فصل دوم آرایه

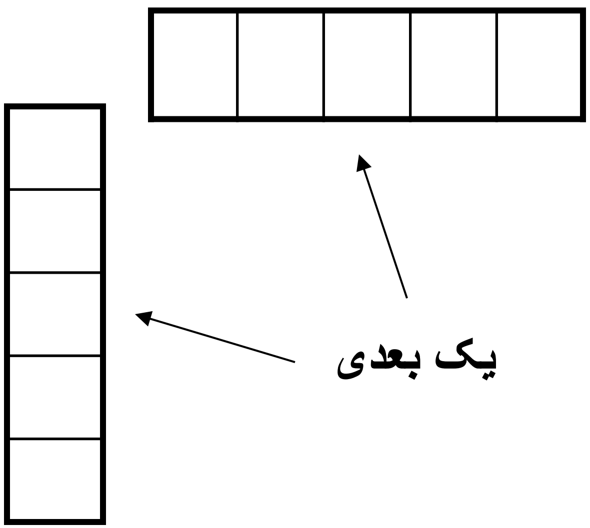
انتخاب عنصر میانه و فلان بهمان، سوال تئوری(استاد برای تمرین قبول نکرد، خودم بزارم حل تمرین)

نجمه منصوری

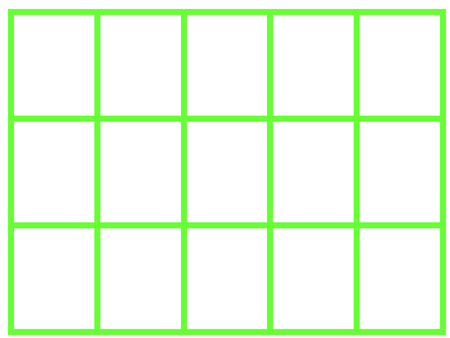
آرایه (Array) :

آرایه یا ماتریس یا جدول یا متغیّر اندیس دار ، مجموعه ای از فضاهای بهم پیوسته یا پی درپی از حافظه است.

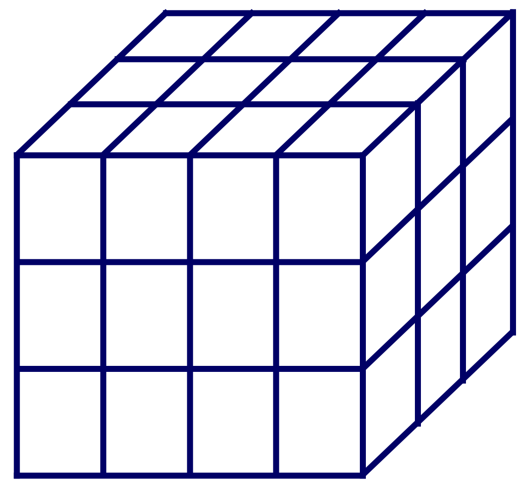
اعضای آرایه از یک نوعند و به یک اندازه حافظه نیاز دارند.



یک بعدی



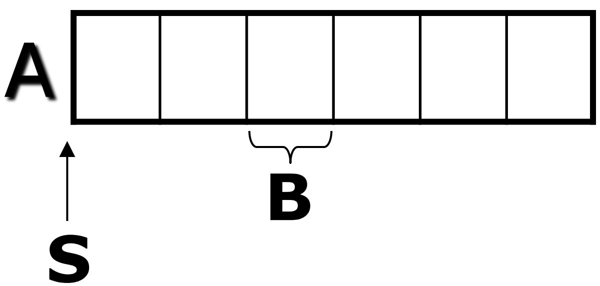
دو بعدی



سه بعدی

آدرس واقعی خانه های آرایه

همه اعضای آرایه درحافظه پشت سرهم ذخیره میشوند. مثلاً آرایه **A** با آدرس شروع **S** که هرخانه آن **B** بایت ازحافظه را نیاز دارد ، در نظر بگیرید.

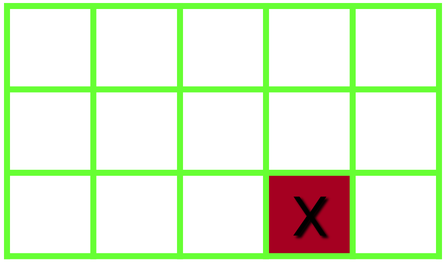


آدرس واقعی خانه اول : **S**

آدرس واقعی خانه دوم : **S+1*B**

آدرس واقعی خانه شماره **n** : **S+(n-1)*B**

درآرایه دو بعدی A_{R*C} آدرس واقعی درایه سطر **row** و ستون **col** :



S + [(row-1) * C + (col-1)]*B

در آرایه مقابل :

Address = S+[(3-1)*5 +(4-1)]*B

محاسبه آدرس خانه $A[i,j,k]$ در آرایه $A[L_1..U_1\ ,L_2\ ..\ U_2\ ,L_3\ ..\ U_3]$

برای محاسبه آدرس خانهٔ $A[i,j,k]$ با توجه به وضعیت (**سطری یا ستونی**) همواره می توانیم بصورت زیر عمل کنیم :

α = **آدرس شروع آرایه** (در صورت عدم بیان آدرس شروع $\alpha = 0$ فرض می‌کنیم)

β = **فضای نوع عناصر آرایه** (در صورت عدم بیان فضای نوع عناصر آرایه $\beta = 1$ فرض می‌کنیم)

(آدرس سطری) $$A[\overline{i,j,k}]=\alpha+\left[(i-L_1)(U_2-L_2+1)(U_3-L_3+1)+(j-L_2)(U_3-L_3+1)+(k-L_3)\right]\times\beta$$

(آدرس ستونی) $$A[\overline{i,j,k}]=\alpha+\left[(k-L_3)(U_2-L_2+1)(U_1-L_1+1)+(j-L_2)(U_1-L_1+1)+(i-L_1)\right]\times\beta$$

دقت کنید:

۱ـ این روش برای حالت **n بعدی** نیز به راحتی قابل تعمیم است.

۲ـ در صورت عدم بیان نوع آدرس (سطری یا ستونی) بطور پیش فرض **آدرس سطری** در نظر گرفته می شود.

مثال۱: آرایه ۳ بعدی $A[1:15,-5:5,10:25]$ برای ذخیره اعداد صحیح به طول 2 بایت به کار گرفته است. اگر آرایه به‌صورت سطری از آدرس 2000 به بعد ذخیره شده باشد آدرس عنصر $A[5,2,15]$ چیست؟

۱) 3868　　۲) 3370　　۳) 3420　　۴) 3642

حل: گزینه ۴ درست است.

$$A:[1..15\ ,-5..5\ ,10..25]\qquad \beta=2\qquad \alpha=2000$$

$$A[\overline{5,2,15}]=\overset{2000}{\cancel{\alpha}}+[(5-1)(5-(-5)+1)(25-10+1)+(2-(-5))(25-10+1)+(15-10)]\times\overset{2}{\cancel{\beta}}=2000+1642=3642$$

مثال۲: آرایه دوبعدی $X[-5..5,3..33]$ در آدرس 400 به بعد حافظه قرار دارد و هر خانه آرایه احتیاج به 4 بایت دارد. آدرس عنصر $X[4\ ,10]$ به روش ستونی کدام است؟

۱) 1444 ۲) 1544 ۳) 844 ۴) 744

حل: گزینه ۴ درست است.

