

ساختمان دادهها**الگوریتم:**

تعریف ۱: مجموعه محدودی از دستورالعمل‌ها که با دنبال کردن آن‌ها هدف خاصی دنبال می‌شود، که لزوماً منحصر به فرد نیست.

تعریف ۲: روش دقیق حل یک مسئله به صورت قدم به قدم که لزوماً منحصر به فرد نیست.

مثال: مطلوب است تغییر مقدار دو متغیر x و y ($x = 10, y = 20$)

روش I	روش II	روش III
30 10 20 $x \leftarrow x + y$	10 10 temp $\leftarrow x$	200 10 20 $x \leftarrow x * y$
10 30 20 $y \leftarrow x - y$	20 20 $x \leftarrow y$	10 200 20 $y \leftarrow x / y$
20 30 10 $x \leftarrow x - y$	10 10 $y \leftarrow temp$	20 200 10 $x \leftarrow x / y$

مثال فرق نشان می‌دهد که روش حل یک مسئله ممکن است منحصر به فرد و یکتا نباشد.

□ خصوصیات الگوریتم:

- ۱) ورودی: حداقل ۰ ورودی (ممکن است الگوریتم ورودی نداشته باشد).
- ۲) خروجی: حداقل ۱ خروجی (هر الگوریتم حداقل یک خروجی خواهد داشت).
- ۳) قطعیت (عدم ابهام): هر کدام از دستورالعمل‌ها باید دقیق و بدون ابهام باشد.
- ۴) محدودیت (پایان پذیر بودن): هر الگوریتم پس از طی مراحل مشخصی باید اتمام پذیرد.
- ۵) کارانی (انجام پذیر بودن): امکان پیاده‌سازی و اجرای الگوریتم روی کاغذ وجود داشته باشد به عبارت بهتر الگوریتم انجام شدنی باشد.

برنامه:

مجموعه‌ای از دستورالعمل‌های یک زبان برنامه‌سازی که امکان پیاده‌سازی و اجرای یک الگوریتم در کامپیوتر را فراهم کند.

□ تفاوت برنامه و الگوریتم:

یک برنامه تمام خصوصیات یک الگوریتم را به جز "شرط پایان‌پذیر بودن" شامل می‌شود.

هر الگوریتم لزوماً پایان‌پذیر است اما هر برنامه لزوماً پایان‌پذیر نیست.

مثال: سیستم‌عامل برنامه‌ای است، که هیچ گاه پایان نمی‌پذیرد و همواره در یک سیکل انتظار قرار دارد تا برنامه بعدی وارد شود.

ساختمان دادهها:

به ساختارهایی که جهت دریافت داده‌های خام به شکل مناسب توسط کامپیوتر برای پیاده‌سازی و اجرای الگوریتم‌های مختلف مورد استفاده قرار می‌گیرند، ساختمان داده‌ها گفته می‌شود.

مسئله: 10 عدد صحیح را گرفته، به صورت مرتب نشان دهد.

ساختمان داده مورد نیاز: آرایه صحیح 10 تابعی

این مثال از ساختمان داده آرایه برای گرفتن 10 داده خام صحیح و حل مسئله استفاده می‌شود.

واع ساختمان داده‌ها:

(۱) اولیه: داده صحیح - داده اعشاری - داده کاراکتری

(۲) غیراولیه: آرایه - رکورد - رشته ...

(۱) پسته

(۲) صفحه

۱- ایستا

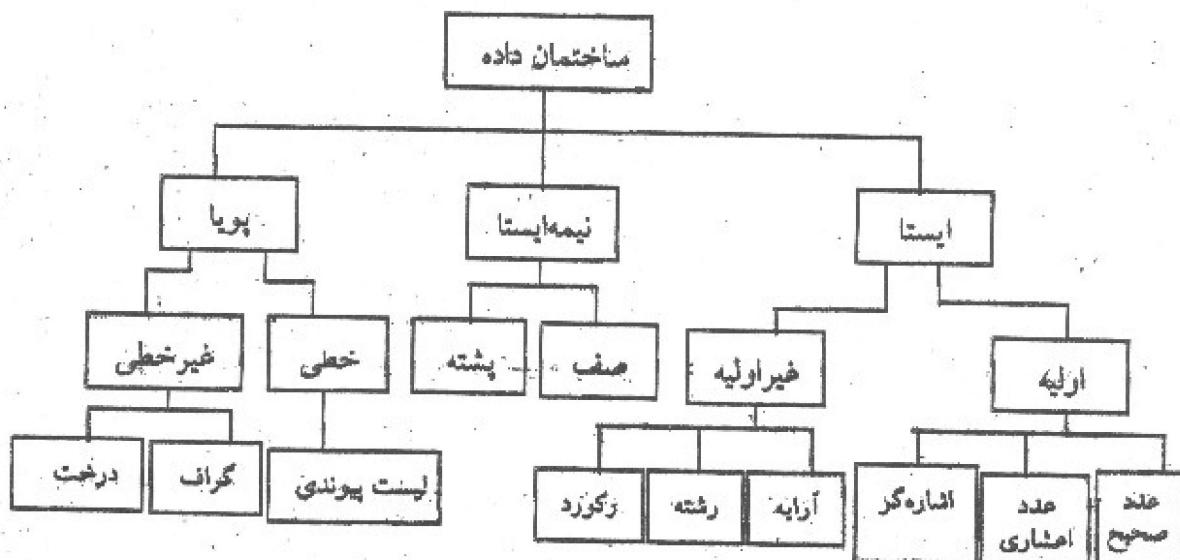
۲- نیمه ایستا

۳- پورا

ع ساختمان داده‌ها:

(۱) خطی: لیست های پرتوانی

(۲) غیرخطی: درخت گراف



۴ داده‌های ایستا:

فضای محدود و از پیش تعریف شده استفاده کرده و این فضای در طول اجرای برنامه ثابت است.

اند:

(داده صحیح) int \leftarrow در طول اجرای برنامه به زبان C، همیشه 2 بایت است

(داده اعشاری) real \leftarrow در طول اجرای برنامه به زبان پاسکال، همیشه 6 بایت است.

۵ داده‌های پورا (Heap):

داده‌های پورا با استفاده از اشاره گرها امکان تغیرات ناصلح و پورا متناسب با داده‌ها را فراهم می‌کند.

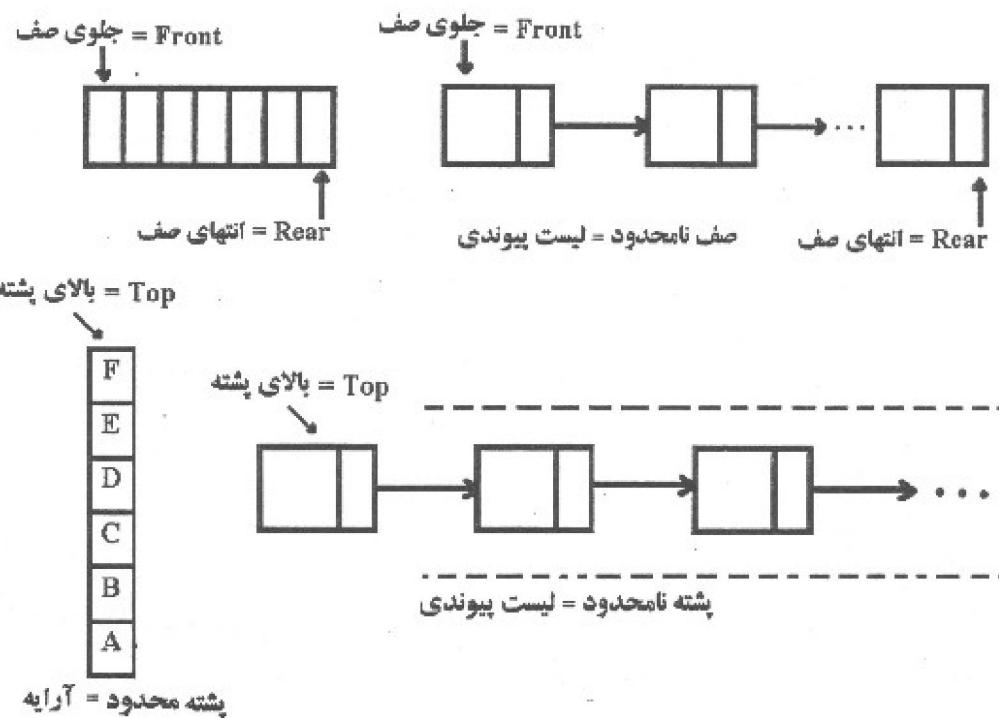
دایدهای نیمه ایستا (پشته و صف):

 **پادآوری:**

۱) صف: لیستی که اولین ورودی در آن، اولین خروجی خواهد بود. (FIFO = First In First Out)

۲) پشته: لیستی که اولین ورودی در آن، آخرین خروجی خواهد بود. (FILO = First In Last out)

لیست مربوط به صف و پشته را می‌توان هم به صورت محدود با استفاده از آرایه‌ها (ساختارهای ایستا) و هم به صورت نامحدود با استفاده از لیست‌های پیوندی (ساختارهای پویا) پاده‌سازی کرد؛ از این رو به آن‌ها نیمه ایستا می‌گویند.

**■ مقدمه‌ای بر «تحلیل پیچیدگی زمانی و مرتبه اجرایی» الگوریتم‌ها:**

در ارزیابی یک برنامه یا الگوریتم دو عامل اصلی وجود دارد، که باید مورد توجه قرار گیرد یکی حافظه مصرفی و دیگری زمان مصرفی الگوریتم است. یعنی الگوریتمی بهتر است که فضای مصرفی و زمان مصرفی کمتری را درخواست کند.

کلکه در الگوریتم‌هایی که در این مجموعه مورد بحث قرار می‌دهیم، «عامل زمان» مهمترین عامل به حساب می‌آید.

کلکه محاسبه زمان اجرایی یک الگوریتم متناسب یا تعداد دفعاتی است که عملیات اصلی انجام می‌شود. در تحلیل زمان اجرای یک الگوریتم تمام دستورات را شمارش نمی‌کنیم زیرا تعداد دستورات به نوع زبان برنامه‌نویسی بستگی دارد.

کلکه برای محاسبه زمان اجرایی یک الگوریتم فقط به عملیات اصلی نیاز داریم که مستقل از کامپیوتر و زبان برنامه‌نویسی هستند، در عین حال هیچ قاعده مشخص برای انتخاب عملیات اصلی وجود ندارد و انتخاب عمل اصلی به صورت تجربی انجام می‌یابد.

۳ شمارش گام‌ها یا قدم‌ها یا مراحل یک برنامه:

در صورتیکه هر عمل اصلی در یک برنامه یک گام اختیار کند. گام‌های هر برنامه با استفاده از قواعد کلی زیر قابل محاسبه خواهد بود.

- تعاریف زیربرنامه‌ها و توابع دارای گام 0 است.

procedure	F(...);	$\rightarrow 0$
Function	P(...); ...;	$\rightarrow 0$
void	F(...);	$\rightarrow 0$
نوع	P(...);	$\rightarrow 0$

- هر بلاک شامل {} یا begin و end دارای گام 0 است.

- تعاریف متغیرها در صورتیکه مقداردهی اولیه برای آنها انجام گیرد، گام 1 و در غیر این صورت گام 0 دارند.

int	x; $\rightarrow 0$	var	
int	x = 3; $\rightarrow 1$	j,i:integer;	$\rightarrow 0$
float	a,b = 5; $\rightarrow 1$		

- در هر دستور اجرائی به ازاء هریار اجرا دارای گام 1 است.

$$y = x * y + z; \quad \rightarrow 1$$

مثال:

(1)

در حلقه زیر تعداد تکرار: 1+ابتدا - انتها = n

$$\begin{aligned} \text{for } i := 1 \text{ to } n \text{ do} &\rightarrow n+1 \\ s := s + 1; &\rightarrow + \frac{n}{2n+1} \end{aligned}$$

(2)

در حلقه زیر تعداد تکرار: 1+ابتدا - انتها = $n - 2 = (n-1) - 2 + 1$

$$\begin{aligned} \text{for } i := 2 \text{ to } n-1 \text{ do} &\rightarrow n-1 \\ s := s + 1; &\rightarrow + \frac{n-2}{2n-3} \end{aligned}$$

(3)

در حلقه زیر تعداد تکرار: 1+ابتدا - انتها = $n - 1 = n - 2 + 1$

$$\begin{aligned} \text{for } i := 2 \text{ to } n \text{ do} &\rightarrow n \\ \text{begin} &\rightarrow + \\ s := s + 3; &\rightarrow + \frac{n-1}{3n-2} \\ r := r + 3; &\rightarrow \frac{n-1}{3n-2} \\ \text{end;} & \end{aligned}$$

معامل مثال‌های ۱ و ۲ و ۳ در زبان C

۱

۲

۳

```
for (i = 1 ; i <= n ; i++)
    s = s + i;
```

```
for(i = 2 ; i <= n - 1 ; i++)
    s = s + i;
```

```
for(i = 2 ; i <= n ; i++)
{
    s = s + 3;
    r = r + 3;
}
```

۵- در دستور شرطی if خیارت شرط ۱ گام دارد اما با توجه به درست یا غلط بودن شرط چون جملات مختلفی ممکن است اجرا شود، گام کل دستور شرط و البته به درست یا غلط بودن شرط خواهد بود (مگر در یک حالت خاص که در ازاه درست یا غلط بودن شرط فقط یک دستور اجرا شود که در این صورت ۲ گام خواهد بود).

مثال:

$$\begin{array}{l} \text{if } (x < y) \rightarrow 1 \\ \quad s = 2; \rightarrow 1 \\ \quad \text{else} \\ \quad \quad s = 5; \rightarrow 1 \end{array} \left. \begin{array}{l} \text{شرط درست} \\ \text{شرط غلط} \end{array} \right\} \begin{array}{l} 1 + 1 = [2] \\ 1 + 1 = [2] \end{array}$$

$$\begin{array}{l} \text{if } (x < y) \rightarrow 1 \\ \quad s = s + 1; \rightarrow 1 \\ \quad \text{else} \\ \quad \quad \{ \\ \quad \quad \quad t = t + 1; \rightarrow 1 \\ \quad \quad \quad r = r + 1; \rightarrow 1 \} \end{array} \left. \begin{array}{l} \text{شرط درست} \\ \text{شرط غلط} \end{array} \right\} \begin{array}{l} 1 + 1 = [2] \\ 2 + 1 = [3] \end{array}$$

۶- دستور حلقه for

مهمنترین قسمت در شمارش گام‌های یک برنامه دستور حلقه for است که به ترتیب زیر گام آن را محاسبه می‌کیم:

الف) ابتدا تعداد تکرار حلقه را محاسبه می‌کیم.

ب) حلقه for به تعداد "تکرار + 1" گام اختیار می‌کند.

ج) جملات تکرار شونده داخل حلقه for به تعداد "تکرار" گام اختیار می‌کنند.

پادآوری:

۱- تعداد تکرار حلقه‌ای زیر "b - a + 1" است.

for i := a to b do $\begin{array}{l} \text{for } (i = a ; i <= b ; i++) \\ \quad \quad \quad \downarrow \end{array}$

۲- تعداد تکرار حلقه زیر "b - a" است.

for (i = a ; i < b ; i++)

تفاوت مورد ۱ و ۲ آن است که در ۱ شرط $b \leq i$ اما در ۲ شرط $b < i$ است.

روابط بالا در شرایطی عمل می‌کنند که به ازاء هر بار تکرار حلقة ۱ واحد به متغیر شمارنده ۱ اضافه شود.

در حلقه زیر چون شرط α است با توجه به قسمت یادآوری، تعداد تکرار "ایندا- انتها" بوده و برابر با 2- α است.

for ($i = 2$; $i < n$; $i++$) \rightarrow $n - 1$
 $s = s + 1;$ \rightarrow $+ \frac{n - 2}{2n - 3}$

七

```

+   0    ← int f(int x)
+   0    ← {
+   1    ← int i, j=0;
+   n    ← for(i = 2; i <=n ; i++)
+   n-1   ← j = j + i;
+   1    ← return i;
+   2n+1   ← }

```

تعداد تکرار در حلقه بالا $n - 2 + 1 = n - 1$ بیار است.

حلقه‌های تو در تو:

برای محاسبه گام حلقه‌های تودرتو به صورت زیر عمل می‌کیم:

۱- از پیروزی قریم خلقه شروع می کیم.

۲- تعداد تکرار هر حلقه را برای تمام حلقه‌ها و دستورات تکرار شونده پائین در نظر گرفته و "تعداد تکرار + 1" را برای خود حلقه در نظر می‌گیریم.

گ-قسمت 2 را برای تمام حلقه‌ها انجام می‌دهیم.

- 112 -

11

$$\begin{aligned}
 \text{for } i := 1 \text{ to } m \text{ do} &\rightarrow (m+1) \\
 \text{for } j := 1 \text{ to } n \text{ do} &\rightarrow + m \times (n+1) \\
 s = s + 1; &\rightarrow + m \times n \\
 &\rightarrow \frac{(m+1) + m(n+1) + mn}{(m+1) + m(n+1) + mn} = 2mn + 2m + 1
 \end{aligned}$$

17

for $i := 1$ to m do $\rightarrow (m + 1)$
 for $j := 2$ to $n - 1$ do $\rightarrow (m) \times (n - 1)$
 for $k := 1$ to L do $\rightarrow (m) \times (n - 2)(L + 1)$
 $s := s + 1;$ $\rightarrow (m) \times (n - 2)(L)$

تکرار حلقه j تکرار حلقه k $L - 1 + 1 = L$ $n - 1 - 2 + 1 = n - 2$

$$m - 1 + 1 = m$$

(۳)

```

for i:=1 to m do → + (m+1)
for j:=2 to n do → + (m)(n)
begin
s := s + l; → + (m)(n-1)
r := r + 1; → + (m)(n-1)
end;
            → 3mn - m + 1

```

تکرار حلقه j

تکرار حلقه i

(۴)

```

procedure add(var a, b, c:matrix; m, n:integer);
var
i, j:integer;
begin
for i:=1 to m do → + (m+1)
for j:=1 to n do → + (m)(n+1)
C[i, j] := a[i, j] + b[i, j]; → + (m)(n)
            → 2mn + 2m + 1
end;

```

(۵)

```

procedure P(x:integer); → 0
var
s, j, i:integer; → 0
begin
→ 0
i := 0; → 1
for i:=1 to m do → + (m+1)
for j:=1 to n do → + (m)(n+1)
s := s + 1; → + (m)(n)
s := s + 1; → + 1
end; → 0
            → 2mn + 2m + 3

```

لکته ۱: تعداد گام حلقه های for مانند while می شوند یعنی گام حلقه یک واحد از تعداد تکرار حلقه بیشتر است.

$n+1 \leftarrow$	for($i=1; i \leq n; i++$)	$i = 1; \rightarrow 1$
$\frac{n}{2n+1} \leftarrow$	$s = s + 1;$	$=$ while($i \leq n$) $\rightarrow n+1$
		{
		$s = s + 1; \rightarrow n$
		$i = i + 1; \rightarrow n$
		}
		$\overline{3n+2}$

نکته ۲: برای محاسبه تعداد گام حلقه های `do ... until` و `repeat ...` باید در نظر بگیریم که چون شرط حلقه انتها حلقه کنترل می شود، تعداد گام حلقه برابر با تعداد تکرار حلقه است. برای محاسبه گام این نوع حلقه ها توصیه می شود برای یک مقدار فرضی n تکرار حلقه را محاسبه کرده و از روی آن گام را بدست می آوریم.

مثال:

$$2n - 1 \leftarrow \begin{cases} 1 & \leftarrow i := 1; \\ & \text{repeat} \\ + & n - 1 \leftarrow i := i + 1; \\ + & n - 1 \leftarrow \text{until } i \geq n; \end{cases}$$

مرتبه اجرائی (O)

طبق تعریف $f(n) = O(g(n))$ می باشد در این حالت می گوئیم مرتبه اجرائی تابع $f(n)$ می باشد. اگر و فقط اگر به ازاء توابع نامنی f و g داشته باشیم:

$$f(n) = O(g(n)) \Leftrightarrow \exists C, n_0 > 0 : \forall n \geq n_0, f(n) \leq Cg(n)$$

برای انتخاب (n) g مناسب به صورت تجربی از (n) f جمله غالب را انتخاب می کیم، به طور مثال:

$$f(n) = 3n + 3 = O(n)$$

$$f(n) = n^2 + 10n = O(n^2)$$

$$f(n) = \log_2 n + 1 = O(\log n)$$

$$\text{نکته ۱: اگر } f(n) = O(n^m) \text{ باشد آن گاه } f(n) = a_m n^m + \dots + a_1 n^1 + a_0$$

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

$$f(n) = \frac{n(n-1)}{2} = \frac{n^2}{2} - \frac{n}{2} = O(n^2)$$

$$f(n) = 4 = 4n^0 = O(n^0) = O(1)$$

$$f(n) = a_0 = a_0 n^0 = O(n^0) = O(1)$$

نکته ۲: جدول زیر مرتبه اجرائی توابع مهم را به ترتیب صعودی مطرح کرده است:

نام تابع	ثابت	خطی	لگاریتمی	مرتبه 2	توانی	فاکتوریل	توانی
مرتبه اجراء	$O(1) <$	$O(\log n) <$	$O(n) <$	$O(n \log n) <$	$O(n^2) <$	$O(2^n) <$	$O(n!) < O(n^n)$

«بصت مرتبه اجرائی به طور کامل در قسمت مرتب شناختی شرح داده می شود»

■ توابع و زیر برنامه‌های بازگشتی (recursion)

استفاده از روند بازگشتی مربوط به مسائلی می‌شود که روند حل آن مسائل به شکل بازگشتی انجام می‌شود. به طور مثال:

$$1) n! = n \times (n - 1)!$$

$$2) x^n = x \times x^{n-1}$$

در مثال ۱ اگر تابع $\text{Fact}(n)$ را برای $n!$ در نظر بگیریم رابطه بازگشتی آن

$$\text{Fact}(n) = n \times \text{Fact}(n - 1)$$

و در مثال ۲ اگر تابع $\text{pow}(x, n)$ را برای x^n در نظر بگیریم رابطه بازگشتی آن

$$\text{Pow}(x, n) = x * \text{pow}(x, n - 1)$$

می‌شود.

تعریف: به هر تابع یا زیر برنامه‌ای که بتواند داخل بدنه‌اش، خودش را دوباره فراخوانی کند یا صدای بزند، تابع یا زیر برنامه بازگشتی می‌گویند.

شرطی:

هر تابع بازگشتی نیاز به ۲ نقطه دارد.

۱- نقطه بازگشت: دستوری که دوباره تابع یا زیر برنامه را فراخوانی کند.

۲- نقطه بنست: دستوری که با یک شرط مشخص باعث اتمام بازگشت‌ها شود.

مراحل بازگشت در ظرفی به نام stack یا پشته نگهداری می‌شود، و چون این ظرف محدود است، در صورت نبودن نقطه بنست بعد از بازگشت‌های مکرر، این ظرف پرشده و با پیغام خطای stack over flow مواجه می‌شود.

مثال: تابع بازگشتی زیر فاکتوریل عدد n را برمی‌گرداند.

(زبان پاسکال)

```
Function Fact(n;integer): integer;
```

```
begin
```

```
    if n <= 1 then Fact:= 1
```

```
    else Fact:= n * Fact(n - 1);
```

```
end;
```

(زبان C)

```
int Fact(int n)
```

```
{
```

```
if (n <= 1)
```

```
    return 1;
```

```
else
```

```
    return n * Fact(n - 1);
```

```
}
```

دیده می‌شود که نقطه بنست نقطه‌ای است که دیگر تابع فراخوانی نشده فقط یک مقدار برگردانده می‌شود و یا در بعضی مواقع در خروجی نمایش داده می‌شود. (در بعضی شرایط نیز با استفاده از دستور exit به روند بازگشت خاتمه داده و به مرحله قبل برگردیم)

مشخصات الگوریتم‌های بازگشتی:

- ۱- **سource** (سورس کد) کوتاه: الگوریتم‌های بازگشتی مانند فاکتوریل را به راحتی می‌توان به صورت بازگشتی با دستورات کمتری نسبت به حالت غیربازگشتی پیاده‌سازی کرد.
- ۲- اتفاف حافظه پیشتر: به علت استفاده از حافظه اضافی **stack** که مراحل بازگشت را نگهداری می‌کند، اتفاف حافظه نسبت به حالت غیربازگشتی پیشتر است.
- ۳- سرعت اجرا کمتر: سرعت اجرا در پیشتر الگوریتم‌های بازگشتی کمتر از الگوریتم غیربازگشتی می‌باشد.

■ مزایا:

(سورس) کوتاه - راحتی پیاده‌سازی برخی الگوریتم‌ها که روند حل آن‌ها بازگشتی است.

■ معایب:

اتفاق حافظه - سرعت اجرای کمتر

■ دلایل استفاده از توابع یا فیروزنامه‌های بازگشتی

- ۱- راحتی تدوین و پیاده‌سازی مسائلی که روند حل آن‌ها به صورت بازگشتی (recursive) انجام می‌شود. مانند: فاکتوریل
- ۲- تعداد کم دستورات استفاده شده نسبت به حالت غیربازگشتی

الگوریتم غیربازگشتی فاکتوریل n

الگوریتم بازگشتی فاکتوریل n

```
int Fact (int n)
{
    int f = 1 , i;
    for (i = 1 ; i <= n ; i++)
        f = f * i;
    return f;
}
```

```
int Fact (int n)
{
    if (n <= 1)
        return 1;
    else
        return n * Fact (n);
}
```

نکته: توابع بازگشتی، معمولاً می‌توانند به ۲ شکل یابان شوند.

۱- ضابطه ریاضی

۲- الگوریتم یا یک برنامه به زبان برنامه‌نویسی

مثال: فاکتوریل

ضابطه ریاضی

برنامه‌نویسی

```

function fact (n: integer): integer;
begin
  if n <= 1. then
    fact := 1
  else
    fact := n * fact ( n - 1 )
  end;

```

$$\text{fact} = \begin{cases} 1 & n \leq 1 \\ n \times \text{fact}(n-1) & (n > 1) \end{cases}$$

در غیر این صورت

■ روش حل تابع و زیر برنامه‌های بازگشتی

۱- روش درختی: این روش زمانی استفاده می‌شود که تابع بازگشتی به صورت ضابطه ریاضی داده شود یا این که از مقدار بازگشتی در خطوط بعدی برنامه استفاده نشود.

۲- روش مرحله به مرحله: زمانی استفاده می‌شود که مقدار بازگشتی در خطوط بعدی زیر برنامه یا تابع بازگشتی به کار برده می‌شود، (معمولًاً مربوط به زیر برنامه‌های بازگشتی است)

مثال ۱: اگر A و B دو عدد صحیح نامنفی باشند و تابع M به صورت زیر تعریف شده باشد، حاصل (20 , 28) M چیست؟

$$M(A, B) = \begin{cases} M(B, A) & A < B \\ A & B = 0 \\ M(B, \text{Mod}(A, B)) & \text{وگرنه} \end{cases}$$

$$M(20, 28)$$

$\downarrow \uparrow$

$$M(28, 20)$$

$\downarrow \uparrow$

$$M(20, 8)$$

$\downarrow \uparrow$

$$M(8, 4)$$

$\downarrow \uparrow$

$$M(4, 0)$$

$\downarrow \uparrow$

بن بست 4

مثال ۲: فرض کنید که a و b اعداد صحیح مثبت بوده و تابع Q به صورت زیر به شکل بازگشتی تعریف شده باشد:

$$Q(a, b) = \begin{cases} 0 & a < b \\ Q(a - b, b) + 1 & a \geq b \end{cases}$$

حاصل (3, 14) Q چیست؟

$$\overbrace{Q(14, 3)}^{\textcircled{4}}$$

$$\downarrow \uparrow$$

$$\overbrace{Q(11, 3)}^{\textcircled{1}} + 1$$

$$\downarrow \uparrow$$

$$\overbrace{Q(8, 3)}^{\textcircled{2}} + 1$$

$$\downarrow \uparrow$$

$$\overbrace{Q(5, 3)}^{\textcircled{3}} + 1$$

$$\downarrow \uparrow$$

$$\overbrace{Q(2, 3)}^{\textcircled{4}} + 1$$

بن بست ۰

مثال ۳: خروجی تابع زیر برای فرآنخوانی $(5, 2)$ Test چقدر است؟

Function test (x, y: integer): integer;

begin

if $(x \leq y)$ or $(y = 0)$ then

test := x

else if $(y = 1)$ then test := test $(x - 1, y) + 1$ else test := test (test $(y, x), y - 1) + 2$

end;

البته می توان شکل ضابطه ریاضی تابع را به صورت زیر در نظر گرفت:

$$\text{test}(x, y) = \begin{cases} x & (x \leq y) \text{ or } (y = 0) \\ \text{test}(x - 1, y) + 1 & y = 1 \\ \text{test}(\text{test}(y, x), y - 1) & \text{وگرنه} \end{cases}$$

$$\overbrace{\text{test}(5, 2)}^{\textcircled{4}}$$

$$\downarrow$$

$$\overbrace{\text{test}(\underbrace{\text{test}(2, 5)}, 1) + 2}^{\textcircled{2}}$$

$$\downarrow \uparrow$$

$$\overbrace{\text{test}(1, 1)}^{\textcircled{1}} + 1$$

بن بست ۱

مثال: خروجی تابع زیر برای فراخوانی $\text{ack}(1, 2)$ چقدر است؟

$$\text{ack}(m, n) = \begin{cases} n + 1 & m = 0 \\ \text{ack}(m - 1, 1) & (m < 0) \text{ And } (n = 0) \\ \text{ack}(m - 1, \text{ack}(m, n - 1)) & \text{وگرنه} \end{cases}$$

$\underbrace{\text{ack}(1, 2)}$

$\downarrow 4$

$\text{ack}(0, \underbrace{\text{ack}(1, 1)})$

$\downarrow 3$

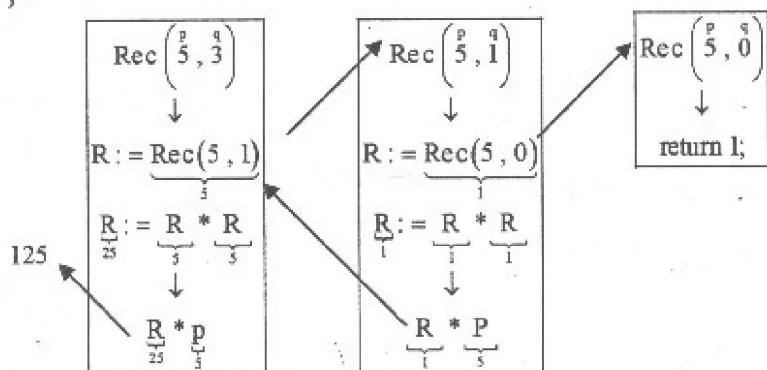
$\text{ack}(0, \underbrace{\text{ack}(1, 0)}_{\downarrow 2})$

$\downarrow 2$

$\underbrace{\text{ack}(0, 1)}_{2}$

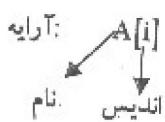
مثال: در روال بازگشتی زیر مقدار $\text{Rec}(5, 3)$ کدام است؟

```
int Rec (int p, int q)
{
    int R;
    if (q <= 0) return 1;
    R = Rec (p, q/2);
    R = R * R;
    if (q mod 2 == 0)
        return R;
    else
        return R * P;
}
```



آرایه‌ها:

مجموعه‌ای از داده‌های پشت سرهم حافظه که همگی از یک نوع بوده و از آدرس مشخص شروع می‌شوند.
تکه ۱: هر آرایه تحت یک نام واحد معروف شده و هر عضو آن با یک اندیس مشخص قابل دستابی است.



نام آرایه = A

۱	۲	۳	۴	۵	۶
10	25	30	17	95	100

برای رسیدن به مقدار خانه چهارم (۱۷) باید از $A[4]$ یا (۴) استفاده کنیم که در اینجا ۴ اندیس خانه چهارم است.

تکه ۲: آرایه‌ها بر حسب وضعیت‌های مختلف نمایش می‌توانند به صورت‌های یک بعدی، دو بعدی و ... مطرح شوند.

