1	2	-7
2	0	0
2	1	0
2	2	0

注：cpot中保存T中各行非零元当前存储位置。

【例】

[illegible]

注：cpot中保存T中各行非零元当前存储位置。

【例】

The diagram illustrates the data structure for the 'M' matrix. It consists of several components:

- M. data**: A 3x3 grid of 3x3 matrices. The matrix at row 4, column 1 (index 4) is highlighted in red.
- M. mu**: A 3x1 vector with values 6, 7, 8.
- M. nu**: A 3x1 vector with values 6, 7, 8.
- M. tu**: A 3x1 vector with values 6, 7, 8.
- num**: A 3x1 vector with values 2, 2, 1.
- cpot**: A 3x1 vector with values 1, 5, 8.
- T. data**: A 3x3 grid of 3x3 matrices. The matrix at row 4, column 1 (index 4) is highlighted in blue.
- T. mu**: A 3x1 vector with values 7, 6, 8.
- T. nu**: A 3x1 vector with values 7, 6, 8.
- T. tu**: A 3x1 vector with values 7, 6, 8.

注：cpot中保存T中各行非零元当前存储位置。

【例】

The diagram illustrates the layout of data for two models, M and T, across seven rows (0 to 6). The data is organized into columns and rows, with specific cells highlighted by red and green boxes.

Model M Data:

	M.data	num	cspot	M.mu
0	1	2	12	6
1	1	3	9	
2	3	1	-3	
3	3	6	14	7
4	4	3	24	
5	5	2	18	
6	6	1	15	8

Model T Data:

	T.data	num	cspot	T.mu
0	1	3	-3	7
1				
2	2	1	12	
3				6
4	3	1	9	
5	3	4	24	5
6				
7	6	3	14	8

The diagram shows the layout of data for the 'M' and 'T' models. The 'M' model has three input columns (M.data, num, cspot) and one output column (M.mu). The 'T' model has three input columns (T.data, num, cspot) and one output column (T.mu). The 'M' model's data is shown in a 7x3 grid, and the 'T' model's data is shown in a 7x3 grid. The 'M' model's output is shown in a 7x1 grid. The 'T' model's output is shown in a 7x1 grid.

注：cpot中保存T中各行
非零元当前存储位置。

【例】

M. data				num	cpot	T. data				
M. mu 6	0	1	2	12	0	2	0	1	0	T. mu 7
	1	1	3	9	1	2	1	3	1	
	2	3	1	-3	2	2	2	6	2	
	3	3	6	14	3	1	3	6	3	
M. nu 7	4	4	3	24	4	0	4	7	4	T. nu 6
	5	5	2	18	5	1	5	8	5	
M. tu 8	6	6	1	15	6	0	6	8	6	T. tu 8
	7	6	4	-7					7	

注：cpot中保存T中各行非零元当前存储位置。

【例】

M. mu

6

M. nu

7

M. tu

8

M. data

num

cput

T. data

T. mu

7

T. nu

6

T. tu

8

注：cpot中保存T中各行非零元当前存储位置。

【例】

The diagram illustrates the layout of data for a 3D convolution operation. It shows three input volumes and one output volume, all with dimensions 8x8x3.

- M.data** (Input 1): A 3D volume with values ranging from -7 to 18. The value 18 is highlighted in red.
- num** (Input 2): A 3D volume with values ranging from 0 to 2. The value 1 is highlighted in red.
- cspot** (Input 3): A 3D volume with values ranging from 1 to 8. The value 3 is highlighted in red.
- T.data** (Output): A 3D volume with values ranging from -3 to 14. The value 7 is highlighted in red.

The diagram also shows the spatial dimensions of the volumes, with the 8x8x3 dimensions indicated by the size of the boxes and the labels M.mu, M.nu, M.tu, T.mu, T.nu, and T.tu.