$X[-5\ ..\ 5\ ,\ 3\ ..\ 33] \qquad \alpha = 400 \qquad \beta = 4$

$$X[\overline{4\ ,10}] = \overset{400}{\cancel{\alpha}} + \left[(10-3)\left(5-(-5)+1\right)+\left(4-(-5)\right)\right] \times \overset{4}{\cancel{\beta}} = 400 + 344 = 744$$

مثال۳: آرایه ۳بعدی $A[1..m,1..n,1..p]$ در یک آرایه یک‌بعدی $B[1..m\times n\times p]$ به روش سطر به سطر ذخیره شده است. آدرس عنصر $A[i,j,k]$ در B کدام است؟

۱) $(i-1)np+(j-1)m+(k-1)$

۲) $(i-1)np+(j-1)p+(k-1)$

۳) $mnp+np\times 1$

۴) $inp+jp+k$

حل: گزینه ۲ درست است.

$A[1..m,1..n,1..p]$

$$A[\overline{i,j,k}] = \overset{0}{\cancel{\alpha}} + [(i-1)(n-1+1)(p-1+1)+(j-1)(p-1+1)+(k-1)]\times \overset{1}{\cancel{\beta}}$$

$$=(i-1)np+(j-1)p+(k-1)\ .$$

محاسبه شماره خانه $A[i,j,k]$ در آرایه $A[L_1 .. U_1, L_2 .. U_2, L_3 .. U_3]$

برای این‌که محاسبه کنیم $A[i,j,k]$ چندمین خانه سطری یا ستونی ماتریس $A[L_1 .. U_1, L_2 .. U_2, L_3 .. U_3]$ است کافیست در شرایطی که $\alpha = 0$ و $\beta = 1$ در نظر گرفته‌ایم به آدرس $A[i,j,k]$ یک واحد اضافه کنیم.

$$A[\overline{i,j,k}] = \left(\overset{0}{\cancel{\alpha}} + \left[(i-L_1)(U_2-L_2+1)(U_3-L_3+1) + (j-L_2)(U_3-L_3+1) + (k-L_3)\right] \times \overset{1}{\cancel{\beta}}\right) + 1$$

$$= \left[(i-L_1)(U_2-L_2+1)(U_3-L_3+1) + (j-L_2)(U_3-L_3+1) + (k-L_3)\right] + 1$$

$$A[\overline{i,j,k}] = \left(\overset{0}{\cancel{\alpha}} + \left[(k-L_3)(U_2-L_2+1)(U_1-L_1+1) + (j-L_2)(U_1-L_1+1) + (i-L_1)\right] \times \overset{1}{\cancel{\beta}}\right) + 1$$

$$= \left[(k-L_3)(U_2-L_2+1)(U_1-L_1+1) + (j-L_2)(U_1-L_1+1) + (i-L_1)\right] + 1$$

مثال: در آرایه $A['a'..'e', 1..4]$، درایه $A['b', 3]$ چندمین خانه آرایه است؟

چون وضعیت سطری یا ستونی مشخص نشده است پیش فرض به صورت سطری عمل می‌کنیم در ضمن فاصله $'a'..'e'$ را به شکل $1..5$ در نظر می‌گیریم بنابراین وضعیت $A[2, 3]$ در $A[1..5, 1..4]$ مدنظر است.

$$A[2, 3] = \left(0 + \left[(2-1)(4-1+1) + (3-1)\right] \times 1\right) + 1 = 7$$

'a', 1	'a', 2	'a', 3	'a', 4
'b', 1	'b', 2	'b', 3	'b', 4
'c', 1	'c', 2	'c', 3	'c', 4
'd', 1	'd', 2	'd', 3	'd', 4
'e', 1	'e', 2	'e', 3	'e', 4

$A[2, 3]$ خانه 7ام به صورت سطری است

به هر آرایه دو بعدی $n * m$ یک ماتریس یا جدول با m سطر و n ستون گفته می‌شود. که تعداد mn خانه در آن وجود دارد.

ماتریس مربع

به هر ماتریس $n * n$ با n^2 خانه ماتریس مربع گفته می‌شود.

در هر ماتریس مربع $n * n$ مانند A روابط زیر برای اندیس خانه A [i , j] وجود دارد:

A [i , j] بالای قطر فرعی است $i + j < n + 1$

A [i , j] روی قطر فرعی است $i + j = n + 1$

A [i , j] پایین قطر فرعی است $i + j > n + 1$

A [i , j] بالای قطر اصلی است $i < j$

A [i , j] روی قطر اصلی است $i = j$

A [i , j] پایین قطر اصلی است $i > j$

قطر اصلی $(i = j)$ ↘ قطر فرعی $(i + j = n + 1)$ ↙

$A(1,1)$	$A(1,2)$	$A(1,3)$
$A(2,1)$	$A(2,2)$	$A(2,3)$
$A(3,1)$	$A(3,2)$	$A(3,3)$

(3×3)

ماتریس بالا مثلث و پایین مثلث

در هر ماتریس مربع n * n

الف) اگر عناصر زیر قطر اصلی 0 باشد ماتریس بالا مثلثی است.

ب) اگر عناصر بالای قطر اصلی 0 باشد ماتریس پایین مثلثی است.

$$\begin{bmatrix} X & 0 & 0 & \cdots & 0 \\ X & X & 0 & \cdots & 0 \\ X & X & X & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ X & X & X & \cdots & X \end{bmatrix}$$

(الف) پایین

$$\begin{bmatrix} X & X & X & \cdots & X \\ 0 & X & X & \cdots & X \\ 0 & 0 & X & \cdots & X \\ \vdots & \vdots & \cdots & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & X \end{bmatrix}$$

(ب) بالا مثلثی

ماتریس‌های بالا مثلثی و پایین مثلثی

در هر ماتریس **بالا مثلث یا پایین مثلث**:

$$
\begin{array}{r}
n \\
+\ n-1 \\
+\ n-2 \\
+\ \vdots \\
1 \\
\hline
\dfrac{n(n+1)}{2}
\end{array}
\qquad
\begin{array}{|c|c|c|c|c|}
\hline
\times & \times & \cdots & \times & \times \\
\hline
0 & \times & \cdots & \times & \times \\
\hline
0 & 0 & \times & \times & \times \\
\hline
0 & 0 & 0 & \times & \times \\
\hline
0 & 0 & 0 & 0 & \times \\
\hline
\end{array}_{\,n\times n}
$$

$\checkmark$

(تعداد خانه‌های مخالف صفر)

تعداد کل خانه‌ها $= n^2$

تعداد خانه‌های مخالف صفر $= \dfrac{n(n+1)}{2}$

تعداد خانه‌های صفر $= \dfrac{n(n-1)}{2}$

ماتریس اسپارس (تُنک ، پراکنده ، خلوت ، sparse)

بنابر تعریف به هر ماتریس $m \times n$ که تعداد خانه‌های صفر و بی ارزش (nonvalue) در آن بیشتر از تعداد خانه‌های مخالف صفر باشد .**ماتریس اسپارس** می‌گویند.