آنچه آرایه A می‌تواند به شکل‌های زیر مطرح شود:

$$A[L_1 \dots U_1] \rightarrow \text{پردار} = \text{لیست} = \text{یک بعدی}$$

$$A\left[\underbrace{L_1 \dots U_1}, \underbrace{L_2 \dots U_2}\right] \rightarrow \text{جدول} = \text{ماتریس} = \text{دو بعدی}$$

بعد دوم = ستون سطر = بعد اول

$$A[L_1 \dots U_1, L_2 \dots U_2, \dots, L_n \dots U_n] \rightarrow \text{بعدی}$$

در حالت خاص اگر آرایه به صورت (U_1, U_2) تعریف شود، اندیس پائین به صورت پیش فرض ۱ در نظر گرفته می‌شود.

تعداد عنصر یک آرایه:

اگر آرایه به صورت یک با چند بعدی تعریف شده باشد، تعداد عنصر آن برابر است با حاصلضرب زیر:
 $(\text{تعداد عنصر بعد ۱ام}) \times \dots \times (\text{تعداد عنصر بعد دوم}) \times (\text{تعداد عنصر بعد اول}) = \text{تعداد عنصر آرایه}$

یادآوری

$$\text{تعداد عنصر بعد ۱ام} = U_i - L_i + 1$$

$$A[L_1 \dots U_1] = \text{تعداد عنصر} = U_1 - L_1 + 1$$

$$A[L_1 \dots U_1, L_2 \dots U_2] = \text{تعداد عنصر} = (U_1 - L_1 + 1)(U_2 - L_2 + 1)$$

$$A[L_1 \dots U_1, L_2 \dots U_2, \dots, L_n \dots U_n] = \text{تعداد عنصر} = (U_1 - L_1 + 1) \times (U_2 - L_2 + 1) \times \dots \times (U_n - L_n + 1)$$

مثال:

$$A(4, 3) = A(1..4, 1..3) = \text{تعداد عناصر آرایه} = (4 - 1 + 1) \times (3 - 1 + 1) = 12$$

$$A(1..4, -1..3, 2..3) = \text{تعداد عناصر آرایه} = (4 - 1 + 1) \times (3 - (-1) + 1) \times (3 - 2 + 1) = 40$$

محاسبه فضای آرایه:

برای محاسبه فضای آرایه کافی است تعداد عناصر آرایه را در فضای نوع عناصر آن ضرب کنیم.

$$\text{فضای نوع آرایه} \times \text{تعداد عناصر آرایه} = \text{فضای آرایه}$$

مثال:

بات ۲

A:Array [-1..3, 2..5] of integer;

$$\text{تعداد عناصر آرایه} = (3 - (-1) + 1) \times (5 - 2 + 1) = 20$$

$$\text{فضای نوع آرایه} \times \text{تعداد عناصر آرایه} = \text{فضای آرایه} = 20 \times 2 = 40 \text{ byte}$$

محاسبه آدرس خانه دلخواه از یک آرایه (مهنم)

مشخص کردن موقعیت عناصر یک آرایه با توجه به نحوه ذخیره‌سازی آنها به یکی از ۲ روش سطري و ستوني متفاوت است.

10	20	30	70
5	15	45	32
12	13	2	24
7	9	8	4
1	2	6	14

مقدار 13 در آرایه (ماتریس)

خانه 10 ام آرایه است = بصورت سطري

خانه 18 ام آرایه است = بصورت ستوني

گاه ممکن است (در ماتریس‌های مربع) وضعیت ستونی و سطري با هم برابر شود.

10	20	30
40	50	60
70	80	90

خانه 15 ام آرایه است = بصورت سطري

مقدار 50 در آرایه (ماتریس)

خانه 5 ام آرایه است = بصورت ستوني

محاسبه آدرس خانه A[i,j,k] در آرایه A[L₁..U₁, L₂..U₂, L₃..U₃]

$\alpha = \text{آدرس شروع آرایه}$ (در صورت عدم بیان آدرس شروع ۰ = α فرض می‌کنیم)

$\beta = \text{فضای نوع عناصر آرایه}$ (در صورت عدم بیان فضای نوع عناصر آرایه ۱ = β فرض می‌کنیم)

حد پایین بعد اول

حد پایین بعد دوم

حد پایین بعد سوم

آدرس سطري

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

از چپ به راست

تعداد عناصر بعد دوم به بعد

تعداد عناصر بعد سوم به بعد

$$\begin{array}{c}
 \text{حد پایین بعد سوم} \\
 \text{آدرس سترنی} \\
 \boxed{A[i,j,k]} \\
 \text{از راست به چپ}
 \end{array}
 \quad
 \begin{array}{c}
 \text{حد پایین بعد دوم} \\
 = \alpha + \left[\begin{array}{c} \uparrow \\ (k-L_3)(U_2-L_2+1)(U_1-L_1+1) + (j-L_2)(U_1-L_1+1) + (i-L_1) \end{array} \right] \times \beta \\
 \text{تعداد عناصر بعد دوم به قبل}
 \end{array}$$

این روش را می‌توان برای حالت ۲ بعدی نیز به راحتی تعمیم داد.

مثال ۱: آدرس $A[i]$ در آرایه $A[L_1..U_1]$

$$\alpha + [(i-L_1)] \times \beta$$

مثال ۲: آدرس $A[i,j]$ در آرایه $A[L_1..U_1, L_2..U_2]$ (چون α و β مشخص نشده، $\alpha = 0$ و $\beta = 1$ در نظر می‌گیریم)

$$A[i,j] = \alpha + [(i-L_1)(U_2-L_2+1) + (j-L_2)] \times \beta = (i-L_1)(U_2-L_2+1) + (j-L_2)$$

$$A[i,j] = \alpha + [(j-L_2)(U_1-L_1+1) + (i-L_1)] \times \beta = (j-L_2)(U_1-L_1+1) + (i-L_1)$$

مثال ۳: آدرس $A[i,j,k]$ در آرایه $A[a,b,c]$ کدام است؟ (سطری و سترنی)

چون حد پایین بعدها مشخص نشده، $A[a,b,c]$ را بصورت $A[1..a, 1..b, 1..c]$ فرض می‌کیم

همچنین چون α و β یاف نشده به ترتیب $0 = \alpha$ و $1 = \beta$ در نظر می‌گیریم.

$$\begin{aligned}
 A[i,j,k] &= \alpha + [(i-1) \times (b-1+1)(c-1+1) + (j-1) \times (c-1+1) + (k-1)] \times \beta \\
 &= (i-1)bc + (j-1)c + (k-1)
 \end{aligned}$$

$$\begin{aligned}
 A[i,j,k] &= \alpha + [(k-1)(b-1+1)(a-1+1) + (j-1) \times (a-1+1) + (i-1)] \times \beta \\
 &= (k-1)ab + (j-1)a + (i-1)
 \end{aligned}$$

مثال ۴: آرایه ماتریسی A که 8×30 است را در نظر بگیرید که آدرس اولیه آن $A(1,1)=200$ و تعداد $w=4$ کلمه در حافظه وجود داشته

باشد، مطلوبست آدرس سطری $A[12,4]$ ؟

$$\alpha = 200 \quad \beta = 4 \text{ byte}$$

چون آرایه بصورت $(30,8)$ مطرح شده، آنرا بصورت $A(1..30, 1..8)$ در نظر می‌گیریم و خواهیم داشت:

$$\begin{aligned}
 A[12,4] &= \alpha + [(12-1)(8-1+1) + (4-1)] \times \beta \\
 &= 200 + [11 \times 8 + 3] \times 4 = 564
 \end{aligned}$$

مثال ۵: آرایه $A[1..3, 1..5]$ از آدرس 3000 به بعد ذخیره شده است و هر خانه 4 بایت اشغال کرده است مطلوبست آدرس $A[2,3]$ بصورت

$$\alpha = 3000, \beta = 4$$

$$A[2,3] = 3000 + [(2-1)(5-1+1) + (3-1)] \times 4 = 3028$$

$$A[2,3] = 3000 + [(3-1)(3-1+1) + (2-1)] \times 4 = 3028$$



نکته: گاهی اوقات یک آرایه چند بعدی را به صورت سطّری یا ستونی در یک آرایه یک بعدی ذخیره می‌کنند، در این صورت آدرس خانه مشخص در آرایه چند بعدی به شکل سطّری یا ستونی در آرایه یک بعدی مورد نظر می‌باشد.

مثال: آرایه سه بعدی $M(10, 20, 30)$ را در آرایه یک بعدی $A(1..10 \times 20 \times 30)$ به صورت سطّری ذخیره کرده‌ایم مطلوب است آدرس

$$\text{خانه } M(2, 1, 3) = \alpha$$

$$\text{آدرس شروع آرایه مشخص نشده} = 0$$

$$\text{فضای نوع عناصر آرایه مشخص نشده} = \beta$$

$$\begin{aligned} M(2, 1, 3) &= \alpha + [(2-1)(20-1+1)(30-1+1) + (1-1)(30-1+1) + (3-1)] \times \beta \\ &= 602 \end{aligned}$$

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

برای این که محاسبه کنیم $A[i, j, k]$ چندمین خانه سطّری 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.

$$\begin{aligned} A[i, j, k] &= \left(\alpha + [(i-L_1)(U_2-L_2+1)(U_3-L_3+1) + (j-L_2)(U_3-L_3+1) + (k-L_3)] \times \beta \right) + 1 \\ &= [(i-L_1)(U_2-L_2+1)(U_3-L_3+1) + (j-L_2)(U_3-L_3+1) + (k-L_3)] + 1 \\ A[i, j, k] &= \left(\alpha + [(k-L_3)(U_2-L_2+1)(U_1-L_1+1) + (j-L_2)(U_1-L_1+1) + (i-L_1)] \times \beta \right) + 1 \\ &= [(k-L_3)(U_2-L_2+1)(U_1-L_1+1) + (j-L_2)(U_1-L_1+1) + (i-L_1)] + 1 \end{aligned}$$

مثال ۱: در آرایه $A['b', 3], A['a' .. 'e', 1 .. 4]$ چندمین خانه آرایه است؟

چون وضعیت سطّری یا ستونی مشخص نشده است پیش فرض به صورت سطّری عمل می‌کنیم در ضمن فاصله 'e' .. 'a' را به شکل ۵ .. ۱

در نظر می‌گیریم بنابراین وضعیت $A[2, 3]$ در $A[1 .. 5, 1 .. 4]$ مدنظر است.

$$\begin{aligned} A[2, 3] &= (0 + [(2-1)(4-1+1) + (3-1)] \times 1) + 1 \\ &= 7 \end{aligned}$$

خانه ۷ام به صورت سطّری است

'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



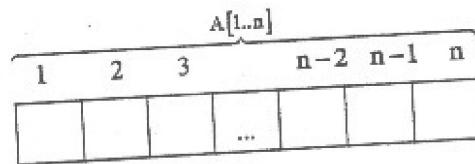
جستجو در آرایه‌ها:

که به ظور کلی برای جستجوی یک عنصر دلخواه در آرایه‌ها ۲ روش اساسی وجود دارد:

- روش جستجوی خطی (ترتبی)
- روش جستجوی دودوئی (باینری)

۱- روش جستجوی خطی (ترتبی)

در این روش برای جستجوی داده x در آرایه $A[1..n]$ ، جستجو را از یکی از دو طرف آرایه (یعنی فرض از ابتداء) آغاز می‌کیم و داده مورد نظر را پشت سر هم با عناصر آرایه مقایسه می‌کنیم، تا این که یا داده مورد نظر پیدا شود یا به طرف دیگر آرایه (نهایی آرایه) برسیم.



الگوریتم جستجوی خطی داده x در آرایه $A[1..n]$ تابی:

```
flag := false;
For i = 1 to n do
    if (A[i] = x) then (یافته و محل آن است)
        x
```

```
begin
    Flag := True;
    break;
end;
if Flag = True then
    write ('Location is.', i);
else
    write ('not found');
```

در صورتی که شرط if برقرار شد ($x = A[i]$) عنصر x یافت شده و محل آن است در نتیجه $Flag$ شده از حلقه خارج می‌شویم.
نکته ۱: در روش جستجوی خطی چون هیچ استراتژی خاصی جز جستجوی پشت سر هم از ابتداء تا انتها وجود ندارد، در نتیجه $sort$ بودن یا

تعداد مقایسات و مرتبه اجرائی در الگوریتم جستجوی خطی مهمترین پارامتر، مقایسه است، در نتیجه تعداد مقایسات مهمترین حامل در محاسبه مرتبه اجرائی الگوریتم جستجوی خطی به حساب می‌آید.

مرتبه اجرائی و زمان جستجوی خطی مربوط به حالت متوسط است.

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

در نتیجه: تعداد مقایسه: $1 - \text{مرتبه اجرائی} (O(1))$

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

وضعیت مقایسات در جستجوی خطی

در نتیجه: تعداد مقایسه: $n - \text{مرتبه اجرائی} (O(n))$

مجموع مقایسات
ج) حالت متوسط: متوسط مقایسات برابر است با:
تعداد عنصر 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 باشد با n مقایسه بدست می‌آید.

در نتیجه:

$$\text{مجموع مقایسات} = 1 + 2 + 3 + \dots + n = \frac{n(n+1)}{2}$$

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

$$\text{مرتبه اجرائی متوسط} = O(n)$$

نکته مهم: زمان جستجوی خطی یا به عبارت بهتر مرتبه اجرائی آن مربوط به حالت متوسط بوده و برابر با $O(n)$ است.

نتیجه گیری جستجوی خطی:

- حداقل مقایسه: 1 مقایسه
- متوسط مقایسهها: $\frac{n+1}{2}$
- زمان (مرتبه اجرائی): $O(n)$
- حداقل مقایسه: n مقایسه

: (binary search)

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

روش جستجو:

داده مورد جستجو (x) را با خانه میانی $A[\text{mid}]$ آرایه مقایسه می‌کنیم، در صورتی که با آن خانه برابر باشد جستجو پایان می‌یابد در غیر این صورت در صورتی که $A[\text{mid}] > x$ باشد به نیمه بالای آرایه می‌رویم و در صورتی که $A[\text{mid}] < x$ 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 بررسیم که محل آن mid خواهد بود یا این که داده x در آرایه وجود ندارد که در این صورت Low (اندیس پائین نیمه آرایه) از high (اندیس بالای نیمه آرایه) بیشتر می‌شود.



الگوریتم جستجوی دودویی:

مفروضات:

داده جستجو شونده = x پیش فرض $False$ و در صورت پیدا شدن x ، $True$ می شود = Flag

اندیس پایین آرایه = Low

اندیس بالای آرایه = High

آرایه $A[1..n]$ مرتب صعودی =

$$mid = \left\lfloor \frac{low + high}{2} \right\rfloor =$$

اندیس وسط آرایه که عامل مقایسه x با $A[mid]$ است.

I) الگوریتم غیر بازگشتی:

Low := 1;

high := n;

Flag := False;

while (low <= high) AND (Flag <> True) do

begin

mid := (low + high) Div 2;

if ($A[mid] = x$) پیدا شده محل آن mid است (

Flag := True

else if ($A[mid] > x$) باید به نیمه پایین برویم (

High := mid - 1

else if ($A[mid] < x$) باید به نیمه بالا برویم (

Low := mid + 1

end;

II) الگوریتم بازگشتی:

```

procedure binary search (a: elementary; x;element; var Left , Right , j:integer);
var
mid: integer;
begin
if (Left <= Right) then
begin
mid:= compare (x , a[mid]) of
'>' : binary search (a , x , mid + 1 , Right , j);
'<' : binary search (a , x , Left , mid - 1,j);
'= ' : j:= mid;
end;
end;

```

مثال:

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

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

مقدار $mid = 1$ است. گرفته بنا بر این محل عنصر x True Flag

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

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)			

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

نکته مهم: در الگوریتم های جستجو دودوئی:

الف) تعداد مقایسات برای یافتن دو عدد متفاوت با نتیجه موفق ممکن است متفاوت باشد.

ب) تعداد مقایسات برای نیافتن دو عدد متفاوت با نتیجه شکست ممکن است متفاوت باشد.

1	2	3	4	5	6	7	8
10	15	20	25	30	45	70	80

به طور مثال برای رسیدن به عدد 70 نیاز به 3 مقایسه و برای رسیدن به عدد 15 نیاز به 2 مقایسه داریم، همین امر نیز برای دو نتیجه ناموفق نیز وجود دارد.

تعداد مقایسات و مرتبه اجرائی

در الگوریتم جستجوی بازگشتی مهمترین پارامتر، مقایسه است، در نتیجه تعداد مقایسه مهمترین عامل در محاسبه مرتبه اجرائی الگوریتم

جستجوی بازگشتی به حساب می آید.

■ مرتبه اجرائی و زمان جستجوی بازگشتی مربوط به حالت متوسط است.

وضعیت مقایسات در جستجوی دودوئی

(الف) بهترین حالت: عنصر در وسط لیست باشد.

در نتیجه: حداقل مقایسه - تعداد مقایسه: ۱ - مرتبه اجرائی: $O(1)$

(ب) بدترین حالت: می‌تواند در دو انتهای هر نیمه وجود داشته باشد.

در نتیجه: حداکثر مقایسه - تعداد مقایسه: $O(n) = \left\lfloor \log_2 n \right\rfloor + 1$ - مرتبه اجرائی: $O(n)$

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

مقایسات

۱ = حداقل مقایسه

۳ = $\left\lfloor \log_2 6 \right\rfloor + 1 = 3$ = حداکثر مقایسه

1	2	3	4	5	6	7	8	9
3	2	3	4	1	3	2	3	4

مقایسات

۱ = حداقل مقایسه

= $\left\lfloor \log_2 9 \right\rfloor + 1 = 4$ = حداکثر مقایسه

(ج) حالت متوسط

برای محاسبه متوسط تعداد مقایسات در روش جستجوی دودوئی ۲ روش وجود دارد.

$$\text{I) متوسط تعداد مقایسه‌ها} = \frac{\text{مجموع مقایسات}}{\text{تعداد عناصر آرایه}}$$

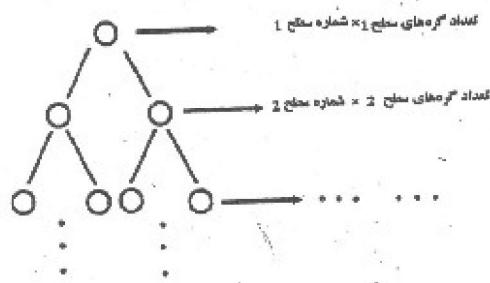
1	2	3	4	5	6	7	8	9
3	2	3	4	1	3	2	3	4

مقایسات

$$\text{متوسط مقایسه} = \frac{3 + 2 + 3 + 4 + 1 + 3 + 2 + 3 + 4}{9} = \frac{25}{9}$$

(II) یک درخت دودوئی به تعداد عناصر آرایه تشکیل می‌دهیم و به شکل زیر حمل می‌کنیم.

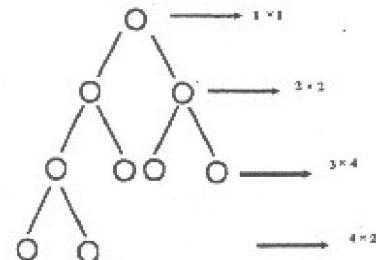
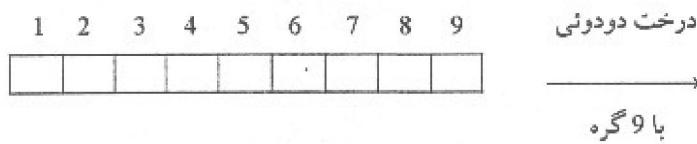
■ سطح ریشه ۱۱ فرض می‌کنیم.



■ متوسط مقایسه‌ها برآورده است با:

$$\text{متوسط مقایسه} = \frac{\sum \text{تعداد گره‌های سطح } n \times \text{شماره سطح } n}{\text{تعداد عناصر آرایه}}$$

ساختمان دادهها



$$\text{متوسط مقایسات} = \frac{1 \times 1 + 2 \times 2 + 3 \times 4 + 4 \times 2}{9} = \frac{25}{9}$$

■ زمان یا مرتبه اجرایی جستجوی دودوئی در حالت متوسط $O(\log_2^n)$ است.

تکنیک: زمان جستجوی دودوئی یا به عبارت بهتر مرتبه اجرایی آن مربوط به حالت متوسط بوده و برابر با $O(\log_2^n)$ است.

مثال: اگر آرایه‌ای مرتب از اعداد صحیح ۱ تا ۱۰۲۴ باشد، الگوریتم جستجوی دودوئی با چند بار تکرار عدد ۴ را پیدا می‌کند؟

15 (۴)

9 (۳)

7 (۲)

1 (۱)

بار اول با مقایسه عدد ۴ با وسط آرایه مترجمه می‌شویم که عدد مذکور باید مابین خانه‌های ۱ تا ۵۱۲ یعنی نیمه پائین آرایه باشد به همین ترتیب:

تعداد تکرار الگوریتم	عدد ۴ در گدام محدود است؟
1	۱ تا ۵۱۲
2	۱ تا ۲۵۶
3	۱ تا ۱۲۸
4	۱ تا ۶۴
5	۱ تا ۳۲
6	۱ تا ۱۶
7	۱ تا ۸

پس از ۷ بار تکرار الگوریتم آرایه زیر در نظر گرفته می‌شود:

Low	mid	high
1	2	3

$$mid = \left\lfloor \frac{1+8}{2} \right\rfloor = 4$$

بار هشتم هنگامی که عدد ۴ با محتواي mid مقایسه می‌شود، پیدامی گردد پس الگوریتم 8 بار تکرار می‌شود.

نتیجه‌گیری جستجوی دودوئی:

- حداقل مقایسه: 1 مقایسه

- حداقل مقایسه: $\lfloor \log_2^n \rfloor + 1$

- زمان (مرتبه اجرایی) جستجوی دودوئی: $O(\log_2^n)$

نتیجه‌گیری انواع جستجوها:

روش جستجو	زمان (متریه اجرایی)	حداقل مقایسه	حداکثر مقایسه	وضعیت آرایه
خطی (تریس)	$O(n)$	1 مقایسه	n مقایسه	مرتب یا نامرتب
باپرسی (دودوئی)	$O(\log_2^n)$	1 مقایسه	$\lfloor \log_2^n \rfloor + 1$	مرتب

دیده می‌شود که روش جستجوی دودوئی به مرتب از روش جستجوی خطی از نظر تعداد مقایسه بهتر و مقرن به صرفه‌تر است.

ماتریس‌ها

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

ماتریس‌های مهم
۱- ماتریس مریع:

به هر ماتریس $n \times n$ با n^2 خانه ماتریس مریع گفته می‌شود.

در هر ماتریس مریع $n \times n$ ماتند A روابط زیر برای اندیس خانه $[i, j]$ وجود دارد:

$$\begin{array}{ll} i < j & A[i, j] \text{ بالای قطر اصلی است} \\ i = j & A[i, j] \text{ روی قطر اصلی است} \\ i > j & A[i, j] \text{ پائین قطر اصلی است} \end{array} \quad \begin{array}{ll} i + j < n + 1 & A[i, j] \text{ بالای قطر فرعی است} \\ i + j = n + 1 & A[i, j] \text{ روی قطر فرعی است} \\ i + j > n + 1 & A[i, j] \text{ پائین قطر فرعی است} \end{array}$$

(i = j) \quad (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

۲- ماتریس بالا مثلث و پائین مثلث

الف) اگر عنصر زیر قطر اصلی 0 باشد ماتریس بالا مثلث است.

در هر ماتریس مریع $n \times n$

ب) اگر عنصر بالای قطر اصلی 0 باشد ماتریس پائین مثلث 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) پائین مثلث

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

(ب) بالا مثلث

شکل ۲ - ۱ : ماتریس‌های بالا مثلثی و پائین مثلثی

نکته مهم: در هر ماتریس بالا مثلث یا پایین مثلث $n \times n$:

$$\text{تعداد کل خانه‌ها} = n^2$$

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

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

n	x	x	...	x	x
+ n - 1	0	x	...	x	x
+ n - 2	0	0	x	x	x
+ :	0	0	0	x	x
	1	0	0	0	x
$\frac{n(n+1)}{2}$					

(تعداد خانه‌های مخالف صفر)

۳- ماتریس L قطری:

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

x	0	0
0	x	0
0	0	x

ماتریس 1 قطری = ماتریس قطری

x	x	0
x	x	x
0	x	x

ماتریس 3 قطری

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 A \times B باید سطون ماتریس A با سطر ماتریس B یکسان باشد.

$$\text{در نتیجه: } C_{m \times k} \leftarrow A_{m \times n} \times B_{n \times k}$$

□ خواص ضرب ماتریس‌ها:

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

ب) ضرب ماتریس‌ها خاصیت شرکت‌پذیری دارد.



حالات شرکت پذیری ۳ ماتریس

C, B, A

۱) A(BC)

۲) (AB)C

حالات شرکت پذیری ۴ ماتریس

D, C, B, A

۱) (AB)(CD)

۲) ((A(BC))D)

۳) (A((BC)D))

۴) (((AB)C)D)

۵) (A(B(CD)))

■ محاسبه تعداد حالات شرکت پذیری ضرب ماتریس‌ها:

بنابر قبیله کاتالان تعداد حالات شرکت پذیری ضرب $n+1$ ماتریس از رابطه زیر بدست می‌آید:

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

مثال: تعداد حالات شرکت پذیری ۴ ماتریس کدام است؟

$$n+1=4 \Rightarrow n=3$$

$$=\frac{\binom{6}{3}}{3+1}=5$$

لکته:

هر گاه یک ماتریس بالا مثلث را در ماتریس بالا مثلث دیگری ضرب کیم، حاصل ماتریس بالا مثلث است.

هر گاه یک ماتریس پائین مثلث را در ماتریس پائین مثلث دیگری ضرب کیم، حاصل ماتریس پائین مثلث است.

هر گاه یک ماتریس قطری را در ماتریس قطری دیگری ضرب کیم، حاصل ماتریس قطری است.

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

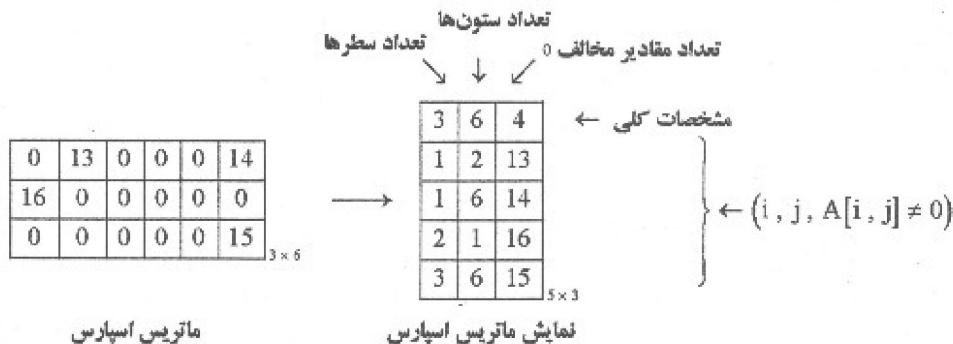
ماتریس اسپارس (ثنک - پر اکنده - خلوت - sparse)

به هر ماتریس $m \times n$ که تعداد خانه‌های 0 و یا بدون ارزش (nonvalue) آن بیشتر از تعداد خانه‌های مختلف 0 آن باشد. ماتریس اسپارس می‌گویند.

■ روش عمومی برای نمایش ماتریس اسپارس: (ذخیره مختصات خانه‌های مختلف 0)

در صورتیکه ماتریس $m \times n$ اسپارس ۳ خانه مختلف 0 داشته باشد، برای نمایش آن از یک آرایه دو بعدی با $3 + 1 + 2$ سطر استفاده می‌شود، به شرح زیر:الف) سطر اول به ترتیب از چپ به راست، m (تعداد سطر) و n (تعداد ستون)، ۰ (تعداد خانه‌های مختلف 0) گذشته می‌شود.ب) از سطر دوم به بعد، هر سطر شامل سه مؤلفه است که همان مختصات و مقدار خانه‌های مختلف 0 است، یعنی: $(i, j, A[i, j] \neq 0)$

لکته: این روش یا عث صرفه جوئی در حافظه می شود چون فقط مختصات خانه های مخالف ۰ نگهداری می شود.



 چون خانه‌های مخالف ۰، ۴ تا است، بنابراین ماتریس اسپارس شامل ۵ سطر و ۳ ستون خواهد بود.

ترانهاده کردن ماتریس (transpose) `tsparse`

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.

$$\begin{array}{|c|c|c|} \hline 3 & 6 & 4 \\ \hline 1 & 2 & 13 \\ \hline 1 & 6 & 14 \\ \hline 2 & 1 & 16 \\ \hline 3 & 6 & 15 \\ \hline \end{array} \xrightarrow{\text{ترانهاده}} \begin{array}{|c|c|c|} \hline 6 & 3 & 4 \\ \hline 1 & 2 & 16 \\ \hline 2 & 1 & 13 \\ \hline 6 & 1 & 14 \\ \hline 6 & 3 & 15 \\ \hline \end{array}$$

نکته مهم: بعضی ماتریس‌ها ماتریس‌های بالا مثبت و پائین مثبت، ماتریس اسپارس نیستند، اما زمانی که بعدهای آن‌ها زیاد شود، تعداد صفرهای ماتریس قابل توجه خواهد بود پرای همین از روش زیر برای تعابیر آن‌ها استفاده می‌کنند:

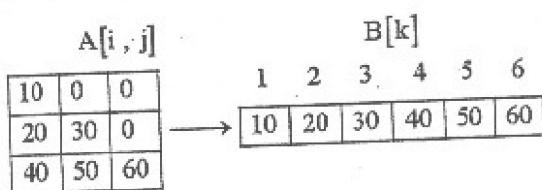
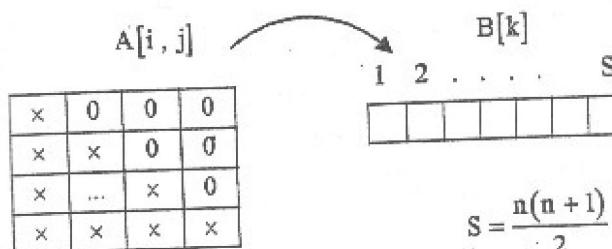
- ۲- یک آرایه باقی مول که محا مقدار مخالف صفر ماتریس را دارد، آرایه مشخص کند.

مثال: ماتریس را شنید $n \times n$ را در نظر می‌گیریم.

- ۱- در این ماتریس $\frac{n(n+1)}{2}$ خانه مخالف صفر وجود دارد، برای همین یک آرایه یک بعدی به تعداد خانه‌های مخالف صفر اختیار می‌کنیم.

۲- برای مرخانه مخالف صفر $[j, i]$ A در ماتریس پائین مثلث، رابطه زیر محل آن خانه را در آرایه یک بعدی B مشخص می کند. $(B[k])$

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



به طور مثال $A[3, 2] = 50$ در آرایه یک بعدی $[5]$ است و این ارتباط از طریق رابطه زیر فراهم می‌شود.

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

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

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

مثال: برای هر خانه مخالف صفر $[i, j]$ در ماتریس بالا مثلاً $n \times n$, رابطه زیر محل آن خانه را در آرایه یک بعدی $B[k]$ مشخص می‌کند.

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

نتیجه گیری:

یعنی دو روش بیان شده

۱- روش عمومی نمایش ماتریس اسپارس (نگهداری مختصات مقادیر مخالف صفر با استفاده از آرایه ۲ بعدی)

۲- نگهداری مقادیر مخالف صفر در آرایه یک بعدی که تعداد عناصر آن به تعداد خانه‌های مخالف ۰ است.

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

لکنه مهم: حاصلضرب، جمع و تفریق ماتریس‌های sparse ممکن است که sparse نباشد.

اسپارس نیست.

$$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}$$

اسپارس نیست.

$$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} - \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ 0 & 0 \end{bmatrix}$$

اسپارس نیست.