注：cpot中保存T中各行
非零元当前存储位置。

【例】

M. data				num		cpot		T. data					
M. mu 6	0	1	2	12	0	2	0	1	1	3	-3	0	T. mu 7
	1	1	3	9	1	2	1	3				1	
	2	3	1	-3	2	2	2	6	2	1	12	2	
	3	3	6	14	3	1	3	6	2	5	18	3	
	4	4	3	24	4	0	4	7	3	1	9	4	
	5	5	2	18	5	1	5	8	3	4	24	5	
	6	6	1	15	6	0	6	8				6	
	7	6	4	-7					6	3	14	7	
M. nu 7													T. nu 6
M. tu 8													T. tu 8

注：cpot中保存T中各行
非零元当前存储位置。

【例】

M. data				num		cpot		T. data			
M. mu 6	0	1	2	12	0	2	0	1	1	3	-3
	1	1	3	9	1	2	1	4			
	2	3	1	-3	2	2	2	6	2	1	12
	3	3	6	14	3	1	3	6	2	5	18
M. nu 7	4	4	3	24	4	0	4	7	3	1	9
	5	5	2	18	5	1	5	8	3	4	24
M. tu 8	6	6	1	15	6	0	6	8			
	7	6	4	-7					6	3	14

注：cpot中保存T中各行
非零元当前存储位置。

【例】

M. data				num	cpot	T. data							
M. mu 6	0	1	2	12	0	2	0	1	1	3	-3	0	T. mu 7
	1	1	3	9	1	2	1	4				1	
	2	3	1	-3	2	2	2	6	2	1	12	2	
	3	3	6	14	3	1	3	6	2	5	18	3	
M. nu 7	4	4	3	24	4	0	4	7	3	1	9	4	T. nu 6
	5	5	2	18	5	1	5	8	3	4	24	5	
M. tu 8	6	6	1	15	6	0	6	8				6	T. tu 8
	7	6	4	-7					6	3	14	7	

注：cpot中保存T中各行非零元当前存储位置。

【例】

The diagram illustrates the data structure for the 'M' and 'T' models. The 'M' model has a 3x8 grid of data points (M.data) and a 3x1 grid of parameters (M.mu, M.nu, M.tu). The 'T' model has a 3x8 grid of data points (T.data) and a 3x1 grid of parameters (T.mu, T.nu, T.tu). The 'num' and 'cpot' vectors are shared between the two models. The 'M' model's data points are highlighted in red, and the 'T' model's data points are highlighted in blue. The 'num' and 'cpot' vectors are highlighted in green.

M. data				num		cpot		T. data			
0	1	2	12	0	2	0	1	1	3	-3	0
1	1	3	9	1	2	1	4	1	6	15	1
2	3	1	-3	2	2	2	6	2	1	12	2
3	3	6	14	3	1	3	6	3	5	18	3
4	4	3	24	4	0	4	7	4	1	9	4
5	5	2	18	5	1	5	8	5	4	24	5
6	6	1	15	6	0	6	8	6			6
7	6	4	-7					7	6	3	14

Parameters:

- M. mu: 6
- M. nu: 7
- M. tu: 8
- T. mu: 7
- T. nu: 6
- T. tu: 8

注：cpot中保存T中各行
非零元当前存储位置。

【例】

M. data				num	cpot		T. data							
M. mu 6	0	1	2	12	0	2	0	2	1	3	-3	0	T. mu 7	
	1	1	3	9	1	2	1	4	1	6	15	1		
	2	3	1	-3	2	2	2	6	2	1	12	2		
	3	3	6	14	3	1	3	6	3	2	5	18		3
	4	4	3	24	4	0	4	7	4	3	1	9		4
	5	5	2	18	5	1	5	8	5	3	4	24		5
	6	6	1	15	6	1	6	8	6					6
	7	6	4	-7		0		8		6	3	14		7

注：cpot中保存T中各行
非零元当前存储位置。

【例】

M. data				num		cpot		T. data			
M. mu 6	0	1	2	12	0	2	0	1	3	-3	0
	1	1	3	9	1	2	1	1	6	15	1
	2	3	1	-3	2	2	2	2	1	12	2
	3	3	6	14	3	1	3	2	5	18	3
M. nu 7	4	4	3	24	4	0	4	3	1	9	4
M. tu 8	5	5	2	18	5	1	5	3	4	24	5
	6	6	1	15	6	0	6				6
	7	6	4	-7				6	3	14	7

T. mu 7				T. nu 6		T. tu 8	
------------	--	--	--	------------	--	------------	--

注：cpot中保存T中各行非零元当前存储位置。

【例】

The diagram illustrates the data layout for a 3D convolution operation. It shows three input volumes and one output volume, each with its dimensions and a corresponding value in a box.