نکته: حاصل ضرب، جمع و تفریق ماتریس‌های sparse ممکن است sparse نباشد.

جمع : $\begin{bmatrix}1 & 0\\0 & 0\end{bmatrix}+\begin{bmatrix}0 & 1\\0 & 0\end{bmatrix}=\begin{bmatrix}1 & 1\\0 & 0\end{bmatrix}$ **اسپارس نیست.**

تفریق : $\begin{bmatrix}1 & 0\\0 & 0\end{bmatrix}-\begin{bmatrix}0 & 1\\0 & 0\end{bmatrix}=\begin{bmatrix}1 & -1\\0 & 0\end{bmatrix}$ **اسپارس نیست.**

ضرب : $\begin{bmatrix}1 & 0 & 0\\1 & 0 & 0\\0 & 0 & 0\end{bmatrix}\times\begin{bmatrix}1 & 1 & 1\\0 & 0 & 0\\0 & 0 & 0\end{bmatrix}=\begin{bmatrix}1 & 1 & 1\\1 & 1 & 1\\0 & 0 & 0\end{bmatrix}$ **اسپارس نیست.**

روش‌های نگهداری ماتریس‌های اسپارس

۱ – «آرایه دوبعدی و ذخیره مختصات خانه های مخالف صفر(روش عمومی)»

۲- «آرایه یک بعدی برای نگهداری مقادیر مخالف صفر به کمک فرمول(رابطه)» در ماتریس‌های بالا مثلث، پایین مثلث، سه قطری و ...

۱ـ «آرایه دوبعدی و ذخیره مختصات خانه های مخالف صفر(روش عمومی)»

در صورتی‌که ماتریس اسپارس $m \times n$ با r خانه مخالف صفروجود داشته باشد، برای ذخیرهٔ آن از یک آرایه دو بعدی با $r+1$سطر و 3 ستون به شرح زیر استفاده می‌شود:

sparse[1..r+1,1..3]

الف) در سطر اول به ترتیب :تعداد سطرها (m) ، تعداد ستونها (n) و تعداد خانه‌های مخالف صفر (r) ماتریس اسپارس قرار داده می‌شود.

ب) از سطر دوم به بعد، هر سطر شامل سه مؤلفه است که همان مختصات و مقدار خانه‌های مخالف صفر است: $(i , j , A[i , j] \neq 0)$

0	13	0	0	0	14
16	0	0	0	0	0
0	0	0	0	0	15

3×6

ماتریس اسپارس

تعداد سطرها — تعداد ستون‌ها — تعداد مقادیر مخالف 0

3	6	4
1	2	13
1	6	14
2	1	16
3	6	15

5×3

مشخصات کلی ←

$\leftarrow (i , j , A[i , j] \neq 0)$

نمایش ماتریس اسپارس

چون خانه‌های مخالف 0، 4 تا است، بنابراین ماتریس اسپارس شامل 5 سطر و 3 ستون خواهد بود.

مقرون به صرفه بودن

روش مطرح شده در بالا در شرایطی مقرون به صرفه است که تعداد خانه های ماتریس نگهدارندهٔ $\mathbf{sparse[1..r+1,1..3]}$ از تعداد خانه‌های ماتریس اسپارس $\mathbf{m \times n}$ به مراتب کمتر باشد :

$$3(r+1) < mn \xrightarrow{3(r+1) \approx 3r} \boxed{r < \frac{mn}{3}}$$

به عبارت بهتر تعداد خانه های مخالف صفر $\mathbf{r}$ کمتر از $\frac{1}{3}$ خانه‌های ماتریس اسپارس باشد.

ماتریس L قطری

هر ماتریس $n * n$ که در آن L قطر آن که قطر اصلی نیز شامل آن است مخالف صفر باشد ماتریس L قطری می‌گویند.

- ماتریس L قطری فقط برای Lهای فرد تعریف می‌شود.

ماتریس	تعداد خانه مخالف صفر	نمایش
ماتریس 1 قطری = ماتریس قطری	n	$\begin{bmatrix} \times & 0 & 0 \\ 0 & \times & 0 \\ 0 & 0 & \times \end{bmatrix}_{n \times n}$
ماتریس 3 قطری	$3n-2$	$\begin{bmatrix} \times & \times & 0 \\ \times & \times & \times \\ 0 & \times & \times \end{bmatrix}_{n \times n}$
. . .	. . .	. . .
ماتریس L قطری	$nL-\frac{L^2-1}{4}=nL$	مخالف صفر

مثال: می‌توان هر ماتریس تنک (Sparse) را به صورت یک آرایه دو بعدی نمایش داد. برای این کار سطر، ستون و مقدار درایه‌های غیرصفر ماتریس را در آرایه ذخیره می‌کنند. یک ماتریس L قطری با n سطر و n ستون داده شده است. برای مقرون به صرفه بودن این نمایش تنک، بزرگ‌ترین L ممکن برابر است با ....(علوم‌کامپیوتر- کارشناسی ارشد ـ ۷۹)

۱) $L<\left\lfloor \frac{n}{3} \right\rfloor$ ۲) $L<\left\lfloor \frac{n}{2} \right\rfloor$ ۳) $L<\left\lfloor \sqrt{n} \right\rfloor$ ۴) $L<\left\lfloor \frac{n}{4} \right\rfloor$

حل: گزینه ۱ درست است.

تعداد خانه های مخالف صفر در یک ماتریس $\mathbf{n\times n}$ بصورت L قطری برابر با $r = nL-\frac{L^2-1}{4} \simeq \mathbf{nL}$ است. بنابراین باید:

$$r<\frac{mn}{3}\xrightarrow[r=nl]{m=n} nL<\frac{n\times n}{3}\rightarrow L<\frac{n}{3}$$

ترانهاده کردن (Transpose)ماتریس اسپارس

برای ترانهاده کردن ماتریس sparse به این شکل عمل می‌کنیم که ابتدا در سطر اول، جای سطر و ستون را عوض کرده و به همراه تعداد خانه‌های مخالف صفر در ماتریس ترانهاده قرار می‌دهیم. سپس از سطر دوم به بعد روی ستون دوم به ترتیب از بالا به پایین کوچک‌ترین عنصر را پیدا کرده، جای سطر و ستون آن‌ها را عوض کرده و به همراه مقدارشان در ماتریس ترانهاده قرار می‌دهیم.