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

محاسبه تعداد ضرب‌های حاصلضرب چند ماتریس:

اگر حاصلضرب دو ماتریس $C_{m \times k} = A_{m \times n} \times B_{n \times k}$ را در نظر بگیریم، قطعاً برنامه زیر این عمل را انجام می‌دهد.

```
for i := 1 to m do
for j := 1 to k do
for L := 1 to n do
C[i, j] := C[i, j] + A[i, L] * B[L, j];
```

■ تعداد ضرب‌های انجام شده به تعداد تکرار $3^{\text{ حلقه}} \times m \times n \times k$ بوده و برابر است با:

■ تعداد جمع‌های انجام شده به تعداد تکرار $3^{\text{ حلقه}} \times m \times n \times k$ بوده و برابر است با:

با این اگر دو ماتریس $A_{m \times n}$ و $B_{n \times k}$ را در هم ضرب کنیم تعداد جمع‌ها $m(n-1)k$ می‌شود اما چون در کامپیوتر این جمع‌ها باید در ماتریس $C_{m \times k}$ که ماتریس حاصل از ضرب دو ماتریس است، قرار گیرد، یک جمع نیز به ازاء هر خانه $C_{m \times k}$ اضافه می‌شود که در نهایت $m(n-1)k + mk = m(n-1)k + mnk$ جمع خواهیم داشت که همان mnk است. (قبل از انجام ضرب تمام خانه‌های ماتریس $C_{m \times k}$ را ۰ قرار می‌دهیم).

مثال (I): تعداد ضرب‌های حاصلضرب ۳ ماتریس $A_{3 \times 10}$, $B_{10 \times 5}$, $C_{5 \times 4}$ با توجه به حالت‌های مختلف شرکت‌پذیری کدام است؟

$$A_{3 \times 10} (BC)_{10 \times 4} = (3 \times 10 \times 4) + (10 \times 5 \times 4) = 120 + 200 = 320$$

$$(AB)_{3 \times 5} C_{5 \times 4} = (3 \times 10 \times 5) + (3 \times 5 \times 4) = 150 + 60 = 210$$

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: برای محاسبه ضرب ۳ ماتریس $A_{m \times n} \times B_{n \times k} \times C_{k \times l}$ برای آن که:

$$A(BC) \longleftrightarrow \frac{1}{m} + \frac{1}{k} > \frac{1}{n} + \frac{1}{l}$$

طبق یک قاعده سرانگشتی می‌توان گفت که هنگام ضرب چند ماتریس، اگر ماتریس‌هایی را زودتر ضرب کنیم که بعد وسط آن‌ها بزرگ‌تر و بعد کناری آن‌ها به نسبت کوچک‌تر است، بدین ترتیب تعداد ضرب‌ها کوچک‌تر می‌شود. البته این قاعده همیشه درست نیست در عین حال قضیه بیان شده همیشه درست عمل می‌کند.

مثال:

(II) تعداد ضرب‌های حاصلضرب ۳ ماتریس $A_{3 \times 5}$, $B_{5 \times 4}$ و $C_{4 \times 2}$ با توجه به حالت‌های مختلف شرکت‌پذیری کدام است؟

$$(AB)_{3 \times 4} C_{4 \times 2} = (3 \times 5 \times 4) + (3 \times 4 \times 2) = 60 + 24 = 84$$

$$A_{3 \times 5} (BC)_{5 \times 2} = (3 \times 5 \times 2) + (5 \times 4 \times 2) = 30 + 40 = 70$$

تحلیل:

در مثال (I)

$$A_{3 \times 10} \times B_{10 \times 5} \times C_{5 \times 4}$$

چون $\frac{1}{3} + \frac{1}{5} > \frac{1}{10} + \frac{1}{4}$ در نتیجه تعداد ضربهای $C(AB)$ کمتر از تعداد ضربهای $A(BC)$ است، در اینجا بعد رسم AB بزرگر بود.

در مثال (II)

$$A_{3 \times 3} \times B_{5 \times 4} \times C_{4 \times 2}$$

چون $\frac{1}{5} + \frac{1}{4} > \frac{1}{3} + \frac{1}{2}$ در نتیجه تعداد ضربهای $A(BC)$ کمتر از تعداد ضربهای $C(AB)$ است. در اینجا با این‌که بعد رسم AB بزرگر از BC بود اما تعداد ضرب $C(AB)$ بیشتر از $A(BC)$ است.

ولی قضیه همیشه درست عمل می‌کند.

نتقه مهم: اگر در صورت یک مسئله برای ضرب چند ماتریس، بهیه بودن (تعداد ضرب کمتر - سرعت اجرا) مطرح نباشد، ماتریس‌ها را از سمت چپ به راست پشت سرهم ضرب می‌کیم.

به طورمثال در ضرب ماتریس‌های $A_{m \times n} \times B_{n \times k} \times C_{k \times l}$ پیش فرض حاصل ضرب رابه صورت $C(AB)$ در نظر می‌گیریم.

مجموعه سوالات

۱ - آرایه دو بعدی زیر در آدرس 3000 به بعد حافظه قرار دارد، آدرس عنصر $[2, 8]$ به روش سطrix کدام است؟

List : Array[-3 .. 4, 5 .. 20] of real;

۶ بایت

3498 (۴)

3258 (۳)

3240 (۲)

3114 (۱)

۲ - بهترین راه نمایش مسیر در مساله Mazing (مسیر پریوج و خم) کدام است؟

(۴) لیست پیوندی

(۳) آرایه یک بعدی

(۲) آرایه یک بعدی

(۱) آرایه $n \times 3$

۳ - تعداد عناصر خیز صفر در یک ماتریس بالامثلی و یا پائین مثلثی کدام است؟

$\frac{n^2 + 2n}{2}$ (۴)

$n(n+1)$ (۳)

$\frac{n(n-1)}{2}$ (۲)

$\frac{n(n+1)}{2}$ (۱)

۴ - تعداد عناصر صفر در یک ماتریس بالا مثلث و یا پائین مثلثی کدام است؟

(۴) هیچ کدام

$n(n+1)$ (۳)

$\frac{n(n-1)}{2}$ (۲)

$\frac{n(n+1)}{2}$ (۱)

۵ - برای نمایش یک ماتریس بالا یا پائین مثلثی در یک آرایه یک بعدی به حداقل چند خانه احتیاج داریم؟

$\frac{n^2 + 2n}{2}$ (۴)

n^2 (۳)

$\frac{n(n+1)}{2}$ (۲)

$n(n+1)$ (۱)

۶ - برای حذف عنصر K از یک آرایه N عنصری چند جابجایی لازم است؟

$N - K + 1$ (۴)

$N - K$ (۳)

K (۲)

$N - K - 1$ (۱)

۷ - در ضرب سه آرایه (۴, ۳, ۲) به ترتیب $C(6, 2), B(4, 6), A(3, 4)$ چند عمل ضرب نیاز است؟

3456 (۴)

2592 (۳)

108 (۲)

25 (۱)

۸ - برای ضرب ۵ ماتریس چند حالت مختلف برای ضرب در شرکت پذیری وجود دارد؟

(۴) هیچ کدام

36 (۳)

14 (۲)

24 (۱)

۹ - آرایه ماتریسی A که 4×30 است را در نظر بگیرید. فرض کنید آدرس اولیه $A(1,1) = 200$ و تعداد $w = 4$ کلمه در حافظه وجود داشته باشد فرض هم بر این است که زیان برنامه سازی آرایه دو بعدی را با روش سطrix ذخیره می کند آدرس $[12, 6] A$ کدام است؟

(۸۲) سراسری

844 (۴)

396 (۳)

356 (۲)

196 (۱)

۱۰ - $A[M', 15]$ چندین عنصر از آرایه تعریف شده زیر محسوب می شود؟

A : Array['K'..'N', 11..50]

125 (۴)

85 (۳)

23 (۲)

19 (۱)



۱۱ - هدف از فشرده کردن آرایه‌ها چیست؟

(۱) نامحدود کردن آرایه

(۲) دسترسی سریع تر به عناصر

(۳) صرفه‌جویی در حافظه

(۴) کوچک شدن آرایه

۱۲ - کدامیک از روش‌های زیر برای نمایش یک ماتریس اسپارس از نظر صرفه‌جویی در حافظه بهتر عمل می‌کند؟

(۱) لیست پیوندی

(۲) ذخیره مختصات

(۳)

ذخیره مختصات با اشاره گر

(۴) الگوی دودویی

۱۳ - آرایه سبعدهی $M[1..a, 1..b, 1..c]$ در یک آرایه یک بعدی $N[1..a \times b \times c]$ به روش سه‌تایی ذخیره شده است. آدرس عنصر $M[i, j, k]$ در آرایه N کدام است؟

$a \times b \times c + b \times c \times 1$ (۱)

$i \times b \times c + j \times c \times k$ (۲)

$(i-1)bc + (j-1)c + (k-1)$ (۳)

$(k-1)ab + (j-1)a + (i-1)$ (۴)

۱۴ - ماتریس اسپارس را توسط کدام گزینه می‌توان تعریف کرد؟ (۱) \max تعداد عناصر غیر صفر

$S[0..\max, 2..3]$ (۱)

$S[1..\max, 1..\max]$ (۲)

$S[0..\max, 1..3]$ (۳)

$S[0..\max, 0..3]$ (۴)

۱۵ - چندین عنصر از آرایه تعریف شده زیر محاسب می‌شود؟

A : array['a'..'z', 10..20]

27 (۱)

24 (۲)

25 (۳)

26 (۴)

۱۶ - در صورتیکه آرایه مورد جستجو در جستجوی دودویی به صورت ۷، ۶، ۵، ۴، ۳، ۲، ۱، ۰ باشد متوجه تعداد مقایسه‌ها برای جستجوی موفق چیست؟

(۱) هیچ‌کدام

$\frac{31}{9}$ (۲)

$\frac{25}{9}$ (۳)

$\frac{27}{9}$ (۴)

۱۷ - در یک آرایه n عدد به ترتیب فردی یا صعودی مرتب قرار دارد، اگر از روش جستجوی دودویی (باینری) برای یافتن حدی استناده کنیم، حداقل تعداد مقایسه چقدر خواهد بود؟

$n - \log_2^n$ (۱)

$\log_2^{(n-1)}$ (۲)

$[\log_2^n] + 1$ (۳)

$n \log_2^n$ (۴)

۱۸ - در یک آرایه n تابی مرتب شده (نژولی - صعودی) یا نامرتب حداقل تعداد جستجو در حالت جستجوی خطی (ترتیبی) کدام است؟

n^2 (۱)

$n/2$ (۲)

$n - 1$ (۳)

n (۴)

(۱) هیچ‌کدام

(۲) تعریف نشده

$[\log_2^n] + 1$ (۳)

\log_2^n (۴)

۱۹ - در یک آرایه تابی نامرتب، حداقل تعداد مقایسه برای جستجوی موفق چیست؟

n^2 (۱)

\log_2^n (۲)

$n/2$ (۳)

$\frac{n+1}{2}$ (۴)

۲۰ - متوسط تعداد مقایسه در جستجوی خطی یک آرایه n حصری کدام است؟

(۱) هیچ‌کدام

$O(n^2), O(\log_2^n)$ (۲)

$O(\log_2^n), O(n)$ (۳)

$O(n^2), O(n)$ (۴)

۲۱ - زمان جستجوی خطی و باینری در یک لیست n تابی از راست به چپ کدام است؟

۲۲ - آدرس عنصر $[x]_A$ از یک آرایه یکبعدی که به صورت $\text{array}[m_1..m_2]$ و از جنس real تعریف می‌شود کدام است؟ (فرض کنید آدرس شروع H باشد.)

$$H + (x - m_2 - m_1) * 6 \quad (۱) \quad H + (x - m_2) * 6 \quad (۲) \quad H + (m_1 - m_2 - x) * 6 \quad (۳) \quad H + (x - m_1) * 6 \quad (۴)$$

۲۳ - آرایه یکبعدی $[B]_{l.m \times n \times p}$ در یک آرایه یکبعدی $A[l..m, l..n, l..p]$ به روش سطر به سطر ذخیره شده است. آدرس عنصر $A[i,j,k]$ در B کدام است؟ (دولتی ۸۰)

$$(i-1)np + (j-1)p + (k-1) \quad (۱) \quad (i-1)np + (j-1)m + (k-1) \quad (۲) \\ imp + jp + k \quad (۳) \quad mnp + np + 1 \quad (۴)$$

۲۴ - می خواهیم حاصلضرب $ABCD$ را بدست آوریم، به طوری که کمترین تعداد عمل ضرب را داشته باشد، ترتیب ضرب ماتریس‌ها کدام است؟

$$A_{13 \times 5} B_{5 \times 89} C_{89 \times 3} D_{3 \times 34}$$

$$\text{۱) } (AB)(CD) \quad (۲) \quad A((BC)D) \quad (۳) \quad ((AB)C)D \quad (۴) \quad \text{هیچکدام}$$

۲۵ - آرایه دو بعدی $[X]_{[4..10, 3..33]}$ در آدرس 400 به بعد حافظه قرار دارد و هر خانه آرایه احتیاج به 4 بایت دارد. آدرس عنصر $X[4, 10, 33]$ به روش ستونی کدام است؟

$$744 \quad (۱) \quad 844 \quad (۲) \quad 1544 \quad (۳) \quad 1444 \quad (۴)$$

۲۶ - دستور حذف شده برنامه جستجوی دودوئی زیر کدام است؟

```
procedure Binsearch (a: elementarray; x: element; var left, Right, j: integer)
```

```
  Vav
```

```
    middle : integer;
```

```
  begin
```

```
    if (Left <= right) then
```

```
      begin
```

```
        middle := (right + Left) Div 2;
```

```
        Case Compare (x , a [middle]) of
```

```
          '>' : Binsearch (a , x , middle +1 , right , j);
```

```
          '=>' : Binsearch (a , x , middle , right , j);
```

```
          '<=' : j := middle;
```

```
      end;
```

```
    end;
```

```
    Binsearch (a , x , middle -1 , left , j);
```

```
Binsearch(a , x , middle - 1 , right , j);
```

هیچکدام

Binsearch (a , x , left , middle -1 , j);

(۱) در لیست ترتیب داده‌ها مهم است.

۲۷ - تفاوت لیست با مجموعه کدام است؟

(۱) در لیست داده تکراری می‌تواند وجود داشته باشد.

۲ و ۱

(۲) در لیست داده‌ها به ترتیب صعودی یا نزولی داشته باشد.

۲۸ - اگر آرایه $A[1..m, 1..n]$ در آرایه $B[1..m \times n]$ به روش سطربه سطر ذخیره شده باشد، آدرس عنصر $[j, i]_A$ در آرایه B کدام است؟

$$(j-1)m+i \quad (2)$$

$$(i-1) m+j \quad (1)$$

$$(j-1) m + i \quad (3)$$

$$(i-1) n + j - 1 \quad (3)$$

۲۹ - آرایه ۳ بعدی $A[1..15, 5..10, 25..200]$ برای ذخیره اعداد صحیح به طول 2 بایت به کار گرفته است. اگر آرایه به صورت سطربی از آدرس 2000 به بعد ذخیره شده باشد آدرس عنصر $[5, 2, 15]_A$ چیست؟

$$3642 \quad (4)$$

$$3420 \quad (3)$$

$$3370 \quad (2)$$

$$3868 \quad (1)$$

۳۰ - کدام روش برای ذخیره ماتریس های پائین مثلثی مناسب تر می باشد؟

$$(4) \text{ ماتریس دو بعدی}$$

$$(1) \text{ ماتریس خلوت}$$

$$(3) \checkmark \text{ آرایه یک بعدی}$$

$$(4) \text{ لیست پیوندی یک طرفه یا در طرفه}$$

۳۱ - فرض کنید یک آرایه 200 عنصری مرتب شده باشد. زمان اجرای بدترین قابع برای پیدا کردن عنصر معلوم x در آرایه A با استفاده از جستجوی دودوئی (باینری) چیست؟

$$12 \quad (4)$$

$$7 \quad (3)$$

$$8 \quad (2)$$

$$200 \quad (1)$$

۳۲ - یک ماتریس بالا مثلث A را با یک آرایه یک بعدی B نمایش داده ایم. اگر عنصر $[i, j]_A$ معادل عنصر $[k]_B$ باشد، بین i, j, k رابطه ای وجود دارد؟

$$\frac{n+1}{2} \quad (4) \text{ همچنان}$$

$$(i-1)\left(n-\frac{1}{2}\right)+j \quad (3)$$

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

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

۳۳ - می نیسم و ما کزیم اعداد ذخیره شده در یک آرایه یک بعدی n خانه، با چند مقایسه یعنی اعداد ذخیره شده در خانه ها بدست خواهد آمد؟

$$\frac{n+1}{2} \quad (4)$$

$$\frac{n}{2} \quad (3)$$

$$\frac{3n}{2}-2 \quad (2)$$

$$\frac{3n}{2} \quad (1)$$

۳۴ - برای پیدا کردن یک item مورد نظر در یک لیست مرتب شده شامل 30000 عنصر حداقل چند مقایسه مورد نیاز است؟

$$15 \quad (4)$$

$$30 \quad (3)$$

$$45 \quad (2)$$

$$100 \quad (1)$$

۳۵ - حاصلضرب، جمع، تفریق در ماتریس اسپارس:

(1) ممکن است اسپارس نباشد.

(2) همواره یک ماتریس اسپارس است.

(3) همواره یک ماتریس 0 است

(4) همواره ماتریس خیر اسپارس است.

۳۶ - یک ماتریس پائین مثلث A را با یک آرایه یک بعدی B نمایش داده ایم اگر عنصر $[i, j]_A$ معادل عنصر $[k]_B$ باشد، بین i, j, k رابطه ای وجود دارد؟

$$k = \frac{i(i-j)}{2} \quad (4)$$

$$k = i+j \quad (3)$$

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

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

۳۷ - تابع $A(m, n)$ به شکل رویرو را درنظر بگیرید. حاصل $(A(1, 3))$ کدام است؟ (سراسری ۸۲)

$$A(m, n) = \begin{cases} n+1 & M=0 \\ A(m-1, 1) & N=0 \\ A(m-1, A(m, n-1)) & \text{بقیه حالات} \end{cases}$$

6 (۴)	5 (۳)	4 (۲)	3 (۱)
-------	-------	-------	-------

۳۸ - مجموع مراحل خطوط در برنامه زیر چند است؟

```
float sum (int num [ ], int n)
{
    int i, temp = 0
    for (i = 0 ; i < n ; i++)
        temp += num [i];
    return temp;
}
```

$n+1$ (۴)	$2n+1$ (۲)	$2n+3$ (۱)
-----------	------------	------------

۳۹ - تعداد مراحل پشتگی به n دارد و نامشخص است.

۴۰ - تعداد مراحل کل خطوط در برنامه زیر چقدر است؟

```
float rsum (float list [ ], int n)
{
    if (n)
        return rsum (list, n - 1) + list [n - 1];
    return list [0];
}
```

$2(n-1)^2$ (۴)	$2n^2$ (۳)	$2n+2$ (۲)	$2n+4$ (۱)
----------------	------------	------------	------------

۴۱ - کدامیک از روابط زیر نشان‌دهنده رابطه صحیح زمان محاسبه الگوریتم‌های مختلف است؟

$$o(\log n) < o(n) < o(n \log n) < o(2^n) < o(n^2) \quad (۱)$$

$$o(n) < o(\log n) < o(n \log n) < o(2^n) < o(n^2) \quad (۲)$$

$$o(n) < o(\log n) < o(n \log n) < o(n^2) < o(2^n) \quad (۳)$$

$$o(\log n) < o(n) < o(n \log n) < o(n^2) < o(2^n) \quad (۴)$$

۴۲ - با توجه به تابع رویرو $func(100)$ چه خواهد شد؟

```
int func ( int n )
{
    if (n == 0) return 0;
    return (n + func (n - 1));
}
```

10000 (۴)	5050 (۳)	200 (۲)	199 (۱)
-----------	----------	---------	---------



۴۲ - الگوریتم عبارتست از مراحل که هر مرحله شده باشد.

(۱) حل یک مسئله / بهخوبی تعریف

(۲) نوشتن یک برنامه / بهخوبی تعریف

(۳) نوشتن یک برنامه / از چند دستور تشکیل

(۴) حل یک مسئله / از چند مرحله فرعی تشکیل

۴۳ - با توجه به تعریف تابع مقابله مقدار $L(25)$ چیست؟

$$L(n) = \begin{cases} 0 & \text{if } n=1 \\ L\left(\left[\frac{n}{2}\right]\right) + 1 & \text{if } n>1 \end{cases}$$

7 (۴)

4 (۳)

5 (۲)

6 (۱)

۴۴ - خروجی تابع بازگشتی زیر در زبان پاسکال به ازاء $y=2$ چه می‌شود.

```
function f(y: integer) : integer;
Begin
  if y <= 0 then f := 2
  Else f := f(y - 1) + f(y - 2);
End;
```

2 (۴)

6 (۳)

8 (۲)

4 (۱)

۴۵ - دلیل اصلی استفاده از توابع Recursive در برنامه‌سازی چیست؟

(۱) ساختار تکرار بعضی از ساختمان دادهها

(۲) راحتی تدوین برنامه‌های Recursive و تعداد کم دستورات استفاده شده در آنها

(۳) هزینه کم از نظر منبع کامپیوتر

(۴) همه انتخاب‌ها.

۴۶ - فرض کید که a, b اعداد صحیح مثبت بتواند و تابع Q به صورت زیر به شکل بازگشتی تعریف شده باشد. مقدادیر $Q(2, 3), Q(14, 3)$ را پیدا کید.

$$Q(a,b) = \begin{cases} 0 & \text{if } a < b \\ Q(a-b, b) + 1 & \text{if } b \leq a \end{cases}$$

$Q(2, 3) = 0, Q(14, 3) = 4$ (۱)

$Q(2, 3) = 4, Q(14, 3) = 3$ (۲)

$Q(2, 3) = 0, Q(14, 3) = 0$ (۱)

$Q(2, 3) = 0, Q(14, 3) = 3$ (۲)

۴۷ - کدام گزینه یک ساختمان داده نیست؟

(۱) گراف جهت دار

(۲) صفحه حلقه ای

(۳) لیست پروردی

(۴) مجتمعه

۴۸- کدامیک از خصوصیات یک برنامه نمی‌باشد؟

(۱) برنامه می‌تواند پایان ناپذیر باشد.

(۲) برنامه اغلب دارای m ورودی ($M \geq 1$), n خروجی ($n \geq 0$) است.

(۳) دستورات برنامه باید بدون ابهام باشد.

(۴) دستورات برنامه توسط کامپیوتر باید قابل اجرا باشد.

۴۹- الگوریتم‌های بازگشتی چه معایین دارند؟

(۱) اتلاف حافظه، سرعت اجرای کمتر

(۲) اتلاف حافظه، طولانی بودن کد (source)

(۳) سرعت اجرای کمتر، طولانی بودن کد

(۴) طولانی بودن کد، اتلاف حافظه، سرعت اجرای کمتر

۵۰- تابع زیر چه خروجی دارد؟

```
void xyz (void){
```

```
    char ch;
```

```
    scanf ("%c", &ch);
```

```
    if(ch! ='\n') xyz();
```

```
    printf ("%c", ch);}
```

(۱) یک رشته را چاپ می‌کند.

(۲) یک رشته ورودی را در خروجی بر عکس چاپ می‌کند.

(۳) یک رشته عددی ورودی را در خروجی به صورت بر عکس چاپ می‌کند.

(۴) یک رشته رقیقی را از ورودی خوانده و در خروجی چاپ می‌کند.

۵۱- کار تابع f بر روی رشته s با n کاراکتر از موقعیت i در رشته s تعداد sub(s,i,n) کاراکتر از کاراکتر چیست؟ تابع (s,i,n) را برمی‌گرداند. (سراسری)

$$f(s,n) = \begin{cases} s & n=1 \\ f(sub(s,1,n-1),n-1)+sub(s,n,1) & n>1 \end{cases}$$

(۱) رشته s را برمی‌گرداند.

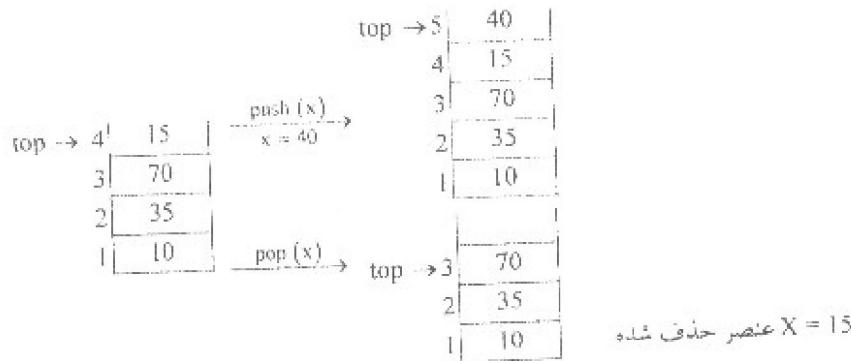
(۲) معکوس رشته s را برمی‌گرداند.

(۳) یک کاراکتر از انتهای رشته s اضافه می‌کند.

(۴) یک کاراکتر به ابتدای رشته s اضافه می‌کند.

(پشتدها) Stack

هر لیست پشت سرهمی از داده‌ها که عمل حذف (pop) و درج (push) در آن از یک طرف لیست که همان عنصر بالا (top) نامیده می‌شود انجام می‌گیرد.

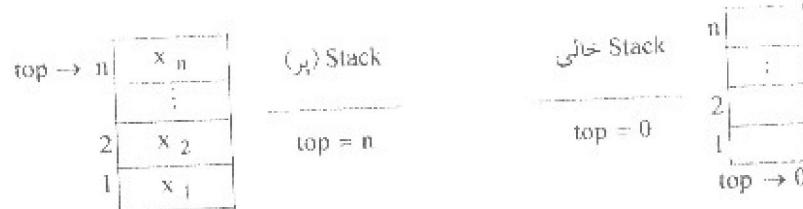


دیده می‌شود که برای درج (push)، باید اشاره گر top یک واحد به سمت بالا حرکت کردد؛ سپس عمل درج انجام می‌شود؛ و در عمل حذف (pop) ابتدا داده‌ای که top به آن اشاره می‌کند حذف شده سپس top یک واحد به سمت پائین حرکت می‌کند.

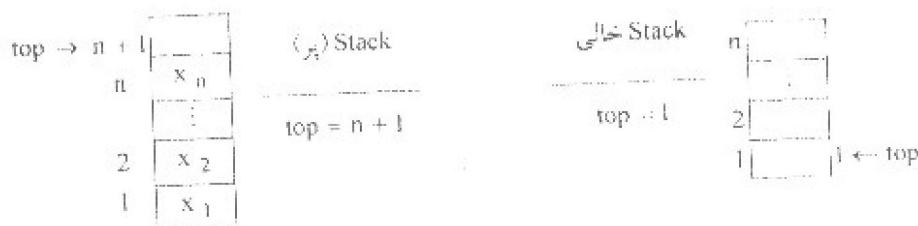
■ وضعیت اشاره گر top در شرایط مرزی خالی یا پر بودن stack

در صورتیک آرایه $[1 \dots n]$ برای نگهداری عناصر پشتنه در نظر گرفته شود؛ در این صورت شرایط مرزی به صورت زیر خواهد بود.

- ۱- با فرض n که top به خانه پر (آخرین خانه پر) اشاره کند.



۲- با فرض آن که top به خانه خالی (اولین خانه خالی) اشاره کند.



نکته مهم: همیشه به طور پیش فرض top به آخرین خانه پر اشاره می‌کند. (حالت ۱)

الگوریتم‌های pop (حذف) و push (درج)

با فرض آن که top به آخرین حالت پر اشاره می‌کند، در صورتیکه بخواهیم عملیات حذف و درج را در stack (پشته) تجام دهیم می‌توانیم از زیر برنامه‌های زیر استفاده کنیم

```
procedure push (item: نوع);
begin
  if top = n then
    write ('stack Full')
  else
    begin
      top := top + 1;
      stack [top] := item;
    end;
```

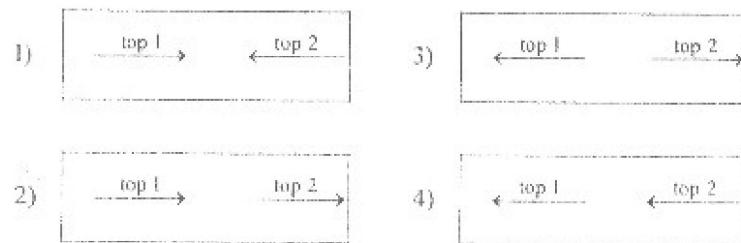
در این الگوریتم در صورتیکه در آرایه $[1..n]$ stack نشده باشد (پشته پر) ابتدا top به بالا حرکت کرده سپس داده (item) در محل خالی جدید درج می‌شود.

```
procedure pop (item: نوع);
begin
  if top = 0 then
    write ('stack Empty')
  else
    begin
      item := stack [top];
      top := top - 1;
    end;
```

در این الگوریتم در صورتیکه در آرایه $[1..n]$ stack نباشد (پشته خالی) ابتدا داده‌ای که top به آن اشاره می‌کند داخل item نگهذاری شده سپس top یک واحد به سمت پائین حرکت می‌کند.

بهینه سازی قرار گرفتن چند stack در یک آرایه:

در صورتیکه بخواهیم 2 stack را داخل یک آرایه قرار دهیم؛ این امکان وجود دارد که 2 پشته در حالت‌های مختلفی کنار هم قرار گیرند،



پاتوچه به حالت‌های نشان داده شده دیده می‌شود که حالت ۱ از تمام حالت‌ها بهتر بیاده‌سازی شده است این به آن علت است که در صورتیکه یکی از دو stack top ۱ و top ۲ درجی نداشته باشد stack دیگر می‌تواند از فضای خالی آن استفاده کند، اما در بقیه حالت‌ها به ترتیب مشکلات زیر وجود دارد:

۱) در این حالت ۱ تا top ۲ تا top ۳ تا انتهای می‌تواند حرکت کند.

۲) در این حالت ۱ تا top ۲ تا از وسط حرکت کرده و به ترتیب به ابتدا و انتهای آرایه محدود می‌شوند.

۳) در این حالت ۲ تا top ۳ تا از top ۱ و ۲ تا از top ۱ ابتدا حرکت می‌کند.

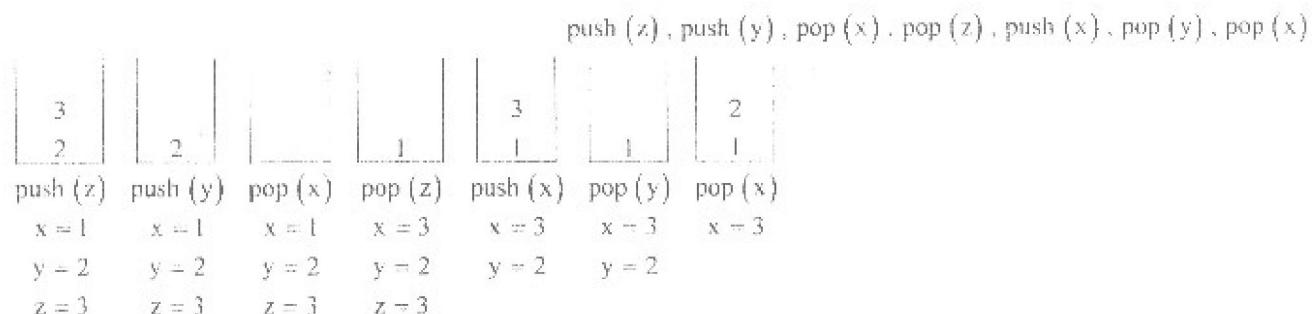
در حالت ۱ که بهترین وضعیت است، ۱ تا top ۲ به سمت هم حرکت می‌کنند و این باعث می‌شود که اگر مثلًا top ۱ هیچ درجی انجام ندهد ۲ تا ابتدای آرایه حرکت کند.

