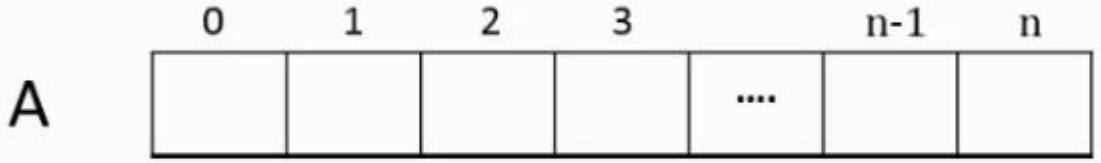
# ساختمان داده ها والگوریتم ها

# فصل دوم

آرایه ها

# تعریف آرایه

به یک مجموعه از مکان های متوالی و به هم پیوسته حافظه که تحت یک نام شناخته می شوند و نوع مشابهی از داده ها را نگهداری می کند آرایه گویند.




دستور تعریف یک آرایه:

```
type  name[size];
```

دسترسی به iامین عنصر آرایه:

```
example [i];
*(example + i);
```

# ذخیره سازی آرایه

برای محاسبه آدرس یک خانه مثل $A[i_1, i_2, i_3, \ldots, i_n]$ در $A[l_1 \ldots u_1, l_2 \ldots u_2, l_3 \ldots u_3, \ldots, l_n \ldots u_n]$ با توجه به وضعیت سطری یا ستونی بودن ماتریس ذخیره شده، میتوان بصورت زیر عمل کرد:

**سطری :**

$$a+\left[(i_1-l_1)(u_2-l_2+1)(u_3-l_3+1)*\ldots*(u_n-l_n+1) + (i_2-l_2)(u_3-l_3+1)*\ldots*(u_n-l_n+1)+\ldots+(i_n-l_n)\right]*b$$

**ستونی :**

$$a+\left[(i_n-l_n)(u_{n-1}-l_{n-1}+1)(u_{n-2}-l_{n-2}+1)*\ldots*(u_1-l_1+1) + (i_{n-1}-l_{n-1})(u_{n-2}-l_{n-2}+1)*\ldots*\right.$$

$$\left.(u_n-l_n+1)+\ldots+(i_1-l_1)\right]*b$$

a : آدرس شروع آرایه

b : فضای نوع عناصر آرایه

# ذخیره سازی آرایه

مثال: آرایه ۳بعدی A[1:15,-5:5,10:25] برای ذخیره اعداد صحیح به طول ۲ بایت به کار گرفته شده است. اگر آرایه به صورت سطری از ۲۰۰۰ به بعد ذخیره شده باشد آدرس A[5,2,15] را محاسبه کنید.

A[5,2,15]= 2000+[ (5-1)(5-(-5)+1)(25-10+1)+(2-(-5))(25-10+1)+(15-10)]*2

= 3642

$$a+[(i_1-l_1)(u_2-l_2+1)(u_3-l_3+1)*\ldots*(u_n-l_n+1) + (i_2-l_2)(u_3-l_3+1)*\ldots*(u_n-l_n+1)+\ldots+(i_n-l_n)]*b$$

# ذخیره سازی آرایه

برای محاسبه اندیس یک خانه مثل $A[i_1, i_2, i_3, ..., i_n]$ در $A[l_1 ... u_1, l_2 ... u_2, ..., l_n ... u_n]$ با توجه به وضعیت سطری یا ستونی بودن ماتریس کافیست در فرمول قبل به جای aوb به ترتیب ۰ و ۱ قرار داد.پس :

**سطری :**

$$[(i_1-l_1)(u_2-l_2+1)(u_3-l_3+1)*...*(u_n-l_n+1) + (i_2-l_2)(u_3-l_3+1)*...*(u_n-l_n+1)+...+(i_n-l_n)]$$