M.data (3x3x3):

0	1	2	12
1	1	3	9
2	3	1	-3
3	3	6	14
4	4	3	24
5	5	2	18
6	6	1	15
7	6	4	-7

num (3x1x3):

0	2
1	2
2	2
3	1
4	0
5	1
6	0

cpot (3x1x3):

0	2
1	4
2	6
3	6
4	7
5	8
6	8

T.data (3x3x3):

0	1	3	-3
1	1	6	15
2	2	1	12
3	2	5	18
4	3	1	9
5	3	4	24
6			
7	6	3	14

The output T.data is calculated as M.data * num * cpot. The values in the boxes represent the corresponding values for each index (0-7) in the input and output volumes.

注：cpot中保存T中各行
非零元当前存储位置。

【例】

M. data				num		cpot		T. data			
M. mu 6	0	1	2	12	0	2	0	1	3	-3	0
	1	1	3	9	1	2	1	1	6	15	1
	2	3	1	-3	2	2	2	2	1	12	2
	3	3	6	14	3	1	3	2	5	18	3
	4	4	3	24	4	0	4	3	1	9	4
	5	5	2	18	5	1	5	3	4	24	5
	6	6	1	15	6	0	6	4	6	-7	6
	7	6	4	-7				6	3	14	7

注：cpot中保存T中各行
非零元当前存储位置。

【例】

M. data				num		cpot		T. data			
M. mu 6	0	1	2	12	0	2	0	1	3	-3	0
	1	1	3	9	1	2	1	1	6	15	1
	2	3	1	-3	2	2	2	2	1	12	2
	3	3	6	14	3	1	3	2	5	18	3
	4	4	3	24	4	0	4	3	1	9	4
	5	5	2	18	5	1	5	3	4	24	5
	6	6	1	15	6	0	6	4	6	-7	6
	7	6	4	-7				6	3	14	7

注：cpot中保存T中各行
非零元当前存储位置。

【例】

M. data				num	cpot	T. data					
M. mu 6	0	1	2	12	0	2	0	1	3	-3	0
	1	1	3	9	1	2	1	1	6	15	1
	2	3	1	-3	2	2	2	2	1	12	2
	3	3	6	14	3	1	3	2	5	18	3
	4	4	3	24	4	0	4	3	1	9	4
	5	5	2	18	5	1	5	3	4	24	5
	6	6	1	15	6	0	6	4	6	-7	6
	7	6	4	-7				6	3	14	7
M. nu 7										T. nu 6	
M. tu 8										T. tu 8	

```
int transpose_quick(triple_matrix *Q, triple_matrix M){  
    int i, t, *num, *position, k;  
    Q->cn = M.rn; Q->rn = M.cn; Q->tn = M.tn;  
    num = (int*) malloc(sizeof(int)*M.cn);  
    for(i=0; i<M.cn; i++) num[i]=0;  
    for(t=0; t<M.tn; t++) num[M.data[t].col-1]++;  
  
    position = (int*) malloc(sizeof(int)*Q.rn);  
    position[0] = 0;  
    for(i=1; i<Q.rn; i++)//计算位置向量  
        position[i] = position[i-1] + num[i-1];  
}
```

```
for(t=0;t<M.tn;t++){//扫描一次三元组表
    k=position[M.data[t].col-1];//k为转置后位置
    Q->data[k].col = M.data[t].row;
    Q->data[k].row = M.data[t].col;
    Q->data[k].value = M.data[t].value;
    position[M.data[t].col-1]++;
}
free(num);
free(position);
return 1;
};//边界等条件自行处理，教材中num, cpot中0号元素空置
```

【算法分析】

该算法中有四个平行的单层for循环，循环次数分别为 n 和 t ，因而总的时间复杂度为 $O(n+t)$ ，因此称快速转置。

当 t 与 $m*n$ 等数量级时，时间复杂度为 $O(n*m)$ ，与经典算法相同。

注意， t 不会超过 $m*n$ 的数量级

损失：仅仅是多用了两个辅助向量的存储空间。

精髓：仍然是要充分利用已知信息（本例通过扫描2次得到）

行逻辑链接的顺序表表示稀疏矩阵

```
typedef struct {  
    Triple data[MAXSIZE];  
    int pos[MAXR]; //即前面算出来的cpos  
    int rn, cn, tn;  
} RLSTMatrix;
```

其余操作如相乘，可参看教材等资料

非随机存取，缺少顺序存储的最大优势

稀疏矩阵的链式存储

思考：如何设计？每个节点，节点间的链接？

可以有多种方式来考虑。
如带头结点的三元组链表方案
（自行设计实现）

这里介绍下十字链表方案。

稀疏矩阵中同一行的非0元素，由一个指针链接成一个行链表，由另一个指针链接成一个列链表。则每个非0元素既是某个行链表中的一个结点，同时又是某个列链表中的一个结点，所有的非0元素构成一个十字交叉的链表。称为十字链表。

