本节主题：

特殊矩阵的压缩存储

矩阵存储与特殊矩阵的压缩存储

- 矩阵的存储方案
  - 直接用二维数组存储
- 特殊矩阵可以“压缩”以降低空间需求
  - （1）对角矩阵
  - （2）对称矩阵
  - （3）上三角矩阵 / 下三角矩阵
- 本节特殊矩阵均为方阵
  - 行数和列数相同

$a_{i,j}$（$i<j$）上三角

$$A = \begin{Bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{Bmatrix}$$

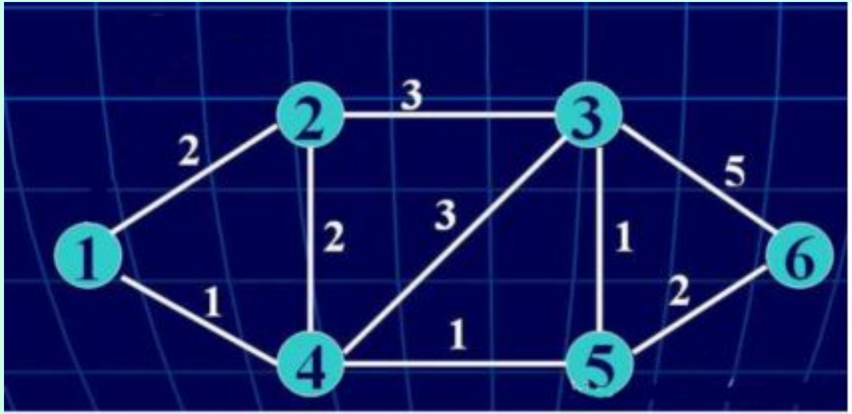
$a_{i,j}$（$i>j$）下三角 $a_{i,i}$（$0\le i\le n$-1）对角线

	0			n-1
0	a[0][0]	a[0][1]	...	a[0][n-1]
	a[1][0]	a[1][1]	...	a[1][n-1]
n-1				

对称矩阵

- 定义
  - 若一个n阶方阵A[n][n]中的元素满足$a_{ij}=a_{ji}$（0≤i，j≤n-1），则称其为**n阶对称矩阵**

$$\begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,n-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,n-1} \\ \cdots & & & \\ a_{n-1,0} & a_{n-1,1} & \cdots & a_{n-1,n-1} \end{bmatrix}$$



0	2	∞	1	∞	∞
2	0	3	2	∞	∞
∞	3	0	3	1	5
1	2	3	0	1	∞
∞	∞	1	1	0	2
∞	∞	5	∞	2	0

对称矩阵的压缩存储

- 策略
  - 存储时可只存储对称矩阵中上三角或下三角中的元素，使得对称的元素共享一个存储空间
- 示例
  - 以行序为主序存储其下三角＋对角线的元素

$$
k = \begin{cases} \dfrac{i(i+1)}{2} + j & i \ge j \\ \dfrac{j(j+1)}{2} + i & i < j \end{cases}
$$

n^2个元素$\longleftrightarrow$ n(n+1)/2个元素

A[0..n-1,0..n-1] $\longleftrightarrow$ B[0..n(n+1)/2-1]

a[i][j] $\longleftrightarrow$ b[k]

$$
\begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,n-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,n-1} \\ \cdots & & & \\ a_{n-1,0} & a_{n-1,1} & \cdots & a_{n-1,n-1} \end{bmatrix}
$$

B

a_{00}	a_{10}	a_{11}	a_{20}	a_{21}	a_{22}	…	a_{ij}	…	$a_{n\text{-}1,0}$	…	$a_{n\text{-}1,n\text{-}1}$
b_0	b_1	b_2	b_3	b_4	b_5	…	b_k	…		…	b_s

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

基本运算(方阵阶数为N)

```
//为N阶对称矩阵初始化存储空间
void Init(int *&b)
{
    b = (int*)malloc(sizeof(int)*(N*(N+1)/2));
}
```

```
//将A[i][j]的值e存储到b中
void Assign(int b[], int e, int i, int j)
{
    if (i>=j)
        b[(i*(i+1))/2+j] = e;
    else
        b[(j*(j+1))/2+i] = e;
    return;
}
```

```
//输出压缩存储在b中的对称矩阵
void Disp(int b[])
{
    int i,j;
    for (i=0; i<N; i++)
    {
        for (j=0; j<N; j++)
            printf("%4d",Value(b,i,j));
        printf("\n");
    }
}
```

```
//返回存储在b[M]中的A[i][j]值
int Value(int b[], int i, int j)
{
    if (i>=j)
        return b[(i*(i+1))/2+j];
    else
        return b[(j*(j+1))/2+i];
}
```