**ستونی :**

$$[(i_n-l_n)(u_{n-1}-l_{n-1}+1)(u_{n-2}-l_{n-2}+1)*...*(u_1-l_1+1) + (i_{n-1}-l_{n-1})(u_{n-2}-l_{n-2}+1)*...*$$

$$(u_n-l_n+1)+...+(i_1-l_1)]$$

# ذخیره سازی آرایه

مثال: آرایه ۳بعدی M[1...a,1...b,1...c] در یک آرایه یک بعدی N[1...a*b*c] به روش ستونی ذخیره شده است. اندیس عنصر M[i,j,k] را در آرایه N محاسبه کنید.

$$M[i,j,k] = (k-1)\,(b-1+1)\,(a-1+1) + (j-1)\,(a-1+1) + (i-1)$$

$$= (k-1)\,ba + (j-1)\,a + (i-1)$$

# ماتریس ها

به هر آرایه دو بعدی m*n یک ماتریس گفته می شود، که تعداد mn خانه در آن وجود دارد.

ماتریس ها:
- ۱- پایین مثلثی
- ۲- بالا مثلثی
- ۳- اسپارس

# ماتریس ها

**ماتریس پایین مثلثی**: ماتریس مربعی که در آن درایه های واقع در بالای قطر اصلی همگی صفر باشد.

$$\begin{bmatrix} a_{11} & 0 & 0 & \dots & 0 \\ a_{21} & a_{22} & 0 & \dots & 0 \\ a_{31} & a_{32} & a_3 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{n3} \end{bmatrix}$$

★در ماتریس پایین مثلثی حداکثر عناصر غیر صفر در سطر اول برابر۱، در سطر دوم۲ و ... و در سطر nام برابر n است بنابراین تعداد کل عناصر غیر صفر:

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

# ماتریس ها

**ماتریس بالا مثلثی**: ماتریس مربعی که در آن درایه های واقع در زیر قطر اصلی همگی صفر باشد.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots. & a_{1n} \\ 0 & a_{22} & a_{23} & \dots. & a_{2n} \\ 0 & 0 & a_{33} & \dots. & a_{3n} \\ \dots. & \dots. & \dots. & \dots. & \dots. \\ \dots. & \dots. & \dots. & \dots. & \dots. \\ 0 & 0 & 0 & \dots. & a_{nn} \end{bmatrix}$$

★در ماتریس بالا مثلثی حداکثر عناصر غیر صفر در سطر اول برابرn، در سطر دوم$n-1$ و ... و در سطر nام برابر یک است بنابراین تعداد کل عناصر غیر صفر:

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

# ماتریس ها

**مثال:** تعداد عناصری که نیاز به ذخیره سازی آن ها در یک ماتریس بالا مثلثی ۱۲*۱۲ نیست را بیابید.

**حل:**

در ماتریس بالا مثلثی n*n حداکثر $\frac{n(n+1)}{2}$ عنصر غیر صفر وجود دارد. پس تعداد عناصری که نیاز به ذخیره سازی آن ها نمی باشد:

$$n*n - \frac{n(n+1)}{2} = 12*12 - \frac{12(12+1)}{2} = 66$$

# ماتریس ها

**ماتریس اسپارس:** به ماتریسی که تعداد عناصر صفر آن زیاد است ماتریس اسپارس گویند. مثلا:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

# ماتریس ها

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

| | Row | Col | Value |
|---|---|---|---|
| a[0] | 6 | 6 | 7 |
| a[1] | 1 | 1 | 1 |
| a[2] | 1 | 5 | 1 |
| a[3] | 2 | 5 | 1 |
| a[4] | 3 | 2 | 1 |
| a[5] | 4 | 1 | 2 |
| a[6] | 5 | 4 | 3 |
| a[7] | 6 | 6 | 1 |