3

2

1

مثال: اگر یک پشته با وضعیت



ورودی‌ها و خروجی‌های مجاز برای یک پشته:

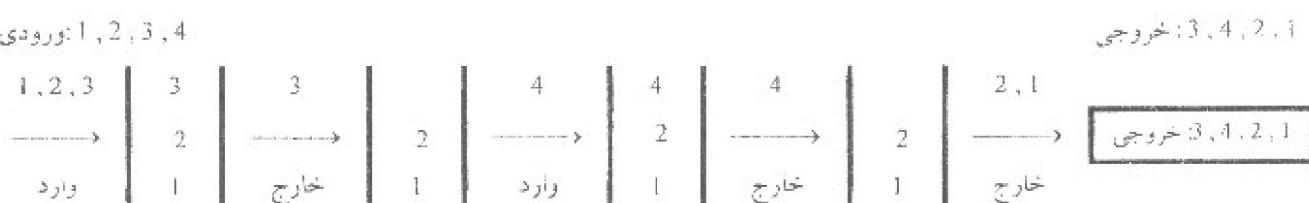
در یک پشته اولی در صورتیکه داده‌های a_1, a_2, \dots, a_n به ترتیب وارد stack شوند، با رعایت قواعد زیر می‌توانیم خروجی‌های مجاز را تعیین کنیم:

الف) هر داده به محض ورود می‌تواند از پشته خارج شود.



ب) در هر مرحله هر گاه بخواهیم داده‌ای را در خروجی پیشم که هنوز وارد stack نشده است می‌باشد داده‌ها را پشت سرهم وارد پشته کنیم تا به این مورد نظر برسیم داده را وارد پشته کرده و سپس خارج می‌کنیم.

ج) در هر مرحله هر گاه بخواهیم داده‌ای را در خروجی پیشم که بالین stack است و عنصر بالای پشته نیست این عمل غیرممکن است.



سافتمن دادهها

۱، ۲، ۳، ۴ ورودی

۳، ۴، ۱، ۲ خروجی



در این مرحله خارج کردن ۱ غیرمجاز است چون در پائین پشته قرار دارد.

تعداد خروجی های مجاز در یک stack تابع از رابطه زیر بدمت می آید:

$$\binom{2n}{n}$$

$n+1$

مثال: در صورتیکه ۲، ۳، ۱ به ترتیب وارد یک پشته شوند تعداد حالت های مجاز کدام هستند.

(6)

$$n=3 \rightarrow \frac{\binom{6}{3}}{3+1} = 5$$

در صورتیکه ورودی ۱، ۲، ۳ باشد خروجی های مجاز و غیرمجاز به شرح زیر است:

	۱ وارد	۱ خارج	۲ خارج	۳ خارج
۱، ۲، ۳ مجاز	→ [] → 1	→ 2 وارد	→ 2 خارج	→ 3 وارد
۱، ۳، ۲ مجاز	→ [] → 1	→ 3 خارج	→ 3 وارد	→ 2 خارج
۲، ۳، ۱ مجاز	→ [] → 2	→ 2 خارج	→ 3 خارج	→ 1، ۳
۲، ۱، ۳ مجاز	→ [] → 2	→ 1 خارج	→ 3 خارج	→ 3 خارج
۳، ۱، ۲ غیر مجاز	→ [] → 3	→ 3 خارج	→ 2 خارج	→ 1 خارج
۳، ۲، ۱ مجاز	→ [] → 3	→ 3 خارج	→ 1 خارج	→ غیر مجاز

■ پسته های مجاز برای ورودی های مختلف

بک slack زیک ورودی دادهها رماني مجاز است که داده های قبلی روی داده های بعدی فراز نگیرند، چرا که داده های قبلی یا باید قبل از وارد شدن باشند یا این که خارج شده باشند و پسین داده های بعدی باشند.



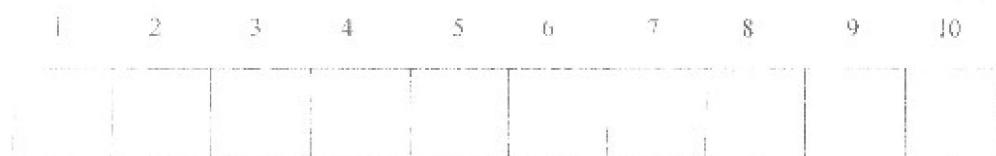
پسته چند خانه

برای ممایش n پسته در یک آرایه stack [1..m] آرایه مورد نظر را به n قسمت مساوی تقسیم می کنیم و به هر پسته خانه های مشخص تخصیص داده می شود.

نکته: سعی داریم که فضای m خانه آرایه را به طور مساوی بین n پسته تقسیم کنیم در این حالت اگر n مصوبی از m نباشد، فضای اضافی از آرایه m کافی به پسته آخر می رسد.

مثال: 3 پسته در آرایه 10 کافی:

$$\text{در این حالت برای هر پسته } \left\lfloor \frac{10}{3} \right\rfloor = 3 \text{ خانه در نظر نگرفته می شود اما یک خانه اضافی به پسته آخر می رسد.}$$



پسته اول

پسته دوم

پسته سوم

پیاده سازی پسته ها

برای هر پسته دلخواه آن از دو اشاره گر h[i] (اشاره به پاسین) و i[1] (اشاره به بالای) پسته استفاده می شود.

اشاره گر h[i] برای این استفاده می شود که اشاره [i-1] مربوط به stack، آن به پسته آن پرخورد نکند.

شوابیط مرزی برای stack (مهندسی)

$$1 \leq i \leq n \quad b[i] = t[i] + (i-1) \left\lfloor \frac{m}{n} \right\rfloor + 1 \quad i = 1, 2, \dots, n$$

$$t[i] = b[i+1]$$

مثال: می خواهیم در یک آرایه 10 تانی 3 بسته را فرار دهیم مطابقیت شوابیط اولیه برای هر بسته؟

$$\begin{cases} i=1 \rightarrow b[1] = t[1] = (1-1) \left\lfloor \frac{10}{3} \right\rfloor + 1 = 1 \\ m=10 \\ n=3 \end{cases} \longrightarrow \begin{cases} i=2 \rightarrow b[2] = t[2] = (2-1) \left\lfloor \frac{10}{3} \right\rfloor + 1 = 4 \\ i=3 \rightarrow b[3] = t[3] = (3-1) \left\lfloor \frac{10}{3} \right\rfloor + 1 = 7 \end{cases}$$

$$\left\lfloor \frac{m}{n} \right\rfloor = \left\lfloor \frac{10}{3} \right\rfloor = 3$$

تعداد خانه های هر بسته

$b[1]$

$t[1]$

1

$b[2]$

$t[2]$

2

$b[3]$

$t[3]$

3

4

5

6

7

8

9

10



دیده می شود که یک خانه اضافی در انتهای آرایه به stack (سوم) نعلق گرفته است.

- عملیات push (درج)

```

procedure push(item: شغ);
if t[i] = b[i+1] then
    write('Full')
else begin
    stack[t[i]] := item;
    t[i] := t[i] + 1;
end;
end;

```

چون در اینجا [i] به خانه خالی اشاره می کند برای درج (push)، باید عمل درج تجزم می شود سپس [i] یک واحد به بالا حرکت می کند.

عمليات pup (حذف)

```
procedure pop (item : &S15 );
```

begin

if $b[i] = t[i]$ then
 write ('Empty')

else

begin

$$t[i] := t[i] - 1;$$

item := stack[t[i]];

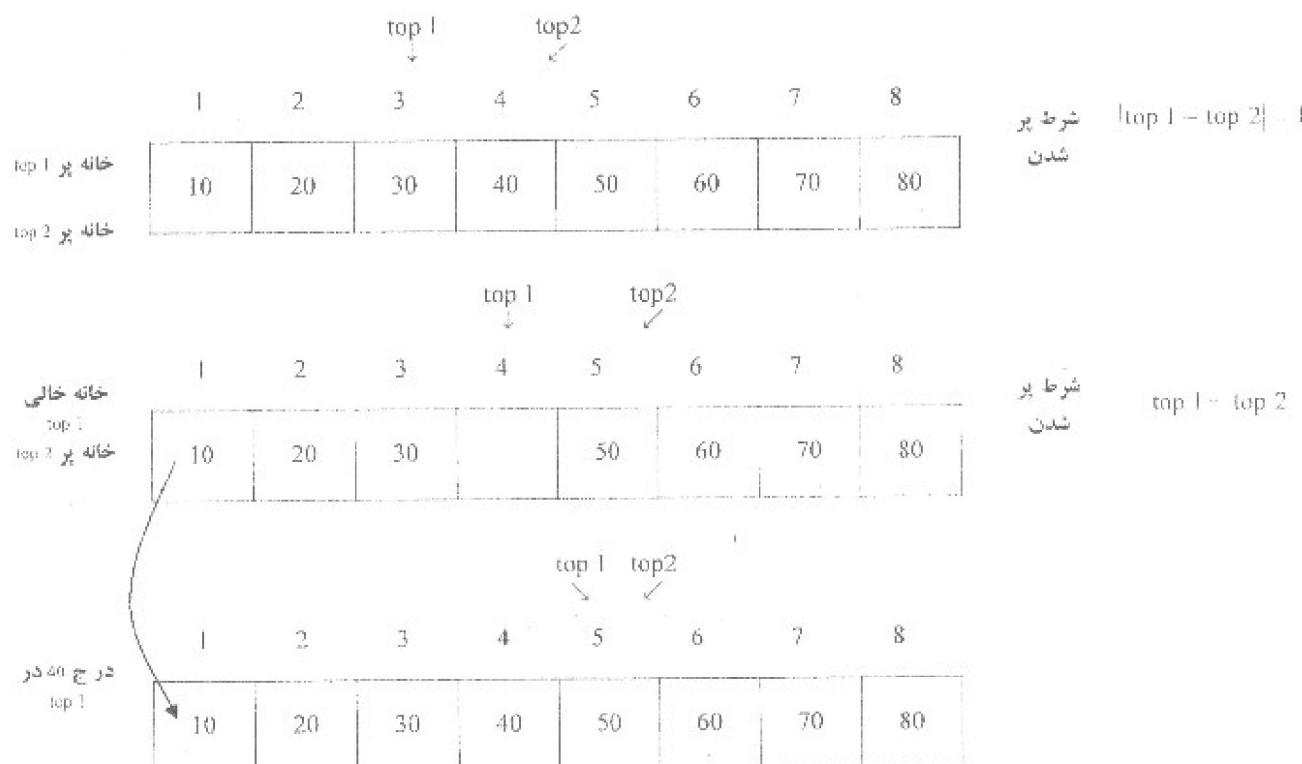
end;

چون در ابتدای [] به حانه خالی اشاره می‌کنند برای حذف (pop) ابتدا [] را یک واحد به پایین حرکت می‌کنند تا به حانه پر بر سرده سپس داده موردنظر حذف شده و در item قرار گیرد.

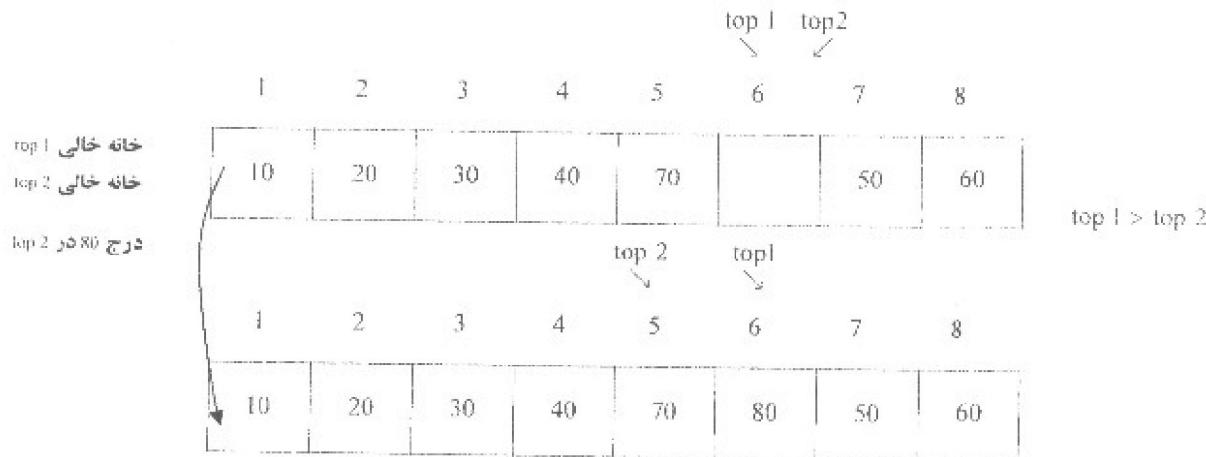
کاربری دشنه ها (۱۰۰)

- ۱- استفاده در توابع بازگشتنی (برج هانوی - تابع Fibonachi - دنباله Ackerman)
 - ۲- ارزیابی عبارت های محاسباتی (prefix - postfix - infix)
 - ۳- الگوریتم مسیر حرکت Maze
 - ۴- پیمايش عمقی درخت ها و گراف ها
 - ۵- استفاده در روش های مرتب سازی mergesort ، quicksort ، insertion sort

■ وضعیت‌های عزیزی در قرار مگر فتن ۲ پیشنهاد رک لست

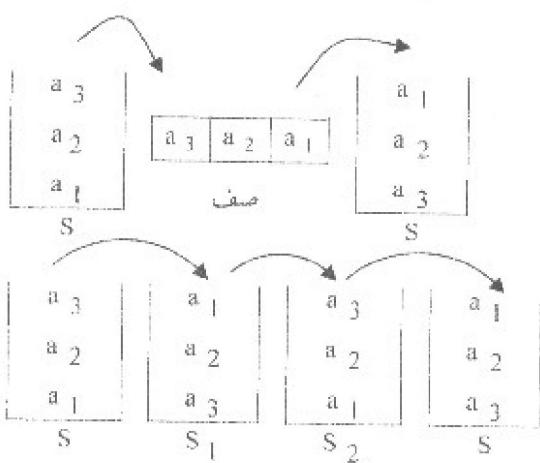


دقت می کنیم که در وضعیتی که ۱ به خانه خالی شاره می کند برای درج ایندا عمل درج انجام شده سپس ۱ واحد به آن اضافه می شود که در این مثال بعد از عمل درج $top_1 = top_2 = 5$ می شود.

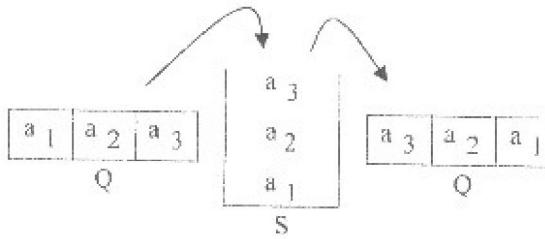


نکات مهم در مورد stack و صف:

- ۱- می خواهیم یک stack را داخل خودش معکوس بیسیم از چه فضای اضافه‌ای استفاده کنیم؛ از یک صف اضافی یا از دو stack می توان استفاده کرد، که استفاده از صف مفروض به صرفه تر است.



- ۲- در صورتی که بخواهیم عناصر یک صف را داخل خودش معکوس کنیم از چه فضای اضافه‌ای استفاده کنیم؛ از یک stack اضافی می توان استفاده کرد.

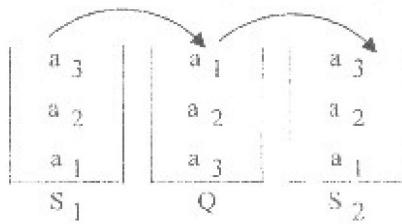


- ۳- می خواهیم یک stack را در stack دیگر معکوس بیسیم از چه فضای اضافه‌ای استفاده کنیم؛ هیچ فضای اضافی نیاز ندارد.

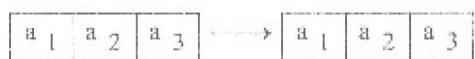




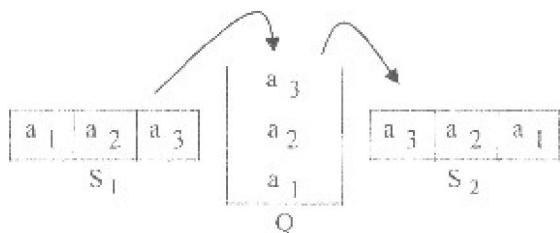
۴. می‌خواهیم بک stack را در دیگر یدون هیچ تغییر بینیم از چه فضای اضافه‌ای استفاده کنیم؛ از بک stack اضافی می‌توان استفاده کرد.



۵. می‌خواهیم بک صف را داخل صف دیگر به همان شکل بینیم از چه فضای اضافه‌ای استفاده کنیم؛ به فضای اضافی نیاز نداوریم.



۶. می‌خواهیم بک صف را داخل صف دیگر معکوس بینیم از چه فضای اضافه‌ای استفاده کنیم؛ از بک stack اضافی استفاده می‌کنیم.



ارزشیابی عبارت‌های محاسباتی

هر عبارت محاسباتی با نوجه به اولویت عملگرهاش قابل ارزیابی و محاسبه است.

بادآوری ۱:

در عبارت محاسباتی



بادآوری ۲:

با نوجه به بادآوری ۱ دیده می‌شود که در عبارت‌های محاسباتی دو نوع عملگر وجود دارد:

۱- یکنای (unary)

۲- بینری (binary)

عملگرها یکنای مانند: - (منفی) و + (ثبت) فقط به یک عملوند نیاز دارند.

عملگرها دودوئی مانند: * (ضرب)، / (تقسیم)، + (جمع) و - (تفريق) و ^ (نوان) به دو عملوند نیاز دارند.

اولویت عملگرها:

به طور کلی صرف نظر از هر زبان برنامه‌سازی اولویت عمومی عملگرها را سی تون به شرح زیر در نظر گرفت:

اولویت	نام عملگر
۱	{ } ()
۲	- (منفی)، + (ثبت)
۳	^ یا ^ (نوان)
۴	* (ضرب)، / (تقسیم)
۵	+ (جمع)، - (تفريق)

نکته مهم: در هر عبارت محاسباتی اگر تعداد عملوندها ۱ واحد بیشتر از تعداد عملگرها باشد، در آن عبارت تمام عملگرها دودوئی است:

$$a * b + c \rightarrow 3 \text{ عملوند و 2 عملگر}$$

$$a * b - c / d \rightarrow 4 \text{ عملوند و 3 عملگر}$$

در عین حال اگر تعداد عملوندها و تعداد عملگرها با هم مساوی باشد، با تعداد عملگرها بیشتر از عملوندها باشد. در آن عبارت حتماً عملگر

یکنای و جزو دارد:

$$- a * b \rightarrow 2 \text{ عملوند و 2 عملگر}$$

$$- a * - b \rightarrow 2 \text{ عملوند و 3 عملگر}$$

مثال: «مطلوب است اولویت‌مندی عبارت‌های زیر»

$$\begin{array}{r} a^4 - b / c \hat{d} + e \\ \underline{\quad 1 \quad} \quad \underline{\quad 2 \quad} \\ \underline{\quad 3 \quad} \\ \underline{\quad 4 \quad} \\ \underline{\quad 5 \quad} \end{array}$$

$$\begin{array}{r} a * \underbrace{(b + c)}_{\underline{\quad 1 \quad}} - \underbrace{k / d}_{\underline{\quad 3 \quad}} + e \\ \underline{\quad 2 \quad} \\ \underline{\quad 4 \quad} \\ \underline{\quad 5 \quad} \end{array}$$

$$\begin{array}{r} a * b - c \hat{d} \hat{e} + f \\ \underline{\quad 1 \quad} \quad \underline{\quad 2 \quad} \\ \underline{\quad 4 \quad} \\ \underline{\quad 5 \quad} \end{array}$$

$$\begin{array}{r} a / \underbrace{(b + c)}_{\underline{\quad 1 \quad}} - v * \underbrace{(f + g)}_{\underline{\quad 2 \quad}} \\ \underline{\quad 3 \quad} \quad \underline{\quad 4 \quad} \\ \underline{\quad 5 \quad} \end{array}$$

$$\begin{array}{r} a - b * c / d + c \hat{f} \hat{g} - h \\ \underline{\quad 1 \quad} \quad \underline{\quad 2 \quad} \\ \underline{\quad 5 \quad} \quad \underline{\quad 6 \quad} \\ \underline{\quad 7 \quad} \end{array}$$

■ در هر عبارت محاسباتی اگر تعداد عملگرهای از تعداد عملوندها بیشتر باشد

1- تعداد عملوندها - تعداد عملگرهای یکانی = تعداد عملگرهای پیکانی

نها عملگر پیکانی - (منفی) است.

تعداد عملگرهای پیکانی = $a * b + c \rightarrow$ پیکانی

نکار عملگرهای پیکانی - (منفی) 2 بار است.

تعداد عملگرهای پیکانی = $a^n + b + c \rightarrow$ پیکانی

پیکانی

■ درخت عبارت محاسباتی (درخت پارس)

برای تشکیل درخت هر عبارت محاسباتی به صورت زیر عمل می‌کنیم:

1- ابتدا عبارت را بر حسب اولویت عملگرهای اولویت‌مندی می‌کنیم.

2- ریشه درخت، کم اولویت‌ترین از نظر اولویت است.

3- ریشه زیر درختان جب و راست به ترتیب کم اولویت‌ترین عملگر از نظر اولویت در جب و راست ریشه درخت است.

4- برگ‌های درخت عملوندها هستند.

مثال ۱:

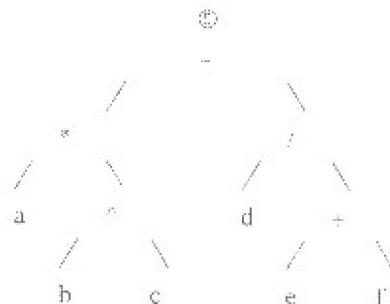
$$\begin{array}{c} \textcircled{1} \\ - \\ \begin{array}{c} a * b - c / d \\ \underline{\quad 1 \quad} \quad \underline{\quad 2 \quad} \\ \backslash \quad / \\ a \quad b \quad c \quad d \end{array} \end{array}$$

کم اولویت‌ترین عملگر - (تفريق) است که در ریشه قرار گرفته است.

مثال ۲:

$$a * b ^ c - d / \{ e + f \}$$

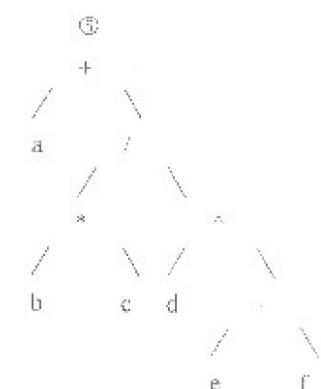
2 1
 3 4
 5



مثال ۳:

$$a + b * c / d ^ \{ e - f \}$$

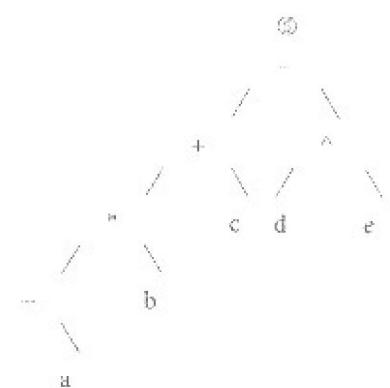
3 1
 2
 4



مثال ۴:

$$- a * b + c - d ^ e$$

1 2
 3
 4



دیده می شود که عضوند مربوط به عملگر یکانی (-)، یعنی a فرزند راست آن قرار گرفته است.

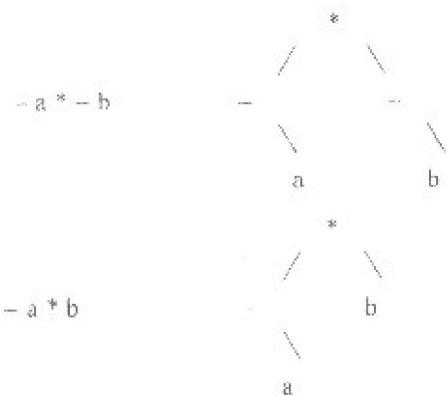
لکنه: تعداد گره های درخت عبارت محاسباتی در صورتیکه همه عملگرها دو دونی باشد فرد است.



درخت روی رو ۵ گره دارد.

کل تعداد گره های درخت عبارت محاسباتی در صورتیکه به تعداد فرد عملگر یکانی داشته باشد زوج و در صورتیکه به تعداد زوج عملگر

یکانی داشته باشد فرد است.



درخت روبرو ۵ گره دارد.

درخت روبرو ۴ گره دارد.

عبارات infix (میانوندی) - postfix (پسوندی) - prefix (پیشوندی)

عبارت infix (میانوندی):

در هر عبارت عمومی محاسباتی هر عملگر دودونی بین عملوندهایش قرار می‌گیرد، که به این نحوه قرار گرفتن عملگرها در بین عملوندها روش میانوندی گفته می‌شود.

عملگر $*$ بین عملوندهای a ، b قرار دارد. $\rightarrow a * b$

نکته ۱: در هر عبارت میانوندی (infix) اولویت عملگرها مطرح بوده و از () برای تغییر اولویت عملگرها استفاده می‌شود.

نکته ۲: در هر عبارت میانوندی (infix) اولویت عملگرها با توجه به جدول اولویت عملگرها مشخص می‌شود.

عبارت postfix (پسوندی):

در این روش هر عملگر بعد از عملوندهایش قرار می‌گیرد.

عملنگر $*$ بعد از عملوندهایش قرار گرفته است. $\rightarrow * ab$

نکته ۱: در عبارت‌های postfix اولویت عملگرها مطرح نبوده و از () نیز استفاده نمی‌شود.

نکته ۲: به روش postfix روش لهستانی معکوس یا polish وارونه (Reverse polish Notation) RPN نیز می‌گویند.

عبارت prefix (پیشوندی):

در این روش هر عملگر قبل از عملوندهایش قرار می‌گیرد.

عملگر $*$ قبل از عملوندهایش قرار گرفته است. $\rightarrow * ab$

نکته ۱: در عبارت‌های prefix اولویت عملگرها مطرح نبوده و از () نیز استفاده نمی‌شود.

نکته ۲: به روش prefix روش لهستانی یا روش polish نیز می‌گویند چون اولین بار این روش توسط ریاضی دان لهستانی معرفی شد.

برای همین به روش postfix روش لهستانی وارونه گفته می‌شود.

نتیجه‌گیری: (مههم)

- ۱- در تبدیل عبارات infix و postfix به prefix به یکدیگر ترتیب عملوندهای هیچ شکل عوض نمی‌شود.

مثال:

$a * b + c$ infix
 $a b * c +$ postfix
 $+ * a b c$ prefix

دیده می شود که در هر سه روش ترتیب عملوندهای a, b, c تغییری نکرده است اما ترتیب عملگرها ممکن است خویش نشود.

۱. در هر عبارت postfix همیشه سمت راست عبارت عملگر است.

۲. در هر عبارت prefix همیشه سمت چپ عبارت عملگر است.

تبدیل عبارت infix به عبارت های prefix و postfix

- | | | |
|-------------------------------|---|--------------------------------|
| ۱- روش پرانتز گذاری | } | برای این عمل سه روش وجود دارد. |
| ۲- استفاده از stack | | |
| ۳- پیمایش درخت عبارت محاسباتی | | |

۱- روش پرانتز گذاری

تبدیل infix به postfix

(الف) عبارت را به طور کامل بر حسب اولویت عملگرها پرانتز گذاری عملگر مربوط به هر پرانتز را روی پرانتز بسته می گذاریم.

(ب) عبارت را از چپ به راست (شامل عملوندها و عملگرهای روی پرانتزهای بسته) در خروجی می نویسیم.

■ قبل از فرمت (الف) ابتدا وضعیت پرانتزهای عبارت را مشخص می کنیم.

مثال ۱:

$$a * b + c \rightarrow ((a * b) + c)' \longrightarrow a, b, *, c, +$$

مثال ۲:

$$a * (b + c) \cdot g / d \longrightarrow ((a * (b + c)) \cdot (g / d))' \rightarrow a, b, c, +, *, g, d, /, \cdot$$

مثال ۳:

$$a + b \uparrow c * d - e \rightarrow ((a + ((b \uparrow c) * d))' - e)' \rightarrow a, b, c, \uparrow, d, *, +, e, -$$

مثال ۴:

$$\sqrt{b^2 - 4ac} = (b \uparrow 2 - 4 * a * c) \uparrow (1 / 2) \rightarrow (((b \uparrow 2)' - ((4 * a)' * c)') \uparrow (1 / 2))'$$

$$\rightarrow b, 2, \uparrow, 4, a, *, c, *, -, 1, 2, /, \uparrow$$

مثال ۵:

$$a * b + c \wedge d \wedge e - f \rightarrow \left[\left((a * b) + \left(c \wedge (d \wedge e) \right) \right) \right] - f$$

$$\rightarrow a, b, +, c, d, \wedge, e, \wedge, -, f, -$$

مثال ۲

$$a * -b \wedge c \uparrow d - e / f \rightarrow \left[\left(a * \left(-b \right) \wedge \left(c \uparrow d \right) \right) \right] - \left(e / f \right)$$

$$\rightarrow a, b, -, c, d, \wedge, \uparrow, e, /, f, -, -$$

تبدیل prefix به infix

الف) عبارت را به طور کامل برحسب اولویت عملگرها پرانتزگذاری می‌کیم در جن پرانتزگذاری عملگر مربوط به هر پرانتز را روی پرانتز باز می‌گذاریم.

ب) عبارت را از چپ به راست (شامل عملوندها و عملگرهای روی پرانتزهای باز) در خروجی می‌نویسیم.

■ قبل از قسمت (الف) ابتدا وصیعت پرانتزهای عبارت را مشخص می‌کیم.

مثال ۳

$$a * b + c \rightarrow \left(\left(a * b \right) + c \right) \rightarrow +, *, a, b, c$$

مثال ۴

$$a * (b + c) - g / d \rightarrow \left(\left(a * \left(b + c \right) \right) - \left(g / d \right) \right) \rightarrow -, *, +, a, b, c, /, g, d$$

مثال ۵

$$a + b \wedge c * d - e \rightarrow \left(+ \left(a + \left(\left(b \wedge c \right) * d \right) \right) - e \right) \rightarrow +, +, a, \wedge, \cdot, b, c, d, e$$

مثال ۶

$$\sqrt{b^2 - 4ac} = (b \uparrow 2 - 4 * a * c) \uparrow (1 / 2) \rightarrow \uparrow \left(\left(\uparrow \left(b \uparrow 2 \right) - \left(\cdot \left(4 * a \right) * c \right) \right) \uparrow \left(1 / 2 \right) \right)$$

$$\rightarrow \uparrow, +, \uparrow, b, 2, *, *, 4, a, c, /, 1, 2$$

مثال ۷

$$a * b + c \wedge d \wedge e - f \rightarrow \left(+ \left(a * b \right) + \wedge \left(c \wedge \wedge \left(d \wedge e \right) \right) \right) - f$$

$$\rightarrow +, *, a, b, \wedge, \cdot, c, \wedge, \wedge, d, e, f$$

مثال ۸

$$a * -b \wedge c \uparrow d - e / f \rightarrow \left(+ \left(a * \left(-b \right) \wedge \uparrow \left(c \wedge d \right) \right) \right) - \left(e / f \right)$$

$$\rightarrow +, *, a, \wedge, -, b, \wedge, \uparrow, c, \wedge, d, /, e, f$$

■ ارزشیابی عبارت‌های postfix (پسوندی) و prefix (پیشوندی) مهم:

برای ارزشیابی عبارت‌های prefix و postfix و تعیین مقدار نهانی آن‌ها به صورت زیر عمل می‌کنیم:

الف) ابتدا کنترل می کنیم که آیا تعداد عملوند هایک واحد از عملگرها بیشتر است یا خیر جون در غیر این صورت در عبارت عملگر بکاری وجود داشته و ارزشیابی آن شرایط خاصی خواهد داشت.

ب) عبارت را از سمتی بررسی می کنیم که با عملوند شروع می شود در postfix از چپ و در prefix از راست عبارت را بررسی می کنیم.