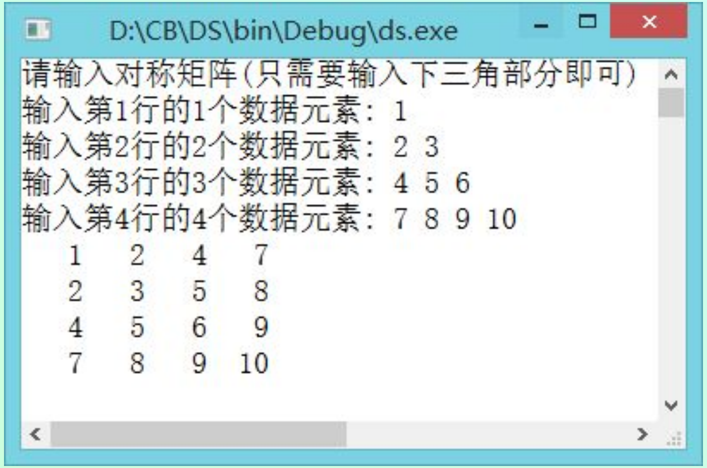
```
//销毁存储空间
void Destroy(int b[])
{
    free(b);
}
```

	a_{00}	a_{10}	a_{11}	a_{20}	a_{21}	a_{22}	…	a_{ij}	…	$a_{n\text{-}1,0}$	…	$a_{n\text{-}1,n\text{-}1}$
B	b_0	b_1	b_2	b_3	b_4	b_5	…	b_k	…		…	b_s

$$k=\begin{cases}\dfrac{i(i+1)}{2}+j & i\ge j\\[2mm] \dfrac{j(j+1)}{2}+i & i<j\end{cases}$$

压缩存储的对称矩阵

```
#define N 4
……
int main() {
    int *b1;
    int i,  j,  v;
    Init(b1);
    printf("请输入对称矩阵(只需要输入下三角部分即可)：\n");
    for(i=0;i<N;i++) {
        printf("输入第%d行的%d个数据元素", i+1, i+1);
        for(j=0; j<=i; j++)  {
            scanf("%d", &v);
            Assign(b1, v, i, j);
        }
    }
    Disp(b1);
    Destroy(b1);
    return 0;
}
```



下三角矩阵的压缩存储

$$\begin{bmatrix} a_{0,0} & & & c \\ a_{1,0} & a_{1,1} & & \\ \cdots & & & \\ a_{n,0} & a_{n,1} & \cdots & a_{n-1,n-1} \end{bmatrix}$$

n^2个元素⟵⟶ n(n+1)/2**+1**个元素

A[0..n-1,0..n-1] ⟵⟶ B[0..**n(n+1)/2**]

a[i][j] ⟵⟶ b[k]

B	a_{00}	a_{10}	a_{11}	a_{20}	a_{21}	a_{22}	…	a_{ij}	…	$a_{n\text{-}1,0}$	…	$\boldsymbol{a_{n\text{-}1,n\text{-}1}}$	c
	b_0	b_1	b_2	b_3	b_4	b_5	…	b_k	…			$b_{s\text{-}1}$	b_s

$$k=\begin{cases} \dfrac{i(i+1)}{2}+j & i \ge j \\ \dfrac{n(n+1)}{2} & i < j \end{cases}$$

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

存放常量c

```
//将A[i][j]的值e存储到b中
void Assign(int b[], int e, int i, int j)
{
    int k;
    k=(i>=j)?((i*(i+1))/2+j):(n*(n+1)/2);
    b[k]=e;
    return;
}
```

```
//返回存储在b[M]中的A[i][j]值
int Value(int b[], int i, int j)
{
    int k;
    k=(i>=j)?((i*(i+1))/2+j):(n*(n+1)/2);
     return b[k];
}
```