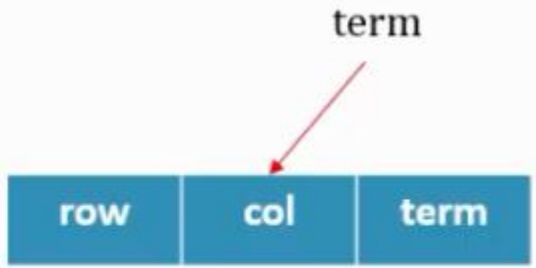
```
Struct spars{
        int row;
        int col;
        int term;
        };

spars  a[maxterm+1];
```

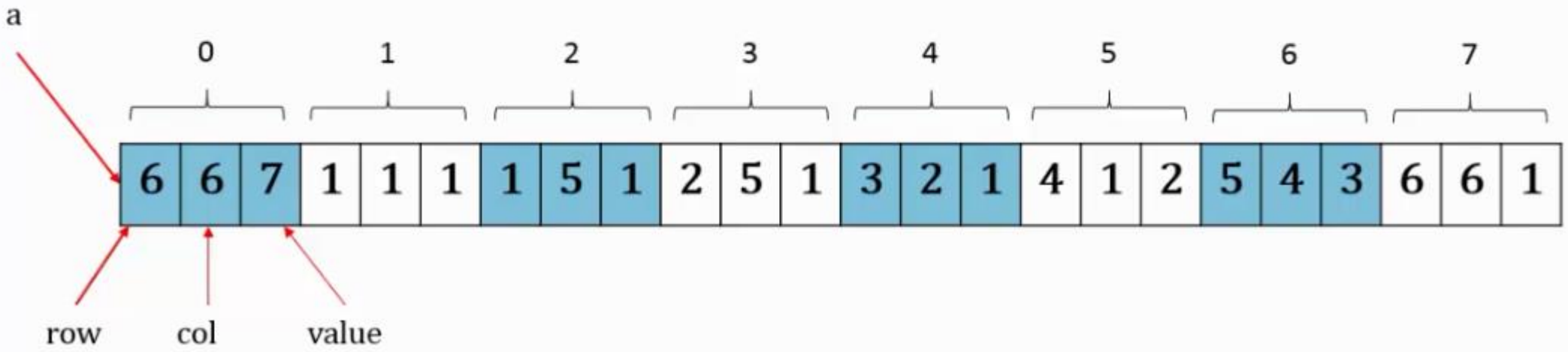
- ★ maxterm: تعداد اعداد غیر صفر ماتریس اسپارس
- ★ a[0].row: تعداد سطرهای ماتریس اسپارس
- ★ a[0].col: تعداد ستون های ماتریس اسپارس
- ★ a[0].term: تعداد عناصر غیر صفر ماتریس اسپارس

# ماتریس ها

```
Int  term=15;
```


```
Struct spars{
        int row;
        int col;
        int term;
    };
spars  term;
```



# ترانهاده ماتریس اسپارس

- عمل ترانهاده کردن ماتریس یعنی عنصر مکان [i,j] به مکان [j,i] منتقل شود.
- محاسبه ترانهاده ماتریس اسپارس با ساختمان داده معرفی شده:

کافیست در هر ردیف جای مقدار ستون col با ستون row را عوض کرده و سپس سه گانه جدید را براساس row و سپس col مرتب کرد.

| | Row | Col | Value |
|---|---|---|---|
| a[0] | 6 | 6 | 7 |
| a[1] | 1 | 1 | 1 |
| a[2] | 1 | 5 | 1 |
| a[3] | 2 | 5 | 1 |
| a[4] | 3 | 2 | 1 |
| a[5] | 4 | 1 | 2 |
| a[6] | 5 | 4 | 3 |
| a[7] | 6 | 6 | 1 |

| | Row | Col | Value |
|---|---|---|---|
| a[0] | 6 | 6 | 7 |
| a[1] | 1 | 1 | 1 |
| a[2] | 5 | 1 | 1 |
| a[3] | 5 | 2 | 1 |
| a[4] | 2 | 3 | 1 |
| a[5] | 1 | 4 | 2 |
| a[6] | 4 | 5 | 3 |
| a[7] | 6 | 6 | 1 |