ج) با رسیدن به هر عملگر دو عملوند برای آن در نظر می گیریم ادر postfix، سمت چپ عملگر و در prefix سمت راست عملگر

د) با توجه به عملگر عملیات را روی عملوند ها انجام می دهیم (چپ به راست)

مثال ۱: از سمت عملوند ها (چپ)

$12, 3, /, 2, *$

$$\rightarrow \left(\underbrace{(12, 3, /)}_3, 2, * \right)$$

مثال ۲: از سمت عملوند ها (چپ)

$2, 4, *, 8, +, 10, 2, -, /$

$$\rightarrow \left(\underbrace{(2, 4, *)}_8, 8, + \right), \left(\underbrace{(10, 2, -)}_{12}, / \right)$$

مثال ۳: از سمت عملوند ها (چپ)

$\rightarrow 12, 7, 3, -, /, 2, 1, 5, +, *, +$

$$\rightarrow \left(\underbrace{12, \left(\underbrace{(7, 3, -)}_2 \right), /}_{15}, \left(\underbrace{2, \left(\underbrace{(1, 5, +)}_4 \right)}_{12}, + \right) \right)$$

:prefix

مثال ۱: از سمت عملوند ها (راست)

$*, /, 12, 3, 2$

$$\rightarrow \left(*, \left(\underbrace{/}_{4}, \underbrace{(12, 3)}_8 \right), 2 \right)$$

مثال ۲: از سمت عملوند ها (راست)

$/, +, *, 2, 4, 8, -, 10, 2$

$$\rightarrow \left(/ , \left(+ , \left(* , \underline{2} , 4 \right) , 8 \right) , \left(- , \underline{10} , 2 \right) \right)$$

$$\begin{array}{c} 8 \\ 16 \\ \hline 2 \end{array}$$

مثال ۳ از سمت عملوند (راست)

$$+ , / , 12 , - , 7 , 3 , * , 2 , + , 1 , 5$$

$$\rightarrow \left(+ , \left(/ , 12 , \left(- , \underline{7} , 3 \right) \right) , \left(* , 2 , \left(- , \underline{1} , 5 \right) \right) \right)$$

$$\begin{array}{c} 1 \\ 12 \\ \hline 15 \end{array}$$

مثال ۴

مطلوب است ارزش عبارت پسوندی زیر: (کنکور ۸۲)

$$6 , 2 , 3 , + , - , 3 , 8 , 2 , / , + , * , 2 , \wedge , 3 , -$$

۷ (۴)

۳۴ (۳)

✓ ۵۲ (۲)

۴۸ (۱)

$$\begin{array}{c} (((6 , \left(2 , 3 , + \right) , -) , (3 , \left(8 , 2 , / \right) , +) *) , 2 , \wedge) , 3 , +) \\ \hline \begin{array}{c} 5 \\ 1 \\ \hline 7 \end{array} \quad \begin{array}{c} 4 \\ 7 \\ \hline 12 \end{array} \\ \hline \begin{array}{c} 49 \\ 12 \\ \hline 52 \end{array} \end{array}$$

حالت خاص: (مههم)

زمینه در یک عبارت محاسباتی (prefix - postfix - infix) تعداد عملگرها بیشتر یا مساوی تعداد عملوندها باشد. حتماً در آن عبارت محاسباتی عملگر یکانی (unary) وجود دارد.

بادآوری:

در هر عبارت محاسباتی تعداد عملگرها بیکانی همیشه ز رابطه زیر بنت می‌آید:

(*) $+ []$ تعداد عملوندها = تعداد عملگرها = تعداد عملگرها بیکانی

در صورتیکه در عبارت محاسباتی همه عملگرها دودونی باشد،

$+ []$ تعداد عملگرها = تعداد عملوندها

و در نتیجه با توجه به رابطه (*) تعداد عملگرها بیکانی (۰) است.

مثال:

$\text{infix : } a * b + c$ $\text{prefix : } + * a b c$ $\text{postfix : } a b * c +$	$\xrightarrow{\quad \cdot \quad}$ $\xrightarrow{\quad \cdot \quad}$ $\xrightarrow{\quad \cdot \quad}$	تعداد عملگرها تعداد عملوندها تعداد عملوندها	$= 2$ $= 3$ $= 3$	$\xrightarrow{\quad \cdot \quad}$ $\xrightarrow{\quad \cdot \quad}$	تعداد عملگر یکانی $= 2 - 3 + 1 = 0$
--	---	---	-------------------------	--	--

 در صورتیکه در عبارت محاسباتی تعداد عملگرها = تعداد عملوندها پاشد، با توجه به رابطه (*) تعداد عملگرها بکاری 1 است.

63

<code>int ix() = a + b - c</code>	تعداد عملگرها	= 3
<code>prefix: + - abc</code>	→ تعداد عملگرها	= 3
<code>postfix: a - b + c</code>	→ تعداد عملوندها	= 3

نکته مهم: همانطور که در مثال ۲ دیده می شود، به راحتی در عبارت Infix می توان فهمید که کدام عملگر یکانی است (a) اما با مشاهده postfix یا prefix نمی توان وضعیت عملگر یکانی را به راحتی مشخص کرد. مگر آن که در عبارت فقط یک عملگر - یا + وجود داشته باشد.

دقت کنید که به طور مثال، اگر عبارت infix به صورت $c - (a - b)$ - مطرح می شد، عبارت prefix آن یاز هم ---abc می شود و این نشان می دهد که بر روی عبارت postfix یا prefix به راحتی نمی توان وضعیت عملگر یکانی را در صورت وجود تعیین کرد.

$$-7 \equiv -(10 - 3) \xrightarrow{\text{prefix}} - , -, 10 , 3$$

د. مثال بالا دیده شد که با آن که عادت infix $\text{b} + \text{c}$ را در حالت prefix $\text{b} + \text{c}$ متفاوت است.

نکته: در خوبی‌بده ترین حالت می‌توانیم امیدوار باشیم که در عبارت فقط یک عملگر - یا + باشد تا در صورتیکه (تعداد عملوندها = تعداد عملگرها) باشد آن گاه آن عملگر - یا + همان عملگر یکتی است. در عین حال تشخیص وضعیت عملگرهای یکانی در عبارت prefix و postfix به راحتی انجام نمی‌شود.

مثال: حاصل عبارت $-(-8, 2, 3, -1, 1, /, 10, /, 5, -, 2, 3)$ کدام است؟ (۸۳)

- 9 (3)

10 of

✓ 1 (3)

اگر عینکیں قلب از ایہ صورت بکھڑے تو فرم کرنا:

$$\begin{array}{r}
 + - * \quad \overbrace{\uparrow 2 \ 3}^{\text{sum}} \quad \overbrace{- 1 \ 1}^{\text{sum}} / \overbrace{10 / 5}^{\text{sum}} \quad \overbrace{+ 2 \ 3}^{\text{sum}} \\
 \underline{2 \uparrow 3 = 8} \quad \underline{- 1} \\
 \hline
 8 * (-1) = -8 \\
 \hline
 -8 - 1 = -9
 \end{array}
 \qquad
 \begin{array}{r}
 \overbrace{2 + 3 = 5}^{\text{sum}} \\
 \hline
 5 / 5 = 1 \\
 \hline
 10 / 1 = 10
 \end{array}$$

و اگر عملگر - قبل از ارائه صورت دو تایی فرض کنیم و - قبل از * رایکنایی فرض کنیم:

$$\begin{array}{r}
 + * \quad \overbrace{\begin{array}{r} 2 \quad 3 \\ \times \quad 1 \end{array}}^{\downarrow} \quad \overbrace{\begin{array}{r} 1 \quad 1 \\ - \quad 1 \end{array}}^{\downarrow} \quad / \quad \overbrace{\begin{array}{r} 10 \quad 5 \\ \times \quad 5 \end{array}}^{\downarrow} \quad + \quad \overbrace{\begin{array}{r} 2 \quad 3 \\ \times \quad 5 \end{array}}^{\downarrow} \\
 \hline
 2 \times 3 = 6 \quad 1 - 1 = 0 \quad \quad \quad 2 \times 5 = 10 \quad \quad \quad 5 \times 5 = 25 \\
 \hline
 8 \times 0 = 0 \quad \quad \quad 5 \times 5 = 25 \\
 \hline
 - 0 = 0 \quad \quad \quad 10 + 25 = 35 \\
 \hline
 0 - 10 = 10
 \end{array}$$

تبدیل infix به postfix و prefix با استفاده از پشتنه:

۱- postfix به infix: برای این تبدیل به ترتیب زیر عمل می کنیم:

(الف) عبارت را از سمت چپ پیمايش می کنیم.

(ب) عملوندها را در خروجی می نویسیم.

(ج) به هر پرانتز () که رسیدیم آن را به راحتی داخل Stack قرار می دهیم.

(د) به هر عملگر که رسیدیم به شرطی که اولویت آن عملگر از عملگر بالای stack بیشتر باشد، آن را داخل stack قرار می دهیم؛ در غیر این صورت آنقدر از بالای stack عملگر خارج کرده و در خروجی می نویسیم تا یا stack خالی شده و یا به عملگری برسیم که بتوانیم عملگر مورد نظر را روی آن در stack قرار دهیم.

پادآوری:

باتوجه به مورد (د)، عملگر + نمی تواند روی + یا - فرار گیرد، همین طور - روی + یا - و / روی * یا / نمی تواند فرار گیرند. اما

توان (\wedge) روی توان (\wedge) می تواند فرار گیرد؛ چون اولویت توانهای پشت سرهم از راست به چپ بود سی می شود.

ه) اگر بالای stack پرانتز () باشد هر عملگری به راحتی روی آن فرار می گیرد.

و) به هر پرانتز () که رسیدیم آنقدر از stack عملگر خارج کرده و در خروجی می نویسیم تا به () برسیم در این وضعیت () و () با هم ختنی می شوند.

ر) زمانیکه به پشتنه عبارت رسیدیم در صورتیکه stack خالی نباشد، آن را به طور کامل در خروجی می نویسیم.

نکته ۱: در این تبدیل (تعداد pop = تعداد push) بوده و تعداد آن از رابطه زیر بدست می آید:

$$\text{تعداد پرانتزهای } ()^2 + \text{تعداد عملگرها} = \text{تعداد pop} = \text{تعداد push}$$

عملت برای تعداد pop و push این است که هر عملگر با پرانتز () که وارد stack می شود (push) بالاخره باید از stack خارج شود. (pop)

نکته مهم: حداقل فضائی که در طول اجرای برنامه استفاده می شود، برای حداقل فضائی است که برای تبدیل مورد نیاز است.

مثال ۱: حداقل تعداد پشتنه برای تبدیل عبارت میانوندی $a / (b - c * (d + e))$ به معادل پسوندی (postfix) کدام است؟

۳ (۴)

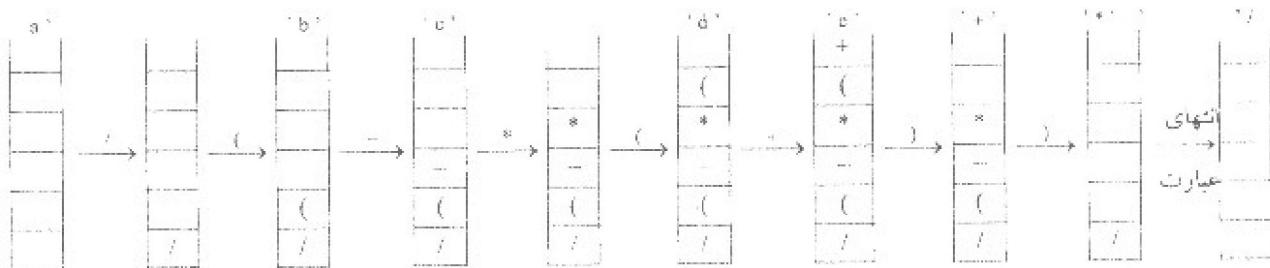
✓ ۶ (۲)

۵ (۲)

۴ (۱)

از چپ به راست عبارت را پویش می کنیم.

ساختمنان داده‌ها

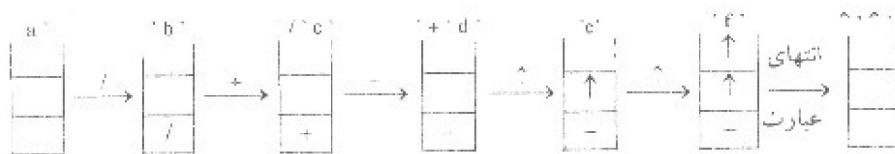


نتیجه را از چپ به راست می‌نویسیم: $a \cdot b \cdot c \cdot d \cdot + \cdot \cdot \cdot$

→ حداقل فضای مورد نیاز 6 بوده که همان حد اکثر فضای استفاده شده است.

$$\text{→ تعداد push - تعداد عملگرها} = \text{تعداد} \quad 6 = 2 + 4 =$$

مثال ۲: حداقل اندازه پشته برای تبدیل عبارت میانوندی $a / b - c - d \uparrow e \uparrow f \uparrow$ به پسوندی کدام است؟



نتیجه از چپ به راست: $\uparrow, \uparrow, \uparrow, \uparrow, \uparrow, \uparrow, a, b, /, -, c, -, d, e, f, -, \uparrow, \uparrow$ خواهد بود.

حداقل فضای مورد نیاز 9 بوده که همان حد اکثر فضای مورد استفاده است.

$$\text{→ تعداد push - تعداد عملگرها} = \text{تعداد} \quad 5 = 0 + 5 =$$

مثال ۳: حداقل فضای پشته مورد نیاز (میانی) برای تبدیل عبارت میانوندی زیر به معادل پسوندی (postfix) کدام است؟

$$\text{الف) } a + b \cdot (c - (d + e)) \cdot f \cdot g \cdot h \rightarrow \text{عبارت میانوندی}$$

$$\text{ب) } a - b - c * ((d - e \wedge f) + g \wedge h) \rightarrow \text{عبارت میانوندی}$$

۳- تبدیل infix به postfix

الف) عبارت را از سمت راست پیماش می‌کنیم.

ب) عملوندها را در خروجی می‌نویسیم.

ج) به هر پرانتز^() که رسیدیم آن را به راحتی داخل stack قرار می‌دهیم.

د) به هر عملگر که رسیدیم به شرطی که اولویت آن عملگر از عملگر بالای stack بیشتر یا مساوی باشد، آن را داخل stack قرار می‌دهیم.

در غیر این صورت آن قدر از بالای stack عملگر خارج کرده و در خروجی می‌نویسیم تا یا خالی شده و یا به عملگری رسیدیم.

که بتوانیم عملگر مورد نظر را روی آن در stack قرار دهیم.

یادآوری:

باتوجه به مورد (د)، عملگر $-$ و $*$ می‌توانند روی $+$ یا $-$ قرار گیرند، همین‌طور $*$ و $/$ می‌توانند روی $*$ یا $/$ قرار گیرند. اما توان \uparrow روی توان \uparrow

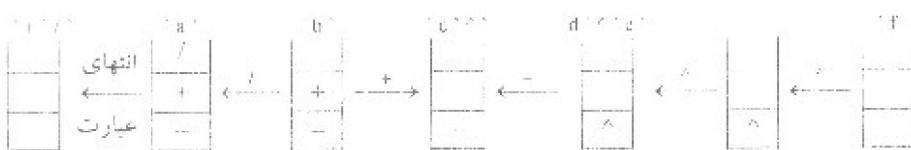
بسیاری توانند قرار گیرد. چون اولویت توان‌های پشت سرهم از راست به چپ بررسی می‌شود.

ه) اگر بالای stack برانزه باشد هر عملگری به راحتی روی آن قرار می‌گیرد.
و) به هر پرنتز $()$ که رسیدیم آن خود از stack عملگر خارج کرده و در خروجی می‌نویسیم تا به $()$ برسیم در این وضعیت $()$ و $()$ با هم خوش می‌شوند.

ر) زمانیکه به این‌دست عبارت رسیدیم (چون عبارات را از راست به چپ بررسی می‌کنیم) در صورتیکه stack خالی باشد؛ آن را به طور کامل در خروجی می‌نویسیم.

نکته: تنها مورد متفاوت (همه) در دو روش infix و postfix به prefix به فرمت (d) در هر دو روش مربوط می‌شود. زمانیکه تبدیل به postfix را بررسی می‌کنیم چون عبارت از جب به راست پوشش می‌شود. به طور مثال در $c - a + b - c$ نمی‌تواند روی $-$ قرار گیرد چون اولویت عملگرهای هم اولویت از جب به راست است و بنابراین اولویت $-$ بیشتر است. اما در تبدیل prefix چون عبارت از راست به چپ پوشش می‌شود، در $c - a + b - c$ نمی‌تواند روی $-$ قرار گیرد چون اولویتش از $-$ بیشتر است. برای همین در تبدیل prefix عملگرهای هم اولویت می‌توانند روی هم قرار گیرند.

مثال ۱: حداقل اندازه بیشتر برای تبدیل عبارت مینووندی $f = a / b + c - d ^ e ^ f$ به عبارت پیشوندی معادل کدام است؟



نتیجه از جب به راست: $a b / c + d e ^ f ^ -$ خواهد بود.

تشخیص صحیح عبارت ریاضی از نظر اطلاعات (یا [] یا []) :

تشخیص صحیح عبارت ریاضی با توجه به علامت (یا [] یا []) با توجه به شرایط زیر حاصل می‌شود:

۱- تعداد ("(" و ")") و تعداد "[" و "]" و تعداد "(" و ")" برابر باشد.

۲- تطبیق علامت ("(" و ")") و ("[" و "]") مطوف این است که برای بستن ("(" از ")" یا "[]" از "]") استفاده نشده باشد و حتی از "(" استفاده گردد باشیم.

مثال:

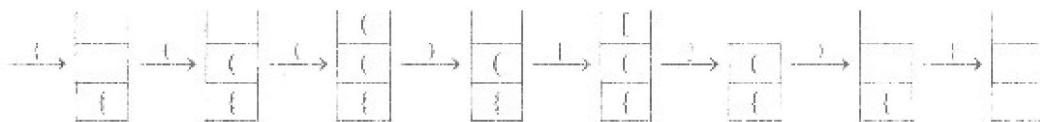
$$\text{عبارت صحیح} \quad \{ ((a + b) + [c / d]) - e \}$$

$$\text{عبارت ناصحیح} \quad \{ ((a + b) + [c / d])] - e \}$$

در عبارت فوق انتهای عبارت اضافی بوده و همچنین برای بستن ("(" و بستن)) از ("(" استفاده شده که غشانده شده عدم تطبیق علامت است.
نحوه کار: عبارت را از سمت چپ به راست پوشش کرده و علامت $\{$ و $\}$ را داخل stack قرار می‌دهیم تا به علامت $\}$ یا $)$ برسیم در این $\{$ برسیم در این صورت حال اگر بالای stack علامت $\}$ یا $)$ مطابق با علامت $\{$ یا $($) وجود داشته باشد علامت را از stack خارج می‌کنیم در غیر این صورت عملیات را متوقف کرده و عبارت صحیح نیست در عین حال اگر به انتهای عبارت رسیدیم و stack خالی باشد عبارت صحیح است در غیر این صورت علامت اضافی در stack وجود داشته و عبارت صحیح نیست

مثال ۱: حداقل فضای پشته برای تشخیص صحت عبارت رویرو کدام است؟

$$\{((a - b) + c - [d * e])\}$$



➡️ حداقل فضای مورد نیاز ۳ بوده که همان حداقل فضای استفاده شده است.

➡️ در انتهای عبارت stack خالی شده است بنابراین عبارت صحیح است.

➡️ تعداد push با تعداد pop برابر بوده و برابر تعداد علامتی از { و } و) که وارد stack شده‌اند می‌باشد.

مثال ۲: حداقل فضای پشته برای تشخیص صحت عبارت رویرو کدام است؟

$$\{((a + b) + c)\}$$



➡️ حداقل فضای مورد نیاز ۳ بوده که همان حداقل فضای استفاده شده است.

➡️ زمینکه به ")" چون بالای (" stack " است بنابراین با عدم تطبيق رویرو می‌شویم و عبارت ناصحیح است.

➡️ تعداد push برابر ۳ برابر علامت { و } و) و تعداد pop ۱ و فقط برای (انجام می‌شود. و با رسیدن به [عملیات متوقف می‌شود.

مثال ۳: حداقل فضای پشته برای تشخیص صحت عبارت زیر کدام است؟

$$[\{((a + b) + c)\}]$$



در انتهای عبارت برای "(" علامت ")" در stack وجود ندارد بنابراین عبارت غلط است.

➡️ حداقل فضای مورد نیاز ۳ بوده که همان حداقل فضای استفاده شده است.

➡️ تعداد push = 3 و تعداد pop = 3 اما در انتهای عبارت "(" علامت در stack وجود ندارد و عبارت ناصحیح است.

تبدیل عبارات infix و postfix به عبارات prefix

۱- روش پرانتز گذاری

۲- استفاده از stack

برای این عمل ۲ روش وجود دارد:

۱- روش پرانتز گذاری

(الف) بتدیل می کنیم که آیا تعداد عملوند ها یک و واحد از عملگرها بیشتر است، چون در غیر این صورت در عبارت عملگر یکتی وجود داشته و ارزشیابی آن شرایط خاصی خواهد داشت.

(ب) عبارت را از سمت چپ و در postfix از چپ و در infix از راست عبارت را بررسی می کنیم.

(ج) با رسیدن به هر عملگر دو عملوند برای آن در نظر می گیریم (در postfix سمت چپ عملگر و در prefix سمت راست عملگر)

(د) برای دو عملوند عملگر را بین آنها قرار داده و برای آنها پرانتز در نظر می گیریم.

(ه) عبارت را از سمت چپ به راست به ترتیب شامل عملوند ها و عملگر های بین عملوند ها ارزیابی می کنیم.

مثال ۱: عبارت را از سمت عملوند ها (چپ) بررسی می کنیم.

 $A, B, C, *, D, /, +$

$$\longrightarrow \left(A + , \left(\left(B ^ * , C , \right) / , D , / \right) , + \right) \longrightarrow A + B ^ * C / D$$

مثال ۲: عبارت را از سمت عملوند ها (راست) بررسی می کنیم.

 $+, -, a, /, b, c, d$

$$\longrightarrow \left(+, \left(-, a ^ - , \left(/, b ^ / , c \right) \right) ^ + , d \right) \longrightarrow a - b / c + d$$

مثال ۳: عبارت prefix زیر داده شده است: (کارشامی ارشد - آزاد ۷۹)

 $+ - * \uparrow ABCD/E/F+GH$

کدامیک از عبارات زیر معادل infix برای این عبارت است؟

$$A^B * (C - D) + \frac{EF}{G} + H \quad (۱)$$

$$A^{B*(C-D)} + E\left(\frac{F}{G+H}\right) \quad (۲)$$

$$A^B * C - D + E / \left(F / \left(G + H \right) \right) \quad (۳)$$

$$A^B * C - D + E / F / G + H \quad (۴)$$

حل: از سمت عملوند ها (راست) عبارت را بررسی می کنیم:

$$\left(+ \left(- \left(* \left(^ \uparrow A \uparrow B \right) ^ * C \right) ^ - D \right) ^ + \left(/ E ^ / \left(F ^ / \left(+ G ^ - H \right) \right) \right) \right)$$

$$= A ^ \uparrow B ^ * C - D + E / \left(F / \left(G + H \right) \right)$$

نحوی نه صدحیع می باشد.

۲- روش stack (پشته)

با فرض آن که عملگرها باینری (دودوئی) هستند (تعداد عملگرهای از عملوندهای یک واحد کمتر باشد) به صورت زیر عمل می‌کیم:

(الف) عبارت را از سمتی بررسی می‌کنیم که با عملوند شروع می‌شود، در از چپ و در postfix از راست عبارت را بررسی می‌کنیم.

ب) عملوندها را در stack قرار می‌دهیم و با رسیدن به هر عملگر (به جز عملگر آخر) دو عملوند خارج کرده و بعد از محاسبه دوباره

حاصل عبارت را داخل stack قرار می‌دهیم. (دقیقت کنید که دو عملوندی که از stack خارج می‌کنیم از سمت چپ به راست عمل می‌کنیم)

ج) به عملگر آخر که رسیدیم بعد از خارج کردن دو عملوند آخر و محاسبه آن را دیگر در stack قرار نمی‌دهیم و حاصل را در خروجی

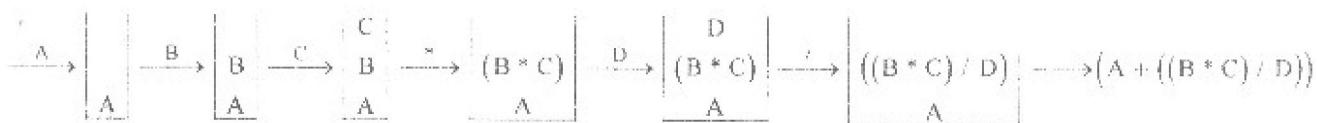
می‌نویسیم.

نکته: تعداد push و pop با هم برابر بوده و برابر با ۲ برابر تعداد عملگرهای می‌باشد.

مثال ۱:

$A, B, C, *, D, /, +$

از سمت چپ



نکته مهم:

۱- حداقل فضای مورد نیاز برای تبدیل بالا ۳ می‌باشد که همان حداقل فضای مورد استفاده است.

۲- تعداد $\text{push} = \text{pop} = \text{تعداد عملگرها}$

مثال ۲: می‌بینیم تعداد متغیرهای مباني در محاسبه عبارت جبری $a \cdot b + c \cdot d \cdot (a + b) / (c \cdot d)$

۴ (۴)

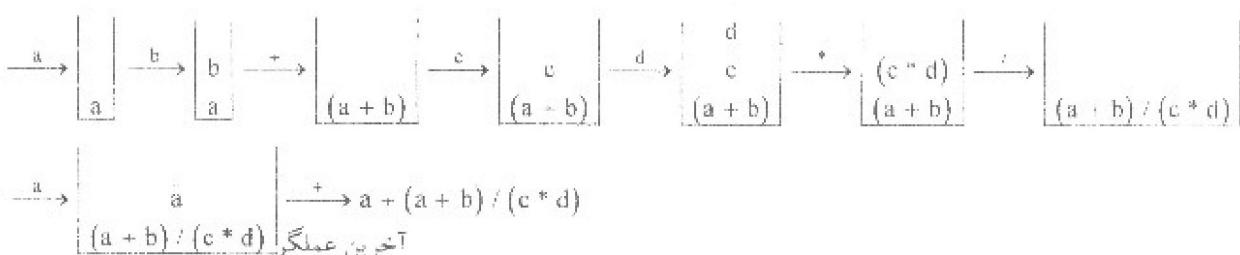
۳ (۳)

۲ (۲)

۱ (۱)

حل: گزینه ۳ صحیح می‌باشد.

از سمت چپ بررسی می‌کنیم:



۱- حداقل فضای مورد نیاز برای تبدیل ۳ می‌باشد که همان حداقل فضای مورد استفاده است.

۲- تعداد $\text{push} = \text{pop} = \text{تعداد عملگرها}$



تبدیل infix به postfix و بالعکس:

۱- اگر در هنگام تبدیل infix به prefix به روش پرانتزگذاری به جای فرار دادن عملگر بین عملوندها، عملگر را روی پرانتز بسته بگذاریم به فرم postfix می‌رسیم.

۲- اگر در هنگام تبدیل infix به postfix به روش پرانتزگذاری به جای فرار دادن عملگر بین عملوندها، عملگر را روی پرانتز باز بگذاریم به فرم prefix می‌رسیم.

مثال ۱: معادل لهستانی عبارت پسوندی زیر کدام است؟

A B C * D / + (لهستانی وارونه)

حل: از سمت عملوندها عبارت را بررسی می‌کنیم (سمت چپ)

$$= ^+ \left(A / \left(^* (B C ^*) D / \right) + \right)$$

$$= + A / ^* B C D$$

مثال ۲: معادل postfix عبارت پیشوندی زیر کدام است؟

+ a / b ^ + d e - a c

حل: از سمت عملوندها عبارت را بررسی می‌کنیم (سمت راست)

$$= \left(+ a \left(/ b \left(^* (+ d e) ^+ (- a c) ^- \right) ^* \right) ^/ \right) ^+$$

$$= a b d e + a c - ^* / +$$

مثال: معادل Prefix Postfix عبارت زیر کدام است؟ (مسابقات آموزشکده‌های فنی ۷۸)

+ a / b - cd / - ab - + c * de / a - bc

$$ab - cd - ab - + / cde * + / abc - / -$$

$$ab + cd - / ab - cd + / e * + ab / c - - + /$$

$$abcd / - - abc - de * abc - - / - / +$$

$$abcd - / + ab - cde * + abc - / - / +$$

حل: از سمت عملوندها عبارت را بررسی می‌کنیم:

$$\left[\left(+ \left(+ a \left(/ b \left(- c d \right) ^- \right) ^/ \right) ^+ \left((- a b) ^- \left(- \left(+ c \left(^* d e \right) ^* \right) ^+ \left(/ a \left(- b c \right) ^- \right) ^/ \right) ^/ \right) ^+ \right] ^+$$

صف (Queue)

هر ساختهای از داده‌های پشت سرهم ذخیره شده که در آن عمل درج زیر یک طرف (نهای) و عمل حذف از طرف دیگر (ابتداء) انجام می‌شود. نکته از به ساختهای صف FIFO (First In First Out) LIFO (Last In Last Out) نیز گفته می‌شود و در بعضی مراجع

گاربرد صف

۱- در صف پردازش‌های سیستم عامل

۲- صف چاپگر

نکته ۳- برای نسبش یک صف از دو ساخته

۱- آرایه

۲- لیست پیوندی

استفاده می‌شود که نمایش پیوندی در بخش لیست پیوندی (link list) (شرح داده می‌شود).

نکته ۴- در صورتیکه از یک آرایه $n \times n$ برای نسبش یک صف استفاده شود همیشه از دو اشاره‌گر Front و Rear که به ترتیب به ابتداء و نهایی صف اشاره می‌کند استفاده می‌کنیم.



: همیشه به خانه قبل از خانه اول صف اشاره می‌کند.

: همیشه به خانه آخر صف اشاره می‌کند.

با توجه به شکل دیده می‌شود که

صف خطی

هر صف خطی یک آرایه $n \times n$ $[1 \dots n] Q$ که عناصر از ابتدای صف (Front) حذف شده و به انتهای صف (rear) اضافه می‌شوند. و حرکت عناصر برای درج به سمت جلو است در عین حال عمل حذف نیز خانه‌ای خالی در ابتدای آرایه بجاد می‌کند که با توجه به حرکت عناصر برای درج به سمت جلو امکان استفاده از این خانه‌ها وجود ندارد.

مثال:



۱) صف 6 عضوی روبرو را در نظر بگیرید.



(۳) حذف ۲ داده (به صور عادی دادهها از اینجا حذف می شوند).

در وضعیت بالا باین که 2 داده از نتیجه حذف شده‌اند اما فقط دو خانه 5، 6 برای درج وجود دارند چون حرکت rear به سمت جلو پردازید امکان استفاده از خانه‌های قبل را ندارد.

شیوه ایندکس ۴۰۰ و اولمه (۱۹۷۰)

همشه به خانه قای از خانه اول عصف اشاره کند.

جامعة الملك عبد الله للعلوم والتقنية (King Abdullah University of Science and Technology)

انتخاب آرایه [1..n] Q ری بادوسری صفت خطه، دارای شرط بسط، ویله و مرزی زیر است:

پیر بودن صرف خطی

خانم بـ دن صحف سخن

خیانت اولیہ صدیق

car == m

front = rear

front - rear = 0

الكتور يقين درج در حف

```

procedure insert (item :  $\Sigma^*$  );
begin
  if rear = n then
    write ('Queue Full')
  else
    begin
      rear := rear + 1;
      Q [rear] := item;
    end;
end;

```

ویژه می شود که جون `rear` به خانه آخر صف انتاره می کند برای عمل درج ابتدا `rear` یک واحد به جلو حرکت کرده میس داده در خانه خانه ای آخر صف درج می شود.

الگوریتم حذف از پشت

```

procedure delete (item : نوع);
begin
  if front = rear then
    write ('Queue Empty')
  else
    begin
      front := front + 1;
      item := Q [front];
    end;
end;

```

دیده می‌شود که چون front همیشه به قبل از اول صفحه اشاره می‌کند برای عمل حذف ابتدا front ۱ واحد به جلو حرکت کرده سپس داده ابتدای صفحه را خارج کرده در item فرار می‌دهد.