上三角矩阵的压缩存储

$$\begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,n-1} \\ & a_{1,1} & \cdots & a_{1,n-1} \\ & \mathbf{c} & & \\ & & & a_{n-1,n-1} \end{bmatrix}$$

n^2个元素 ⟷ $n(n+1)/2$**+1**个元素

A[0..n-1,0..n-1] ⟷ B[0..**n(n+1)/2**]

a[i][j] ⟷ b[k]

A	a_{00}	…	$a_{0,n-1}$	a_{11}	…	$a_{1,n-1}$	a_{22}	…	a_{ij}	…	$\boldsymbol{a_{n-1,n-1}}$	c
B	b_0	…	b_{n-1}	b_n		…			b_k		b_{s-1}	b_s

$$k=\begin{cases} \dfrac{i(2n-i+1)}{2}+j-i & i\le j \\ \dfrac{n(n+1)}{2} & i>j \end{cases}$$

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

存放常量c

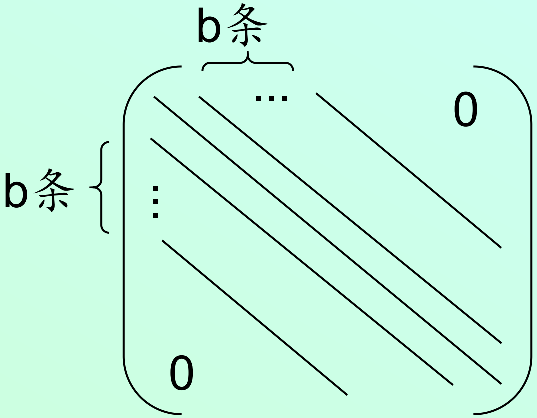
```
//将A[i][j]的值e存储到b中
void Assign(int b[], int e, int i, int j)
{
   …
}
```

```
//返回存储在b[M]中的A[i][j]值
int Value(int b[], int i, int j)
{
    …
}
```

对角矩阵

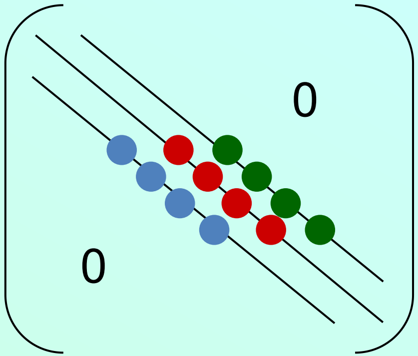
- 定义
  - 若一个n阶方阵A满足，其所有非零元素都集中在以主对角线为中心的带状区域中，则称其为**n阶对角矩阵**
- 术语
  - b——矩阵半带宽：主对角线上下方各有b条次对角线
  - (2b+1)——矩阵的带宽
- 特征
  - 对于半带宽为b（0≤b≤(n-1)/2）的对角矩阵，其|i-j|≤b的元素$a_{i,j}$不为零，其余元素为零。

$$\begin{bmatrix} 1 & 2 & 0 & 0 & 0 & 0 \\ 1 & 2 & 4 & 0 & 0 & 0 \\ 0 & 3 & 3 & 6 & 0 & 0 \\ 0 & 0 & 5 & 4 & 8 & 0 \\ 0 & 0 & 0 & 7 & 5 & 0 \\ 0 & 0 & 0 & 0 & 9 & 6 \end{bmatrix}$$



半带宽为*b*的对角矩阵

三对角矩阵(b=1时)的压缩存储



B	a_{00}	a_{01}	a_{10}	a_{11}	a_{12}	a_{21}	a_{22}	a_{23}	…	a_{ij}	…	$a_{n-1,n-1}$
	b_0	b_1	b_2	b_3	b_4	b_5	b_6	b_7	…	b_k	…	b_{3n-3}

k=2i+j

- 在三对角矩阵中，除了第一行和最后一行各有两个元素，其余各行非0元素a_{ij}均有三个，所以共有3n-2个非0元素
  - $(a_{10}, a_{21}, a_{32}, \ldots)$，有：j=i-1
  - $(a_{00}, a_{11}, a_{22}, \ldots)$，有：j=i
  - $(a_{01}, a_{12}, a_{23}, \ldots)$，有：j=i+1

- 用ij确定k——前i行元素个数：2+3(i-1)=3i-1
  - a_{ij}是本行第1个非零元素，k=3i-1 =2i+j
  - a_{ij}是本行第2个非零元素，k=3i =2i+j
  - a_{ij}是本行第3个非零元素，k=3i+1 =2i+j