3	6	4
1	2	13
1	6	14
2	1	16
3	6	15

$\xrightarrow{\text{ترانهاده}}$

6	3	4
1	2	16
2	1	13
6	1	14
6	3	15

تحلیل الگوریتم ترانهاده

برای ترانهاده کردن ماتریس $A_{rows \times columns}$ در ماتریس $B_{columns \times rows}$ می‌توان از الگوریتم زیر در زمان $O(columns \times rows)$ استفاده کرد:

```
for i := 1   to   columns     do
for j := 1   to   rows        do
B[j][i] := A[i][j];
```

مثال: ماتریس خلوت ماتریسی است دو بعدی مانند $A[1..m, 1..n]$ که اکثر عناصر آن صفر می‌باشد. برای صرفه‌جویی در حافظه فقط عناصر غیرصفر را به همراه شماره سطر و شماره ستون و مقدار عنصر در آرایه‌ای با تعریف زیر ذخیره می‌نماییم. (به ترتیب سطری)

type sparse matrix = arry[0..max terms, 1..3] of integer;

کدام‌یک از گزینه‌های زیر صحیح می‌باشد؟ (t تعداد عناصر غیرصفر می‌باشد.)

۱) سریع‌ترین زمان برای به دست آوردن ترانهاده ماتریس خلوت به ترتیب سطری $O(n+t)$ است.

۲) حاصل ضرب دو ماتریس خلوت الزاماً یک ماتریس خلوت است.

۳) سریع‌ترین زمان برای به دست آوردن ترانهاده ماتریس خلوت به ترتیب سطری $O(m+t)$ است.

۴) سریع‌ترین زمان برای به دست آوردن ترانهاده ماتریس خلوت به ترتیب سطری $\theta(nt)$ است.

حل: گزینه ۱ درست است.

$$O(\text{columns} + \text{elements}) \xrightarrow{\text{columns}=n\,,\ \text{elements}=t} O(n+t)$$

?

۲- «آرایه یک بعدی برای نگهداری مقادیر مخالف صفر به کمک فرمول(رابطه)» در ماتریس‌های بالا مثلث، پایین مثلث ، سه قطری و ...

بعضی ماتریس‌ها مانند: **بالا مثلث ، پایین مثلث و ... ، اسپارس نیستند،** اما زمانی که تعداد عناصر در آنها زیاد می شود، تعداد صفرهای ماتریس قابل توجه خواهد شد در این وضعیت بهتر است مقادیر مخالف صفر ماتریس به صورت زیر ذخیره شود:

الف- **یک آرایه یک بعدی** برای نگهداری **مقادیر مخالف صفر**ماتریس به صورت سطری یا ستونی ، که **تعداد عناصر آرایه یک بعدی به تعداد مقادیر مخالف صفر ماتریس می باشد.**

ب - **یک رابطه یا فرمول** که **محل مقادیر مخالف صفر** ماتریس را در آرایه یک بعدی مشخص کند.

مثال: ماتریس **پایین مثلث** $A[1..n,1..n]$ را در نظربگیریدکه عناصر مخالف صفر آن را بصورت **سطری** در یک آرایه **یک بعدی** نگهداری کرده‌ایم در این وضعیت:

۱ـ در ماتریس $S=\frac{n(n+1)}{2}$ خانهٔ مخالف صفر وجود دارد،برای همین به یک **آرایه یک بعدی** ($B[1..S]$) نیاز داریم.

۲ـ هرخانه $A[i\,,\,j]\neq 0$ در ماتریس پایین مثلث با ، **رابطه**(**فرمول سطری**) زیر محل خود را در آرایه یک بعدی (**$B[k]$**)مشخص می‌کند.

$$\boxed{k=\frac{i(i-1)}{2}+j}$$

$$A[i\,,\,j]\neq 0 \longrightarrow B[k]$$

×	0	0	0
×	×	0	0
×	...	×	0
×	×	×	×

1	2	.	.	.	.	S

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

$A[i\,,j]$

10	0	0
20	30	0
40	50	60

$\longrightarrow$

$B[k]$

1	2	3	4	5	6
10	20	30	40	50	60

به طور مثال موقعیت خانهٔ $A[3\,,2]=50$ در آرایه یک بعدی $B\,[5]$ است ، رابطه بین $(i=3, j=2)$ در ماتریس و $(k=5)$ در آرایه یک بعدی توسط فرمول زیر تعیین می شود:

$$k=\frac{i(i-1)}{2}+j$$

که در این‌جا داریم:

$$\underbrace{A[3\,,2]}_{A[i\,,j]} \xrightarrow[j=2]{i=3} \underbrace{k=\frac{3(3-1)}{2}+2=5}_{k=\frac{i(i-1)}{2}+j} \longrightarrow \underbrace{B[5]}_{B[k]}$$

ماتریس	تعداد خانه مورد نیاز برای ذخیره مقادیر مخالف صفر در آرایه یک بعدی	فرمول سطری	فرمول ستونی
پایین مثلث	$\frac{n(n+1)}{2}$	$k=\frac{i(i-1)}{2}+j$	$k=(j-1)\left(n-\frac{j}{2}\right)+i$
بالا مثلث	$\frac{n(n+1)}{2}$	$k=(i-1)\left(n-\frac{i}{2}\right)+j$	$k=\frac{j(j-1)}{2}+i$

فرض کنید ماتریس ۴*۴ به صورت زیر باشد...

موقعیت خانه های A[i , j] در آرایه ی یک بعدی ستونی B[k] را با توجه به فرمول مربوطه توسط فرمول زیر تعیین میشود.

A[i , j]

25	0	0	0
52	21	0	0
29	24	35	0
99	61	17	58

ماتریس پایین مثلث

فرمول ستونی ماتریس پایین مثلث

$$K = (j-1)\left(n - \frac{j}{2}\right) + i$$

B[K]

25	1
52	2
29	3
99	4
21	5
24	6
61	7
35	8
17	9
58	10

$$K = (3-1)\left(4 - \frac{3}{2}\right) + 4 = 9$$

B[9]

(i=4 , j=3)

n=4

به عنوان مثال :
موقعیت
A[4,3]=17
در آرایه ی یک بعدی ستونی
B[10]
با فرمول فوق در مکان
(K=9)
قرار میگیرد

فرض کنید ماتریس ۴*۴ به صورت زیر باشد...
موقعیت خانه های A[i, , j] در آرایه ی یک بعدی ستونی B[k] را با توجه به فرمول مربوطه ، توسط فرمول زیر تعیین میشود...

A[i , j]

41	98	87	91
0	25	85	56
0	0	31	30
0	0	0	11

ماتریس بالا مثلث

فرمول ستونی ماتریس بالا مثلث

$$K=\frac{j(j-1)}{2}+i$$

B[K]

B[K]	k
41	1
98	2
25	3
87	4
85	5
31	6
91	7
56	8
30	9
11	10

$$K=\frac{4(4-1)}{2}+2=8$$

B[8]

به طور مثال :
موقعیت
A[2,4]=56
در آرایه ی یک بعدی ستونی
B[10]
با فرمول فوق در مکان
(K=8)
قرار میگیرد

(i=2 , j=4)

n=4

فرض کنید ماتریس ۴*۴ به صورت زیر باشد...
موقعیت خانه های A[i, , j] در آرایه ی یک بعدی سطری B[k] را با توجه به فرمول مربوطه ، توسط فرمول زیر تعیین میشود...