مشکل صفحه‌های خطی

مشکل هر صفحه خطی حرکت تدریجی عناصر به سمت تنهای صفحه و عدم امکان استفاده از خانه‌های ابتدای صفحه که در اثر حذف خالی شده‌اند.

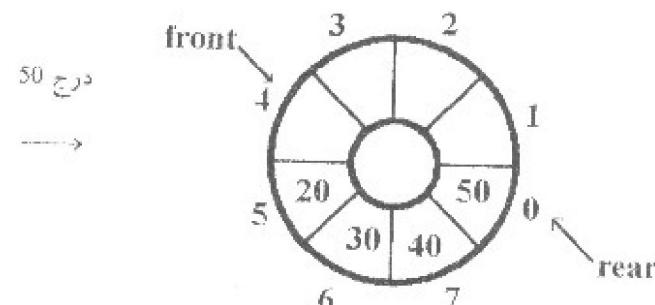
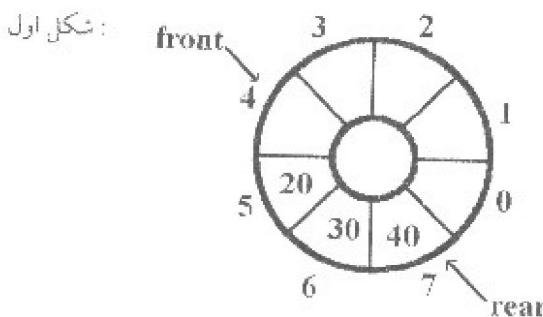
راه حل رفع مشکل صفحه‌های خطی:

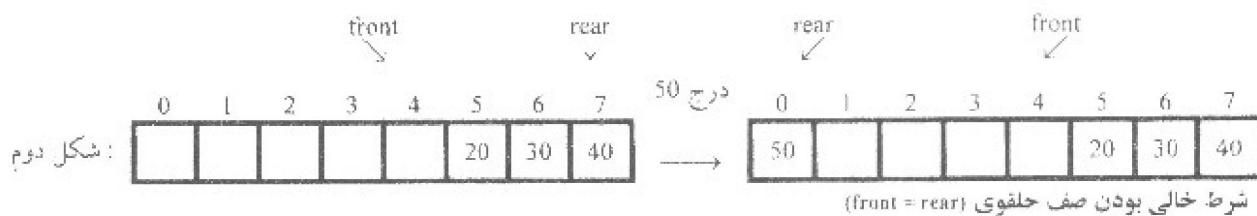
۱- شیفت خانه‌های انتهای صفحه به سمت انتهای صفحه که این عمل در صورتی که تعداد خانه‌های صفحه زیاد باشد هزینه بالایی خواهد داشت.

۲- استفاده از صفحه حلقوی به این مفهوم که زمانی که به انتهای صفحه رسیدیم عمل درج را دوباره از ابتدای صفحه انجام می‌دهیم و این حرکت بجز خشی می‌باشد.

صف حلقوی

آرایه ای از $[0..n-1]$ را می‌توان به صورت یک صفحه حلقوی در نظر گرفت به صورتی که در این صفحه زمانیکه rear برابر n می‌شود، عنصر بعدی در خانه ۰ فرار می‌گیرد.





به قبل از اول صف اشاره کند، در صف حلقوی شرط خالی بودن صف (front = rear) خواهد بود.

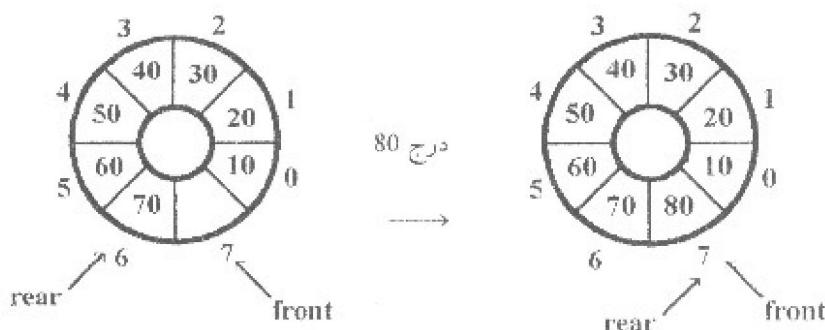
به خانه آخر صف اشاره کند

در شرایطی که

در هنگام شروع کار با صف هنگامی که صف حلقوی خالی بانشد، $front = rear = 0$ است که یکی از حالت‌های خالی بودن صف در حین کار با صف می‌باشد.

نکته مهم: در هر صف حلقوی n عضوی همیشه یک خانه خالی وجود دارد، و حداقل از $1 - n$ خانه صف «ی» نوانیم استفاده کیم، این به آن علت است که بتوانیم وضعیت پر با خالی بودن صف حلقوی را تشخیص بدهیم.

در صورتیکه در یک صف حلقوی n عضوی از همه خانه‌های صف بخواهیم استفاده کیم و یک خانه را خالی نگذاریم وضعیت پر با خالی بودن صف حلقوی قابل تشخیص نخواهد بود.



در مثال بالا بعد از درج 80 در صف حلقوی و استفاده از همان یکی خانه‌ای که باید در صف خالی می‌ماند $front = rear = 7$ می‌شود که همان شرط خالی بودن صف است در صورتیکه صف بر شده و این یک تناقض است برای نبودن تناقض فوق و امکان تشخیص وضعیت پر با خالی بودن صف حلقوی یک خانه را خالی نگه می‌داریم.

شرط پر بودن صفت حلقوی $((rear - 1) \mod n = front)$

در صورتیکه صفت حلقوی n عضو نوسط آزایه $[0 .. n - 1]$ پیاده‌سازی شود، شرط پر شدن صفت حلقوی با در نظر گرفتن این نکته که حد اکثر از $1 .. n - 1$ خانه می‌توانیم استفاده کنیم و همیشه یک خانه صفت باید خالی بماند از رابطه زیر بدست می‌آید:

شرط پر بودن صفت حلقوی

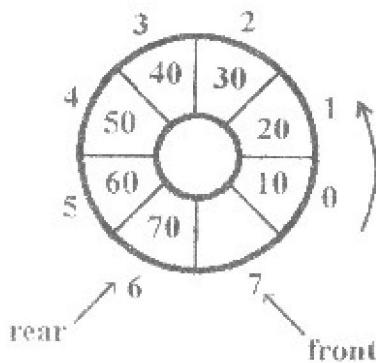
$$1) (rear + 1) \mod n = front$$

یا

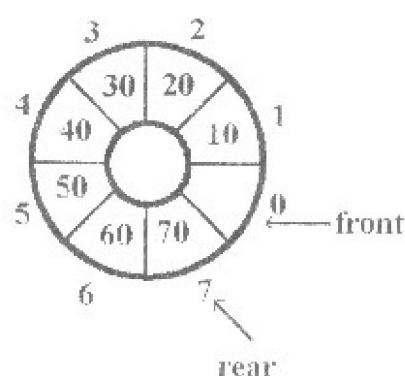
$$2) ((rear + 1) = front) \text{ or } (rear = n - 1, front = 0)$$

مثال: در صفت حلقوی زیر با $n = 8$ خانه:

(الف)



(ب)



الف) در این وضعیت $\left. \begin{array}{l} rear = 6 \\ front = 7 \end{array} \right\}$ و $((rear + 1) \mod n = front)$ می‌شود صفت پر است.

$$(6 + 1) \mod 8 = 7$$

↓ ↓ ↑
rear n front

ب) در این وضعیت $\left. \begin{array}{l} rear = 7 \\ front = 0 \end{array} \right\}$ و $((rear + 1) \mod n = front)$ می‌شود صفت پر است.

$$(7 + 1) \mod 8 = 0$$

↓ ↓ ↑
rear n front

رابطه ۱ یعنی $((rear + 1) \mod n = front)$ رابطه‌ای است که هر دو وضعیت رابطه ۲ را پوشش می‌دهد. برای همین کامترین رابطه می‌باشد.

علت استفاده از \mod در رابطه بالا در شرایطی دیده می‌شود که در وضعیت عالی‌تر حالت تکل (ب) قرار داریم که با حرکت $rear$ به اندازه واحد به جو باید از انتهای ایندا برگردیم.

نتیجه گیری:

شرط پر و خالی بودن صفت حلقوی n عضوی:

{ شرط خالی بودن: $front = rear$

شرط پر بودن: $((rear + 1) \mod n = front)$



مثال: کدام مورد در ساختار یک صف حلقوی 1000 عضوی بیان گشته خالی و پر بودن صف است؟ (کنکور ۸۳)

(۱) front = (rear + 1) mod n front = 0, rear = 0 (خالی) و پر بودن صف (صحیح)

(۲) front = 0, rear = 1000 (خالی) و front = 0, rear = 0 (پر بودن صف)

(۳) rear = (rear + 1) mod 1000 front = 0, rear = 1 (خالی) و rear = 0 (پر بودن صف)

(۴) rear = 999, front = 0 (خالی) و front = 1000, rear = 1001 (پر بودن صف)

} گزینه ۱ صحیح می باشد.

خالی بودن front = rear = 0
پر بودن صف front = (rear + 1) mod n

خالی بودن صف (صحیح) front = rear = 0

پر بودن صف (غلط) front = 0, rear = 1000

گزینه ۲ غلط می باشد.

(۵) $(\text{rear} + 1) \mod n = (\underbrace{1000 + 1} \mod 1000) \neq \underbrace{\text{front}}_0$

خالی بودن صف (غلط) front = 0, rear = 1

پر بودن صف (غلط) rear = (rear + 1) mod 1000

گزینه ۳ غلط می باشد.

خالی بودن صف (غلط) front = 1000, rear = 1001

پر بودن صف (صحیح) rear = 999, front = 0

گزینه ۴ غلط می باشد.

$\frac{(\text{rear} + 1) \mod n}{(999 + 1) \mod 1000} = 0 = \text{front}$

عملیات حذف از صف حلقوی

procedure delqueue (item):

begin

if front = rear then

write ('Empty')

else

begin

front := (front + 1) mod n;

item := Q [front];

end;

end;

عملیات درج در صف حلقوی

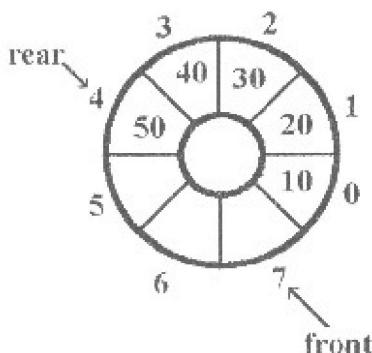
```

procedure Addqueue (item)
begin
if  $(\text{rear} - 1) \bmod n = \text{front}$  then
    write ('Full')
else
begin
     $\text{rear} := (\text{rear} + 1) \bmod n;$ 
    Q [rear] := item;
end;

```

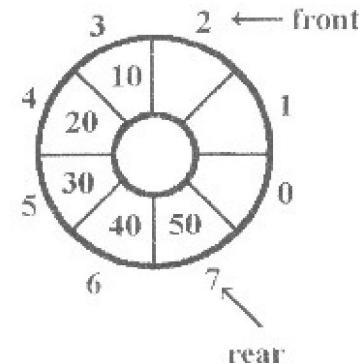
در هر دو روش حذف و درج به ترتیب از جملات $\text{rear} := (\text{rear} + 1) \bmod n$ و $\text{front} := (\text{front} + 1) \bmod n$ است که علت استفاده از \bmod در شرایطی است که rear یا front به خانه آخر اشاره کرده باشد و حرکت به جلو آنها را به ابتدای صف منتقل می‌کند.

این حالت‌ها در شکل زیر دیده می‌شود.



$$\text{front} := \frac{(\text{front} + 1)}{8} \bmod 8;$$

برای حذف یک واحد به جلو حرکت کرده به 0 اشاره می‌کند



$$\text{rear} := \frac{(\text{rear} + 1)}{8} \bmod 8;$$

Rear برای درج یک واحد به جلو حرکت کرده به 0 اشاره می‌کند

سبس مقدار جدید در فضای خالی درج می‌شود.

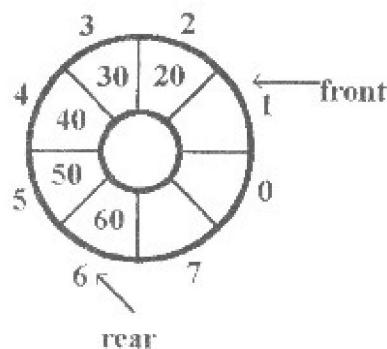
تعداد خانه‌های پر و خالی یک صف حلقوی (n-1)

یه قبل ز اول صف اشاره می‌کند.

ز rear به انتهای صف اشاره می‌کند.

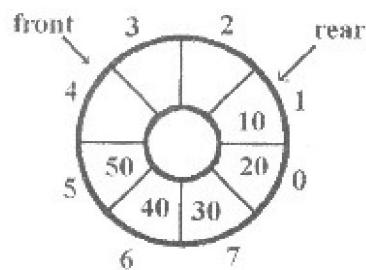
(الف)

$$\text{rear} \geq \text{front} \rightarrow \left\{ \begin{array}{l} \text{rear} - \text{front} : \text{تعداد خانه‌های پر} \\ n - (\text{rear} - \text{front}) : \text{تعداد خانه‌های خالی} \end{array} \right.$$



$$\left. \begin{array}{l} n = 8 \\ rear = 6 \\ front = 1 \end{array} \right\} \longrightarrow \begin{array}{l} \text{تعداد خانه‌های پر: } rear - front = 6 - 1 = 5 \\ \text{تعداد خانه‌های خالی: } n - (rear - front) = 8 - 5 = 3 \end{array}$$

$$\text{rear} < \text{front} \longrightarrow \begin{cases} \text{تعداد خانه‌های پر: } n - (\text{front} - \text{rear}) \\ \text{تعداد خانه‌های خالی: } \text{front} - \text{rear} \end{cases}$$



$$\left. \begin{array}{l} n = 8 \\ rear = 1 \\ front = 4 \end{array} \right\} \longrightarrow \begin{array}{l} \text{تعداد خانه‌های پر: } n - (\text{front} - \text{rear}) = 8 - 3 = 5 \\ \text{تعداد خانه‌های خالی: } \text{front} - \text{rear} = 4 - 1 = 3 \end{array}$$

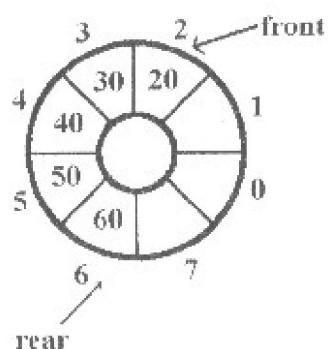
به ابتدای صفحه می‌کند front

به انتهای صفحه می‌کند rear

ب)

$$\text{rear} \geq \text{front} \longrightarrow \begin{cases} \text{تعداد خانه‌های پر: } \text{rear} - \text{front} + 1 \\ \text{تعداد خانه‌های خالی: } n - (\text{rear} - \text{front} + 1) \end{cases}$$

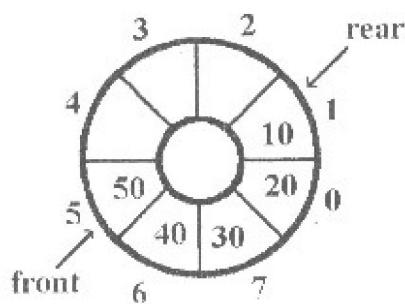
$$\left. \begin{array}{l} n = 8 \\ front = 2 \\ rear = 6 \end{array} \right\} \longrightarrow \begin{array}{l} \text{تعداد خانه‌های پر: } 6 - 2 + 1 = 5 \\ \text{تعداد خانه‌های خالی: } 8 - 5 = 3 \end{array}$$



$\text{rear} < \text{front} \longrightarrow$

$$\left\{ \begin{array}{l} \text{تعداد خانه‌های پر}: n - (\text{front} - \text{rear} - 1) \\ \text{تعداد خانه‌های خالی}: (\text{front} - \text{rear} - 1) \end{array} \right.$$

$$\left. \begin{array}{l} n = 8 \\ \text{front} = 5 \\ \text{rear} = 1 \end{array} \right\} \longrightarrow \left. \begin{array}{l} \text{تعداد خانه‌های پر}: 8 - 3 = 5 \\ \text{تعداد خانه‌های خالی}: 5 - 1 - 1 = 3 \end{array} \right.$$



لیست‌های پیوندی (Linked list)

به طور کلی برای ذخیره‌سازی انواع داده‌ها در حافظه احصی (RAM) از دو نوع ساختار می‌توان استفاده کرد:

۱- آرایه‌ها

۲- لیست‌های پیوندی

■ آرایه

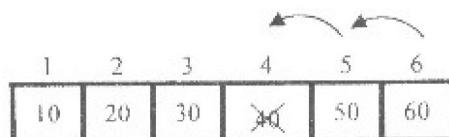
هر آرایه لیست یشت سره‌همی از داده‌ها است که همگنی داده‌ها از یک نوع بوده و ناحیه‌ای پیوسته از حافظه را در اختیار دارد.
نکات مهم در آرایه‌ها:

۱- تعداد عناصر هر آرایه همیشه محدود و مشخص و ایست است و در تمام طول برنامه ثابت می‌باشد. (عیب)

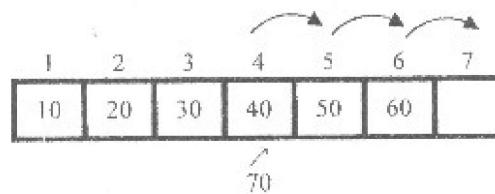
۲- عملیات حذف و درج یک داده دلخواه در ناحیه‌ای مشخص نیاز به شیفت دارد. (عیب)

مثال: ۱- (الف) درج یک داده در محل k ام یک آرایه N تائی نیاز به $1 + N - k$ شیفت دارد.

۲- حذف یک داده از محل k ام یک آرایه N تائی نیاز به $k - N$ شیفت دارد.



به طور مثال برای حذف 40 از محل 4 ام نیاز به $4 - 6 = 2$ شیفت است.



به طور مثال برای درج 70 در محل 4 ام نیاز به $4 - 6 = 3$ شیفت داریم.

پادآوری:

عملیات همراه با شیفت عناصر همیشه هزینه بالاتری دارد.

کلکر زمان حاصل از شیفت برای درج و حذف $O(n)$ است، چون به طور متوسط برای درج $\frac{n+1}{2}$ و برای حذف $\frac{n-1}{2}$ عمل شیفت نیاز داریم.

۳- عملیات جستجو به دنبال یک داده دلخواه در آرایه‌ها با بکی از دورش زیر تجمم می‌گیرد (تریت)

(الف) جستجوی خطي (تریتی) در آرایه‌های مرتب یا تامرت با زمان $O(n)$

(ب) جستجوی دودونی (باینری) در آرایه‌های مرتب با زمان $O(\log_2 n)$



۳- مزیت این خصوصیت، تنوع روش‌های جستجو در آرایه است.

۴- دسترسی به داده‌های هر آرایه به صورت تصادفی و دلخواه به راحتی توسط اندیس آرایه انجام شده و دسترسی مستقیم و با زمان (O) ۰ خواهد بود. (مزیت)

۵- روش‌های مختلفی برای مرتب‌سازی آرایه‌ها از قبیل: مرتب‌سازی حبابی - مرتب‌سازی انتخابی - مرتب‌سازی سریع - مرتب‌سازی درجی - ... وجود دارد. (مزیت)

■ لیست‌های پیوندی

نیست پیوندی؛ دارای عناصر (رکوردها) مختلفی از حافظه پویا (Heap) بوده که لزوماً عناصر آن کنار هم قرار نگرفته‌اند.

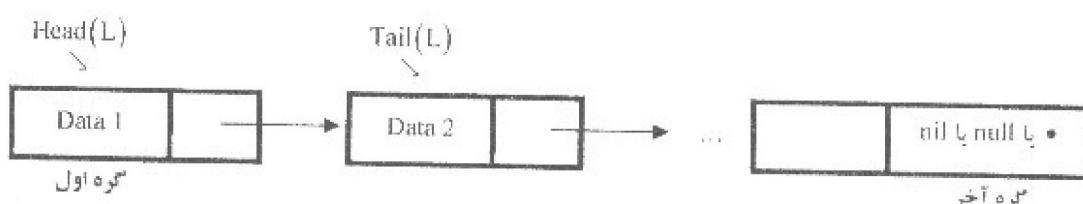
نکته:

کلید هر عنصر از یک لیست پیوندی که معمولاً یک رکورد است به نام گره یا Node شناخته می‌شود و حداقل دارای ۲ فیلد (قسمت) است

۱- داده (Data)

۲- اشاره گر (link)

که اشاره گر آدرس عنصر بعدی از لیست پیوندی را نگهداری می‌کند.



: ۷۶۰

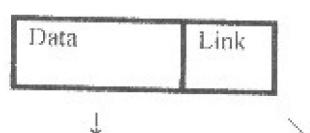
کلید اشاره گر (link) گره آخر معمولاً با nil یا null یا * مقداردهی می‌شود.

کلید Head (L) تابعی که اشاره گر به گره اول نیست را برمی‌گرداند.

کلید Tail (L) تابعی که اشاره گر به گره بعد از گره اول نیست را برمی‌گرداند.

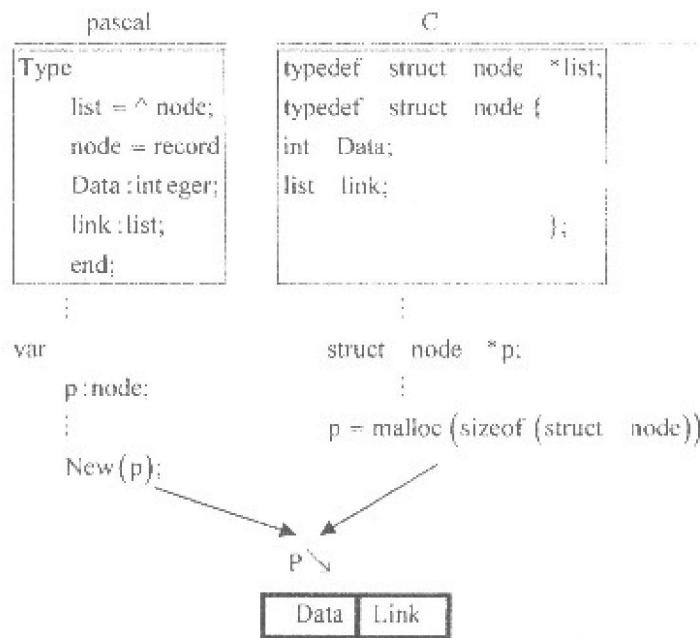
نماینده شکل کلی هر گره در ساختار لیست پیوندی در زبان‌های C و Pascal:

در صورتیکه هر گره (node) به صورت:



اشارة گر به گره بعدی از نوع صحیح

باشد برای پیاده‌سازی آن در زبان‌های برنامه‌سازی C و pascal به شکل زیر عمل می‌کنیم:



پیاده‌سازی یک گره به زبان الگوریتم به صورت زیر است:

1) if Avail = Null then write 'overflow' and exit

اگر فضای خالی در heap وجود نداشته باشد Avail = null شده و پیغام overflow ظاهر می‌شود. در غیر این صورت به خط بعدی می‌رویم و تخصیص انجام می‌شود.

2) $p = Avail$, $Avail = link(Avail)$

در صورتیکه فضای خالی در Heap وجود داشته باشد، Avail به عنوان یک گره آزاد در اختیار p قرار می‌گیرد. در این حالت گره مورد نظر می‌تواند نویسندگان p مورد دسترسی واقع شود.

■ نحوه دسترسی به فیلدات گره دلخواه

در صورتیکه p را اشاره گری به رکورد (گره یا node) دلخواه بدانیم در این صورت نحوه دسترسی به فیلدات آن گره (node) فقط از طریق p به صورت زیر خواهد بود:

زبان الگوریتمی	زبان C	زبان پاسکال
Data(p)	$P \rightarrow Data$	$P^.Data$
link(p)	$P \rightarrow link$	$P^.link$
	$(* P).Data$	
	$(* P).link$	

در بعضی مراجع یا تست‌ها به جای Data از کلمه info استفاده می‌کنند.

مثال: برای آماده‌سازی گره p به صورت 20 nil می‌توانیم به صورت زیر عمل کنیم.

زبان C

زبان پاسکال

زبان الگوریتمی

 $P \rightarrow Data = 20$ $P^.Data := 20;$ $Data(P) = 20,$ $P \rightarrow link = null$ $P^.link := nil;$ $link(P) = 0;$

• $null$ و nil یک مفهوم را دارند.

عملیات و انتساب‌ها در اشاره‌گرهای لیست‌های پیوندی (مهمن)

اگر p و q دو اشاره‌گر به گره‌های یک لیست پیوندی باشد وضعیت‌های زیر بروز می‌کنند:

(الف)

$p = q$ اشاره کند به همان محلی که q اشاره می‌کند.



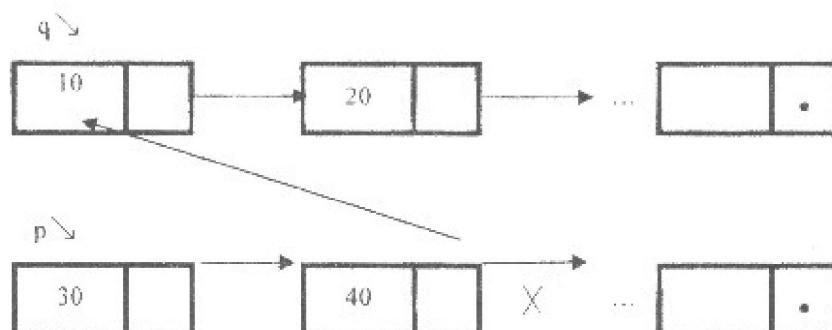
(ب)

$$p := q^.link^.link;$$

$$\frac{20}{30}$$

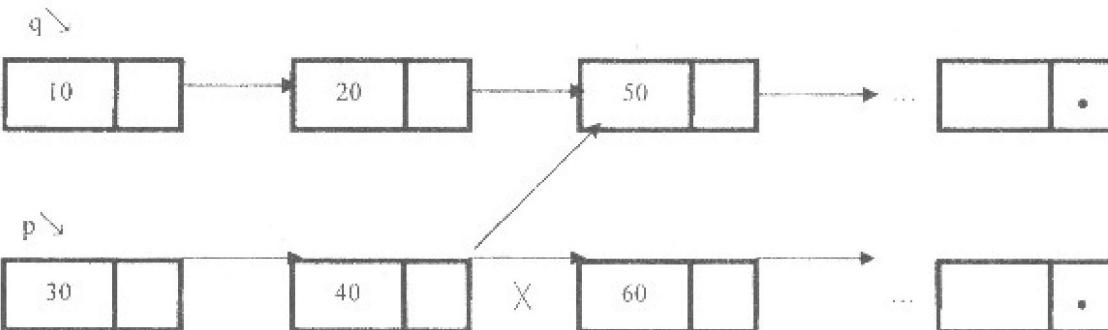

(ج)

$$p^.link^.link := q;$$

$$\frac{40}{40}$$


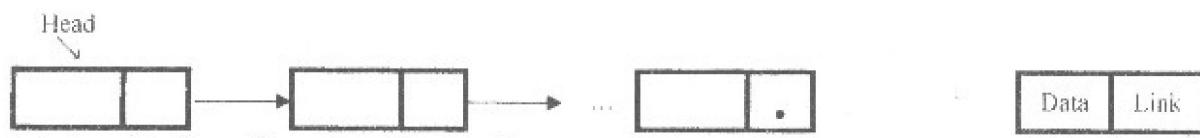
$$p \wedge link \wedge link = q \wedge link \wedge link;$$

40 20
 ↓ ↓
 50

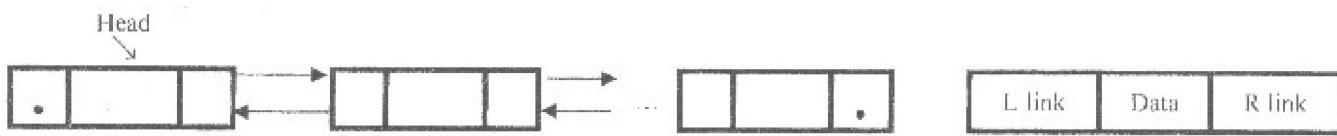


دیده می‌شود که هنگام انتساب اشاره گرهای لیست پیوندی در صورتیکه در سمت راست حکم انتساب مجموعه‌ای از linkها وجود داشته باشد، همه linkها محاسبه می‌شود ولی در سمت چپ حکم انتساب آخرین link را محاسبه نمی‌کنیم.
لکنه: لیست‌های پیوندی می‌توانند به دو شکل عمومی یک طرفه و دو طرفه تعریف و استفاده شوند.

۱- لیست پیوندی یک طرفه (خطی): در هر گره فقط یک اشاره گر وجود دارد که آن هم آدرس گره بعدی را دارد.



۲- لیست پیوندی دو طرفه (خطی): در هر گره دو اشاره گر وجود دارد که یکی به گره بعدی و دیگری به گره قبلی اشاره می‌کند.

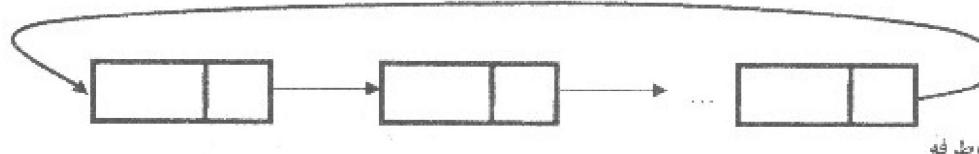


لیست‌های دوری (حلقوی)

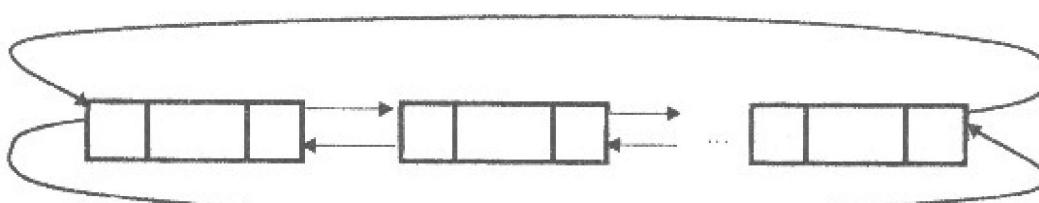
۱- در لیست حلقوی یک طرفه اشاره گر link گره آخر به گره اول اشاره می‌کند.

۲- در لیست حلقوی دو طرفه اشاره گر R link گره آخر به گره اول و اشاره گر L link گره اول به گره آخر اشاره می‌کند.

۱- لیست حلقوی یک طرفه

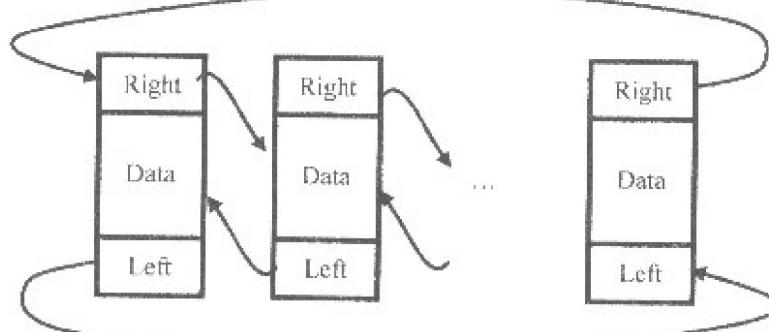


۲- لیست حلقوی دو طرفه



ساختهای دادهها

نکته مهم: بین لیست‌های دوری و غیردوری تفاوتی از نظر انلاف حافظه وجود ندارد، جون در لیست‌های دوری در حقیقت از اشاره‌گر (link) گره‌هایی استفاده می‌شود، که در لیست‌های خطی خطي null با هستند.



نکته مهم: یکی از مزایای مهم لیست‌های دوری یک طرفه نسبت به لیست‌های خطی یک‌طرفه این است که در لیست‌های دوری یک‌طرفه بدون استفاده از حافظه اضافی امکان رسیدن به گره قبلی زیر گره دلخواه وجود دارد که این عمل با یک دور کامل انجام می‌شود.