链式表示（十字链表）

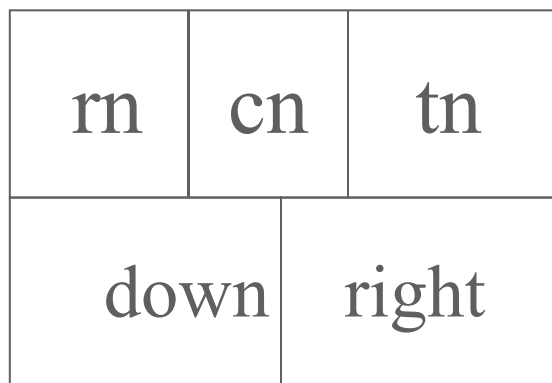
矩阵中非0元素的结点所含的域有：**行、列、值、行指针**(指向同一行的下一个非0元)、**列指针**(指向同一列的下一个非0元)。

row	col	value
down		right

(a) 结点结构

```
typedef struct node {  
    int row , col ;  
    elem_type value ;  
    struct node *down , *right ;  
}cross_node ;  
/* 非0元素结点 */
```

还有一个头结点，除了保存总体信息外，还用两个一维数组分别存储行链表的头指针和列链表的头指针。结点的结构如图所示。(十字链表还可用来表示有向图，暂不做详细介绍)



(b) 头结点结构

```
typedef struct head_node {  
    int  rn;      /* 矩阵的行数 */  
    int  cn;      /* 矩阵的列数 */  
    int  tn;      /* 非0元素总数 */  
    cross_node *rhead ;  
    cross_node *chead ;  
} cross_list ;
```

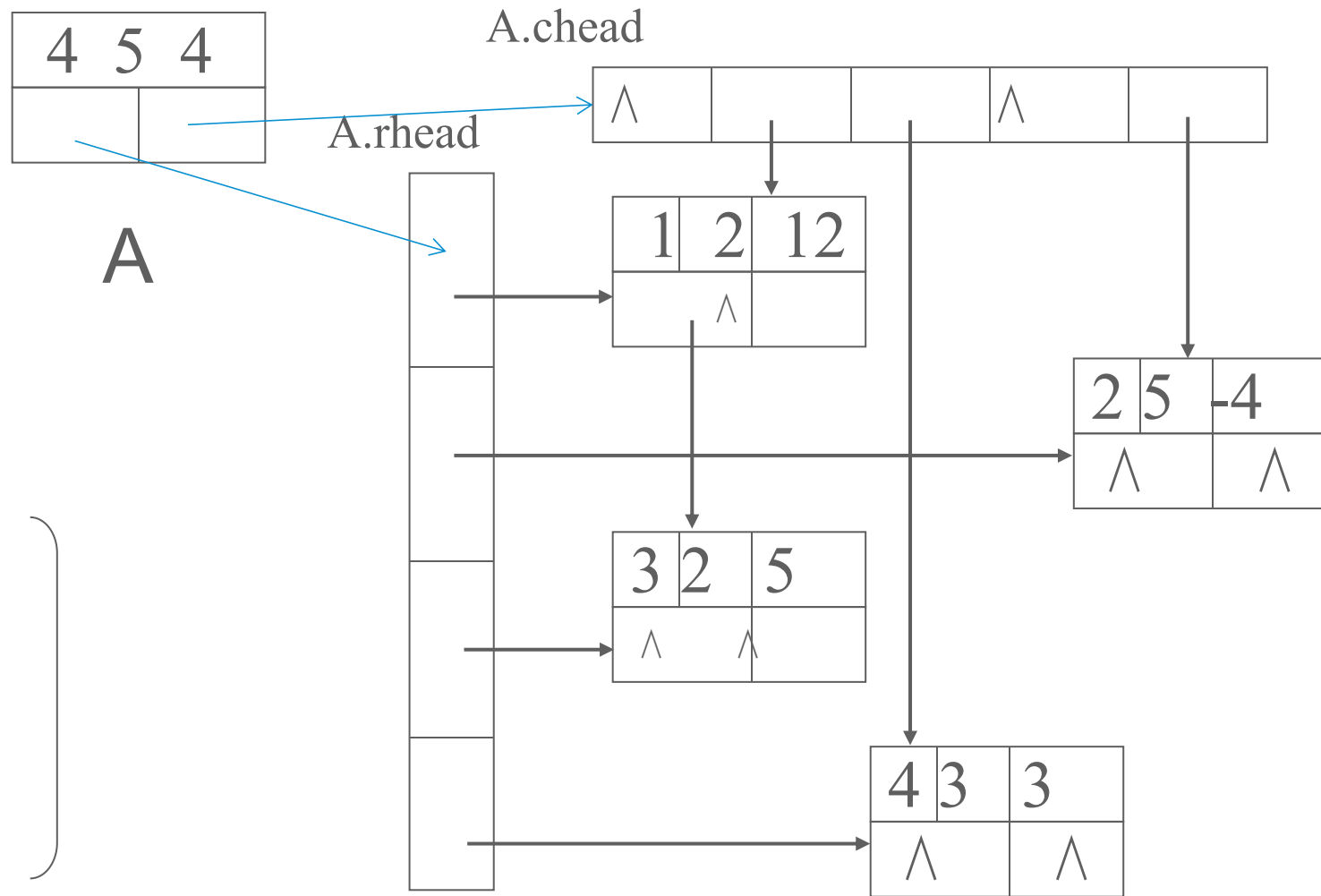
```

cross_list A;
A.chead=malloc...
A.rhead=malloc...
A.cn=4;
...