A[i , j]

25	98	87	91
0	16	85	19
0	0	70	30
0	0	0	11

ماتریس بالا مثلث

فرمول سطری ماتریس بالا مثلث

$$K = (i-1)\left(n - \frac{i}{2}\right) + j$$

$$K = (1-1)\left(4 - \frac{1}{2}\right) + 3 = 3$$

B[3]

	1	2	3	4	5	6	7	8	9	10
B[K]	25	98	87	91	16	85	19	70	30	11

(i=1 , j=3)

n=4

به عنوان مثال :
موقعیت
A[1,3]=87
در آرایه ی یک بعدی ستونی
B[10]
با فرمول فوق در مکان
(K=3)
قرار میگیرد

نتیجه‌گیری:

بین دو روش بیان شده یعنی:

۱ـ روش عمومی نمایش ماتریس اسپارس (نگهداری مختصات مقادیر مخالف صفر با استفاده از آرایه 2 بعدی)

۲ـ نگهداری مقادیر مخالف صفر در آرایه یک بعدی که تعداد عناصر آن به تعداد خانه‌های مخالف 0 است.

روش دوم از نظر حافظه مقرون به صرفه‌تر است البته به شرط آن‌که بتوان رابطه یا فرمولی بین اندیس‌های آرایه B [k] و اندیس‌های مقادیر مخالف صفر A [i , j] پیدا کرد.

ضرب ماتریس‌ها

بر روی هر ماتریس عملیات مختلفی می‌تواند انجام گیرد. مانند: جمع، تفریق، ضرب، ... که یکی از مهمترین آن‌ها عملیات ضرب ماتریس‌ها است.

شرایط ضرب دو ماتریس

برای آن‌که ضرب دو ماتریس A و B امکان‌پذیر باشد باید بعد وسط آن‌ها یکسان باشد به عبارت بهتر برای امکان‌پذیر بودن حاصلضرب B * A باید ستون ماتریس A با سطر ماتریس B یکسان باشد. در نتیجه:

$$A_{m \times n} \times B_{n \times k} \rightarrow C_{m \times k}$$

خواص ضرب ماتریس‌ها

الف) ضرب ماتریس‌ها خاصیت جابجایی ندارد. $AB \neq BA$

ب) ضرب ماتریس‌ها خاصیت شرکت‌پذیری دارد.

حالت‌های شرکت‌پذیری ۴ ماتریس

A، B، C و D

۱) $(AB)(CD)$

۲) $((A(BC))D)$

۳) $(A((BC)D))$

۴) $(((AB)C)D)$

۵) $(A(B(CD)))$

حالت‌های شرکت‌پذیری ۳ ماتریس

A، B و C

۱) $A(BC)$

۲) $(AB)C$

محاسبه تعداد حالات شرکت‌پذیری ضرب ماتریس‌ها

تعداد حالات شرکت‌پذیری ضرب $n+1$ **ماتریس** برابر عدد **کاتالان** بوده و از رابطه زیر به‌دست می‌آید:

$$\frac{\binom{2n}{n}}{n+1}$$

مثال: تعداد حالات شرکت‌پذیری ضرب ۴ ماتریس کدام است؟

$$n+1=4 \Rightarrow n=3 \longrightarrow \text{تعداد حالات شرکت‌پذیری ضرب} = \frac{\binom{6}{3}}{3+1} = 5$$

هرگاه یک ماتریس بالا مثلث را در ماتریس بالا مثلث دیگری ضرب کنیم، حاصل ماتریس بالا مثلث است.

هرگاه یک ماتریس پایین مثلث را در ماتریس پایین مثلث دیگری ضرب کنیم، حاصل ماتریس پایین مثلث است.

هرگاه یک ماتریس قطری را در ماتریس قطری دیگری ضرب کنیم، حاصل ماتریس قطری است.

به عبارت دیگر ضرب یک ماتریس در ماتریس دیگر با همان مشخصه ماتریس اول خاصیت ماتریس را عوض نمی‌کند.

الگوریتم ضرب $(C_{m \times k} = A_{m \times n} \times B_{n \times k})$

```
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 ضرب‌های انجام شده $= m \times n \times k$

تعداد جمع‌های انجام شده $= m \times n \times k$

مثال ۱: تعداد ضرب‌های حاصلضرب 3 ماتریس $A_{3\times10}$, $B_{10\times5}$, $C_{5\times4}$ با توجه به حالت‌های مختلف شرکت‌پذیری کدام است؟

$$A_{3\times10}\,(BC)_{10\times4} = (3\times10\times4)+(10\times5\times4)=120+200=320$$

$$(AB)_{3\times5}\,C_{5\times4} = (3\times10\times5)+(3\times5\times4)=150+60=210$$

دیده می‌شود که حالت‌های مختلف شرکت‌پذیری، تعداد ضرب‌های متفاوتی بوجود می‌آورد.

محاسبه بهینه‌ترین تعداد ضرب در ضرب چند ماتریس

در ضرب چند ماتریس بهینه‌ترین حالت در مورد تعداد ضرب‌ها داشتن کمترین تعداد ضرب است تا عمل ضرب ماتریس‌ها سریع‌تر انجام شود.

قضیه : در هنگام ضرب 3 ماتریس $A_{m\times n}$, $B_{n\times k}$, $C_{k\times L}$ برای آن‌که:

$$\boxed{\frac{1}{m}+\frac{1}{k}>\frac{1}{n}+\frac{1}{L}} \longleftrightarrow \text{تعداد ضرب } A(BC) < \text{تعداد ضرب } (AB)C$$

نتیجه: طبق یک قاعده سرانگشتی می‌توان گفت که هنگام ضرب چند ماتریس، اگر ماتریس‌هایی را زودتر ضرب کنیم که بُعد وسط آن‌ها بزرگتر و بعد کناری آن‌ها به نسبت کوچک‌تر است، بدین ترتیب تعداد ضرب‌ها کوچک‌تر می‌شود.

تحلیل مثال ۱:

در هنگام ضرب ماتریسهای $A_{3\times 10}\times B_{10\times 5}\times C_{5\times 4}$ از آنجایی‌که $\frac{1}{3}+\frac{1}{5}>\frac{1}{10}+\frac{1}{4}$ در نتیجه تعداد ضرب‌های $(AB)C$ کمتر از تعداد ضرب‌های $A(BC)$ است. در این‌جا بعد وسط AB بزرگ‌تر بود.

مثال۲: اگر A یک ماتریس 13×5، B ماتریس 5×89، C ماتریس 89×3 و D یک ماتریس 3×34 باشد کمترین تعداد ضرب برای به‌دست آوردن حاصلضرب $A \times B \times C \times D$ چقدر است؟

۱) 4055 ۲) 2856 ۳) 2820 ۴) 1742

حل: گزینه ۲ درست است.

تعداد ضرب‌ها	ترتیب(شرکت پذیری) ماتریس‌ها برای داشتن کمترین ضرب
$5 \times 89 \times 3 = 1335$	$B_{5 \times 89} \times C_{89 \times 3} = (B \times C)_{5 \times 3}$
$13 \times 5 \times 3 = 195$	$A_{13 \times 5} \times (B \times C)_{5 \times 3} = (A \times (B \times C))_{13 \times 3}$
$13 \times 3 \times 34 = 1326$	$(A \times (B \times C))_{13 \times 3} \times D_{3 \times 34} = ((A(BC))D)_{13 \times 34}$
جمع ضرب‌ها $= 2856$	

مثال۳: می‌خواهیم برای ماتریس‌های $M1_{(10 \times 20)}$ و $M2_{(20 \times 50)}$ و $M3_{(50 \times 1)}$ و $M4_{(1 \times 100)}$ ترکیب بهینه پرانتزبندی پیدا نماییم تا تعداد ضرب‌های کل جهت محاسبه عبارت ذیل حداقل گردد:

$$M = M_1 \times M_2 \times M_3 \times M_4$$

این ترکیب بهینه عبارت است از؟

۱) $\left(M_1 \times \left(M_2 \times M_3\right)\right) \times M_4$

۲) $\left(\left(M_1 \times M_2\right) \times M_3\right) \times M_4$

۳) $M_1 \times \left(\left(M_2 \times M_3\right) \times M_4\right)$

۴) هیچکدام

حل: گزینه ۱ درست است.

مثال۴: قطعه برنامه زیربرای ضرب دو ماتریس مربع A و B با مرتبه n موردنظر است: (کارشناسی ارشد ـ علوم‌کامپیوتر ۷۹)

برای تکمیل این قطعه برنامه، دستور حذف شده به صورت .... است.