به صور مثال در لیست خطی بالا امکان رسیدن به گره 10 از موقعیت گره 20 وجود ندارد، در صورتیکه در لیست دوری امکان رسیدن از موقعیت گره 20 به گره 10 با رفتن به انتهای لیست و برگشت به ابتدای لیست وجود دارد.

نکته: بادآوری:

در لیست‌های دوطرفه رسیدن از یک گره به گره قبلی راحت‌تر از لیست ورودی انجام می‌شود، چون از هر گره یک اشاره‌گر link به گره قبل وجود دارد و مانند لیست حلقوی نباری به رفتن به انتهای لیست و برگشت به ابتدای لیست نخواهد بود.

لیست‌های خطی یک‌طرفه

در این نوع لیست پیوندی، هر گره با یک اشاره‌گر link فقط آدرس گره بعدی را نگهداری می‌کند و امکان دسترسی به گره قبل را ندارد، برای همین به آن لیست خطی یک‌طرفه می‌گویند.



نکته: بادآوری:

first را معادل head یا start می‌توانیم در نظر بگیریم.

در این ساختار اشاره گر first به گره اول لیست اشاره می‌کند و فقط از طریق این اشاره گر می‌توانیم به گره‌های لیست دسترسی پیدا کنیم، در این وضعیت اگر first را از دست بدهیم یا به هر نحوی آدرس آن را گم کنیم دیگر امکان دسترسی به عناصر لیست وجود ندارد.

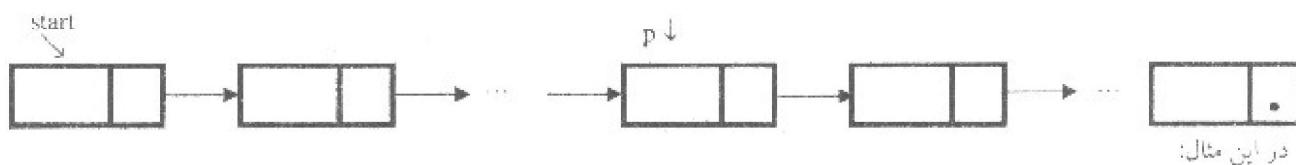
نکات مهم در لیست‌های پیوندی خطی یک طرفه:

۱- امکان پیمایش لیست فقط از Head به وجود دارد، آن‌هم به این علت که نیست یک‌طرفه است، در عین حال امکان پیمایش لیست از Tail به Head وجود ندارد.

۲- در صورتیکه شرط Head = null برقرار باشد، یا به عبارت بهتر متغیر Head مقدار null را داشته باشد لیست نهی خواهد بود، در این حالت ابزاری به کنترل شرط Head = null = Tail نداریم، چون لیست یک‌طرفه است.

۳- از هر گره دلخواه p به راحتی می‌توان به گره بعدی دسترسی پیدا کرد، چون آدرس گره بعدی در اشاره گر link گره p ذخیره شده است، در عین حال امکان دسترسی به گره قبلی از گره p وجود نداشته و برای این کار باید از ابتدای لیست عمل پیمایش را تا رسیدن به گره قبل از p انجام دهیم.

مثال:



در این مثال:

(الف) با استفاده از link → p به راحتی به گره بعد از p دسترسی خواهیم داشت.

(ب) برای رسیدن به گره قبل از p از موقعیت گره p نمی‌توان عمل کرد و باید از ابتدای لیست عمل پیمایش را تا رسیدن به موقعیت قبل از p انجام داد.

```
q = start;
while q^.link <> p do
    q = q^.link;
```

در این قطعه برنامه ابتدا q به گره اول اشاره کرده؛ سپس با شروع کار حلقه نازمانی که به گره‌ای بررسیم که آن گره p باشد، گره‌های لیست توسط q پیمایش می‌شوند؛ هنگام خروج از حلقه q به گره قبل از p اشاره می‌کند.

۴- برای حذف یک گره دلخواه نیاز به ۱ عمل جایگزینی داریم با زمان (1) O، البته برای حذف هر گره باید از موقعیت قبل از آن این عمل را انجام دهیم.

۵- برای درج یک گره دلخواه نیاز به ۲ عمل جایگزینی داریم با زمان (1) O، البته برای درج هر گره باید از موقعیت قبل از آن گره این عمل را انجام دهیم.

۶- برای حذف یک گره دلخواه در صورتیکه نیست خانه‌های حذف شده نیاز باشد، بعد از عمل حذف هر گره باید آن را به لیست مورد نظر اضافه کنیم که در این صورت نیاز به ۳ عمل جایگزینی داریم، ۱ عمل جایگزینی برای حذف و ۲ عمل جایگزینی برای درج در لیست خانه‌های حذف شده نیاز است.

۷. به صور متوسط برای رسیدن به یک گره دلخواه در یک لیست پیوندی یک طرفه n عضوی، نیاز به $\frac{n}{2}$ پیمایش داریم.

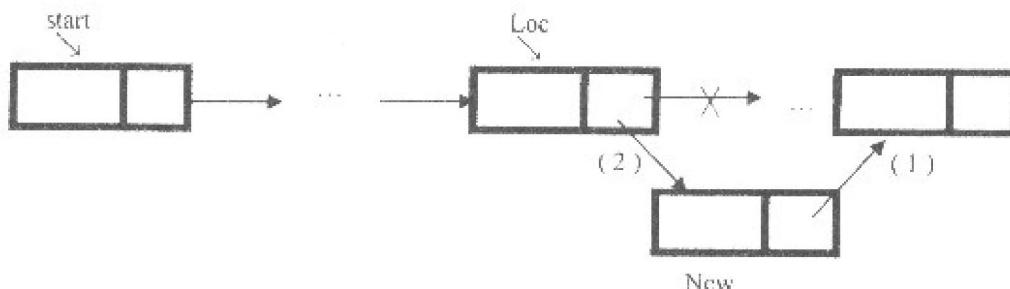
■ درج گره در یک لیست پیوندی یک طرفه

برای درج یک گره دلخواه در موقعیت مشخص باید به قبیل از موقعیت مورد نظر رسیده و با ۲ عمل جایگزینی عمل درج را انجام داد. می خواهیم گره New را بعد از گره معلوم Loc از لیستی که start به ابتدای آن اشاره می کند درج کنیم.

```

New := getnode();
New^.data := item;
if first = nil then
begin
  New^.link := start;
  start := New;
end
else
begin
  1) New^.link := Loc^.link;
  2) Loc^.link := New;
end;
  
```

لیست تهی بوده و گره New به ابتدای لیست اضافه می شود.



در صورتیکه دستورات (1) و (2) را جابجا کنیم فرمی از نیست که start به آن اشاره می کند گم می شود. شکل الگوریتمی درج گره New می تواند به شکل زیر برو باند:

- 1) if Avail = null then 'overflow' and 'exit'
- 2) New = Avail , Avail = Link(Avail)
- 3) data(New) = item
- 4) if (start = null) then link(New) = start , start = New
- 5) else
 - Link(New) = Link(Loc)
 - Link(Loc) = New
- 6) exit

نکته: قطعه برنامه زیر در هر وضعیتی که لیست نهی با غیر نهی باشد گره New را به ابتدای لیست پیوندی اضافه می کند.

- 1) New^.link = start
- 2) start = New



■ حذف گره از یک لیست پیوندی یک طرفه

برای حذف گره با آدرس x باید لیست را یافتن گره قبل از آن پیماش شود، سپس با یک عمل جا به جایی گره مورد نظر حذف می شود.

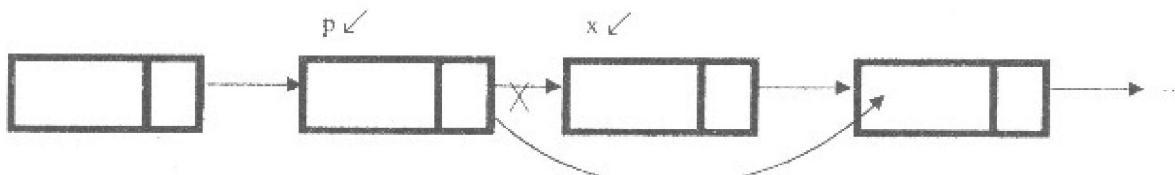
```
p = start;
while (p^.link < > x) do
  p = p^.link;
```

رسیدن به گره قبل از گره حذف شونده x

1) $p^.link = x^.link$;

یا

2) $p^.link = p^.link^.link$;



بعد از رسیدن گره p به گره قبل از x توسط یکی از ۲ عمل (1) و (2) گره مورد نظر حذف می شود.

لکته: با استفاده از دستور $(x) \text{ free}$ یا $(x) \text{ dispose}$ می توانیم بعد از حذف گره x آن را به طور کامل تخریب کنیم.

پیماش لیست های پیوندی خطی یک طرفه

پیماش گره های یک لیست پیوندی خطی یک طرفه از ابتدای لیست انجام شده و به دو صورت بازگشتی و غیر بازگشتی می تواند انجام گیرد.

الف) غیر بازگشتی:

```
if start < > nil then
begin
  p := start;
  while p < > nil do
    begin
      write (p^.Data);
      p := p^.link;
    end;
end;
```

در شرط حلقه while $p^.link < > nil$ اگر از شرط $p^.link < > nil$ استفاده کنیم پیماش لیست تا قبل از گره آخر انجام شده و با رسیدن اشاره گر p به

گره آخر از حلقه خارج می شویم.

ب) بازگشتی:

```

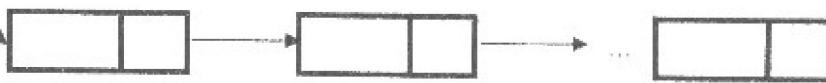
procedure show (L>List);
begin
  if (L <> nil)
  begin
    write(L^.Data);
    show(L^.link);
  end;
end;

```

در شرط دستور `if` اگر از شرط `L^.link <> nil` استفاده کنیم پیمایش لیست تا قلی از گره آخر انجام شده و با رسیدن اشاره گر `L` به گره آخر از تابع بازگشتی خارج می‌شویم.

لیست حلقوی (لیست پیوندی یک طرفه دوری)

یک لیست پیوندی خطی یک طرفه که اشاره گر `Link` گره آخر به جای آن که مقدار `null` را اختیار کند به گره نول نیست اشاره می‌کند.



نکته ۱: مزیت لیست حلقوی (دوری) بر لیست خطی یک طرفه این است که در لیست حلقوی بدون نیاز به هیچ حافظه اضافی با داشتن آدرس یک گره، دلخواه امکان دسترسی به گره قبلی یا به عبارت بهتر کلیه گره‌ها با حداقل ۱ - ۱ پیمایش در لیست n عضوی وجود دارد ولی در لیست‌های یک طرفه این امکان وجود نداشته و دسترسی به گره‌های قبل از یک گره فقط از طریق آدرس شروع لیست و پیمایش لیست می‌پرسید.

خواهد بود.

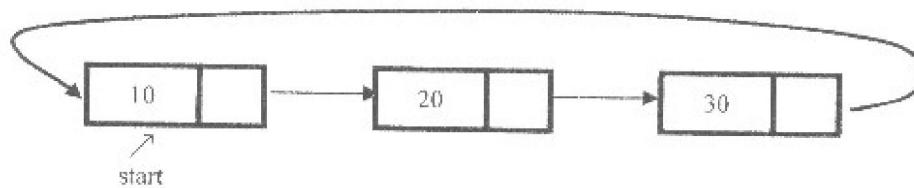
نکته ۲: پیمایش یک لیست حلقوی در صورتیکه `start` نشانه گری به ابتدای لیست حلقوی باشد به شرح زیر خواهد بود:

```

if start <> nil then
  begin
    p := start;
    repeat
      write (p^.Data);
      p := p^.link;
    until p = start;
  end;

```

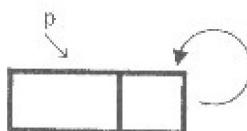
خروجی	
10	
20	
30	



دنباله می‌شود که شرط پایان حلنه آن است که اشاره گر `p` بعد از پیمایش کل لیست دوباره به ابتدای لیست اشاره کند.

نکته: در صورتیکه در شرط رو بروی until عبارت $p^.link = start$ ذکر می‌شد، آن‌گاه عمل پیمایش با رسیدن به گره آخر متوقف می‌شد و خروجی به صورت 10, 20 نمایش داده می‌شد.

نکته ۳: در صورتیکه p اشاره‌گری به یک لیست حلقوی باشد جمله $p^.link = p$ یک لیست حلقوی یک عضوی را پیاده‌سازی می‌کند.



عملیات در لیست‌های حلقوی

عملیات در لیست‌های حلقوی مانند عملیات در لیست‌های خطی غیردوری است با این تفاوت که باید شرط بایان جلفه‌ها و نحوه اصلاح اشاره گر گره آخر بازوجه به دوری بودن لیست تغییر پابند.

مثال: فرض کنید a اشاره‌گر انتهای یک لیست حلقوی و یک لیست غیرحلقوی باشد و بخواهیم گره x را بعد از آن درج کنیم.

لیست غیر دوری

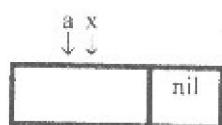
```
if (a < > nil) then
begin
    x^.link := a^.link;
    a^.link := x;
end
else
begin
    a := x;
    x^.link := nil;
end;
```

لیست دوری

```
if (a < > nil) then
begin
    x^.link := a^.link;
    a^.link := x;
end
else
begin
    a := x;
    x^.link := x;
end;
```

دیده می‌شود برای درج درحالی که لیست تهی نیست دو الگوریتم شبیه هم عمل می‌کنند اما زمانی که لیست تهی باشد دو وضعیت متفاوت به

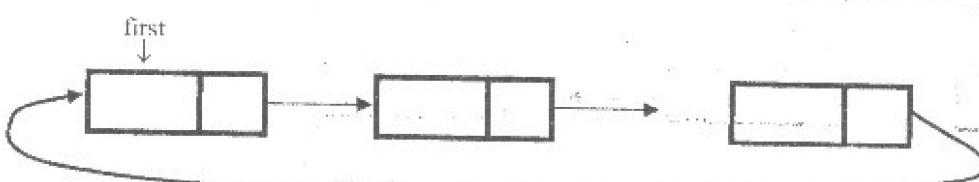
صورت زیر بعد از درج گرده به وجود می‌آید:



نکته مهم:

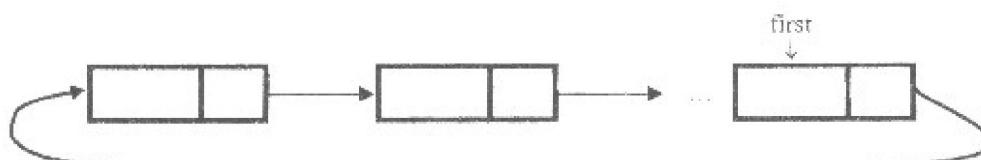
در صورتیکه اشاره‌گر $first$ در لیست‌های دوری به گره اول یا آخر اشاره کند وضعیت‌های متفاوتی از نقطه نظر زمان انجام بعضی از عملیات‌ها بوجود می‌آید که به شرح زیر آن‌ها را بررسی می‌کنیم.

الف) اشاره‌گر $first$ به ابتدای لیست اشاره کند:



در این وضعیت برای انجام عملیات: درج قبل از ابتدا - حذف از ابتدا - برای درج در انتهای نیاز به عمل پیمایش داریم تا به گره آخر برسیم و توانیم عملیات فوق را از روی موقعیت گره آخر انجام دهیم.

ب) اشاره گر first به انتهای لیست اشاره کند:



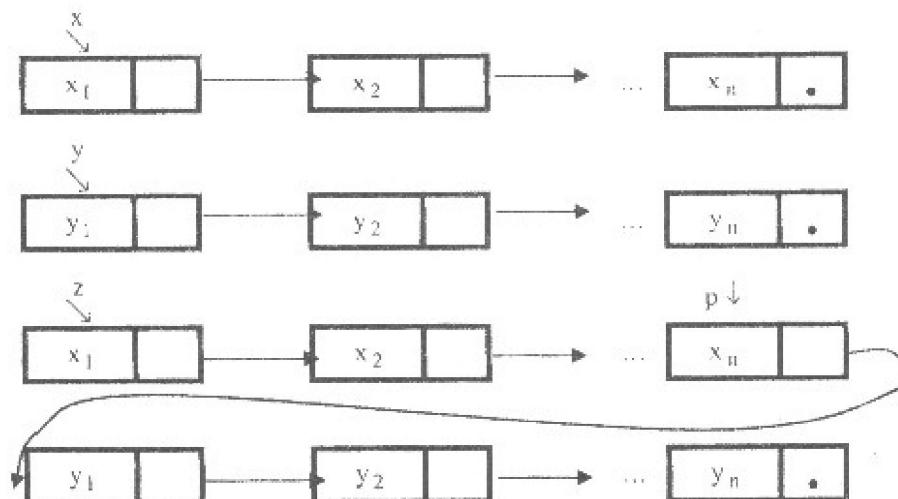
در این وضعیت برای انجام عملیات: درج قبل از ابتدا - حذف از ابتدا - برای درج در انتهای نیاز به عمل پیمایش نداریم، و به راحتی می‌توانیم عملیات فوق را از روی گره آخر که first به آن اشاره می‌کند انجام دهیم.

اتصال دو لیست پیوندی

اگر دو لیست پیوندی x و y به شرح زیر وجود داشته باشند، با استفاده از قطعه برنامه زیر می‌توان این دو لیست را به هم متصل کرده و اشاره گر z را به ابتدای لیست حاصل از اتصال دو لیست x ، y اشاره داد.

```

if x = nil then z := y
else
begin
  z := x;
  if y < > nil then
    begin
      p := x;
      while p^.link < > nil do
        p := p^.link;
      p^.link := y;
    end;
end;
  
```





۲- در شرط $x = \text{nil}$ اگر x تهی باشد z به y اشاره خواهد کرد در غیر این صورت در قسمت else : $x := z$ شده و z به ابتدای لیست x اشاره خواهد کرد.

۳- در قسمت else بعد از آن که $x := z$ انجام شد، با فرض آن که z تهی نباشد (در شرط $\text{nil} < y$)، اشاره گر p در حلقه while لیست x را پیمایش کرده تا روی گره آخر فرار گیرد، در این حالت از حلقه خارج شده و با جمله $y := p^.link$ اشاره گر (link) گره آخر x به گره اول y اشاره کرده و الگوریتم پایان می‌پذیرد.

لیست‌های پیوندی دوطرفه

در لیست‌های پیوندی دوطرفه در هر گره دو اشاره گر Left و Right وجود دارد که به ترتیب به گره قبلی و بعدی اشاره می‌کنند، با داشتن آدرس هر گره به راحتی می‌توان به گره قبلی یا بعدی دسترسی پیدا کرد.

مشخصات مهم لیست‌های پیوندی دوطرفه:

۱- امکان پیمایش لیست از Head به Tail و از Tail به Head به عنوان دوطرفه بودن وجود دارد.

۲- شرط تهی بودن لیست پیوندی دوطرفه $\text{Head} = \text{Tail} = \text{null}$ است.

`if (head = null) And (Tail = null) then write('empty')`

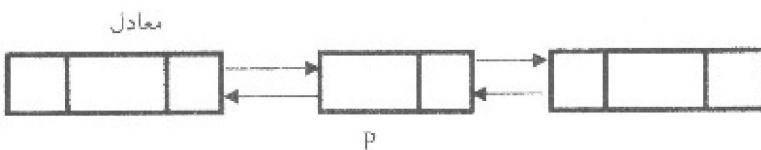
۳- از هر گره دلخواه p می‌توان به راحتی به گره بعدی $p^.Right$ یا گره قبلی $p^.Left$ دسترسی داشت.

۴- امکان درج و حذف هر گره در هر موقعیت دلخواه وجود دارد.

۵- در هر لیست پیوندی دوطرفه برای هر گره مانند p رابطه زیر برقرار است:

$$p^.Right^.left = p^.Left^.Right = p$$

معادل



۶- برای حذف یک گره دلخواه نیاز به ۲ عمل جایگزینی داریم؛ با زمان $O(1)$

۷- برای درج یک گره دلخواه نیاز به ۴ عمل جایگزینی داریم، با زمان $O(1)$

۸- برای حذف یک گره دلخواه در صورتیکه لیست خاله‌های حذف شده نیز نیاز باشد بعد از عمل حذف هر گره باید آن را به لیست مورد نظر اضافه کیم که در این صورت نیاز به ۶ عمل جایگزینی داریم.

۹- به طور متوسط برای رسیدن به یک گره دلخواه در یک لیست دوطرفه n عضوی نیاز به $\frac{n}{2}$ پیمایش است، که این عمل می‌تواند از ابتدای لیست آغاز شود.

عمل درج در لیست‌های دوپیوندی

با داشتن ۲ درس یک گره دلخواه مانند p در یک لیست دوپیوندی می‌توان گره جدید New را در سمت راست با چپ گره p درج کرد.

نکته ۱: (بهم)

هر عمل درج در لیست دوپیوندی نیاز به ۴ عمل جابجایی دارد.

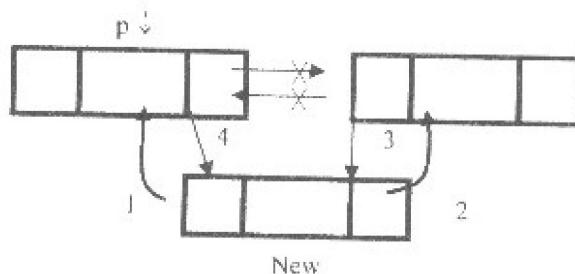
نکته ۲:

در صورتیکه عملیات لازم برای درج گره New در سمت راست با چپ گره p نوشته شود کافیست در داخل عملیات به جای Left از Right و

به جای Right از Left استفاده کنیم تا عملیات برای سمت دیگر بدست آید.

 اضافه کردن گره New در سمت راست گره p

- 1) $New^.Left := p;$
- 2) $New^.Right := p^.Right;$
- 3) $p^.Right^.Left := New;$
- 4) $p^.Right := New;$

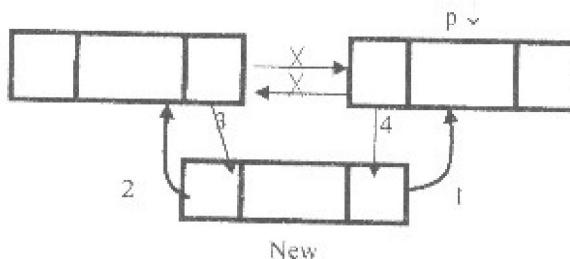


بادآوری:

در لیست‌های دوپیوندی اصطلاح‌های Next و Right و R link که اشاره‌گر به گره بعدی هستند با هم معادلند. همین طور Left و Prev و L link که اشاره‌گر به گره قبلی هستند نیز با هم معادلند.

 اضافه کردن گره New در سمت چپ گره p

- 1) $New^.Right := p;$
- 2) $New^.Left := p^.Left;$
- 3) $p^.Left^.Right := New;$
- 4) $p^.Left := New;$



نکته: همانطور که دیده می‌شود برای الگوریتم اضافه کردن گره New در سمت جب گره p فقط جای Left و Right را در الگوریتم اضافه کردن گره New در سمت راست گره p عرض کردیم.

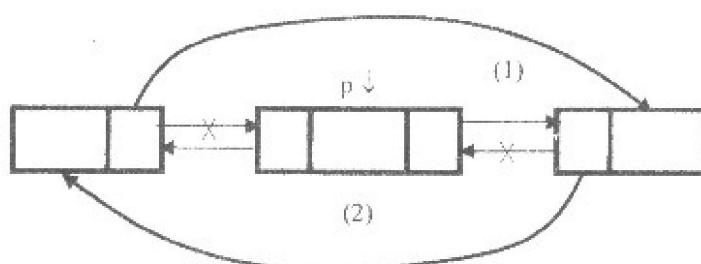
عمل حذف در لیست‌های دو پیوندی

با داشتن آدرس گره دلخواه p در یک لیست دوپیوندی می‌توان گره مورد نظر را حذف نمود.

نکته ۱: (مهمن)

هر عمل حذف در لیست دوپیوندی نیاز به ۲ عمل جایگزینی دارد.

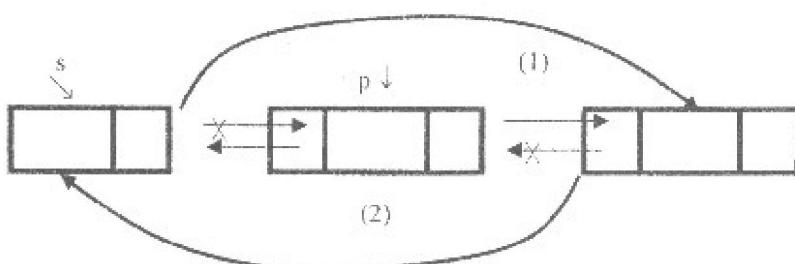
کلیه عملیات لازم برای حذف گره p در لیست دوپیوندی:



- (1) $p^.Left^.Right := p^.Right;$
- (2) $p^.Right^.Left := p^.Left;$

در عین حال عمل حذف گره می‌تواند از گره قبلی یا بعدی گره p انجام شود.

الف) در صورتیکه اشاره گر da قبل از گره حذف شونده p باشد.



- (1) $s^.Right := p^.Right;$
- (2) $p^.Right^.Left := s;$

پیاده‌سازی پیوندی ساختارهای صف و پشته

ساختارهای صف و پشته را می‌توان با استفاده از لیست‌های پیوندی پیاده‌سازی کرد.

نکته: استفاده از لیست‌های پیوندی در مقایسه با آرایه‌ها برای پیاده‌سازی صف و پشته این مزیت را دارد که با استفاده از حافظه پویا و تخصیص

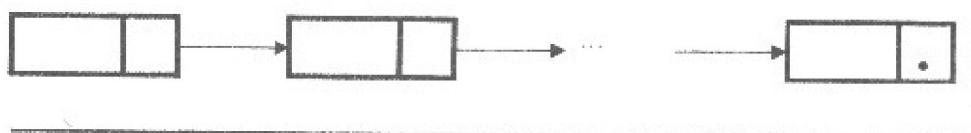
ابن حافظه به صورت متناسب با داده‌ها از فضای حافظه به طور بهینه استفاده می‌شود، این در حالی است که در آرایه‌ها فضای به صورت محدود و

ایستاده از خیلی ساختارهای صف و پشته قرار می‌گیرد.

■ پشته پیوندی:

هر لیستی که در آن عمل درج و حذف عناصر از یک طرف لیست (ابتدای لیست) انجام شود پشته پیوندی نامیده می‌شود.

`start = top \`

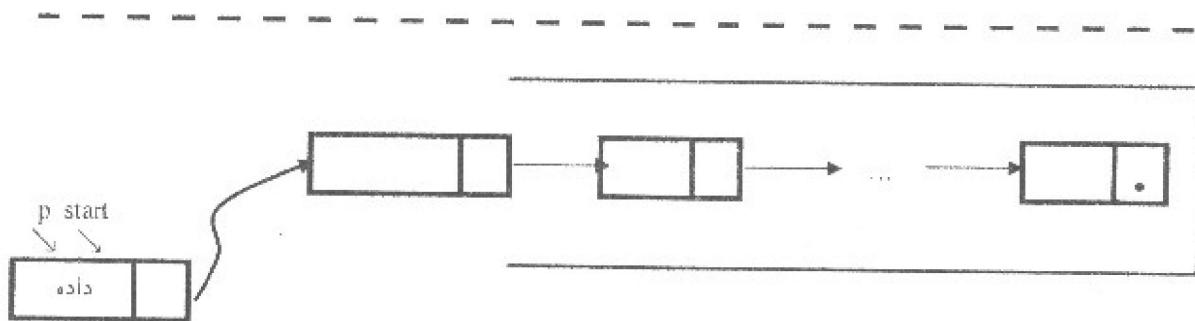
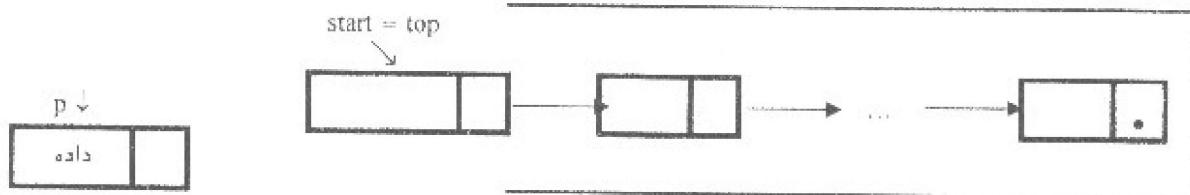


عملیات pop (حذف داده) و push (درج داده) به صورت زیر انجام می‌شود:

(الف) push (درج داده)

این عمل در ابتدای لیست انجام می‌شود و به صورت مقابل:

- 1) `New(p);`
- 2) `p^.data = داده;`
- 3) `p^.link := start;`
- 4) `start := p;`



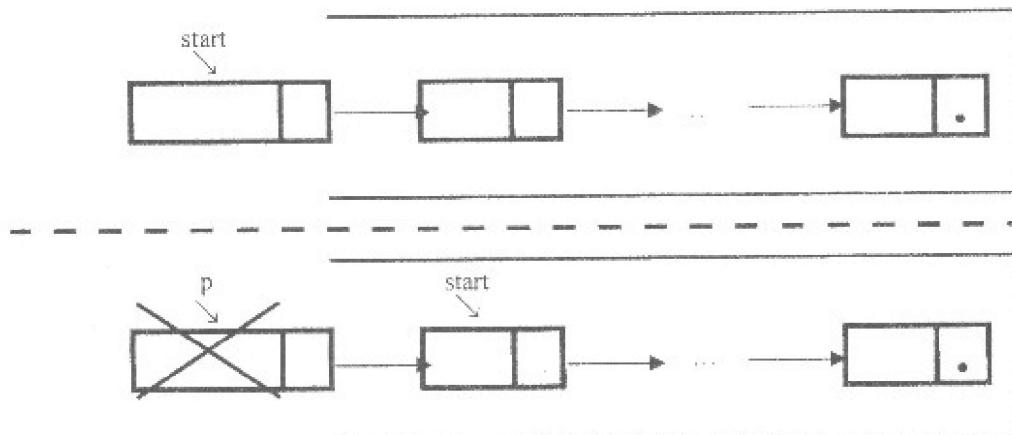
عمل درج در لیست پیوندی همان‌طور که قبل اشاره شد، 2 عمل جایگزینی نیاز دارد که در اینجا عملیات 3، 4 می‌باشد.

عملیات 1، 2 برای آماده سازی گره `p` شامل بیجاد گردن (`New(p);`) و مقداردهی آن (`(p^.data := داده)` می‌باشد.

(ب) pop (حذف داده)

این عمل مانند Push از ابتدای لیست پیوندی انجام شده و به صورت زیر در نظر گرفته می‌شود.

- 1) `p := start;`
- 2) `start := start^.link;`
- 3) `x := p^.data;`
- 4) `free(p);`



عمل حذف در لیست‌های پیوندی همان‌طور که قبل اشاره شد، فقط نیاز به یک عمل جایگزینی دارد. که در اینجا دستور دوم ($start := start^.link$) همان دستور جایگزینی می‌باشد که اشاره گر $start$ را برای حذف گره از ابتدای لیست یک گره به سمت جلو هداخت می‌کند.

دستورات ۱، ۳، ۴ به ترتیب برای اشاره گر به گره حذف شونده ($p := start$) نگهداری داده گردد حذف شونده ($x := p^.data$) و خریب کامل گردن حذف شونده ($free(p)$) در نظر گرفته شده است.

صف پیوندی

هر لیستی که عمل حذف از آن از یک طرف (ابتدای لیست) و عمل درج در آن در طرف دیگر لیست (نهایی لیست) انجام شود به عنوان صف پیوندی نامیده می‌شود.



(الف) عمل حذف از ابتدای لیست (front) به شکل زیر انجام می‌شود:

```

if (front = null) then 'queue empty'
else
begin
  1) p := front;
  2) front := front^.link;
  3) x := p^.data;
  4) if (front = null) then rear := nil;
  5) free(p);
end;
```

نوضیحات:

که در صورتیکه شرط $front = null$ برقرار باشد صف خالی بوده و امکان حذف از آن وجود ندارد.