| | Row | Col | Value |
|---|---|---|---|
| a[0] | 6 | 6 | 7 |
| a[1] | 1 | 1 | 1 |
| a[2] | 1 | 4 | 2 |
| a[3] | 2 | 3 | 1 |
| a[4] | 4 | 5 | 3 |
| a[5] | 5 | 1 | 1 |
| a[6] | 5 | 2 | 1 |
| a[7] | 6 | 6 | 1 |

# ضرب ماتریس ها

**شرط ضرب دو ماتریس:** برای آن که ضرب دو ماتریس A و B امکان پذیر باشد باید بعد وسط آن ها یکسان باشد.

$$A_{m*n} * B_{n*k} = c_{m*k}$$

**خواص ضرب ماتریس ها:**

۱) ضرب ماتریس ها خاصیت جا به جایی ندارد. $BA \neq AB$

۲) ضرب ماتریس ها خاصیت شرکت پذیری دارد. $ABC=(AB)C=A(BC)$

# ضرب ماتریس ها

الگوریتم ضرب دو ماتریس:($A_{m*n} * B_{n*k} = C_{m*k}$)

```
For   i=1   to   m   do
For   j=1   to   k   do
Begin
c[i,j]=0
For   L=1   to   n   do
c[i,j]=c[i,j]+A[i,L]*B[L,j];
end;
```

تعداد ضرب های انجام شده: m*n*k

تعداد جمع های انجام شده: m*(n-1)*k

# ضرب ماتریس ها

**مثال:** تعداد ضرب های حاصل ضرب ۳ ماتریس $C_{5*4}$ و $B_{10*5}$ و $A_{3*10}$ با توجه به حالت های مختلف شرکت پذیری را محاسبه کنید.

$$(\mathrm{AB})\mathrm{C}=(AB)_{3*5} * C_{5*4}= (3*10*5) + (3*5*4) = 210$$

$$\mathrm{A}(\mathrm{BC})=A_{3*10} * (BC)_{10*4}= (3*10*4) + (10*5*4)=320$$

# محاسبه بهینه ترین تعداد ضرب در ضرب چند ماتریس

بهینه ترین حالت، زمانی خواهد بود که کمترین تعداد ضرب را در محاسبه حاصل ضرب ماتریس داشته باشیم.

**قضیه:** در هنگام ضرب ۳ماتریس $A_{m*n}$ و $B_{n*k}$ و $C_{k*L}$ برای آن که تعداد ضرب (AB)C از A(BC) کمتر شود باید:

$$\frac{1}{m}+\frac{1}{k}>\frac{1}{n}+\frac{1}{L}$$

$$(AB)C < A(BC) \text{ ----> } mnk + mkL < mnL + nkL \xrightarrow{\div mnkL} \frac{1}{m}+\frac{1}{k}>\frac{1}{n}+\frac{1}{L}$$

**نتیجه:** می توان گفت که هنگام ضرب چند ماتریس اگر ماتریس هایی را که بعد وسط آن ها بزرگتر و بعد کناری آن ها به نسبت کوچک تر باشد را زودتر ضرب کنیم، تعداد ضرب ها کوچکتر می شود.

# محاسبه بهینه ترین تعداد ضرب در ضرب چند ماتریس

مثال: کمترین تعداد ضرب برای به دست آوردن حاصل ضرب $A_{13*5}*B_{5*89}*C_{89*3}*D_{3*34}$ چقدر است؟

AB=[13*5] * [5*89]

BC=[5*89]*[89*3]

CD=[89*3]*[3*34]

B*C = E

AE=[13*5]*[5*3]

ED=[5*3]*[3*34]

(A(BC))D

(5*89*3) + (13*5*3) + (13*3*34)=2856