```
for  i:=1  to  n  do
     for  j:=1  to  n  do  begin
         c[i , j]:=0;
         for k:=1  to  n  do
```

دستورات حذف شده

```
end:
```

۱) C[i , j]:= A[i , k]*B[k , j] + C[i , j];

۲) C[i , j]:=A[i , j]*B[k , j];

۳) C[i , j]:=A[i , k]*B[k , j];

۴) C[i , j]:= A[i , j]*B[k , j] + C[i , j];

حل: گزینه ۱ درست است.

جستجو در آرایه‌ها

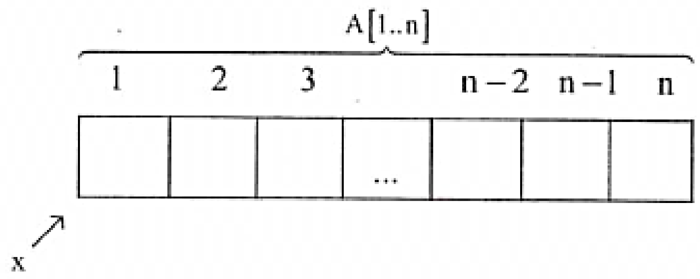
به طور کلی برای جستجوی یک عنصر دلخواه در آرایه ها ۲ روش اساسی وجود دارد:

۱ـ روش جستجوی خطی (ترتیبی)

۲ـ روش جستجوی دودویی (باینری)

۱ـ روش جستجوی خطی (ترتیبی)

در این روش برای جستجوی دادهٔ x در آرایه n تایی A [1..n] ، جستجو را از یکی از دو طرف آرایه (پیش فرض از ابتدا) آغاز می‌کنیم و داده مورد نظر را پشت سر هم با عناصر آرایه مقایسه می‌کنیم، تا این‌که یا داده مورد نظر پیدا شود یا به طرف دیگر آرایه (انتهای آرایه) برسیم.



دقت کنید: در روش جستجوی خطی چون هیچ استراتژی خاصی جز جستجوی پشت سر هم از ابتدا تا انتها وجود ندارد، در نتیجه sort بودن یا نبودن آرایه تأثیری در روند جستجو ندارد.

الگوریتم جستجوی خطی

برای پرس و جوی(جستجوی) داده x در آرایه n تایی A [1.. n] :

```
flag : = false;
For  i: = 1  to  n  do
    if (A [ i ] = x ) then
        begin
            Flag: = True;
            break;
        end;
if Flag = True then
            write (' Location is:' , i ) ;
    else
            write (' not found');
```

در صورتی که شرط if برقرار شود ($x = A[i]$) عنصر x پیدا شده و محل آن i است در نتیجه Flag ، True شده از حلقه خارج می‌شویم.

تحلیل جستجوی خطی

در الگوریتم جستجوی خطی مهمترین پارامتر، مقایسه است، در نتیجه تعداد مقایسات مهمترین عامل در محاسبه مرتبه اجرایی الگوریتم جستجوی خطی به حساب می‌آید.

الف) بهترین حالت: داده مورد جستجو (x) در ابتدای لیست باشد (با فرض شروع جستجو از ابتدا). در این حالت حداقل مقایسه را داریم.

ب) بدترین حالت: داده مورد جستجو (x) در انتهای لیست باشد (با فرض شروع جستجو از ابتدا). در این حالت حداکثر مقایسه را داریم.

ج) حالت متوسط: متوسط مقایسات برابر است با: $\dfrac{\text{مجموع مقایسات}}{\text{تعداد عناصر آرایه‌ها}}$

در حالت کلی اگر داده x عنصر اول آرایه باشد با 1 مقایسه، اگر عنصر دوم باشد با 2 مقایسه و ... و اگر عنصر n ام باشد با n مقایسه بدست می‌آید،در نتیجه:

$$\text{مجموع مقایسات} = 1 + 2 + \cdots + n = \frac{n(n+1)}{2} \rightarrow$$

$$\text{متوسط} = \frac{\frac{n(n+1)}{2}}{n} = \frac{n+1}{2} \rightarrow \text{مقایسه}$$

$$\text{زمان} = O(n)$$

نتیجه‌گیری جستجوی خطی:

بهترین حالت	حالت متوسط	بدترین حالت
حداقل مقایسه ۱ در زمان $o(1)$	$\frac{n+1}{2}$ مقایسه در زمان $O(n)$	حداکثر مقایسه n در زمان $O(n)$

۲ـ جستجوی دودویی (Binary Search)

شرط اولیه در این روش جستجو **مرتب (sort) بودن آرایه** (پیش‌فرض صعودی) است، در غیر این صورت این روش غیرقابل استفاده و تعریف نشده خواهد بود.

روش جستجو

داده مورد جستجو (x) را با خانه میانی A [mid] آرایه مقایسه می‌کنیم، در صورتی که با آن خانه برابر باشد جستجو پایان می‌پذیرد در غیر این صورت در صورتی که x > A [mid] باشد به نیمه بالای آرایه می‌رویم و در صورتی که x < A [mid] باشد به نیمه پایین آرایه می‌رویم و دوباره با قسمت میانی آن نیمه عمل مقایسه را انجام می‌دهیم این عمل را تا زمانی انجام می‌دهیم که یا به داده مورد نظر برسیم که محل آن mid خواهد بود یا این‌که داده x در آرایه وجود ندارد که در این صورت Low (اندیس پایین نیمه آرایه) از high (اندیس بالای نیمه آرایه) **بیشتر** می‌شود.

الگوریتم جستجوی دودویی

x = داده جستجو شونده

Flag = پیش‌فرض False و در صورت پیدا شدن x ، True می‌شود

Low = اندیس پایین آرایه

High = اندیس بالای آرایه

A [1.. n] = آرایه n تایی مرتب صعودی

$mid = \left\lfloor \frac{low + high}{2} \right\rfloor$ = اندیس وسط آرایه که عامل مقایسه x با A [mid] است

الگوریتم (غیر بازگشتی)