```

$$A = \begin{pmatrix} 0 & 12 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -4 \\ 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \end{pmatrix}$$

(a) 稀疏矩阵



(b) 稀疏矩阵的十字交叉链表

稀疏矩阵及其十字交叉链表

操作

类似链表操作，注意由两个边链表出发，节点两个方向指针

```
int init(cross_list &M);
```

```
int insert_node(cross_list &M, node p);
```

```
int creat(cross_list &M);
```

```
int destory(cross_list &M);
```

教材104页把init与creat合在一起

其他应用：

```
int add(...),sub(...),T().....有兴趣自行完成
```

广义表简介

广义表 (Lists, 又称列表) 是线性表的推广。线性表定义为 $n \geq 0$ 个元素 $a_1, a_2, a_3, \dots, a_n$ 的有限序列。如果允许线性表的元素具有其自身结构, 这样就产生了广义表的概念。

广义表是 n ($n \geq 0$) 个元素 $a_1, a_2, a_3, \dots, a_n$ 的有限序列, 其中 a_i 或者是原子项, 或者是一个广义表。

通常记作 $LS = (a_1, a_2, a_3, \dots, a_n)$ 。

LS 是广义表的名字, n 为它的长度。

若 a_i 是广义表, 则称它为 LS 的子表。

通常用圆括号将广义表括起来，用逗号分隔其中的元素。

为了区别原子和广义表，书写时用大写字母表示广义表，用小写字母表示原子。

若广义表LS ($n \geq 1$)非空，则a1是LS的表头，其余元素组成的表(a2,...an)称为LS的表尾。

广义表是递归定义的。LISP语言中，广义表作为基本数据结构，广泛应用。

- (1) $A = ()$ —— A 是一个空表，其长度为零。
- (2) $B = (e)$ ——表 B 只有一个原子 e ， B 的长度为1。
- (3) $C = (a, (b, c, d))$ ——表 C 的长度为2，两个元素分别为原子 a 和子表 (b, c, d) 。
- (4) $D = (A, B, C)$ ——表 D 的长度为3，三个元素都是广义表。显然，将子表的值代入后，
则有 $D = ((), (e), (a, (b, c, d)))$ 。
- (5) $E = (a, E)$ ——这是一个递归的表，它的长度为2， E 相当于一个无限的广义表 $E = (a, (a, (a, (a, \dots))))$ 。

广义表的ADT见P107

广义表的存储结构

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	value	标志tag=1	表头指针hp	表尾指针tp
---------	-------	---------	--------	--------

(a) 原子结点

(b) 表结点

广义表的链表结点结构示意图

相应的数据结构定义如下：

```
typedef struct GLNode
```

```
{ int tag ;    /* 标志域，为1：表结点；为0：原子结点 */
```

```
union
```

```
{ elemtype value;    /* 原子结点的值域 */
```

```
struct { struct GLNode *hp , *tp ; }ptr ;
```

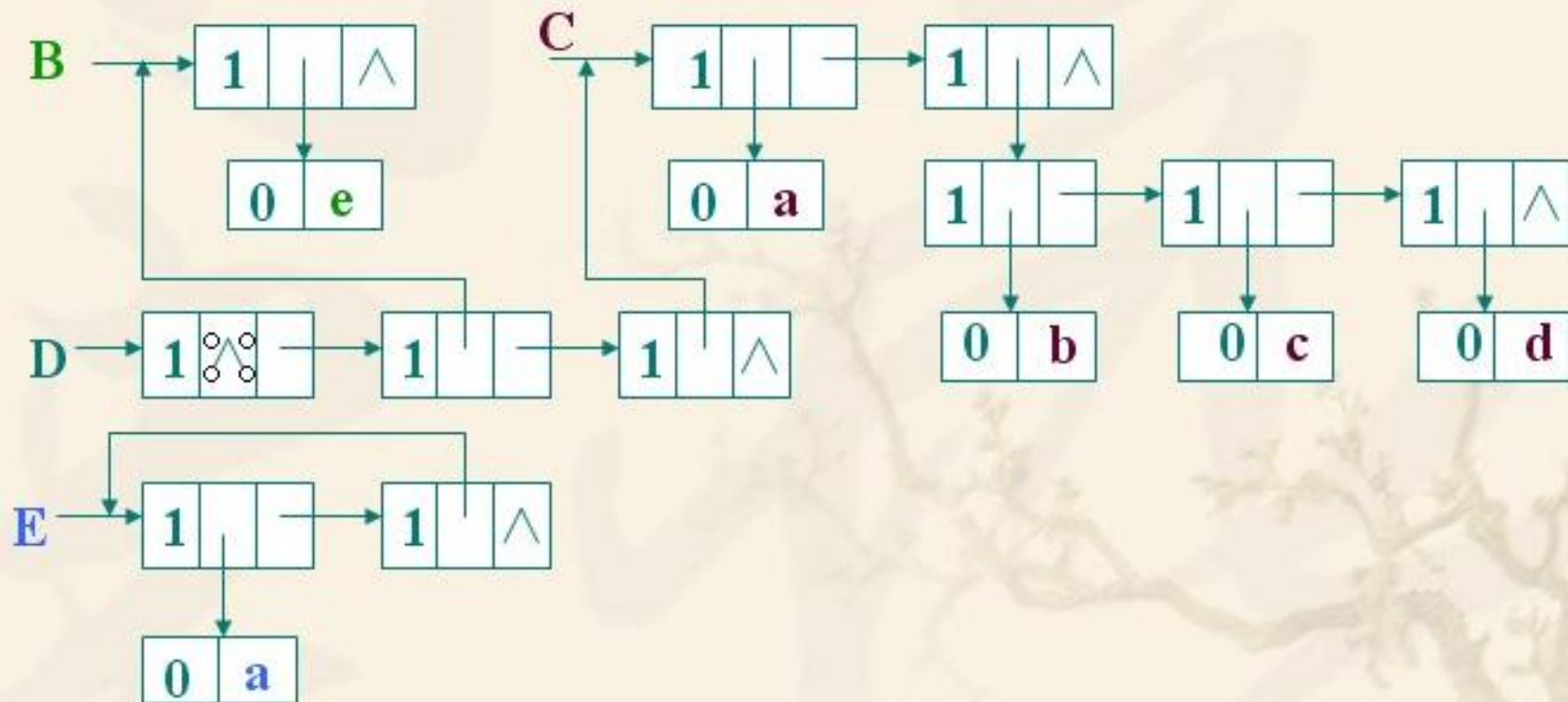
```
/* ptr和atom两成员共用联合体 */
```

```
};
```

```
} GLNode, *GList ;    /* 广义表结点及广义表 */
```

$A = ()$
$$B = (e)$$
$$C = (a, (b, c, d))$$
$$D = (A, B, C)$$
$$\mathbf{E} = (\mathbf{a}, \mathbf{E})$$

A=NIL



广义表的另一种存储结构见教材P110图。
仿照该结构可用于表示 m 元多项式。

该结构某种程度上与下一章《树》的孩子兄弟表示法相似

下午上机：

1、上三角矩阵压缩存储

要求，给定一个矩阵，压缩成一维数组并打印输出，然后由这个一维数组再还原出原未压缩矩阵

2、稀疏矩阵转置及快速转置

要求，给定一个稀疏矩阵，将其用三元组顺序表表示，打印该表，然后进行转置及快速转置，打印转置后的三元组表。