```
Low:  = 1 ;
high :  = n ;
Flag :  = False;
while (low < = high) AND (Flag < > True ) do
begin
mid : = (low + high) Div 2;
if (A [mid] = x)  then
        Flag: = True
 else if (x < A [mid]) then
        High: = mid - 1
     else if ( x>A [mid] ) then
          Low : = mid + 1
end;
```

الگوریتم (بازگشتی)

```
procedure binary search (a: elementary; x:element; var Low , High, j:integer);
var
mid: integer;
begin
if (Low < = High)  then
begin
mid: = (Low + High) Div 2;
case compare (x , a[mid]) of
'>': binary search (a , x , mid + 1 , High, j);
'<' : binary search (a , x , Low , mid - 1, j);
' = ' : j:=  mid;
end;
end;
```

مثال:

1	2	3	4	5	6	7	8
10	20	30	40	50	60	70	80
3	2		1				

Low	high	mid	A[mid]	x	Flag
1	8	4	40	10	False
1	3	2	20	10	False
1	1	1	10	10	True

Flag مقدار True گرفته بنابراین محل عنصر x، mid = 1 است.

1	2	3	4	5	6	7	8
10	20	30	40	50	60	70	80
			1		2	3	

Low	high	mid	A[mid]	x	Flag
1	8	4	40	75	False
5	8	6	60	75	False
7	8	7	70	75	False
8	8	8	80	75	False
8	7	(Low > high)			False

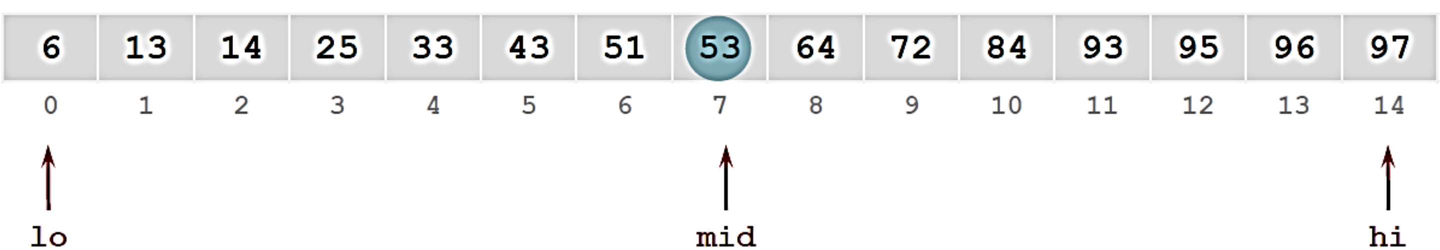
شرط اتمام حلقه اتفاق افتاده و Flag مقدار False داشته در نتیجه x در آرایه وجود ندارد.

جست جوی دودویی

- هدف. با داشتن یک آرایه‌ی مرتب و یک کلید، اندیس کلید را در آرایه پیدا کنید.

33

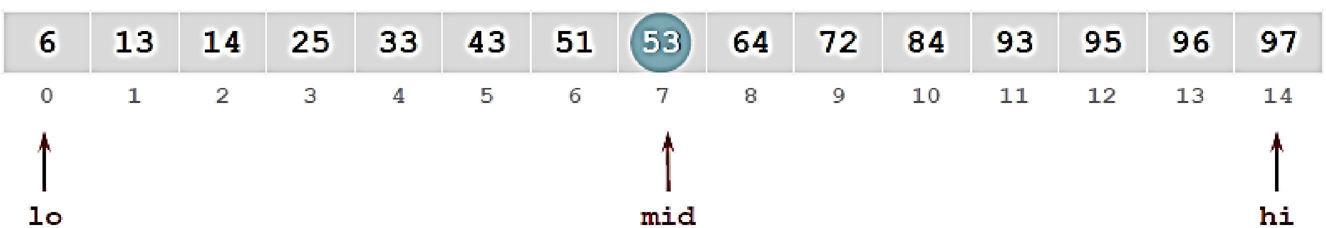
- جستجوی دودویی. کلید مورد نظر را با کلید وسط مقایسه کن:
  - اگر کوچک‌تر است، نیمه‌ی چپ را جستجو کن.
  - اگر بزرگ‌تر است، نیمه‌ی راست را جستجو کن.
  - اگر مساوی است، کلید پیدا شده است.



جست جوی دودویی

- هدف. با داشتن یک آرایه‌ی مرتب و یک کلید، اندیس کلید را در آرایه پیدا کنید.
- جستجوی دودویی. کلید مورد نظر را با کلید وسط مقایسه کن:
  - اگر کوچک‌تر است، نیمه‌ی چپ را جستجو کن.
  - اگر بزرگ‌تر است، نیمه‌ی راست را جستجو کن.
  - اگر مساوی است، کلید پیدا شده است.

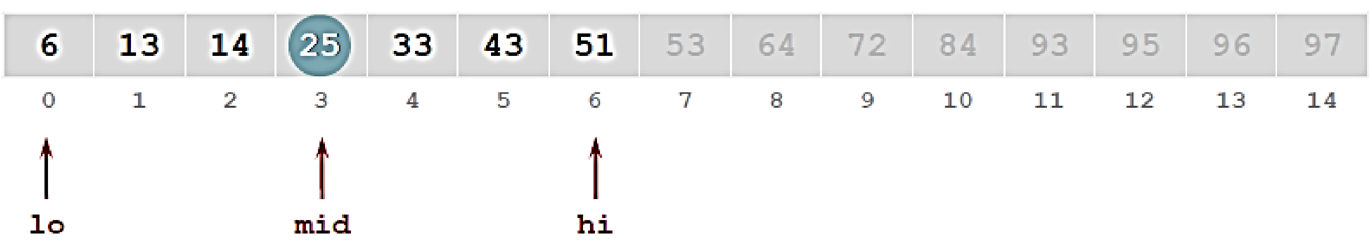
جستجوی موفق ۳۳



جست جوی دودویی

- هدف. با داشتن یک آرایه‌ی مرتب و یک کلید، اندیس کلید را در آرایه پیدا کنید.
- جستجوی دودویی. کلید مورد نظر را با کلید وسط مقایسه کن:
  - اگر کوچک‌تر است، نیمه‌ی چپ را جستجو کن.
  - اگر بزرگ‌تر است، نیمه‌ی راست را جستجو کن.
  - اگر مساوی است، کلید پیدا شده است.

جستجوی موفق ۳۳



جست جوی دودویی

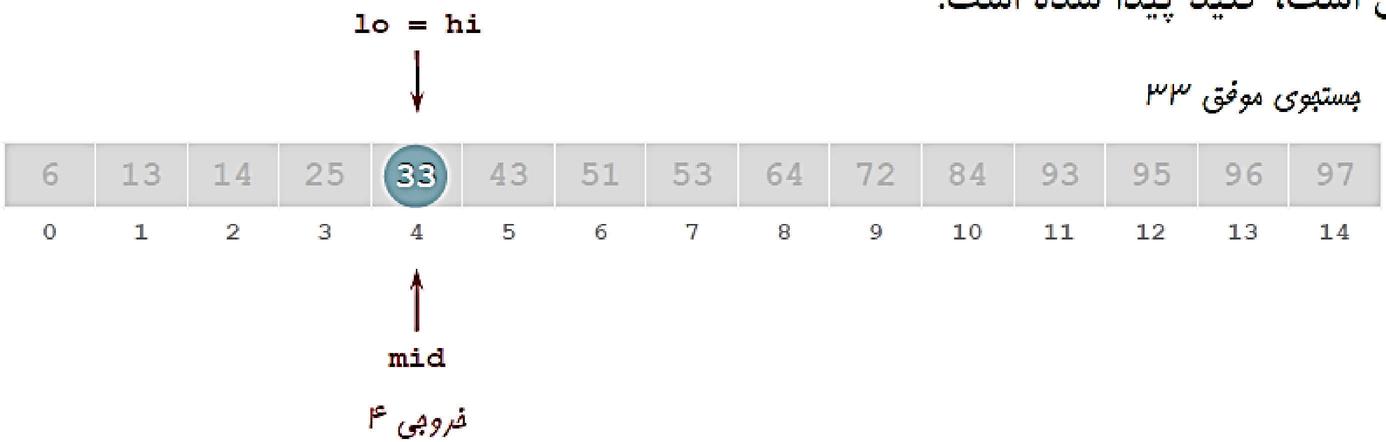
- هدف. با داشتن یک آرایه‌ی مرتب و یک کلید، اندیس کلید را در آرایه پیدا کنید.
- جستجوی دودویی. کلید مورد نظر را با کلید وسط مقایسه کن:
  - اگر کوچک‌تر است، نیمه‌ی چپ را جستجو کن.
  - اگر بزرگ‌تر است، نیمه‌ی راست را جستجو کن.
  - اگر مساوی است، کلید پیدا شده است.

جستجوی موفق ۳۳

6	13	14	25	33	43	51	53	64	72	84	93	95	96	97
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
				lo	mid	hi								

جست جوی دودویی

- هدف. با داشتن یک آرایه‌ی مرتب و یک کلید، اندیس کلید را در آرایه پیدا کنید.
- جستجوی دودویی. کلید مورد نظر را با کلید وسط مقایسه کن:
  - اگر کوچک‌تر است، نیمه‌ی چپ را جستجو کن.
  - اگر بزرگ‌تر است، نیمه‌ی راست را جستجو کن.
  - اگر مساوی است، کلید پیدا شده است.



تحلیل جست جوی دودویی

- گزاره. جستجوی دودویی برای جستجو در یک آرایه‌ی مرتب با اندازه‌ی **N**، حداکثر **lg N + 1** مقایسه انجام می‌دهد.

- تعریف.
  - `T(N)` = حداکثر تعداد مقایسه‌های جستجوی دودویی در یک زیرآرایه‌ی مرتب با اندازه‌ی N.

- رابطه‌ی بازگشتی. $T(N) \le T(N/2) + 1, \quad T(1) = 1$

$$
\begin{aligned}
T(N) &\le T(N/2) + 1 \\
&\le T(N/4) + 1 + 1 \\
&\le T(N/8) + 1 + 1 + 1 \\
&\;\;\cdots \\
&\le T(N/N) + 1 + 1 + \ldots + 1 \\
&= 1 + \lg N
\end{aligned}
$$

تحلیل الگوریتم

به آرایه زیر و تعداد مقایسهٔ بدست آمده از جستجوهای **موفق و ناموفق** دقت کنید:

	1	2	3	4	5	6	7	8	9	10	
	12	30	34	40	45	50	56	60	64	70	
تعداد مقایسات برای جستجوی موفق	3	2	3	4	1	3	4	2	3	4	
تعداد مقایسات برای جستجوی نا موفق	3	3	3	4	4	3	4	4	3	4	4

بطور مثال **در جستجو ی موفق:**

$x = 45$ با 1 مقایسه بدست می آید.

$x = 40, 56, 70$ هر کدام با $4 = 1 + \lfloor \log_2 10 \rfloor$ مقایسه بدست می آیند.

بطور مثال **در جستجو ی نا موفق:**

$x = 20$ بین $12,30$ با $3 = \lfloor \log_2 10 \rfloor$ مقایسه ناموفق تمام می‌شود.

$x = 66$ بین $64,70$ با $4 = 1 + \lfloor \log_2 10 \rfloor$ مقایسه ناموفق تمام می‌شود.

جستجوی موفق

۱- حداقل تعداد مقایسه : 1

۲- حداکثر تعداد مقایسه : $\lfloor \log_2 n \rfloor + 1$

۳- اگر x در $A[1..N]$ باشد، همواره با **حداکثر** $O(\log_2 N)$ پیدا می‌شود.

٤- در جستجوی موفق برای دو مقدار متفاوت تعداد مقایسه‌های لازم **لزوماً برابر نیست**.

جستجوی ناموفق

۱- حداقل تعداد مقایسه : $\lfloor \log_2 n \rfloor$

۲- حداکثر تعداد مقایسه : $\lfloor \log_2 n \rfloor + 1$

مثال۳: اگر A **آرایه‌ای مرتب** از اعداد صحیح **1 تا 1024** باشد، الگوریتم جستجوی دودویی با چند بار تکرار **عدد 4** را پیدا می‌کند؟

۱) 8 ۲) 7 ۳) 9 ۴) 15

بار اول با مقایسه عدد 4 با وسط آرایه متوجه می‌شویم که عدد مذکور باید مابین خانه‌های 1 تا 512 یعنی نیمه پایینی آرایه باشد به همین ترتیب:

شمارهٔ مقایسه	عدد 4 در کدام محدوده است؟
1	1 تا 512
2	1 تا 256
3	1 تا 128
4	1 تا 64
5	1 تا 32
6	1 تا 16
7	1 تا 8

پس از 7 بار تکرار الگوریتم آرایه زیر در نظر گرفته می‌شود:

Low			mid			high
1	2	3	4	5	6	7

$$mid = \left\lfloor \frac{1+7}{2} \right\rfloor = 4$$

بار هشتم هنگامی که عدد 4 با محتوای mid مقایسه می‌شود، پیدا می‌گردد پس الگوریتم 8 بار تکرار می‌شود.

نتیجه‌گیری جستجوی دودویی:

بهترین حالت	حالت متوسط	بدترین حالت
حداقل مقایسه ۱ در زمان $o(1)$	کمتر از $\lfloor \log_2 n \rfloor + 1$ در زمان $O(\log_2 n)$	حداکثر مقایسه $\lfloor \log_2 n \rfloor + 1$ در زمان $O(\log_2 n)$

دیده می‌شود که روش جستجوی دودویی به مراتب از روش جستجوی خطی از نظر تعداد مقایسه بهتر و مقرون به صرفه‌تر است.