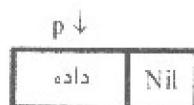
که در صورتیکه به قسمت else برویم صف خالی نبوده اما اگر بعد از عمل حذف با دستور (2) لیست خالی شود دستور شماره (4) به علت خالی شدن لیست rear بین null می‌کند.

گرّه در هر صفت پیوندی بعد از هر عمل حذف ارزش گرّه حذف شده مورد ارزیابی قرار می‌گیرد برای همین در دستورات (۱) و (۳) مقدار گرّه حذف شده ارزیابی می‌شود.

گرّه آزادسازی گرّه حذف شده توسط دستور `free(p)` انجام می‌شود.

ب) عمل درج در صفت پیوندی در انتهای (rear) به شکل زیر انجام می‌شود:

- 1) `p = getnode();`
- 2) `p^.data = ...` آماده سازی گرّه p برای درج در انتهای داده
- 3) `p^.link = nil;`
- 4) `if front = nil then front = p;`
- 5) `else rear^.link = p;`
- 6) `rear = p;`

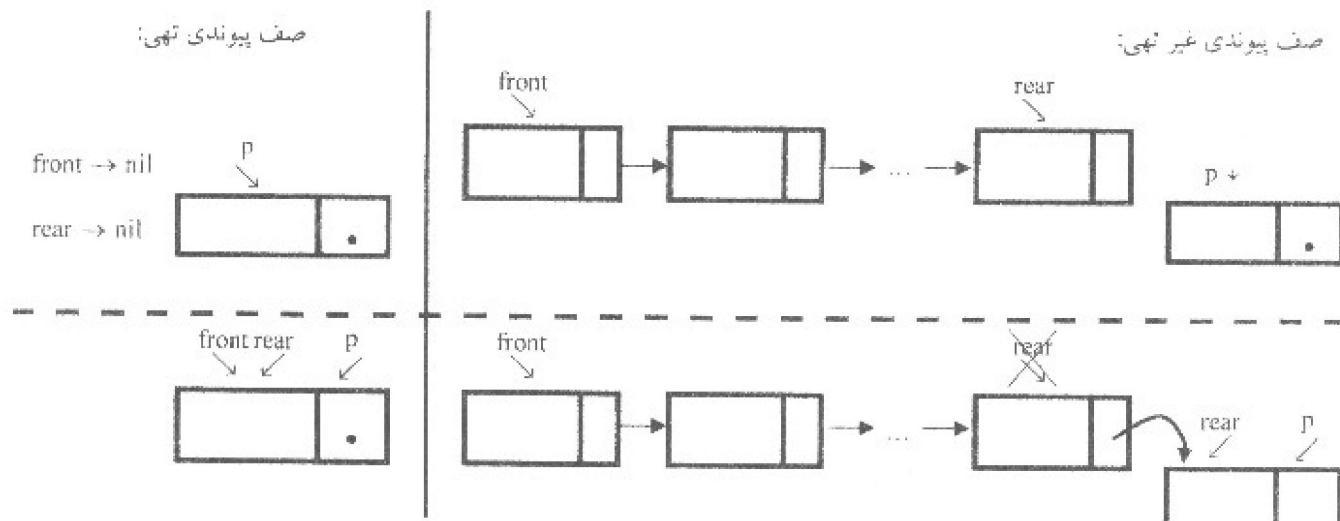


توضیحات:

(۴) در صورتیکه صفت پیوندی تهی باشد (برقراری شرط `front = nil`) در نتیجه گرّه درج شونده تنها گرّه صفت بوده و `front` هم باید به آن اشاره کند.

(۵) در صورتیکه صفت پیوندی تهی نباشد (اجرای قسمت else) گرّه p انتهای لیست درج می‌شود جمله `rear^.link = p` باعث می‌شود اشاره گر Link آخر به p اشاره کند.

(۶) در هر وضعیتی بعد از درج p در انتهای صفت پیوندی، اشاره گر rear باید به گرّه آخر که همان گرّه اضافه شده p است اشاره کند این عمل با دستور `rear = p` انجام می‌شود.





معکوس کردن یک لیست پیوندی

برای معکوس ہر لیست پیوندی با هر تعداد گره فقط با استفاده از ۳ اشاره گر می‌توان این عمل را انجام داد. در صورتیکه first اشاره گری به ابتدای یک لیست پیوندی باشد با استفاده از ۳ اشاره گر p, q, r می‌توانیم لیست مورد نظر را معکوس کنیم: در انتها بر نامه:

کلچر اشاره گر ۴ به ابتدای لیست معکوس شده اشاره می‌کند.

کلچر اشاره گر ۵ null می‌شود.

کلچر اشاره گر ۶ به یک گره بعد از ۴ اشاره می‌کند.

```
p := first;
q := nil;
while p < > nil do
begin
1) r := q;
2) q := p;
3) p := p^.link;
4) q^.link := r;
end;
first := q;
```

مثال: تابع زیر جهت عملی انجام می‌دهد.

```
fun(s:list)
begin
if (s = nil) then return (nil)
else
return(cons(fun(tail(s)), head(s)));
end;
```

cons(x, y) : لیست ۲ را به انتها لیست x متصل می‌کند.

حل: تابع بالا یک لیست پیوندی را معکوس می‌کند.

نتیجه‌گیری از لیست‌های پیوندی

هزایا:

۱- تخصیص بروبا و متناسب برای داده‌ها برخلاف آرایه‌ها که نخصیص ایستا و محدود دارند.

۲- عملیات درج و حذف بدون نیاز به شیفت با O(n) انجام می‌شود برخلاف آرایه‌ها که نیاز به شیفت برای این عمل دارند با O(n).

معایب:

۱- اثلاف حافظه سبب به آرایه‌ها به علت استفاده از فیلد اشاره گر بیشتر است.

۲- تنها روش جستجو: جستجوی خطي است و با زمان O(log₂ n) برخلاف آرایه‌ها که جستجو دودوئی را نیز دارند با زمان O(n).

۳- دسترسی به هر داده ترتیبی و از ابتدای لیست است با زمان O(n) برخلاف آرایه‌ها که امکان دسترسی تصادفی با زمان O(1) را دارند.

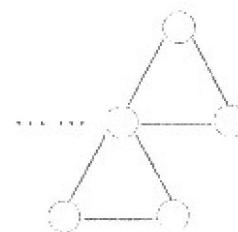
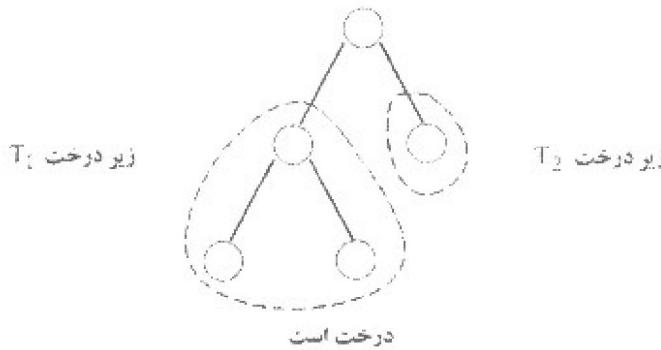
درخت (tree): ساختمان داده غیرخطی

درخت: درخت مجموعه محدودی از یک یا چند گره به صورت زیر می‌باشد:

۱) دارای گره خاصی به نام ریشه

۲) بقیه گره‌های $n > 0$ مجموعه مجزا T_1 و T_2 و ... و T_n تقسیم شده که هر یک از این مجموعه‌ها خود یک درخت هستند و T_1 و T_2 و ... و T_n زیر درختان ریشه نامیده می‌شوند.

شرط مجزا بودن زیر درختان T_1 و T_2 و ... و T_n آن است که هیچ اتصالی بین زیر درختان وجود ندارد.



(حلقه دارد) درخت نیست

☒ نکات و اصطلاح‌های مهم در مورد درختان:

- درجه یک گره: تعداد زیر درخت‌های یک گره درجه آن نام دارد.

- درجه درخت: حداکثر درجه گره‌های درخت (بزرگترین درجه درخت)

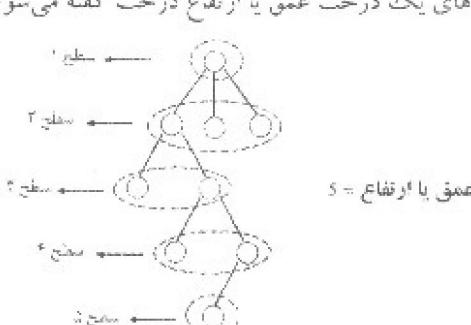
- برگ (گره‌های پایانی): گره‌ها با درجه ۰ برگ نام دارد.

- گره‌های هم‌زاد (هم‌بنا): فرزندان یک گره، گره‌های هم‌زاد نامیده می‌شوند.

- اجداد یک گره: گره‌هایی هستند که در مسیر پیش‌شده از ریشه تا گره دلخواهی وجود دارند.

- سطح یک گره: در صورتی که ریشه را در سطح ابدانیم بقیه گره‌ها به تعداد کمتری که از ریشه فاصله دارند شماره سطح می‌گیرند.

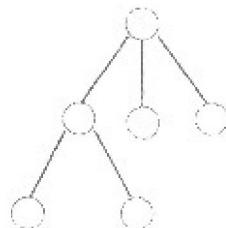
- ارتفاع یا عمق درخت: به بیشترین سطح گره‌های یک درخت عمق یا ارتفاع درخت گفته می‌شود:



در بعضی مراجع ریشه را در سطح ۰ در نظر می‌گیرند، در این حالت ارتفاع درخت رویه‌رو به طور مثال ۴ است.

- بال: هر خط یا اتصال از یک گره به گره دیگر، یا به هر انشعاب از هر گره به گره دیگر.

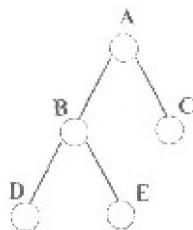
$$\text{لکته مجهه: } 1 + \text{تعداد بال‌ها} - \text{تعداد گره‌ها}$$



$$\text{تعداد بال‌ها} = 5 + 1 = 6 \quad \text{تعداد گره‌ها}$$

- مسیر: دنباله‌ای از بال‌های متوالی از یک گره به گره دیگر

- شاخه: مسیری که به یک برگ ختم شود شاخه نام دارد.

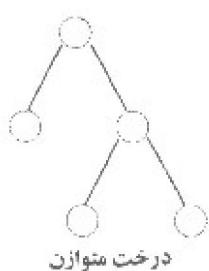


مسیر: A - B

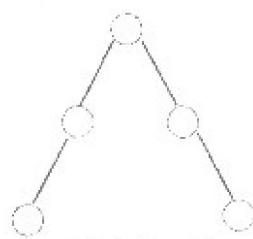
شاخه: A - B - D

درخت متوازن: درختی است که اختلاف سطح برگ‌های آن حداقل ۱ است.

درخت کاملاً متوازن: درختی است که اختلاف سطح برگ‌های آن ۰ است.



درخت متوازن



درخت کاملاً نمتوازن

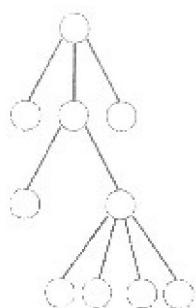
درخت متوازن

هر درخت کاملاً متوازن، حتماً متوازن است.

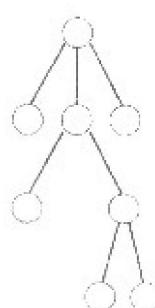
هر درخت متوازن، ممکن است کاملاً متوازن نباشد.

درخت کاملاً نمتوازن: درختی که تعداد فرزندان هر گره حداقل ۰ است.

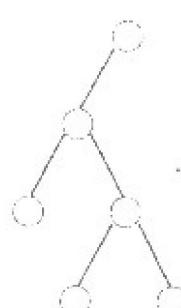
درخت ۴ تا



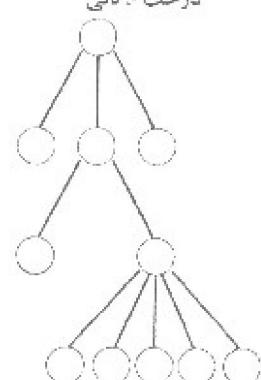
ساخت و ناشر



- 30 -



¹⁰ See also *ibid.*, pp. 11–12.



نکته مهم: در یک درخت K_n تا نی با n گره

نک تعداد کل اتصالات

نیز تعداد اتصال‌های پر

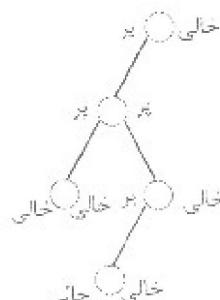
$$\text{تعداد اتصال‌های خالی} \rightarrow nk - (n - 1)$$

مثال: $K = 2$ درجه گرادیت $n = 5$:

$$nK = 2 \times 5 = 10$$

$$n - 1 = 4$$

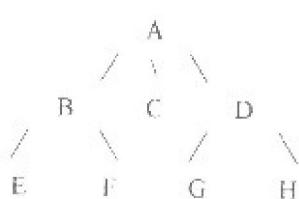
$$nK - (n - 1) = 6$$



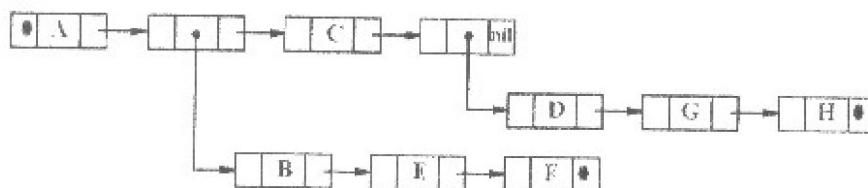
نماش درخت عمومی:

- ۱ -

(A(B(E,F),C,D(G,H))))



۴ - پیوندی:



در این تماش فرزندان هر گره سمت راست آن هستند، اگر برگ باشد سمت راست و اگر برگ باشد در یک سطح پایین تر سمت راست یکده می‌شوند.

درخت K نامی و ارتباط تعداد گره‌های یک فرزندی - دو فرزندی ... با برگ‌ها



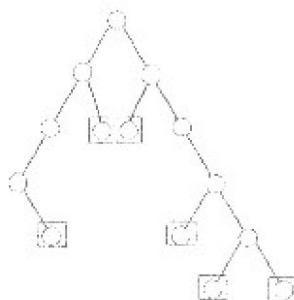
در هر درخت تعداد برگ‌ها (n) مستقل از تعداد گره‌های تک فرزندی n_1 است و رابطه زیر برای آن‌ها در یک درخت با n گره از درجه K وجود دارد.

$$n_0 = (K-1)n_K + (K-2)n_{K-1} + \dots + 2n_3 + n_2 + 1$$

$$\boxed{n_0 = n_2 + 1} \quad \text{در هر درخت با } n \text{ گره از درجه } 2 \quad \text{مثال: } n_0 = n_2 + 1$$

$$\text{مثال ۱: در یک درخت دودوئی با } n \text{ گره از درجه } 2 \text{ تعداد برگ‌ها کدام است?}$$

$$\begin{cases} n_2 = 5 \\ n_1 = 3 \\ n_0 = ? \end{cases}$$



$$n_0 = n_2 + 1 \rightarrow n_0 = 5 + 1 = \boxed{6}$$

$$\boxed{1 + \text{تعداد گره‌ها با درجه } 2 = \text{تعداد برگ‌ها}} \quad \text{در هر درخت دودوئی همیشه:}$$

مثال ۲: در یک درخت از درجه 4 با $n_1 = 5$ و $n_2 = 6$ و $n_3 = 7$ و $n_4 = 9$ مطابقت تعداد برگ‌ها!

$$n_0 = 3n_4 + 2n_3 + n_2 + 1 = 3 \times 9 + 2 \times 7 + 6 + 1 = \boxed{48}$$

■ درخت دودوئی (binary)

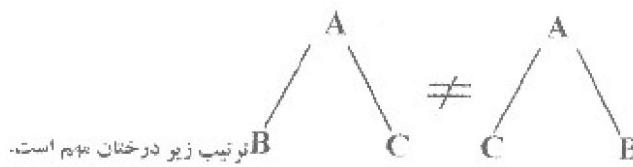
یک درخت دودوئی یا تهی است یا حاوی مجموعه‌ای محدود از گره‌ها شامل یک ریشه و هر گره در آن دارای حداقل 2 فرزند است، برای هر گره دو زیر درخت چپ و راست می‌تواند وجود داشته باشد. (تریبیشن مهم است)

در هر درخت دودوئی درجه هر گره حداقل 2 است.

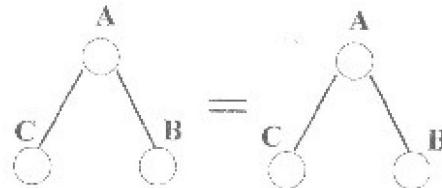
در هر درخت عادی ترتیب زیر درختان مهم نیست اما در درخت دودوئی ترتیب زیر درختان مهم است.

در هر درخت عادی گره صفر (نهی) وجود ندارد، اما در هر درخت دودوئی گره صفر (نهی)، می‌تواند وجود داشته باشد.

✓ در هر درخت دودویی:

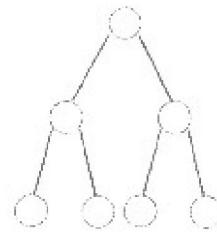


✓ در درخت عادی:

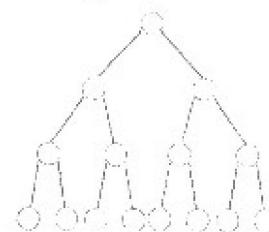


■ درخت پر؛ درخت دودویی، که همه گره‌های تمام سطوح در آن حد کثر فرزند یعنی دقیقاً ۲ فرزند دارد

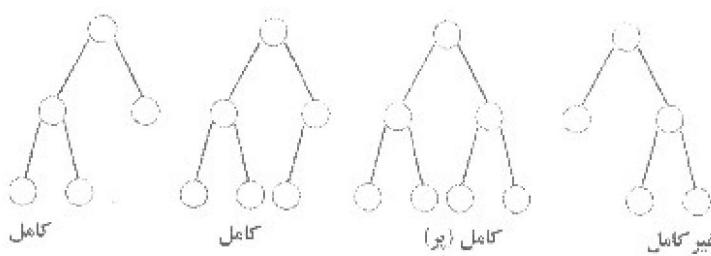
درخت دودویی پر به عمق ۳



درخت ۵ و دویی پر به عمق ۴



■ درخت کامل: درختی را کامل گویند، اگر تمام سطوح های آن به جز احتمالاً آخرین سطح حد، اکثر فرزندان را داشته باشد، و سطح آخر نیز از سمت چپ گره‌ها را بپرسی کند.



درخت کامل

درخت کاملاً متوزن

درخت پر

درخت متوزن

درخت کامل

متوزن

کاملاً متوزن

درخت پر هم کامل است و هم کاملاً متوزن است.



☒ هر درخت پر هم کامل است، هم کاملاً متوزن، هم متوزن

☒ هر درخت کامل، متوزن است. ولی همیشه پر یا کاملاً متوزن نیست.

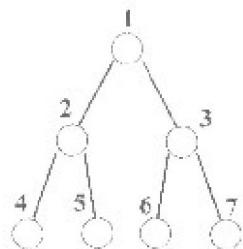
- درخت مورب (اریب) در درخت اریب به راست هر گره فرزند راست گره پدر خود است و در درخت اریب به چپ هر گره فرزند چپ گره پدر خود است.



نمایش درختان دودوئی:

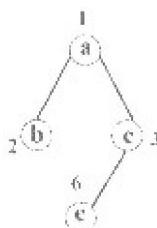
برای نمایش درختان دودوئی از دو روش ۱- آرایه ۲- لیست پیوندی استفاده می‌شود.

- آرایه: در این نمایش با فرض آن که ریشه را در سطح ۱ شماره ۱ بدهیم بقیه گره‌ها از سمت چپ به راست در سطوح بعدی می‌توانند شماره بگیرند.



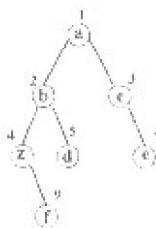
برای هر گره i
فرزند راست $\frac{i}{2i}$ فرزند چپ $\frac{i+1}{2i}$

مثال ۱:



1	2	3	4	5	6
a	b	c			e

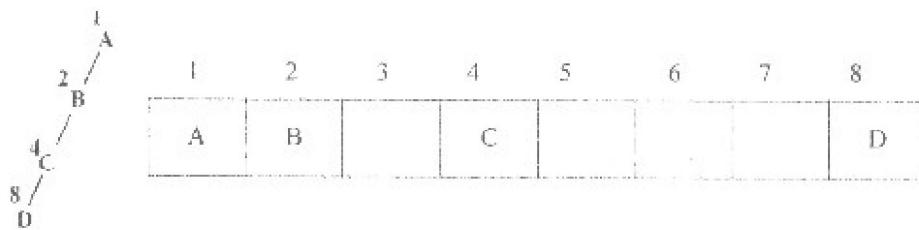
مثال ۲:



1	2	3	4	5	6	7	8	9
a	b	c		d		e		f

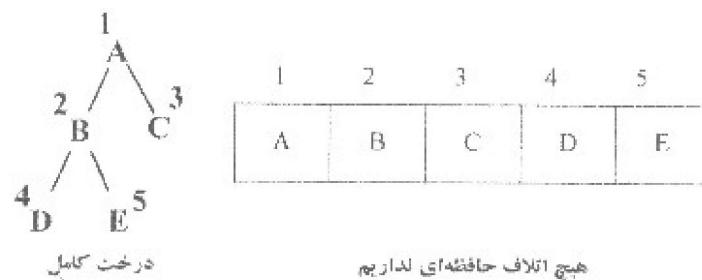
بهترین نمایش درخت با آرایه استفاده از درخت پر با کامل است چون اتفاق حافظه و حالت بلا استفاده ندارد.

بدترین نمایش درخت با آرایه استفاده از درخت اریب به چپ با راست است، که بیشترین اتفاق حافظه را در برداشته است.



از ب
جب

* اتلاف حافظه یا خانه با استفاده داریم (7, 6, 5, 3, 0)

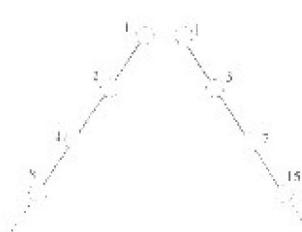


درخت کامل

بینج اتلاف حافظه‌ای داریم

گرچه در درخت اربی به جب با n گره احتیاج به $2^{n+1} - 1$ خانه داریم که فقط از n خانه آن استفاده می‌شود.

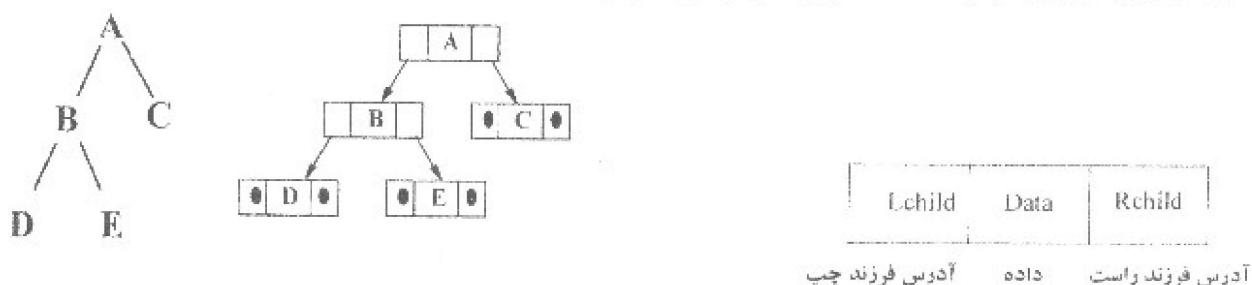
گرچه در درخت اربی به راست با n گره احتیاج به $1 - 2^n$ خانه داریم که فقط از n خانه آن استفاده می‌شود. (اتلاف حافظه در این حالت پیشتر از اربی به جب است).



۱- نمایش پیوندی:

در نمایش آرایه دسترسی به خانه‌ها به راحتی انجام می‌شود اما حذف و درج نیاز به شبکت داشته و هزینه بزرگی بالا می‌شود.

در نمایش پیوندی برای هر گره سه فسمت با فیند در نظر گرفته می‌شود. به صورت زیر:



آدرس فرزند و است داده آدرس والد

✓ مهم: در بعضی شرایط اگر نیاز داشته باشیم از هر گره آدرس والد (پدر) آن را داشته باشیم در این صورت هر چهار فیلد خواهد داشت.

Parent	Lchild	Data	Rchild
آدرس والد	آدرس فرزند	داده	آدرس والد

جب

داده

آدرس فرزند

آدرس والد

آدرس والد

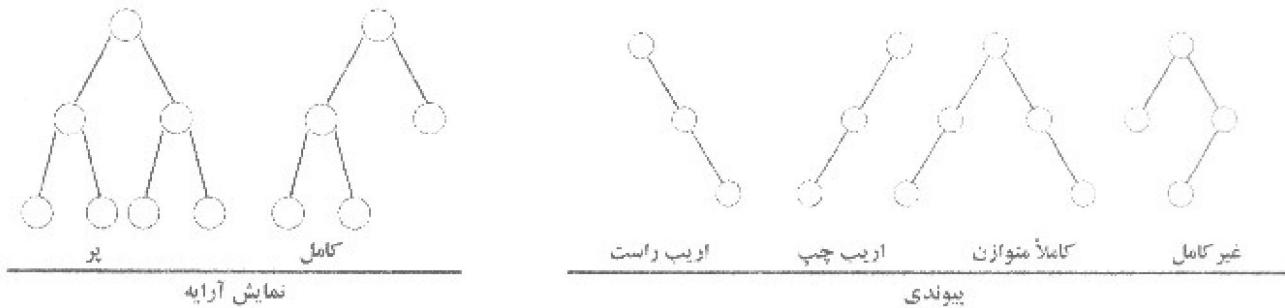
آدرس والد

لکنه تستی عهیم: در تماش درخت‌ها بهتر است از راه حل زیر حتماً استفاده کنیم.

نهاش آرایه: درخت کاملاً - درخت پی - درخت Heap (max Heap - min Heap)

نهایش پیوندی:	درخت اریب	BST (دودونی جستجوی)
متوازن - کاملاً متوازن	mintree	maxtree

١٦



نکات مهم درخت‌های دودوپی:

- حداکثر تعداد گرهها در سطح A یک درخت دودوئی $= 2^A$ (ریشه در سطح ۱)
 - حداکثر تعداد کل گرهها در یک درخت دودوئی به عمق K برابر با $= 2^K$ می‌باشد (ریشه در سطح ۱)
 - حداقل تعداد کل گرهها در یک درخت دودوئی به عمق K است.
 - حداقل تعداد کل گرهها در یک درخت دودوئی کامل به عمق K ، $= 2^{K-1}$ است.
 - در هر درخت دودوئی در صورتی که:

$$n_2 = \text{تعداد گره‌های 2 فرزندی} \quad \Rightarrow \quad n_0 = n_1 + 1$$

⁹ عمومی درخت یک داده است که هر گره را ممکن است یک گره از دو گره داشته باشد.

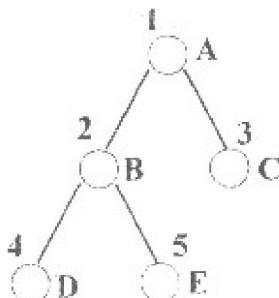
۷- عمق درخت دودوئی کمال یا پر:

۸- اگر یک درخت دودو نی کامل با n گره داشته باشیم آنگاه برای هر $1 \leq i \leq n-1$

الف) اگر $A \neq \emptyset$ باشد، پدر A در $\left\{ \begin{array}{l} \frac{1}{2} \\ \frac{1}{2} \end{array} \right\}$ است. (۱) ریشه است و پادری تدارد.

ب) اگر $n \leq 21$ باشد، آنگاه فرزند جیب از در 21 وجوددارد و گفته اگر $n > 21$ باشد، افرزند جیب ندارد.

ج) اگر $n \leq 2i+1$ آنگاه فرزند راست آن $i+1$ است و گره اگر $n > 2i+1$ باشد، افرزند رامت ندارد.



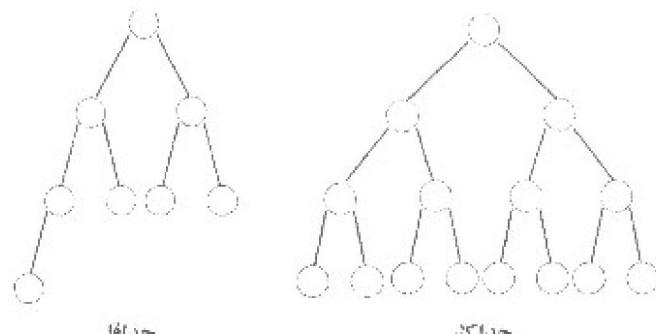
مثال ۱: $5 = 11$ گره در یک درخت کامل به صورت روبرو وجود دارد.

گره ۵ با شماره ۳ = ۱ فرزند چپ یا راست ندارد.

چون $2i + 1 = 7 > \boxed{5}$ یا $2i = 6 > \boxed{5}$ است.

گره B با شماره ۲ = ۱ فرزند چپ و راست دارد. چون $2i + 1 = 5 < = \boxed{5}$ و $2i = 4 < = \boxed{5}$ است.

مثال ۲: در یک درخت دودوئی کامل به عمق ۴ حداقل و حداکثر گره کدام است؟

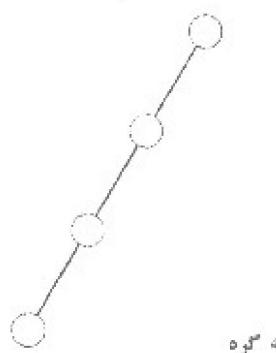


مثال ۳: در یک درخت دودوئی کامل با ۱۰۰۰ گره، عمق چقدر است؟

$$\log_2 1000 + 1 = \boxed{9.97} + 1 = 10$$

مثال ۴: در یک درخت دودوئی با ۱۰۰۰ گره، عمق چقدر است؟ نامعلوم

مثال ۵: در یک درخت دودوئی با عمق ۴ حداقل گره، چقدر است؟ (کامل نیست)



نکته مهم: در یک درخت K تائی کامل با n گره:

$$\text{اتصال پر} \quad \text{کل اتصالات} \\ = \left\lceil \frac{nK - (n-1)}{K} \right\rceil \quad = \text{تعداد برگهای}$$

در حالت خاص زمانی که درخت کامل از درجه دودوئی باشد:

$$\text{تعداد برگ‌ها} = \left\lfloor \frac{n+1}{2} \right\rfloor$$

مثال ۱: در یک درخت دودوئی کامل با ۲۰ گره تعداد برگ‌ها کدام است؟

$$\text{تعداد برگ‌ها} = \left\lfloor \frac{20+1}{2} \right\rfloor = 10$$

مثال ۲: در یک درخت کامل از درجه ۳ K-3 با ۱۰ گره تعداد برگ‌ها کدام است؟

$$\left\lfloor \frac{10 \times 3 - (10 - 1)}{3} \right\rfloor = \left\lfloor \frac{21}{3} \right\rfloor = 7$$

با آوری: تعداد برگ‌ها همیشه مستقل از گره‌های نکفروزندی است.

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

لکنه مهم: همیشه می‌توان رابطه را برای موزد زیر استفاده کرد:

تعداد حالت‌های مختلف ضرب ۳ = تعداد درخت‌های مختلف خروجی ۲ = تعداد حالت‌های مختلف خروجی ۱ = تعداد ماتریس با استفاده از کده می‌توان برای n ورودی یک می‌توان با n گره ساخت. stack در نظر گرفت. خاصیت شرکت‌پذیری.

مثال: تعداد درختان دودوئی که می‌توان با ۳ گره ساخت:

$$\frac{\binom{2n}{n}}{n+1} = \frac{\binom{6}{3}}{3+1} = \frac{6!}{3!3!} = 5$$

تعداد حالت‌های مختلف خروجی پشته با ۳ ورودی

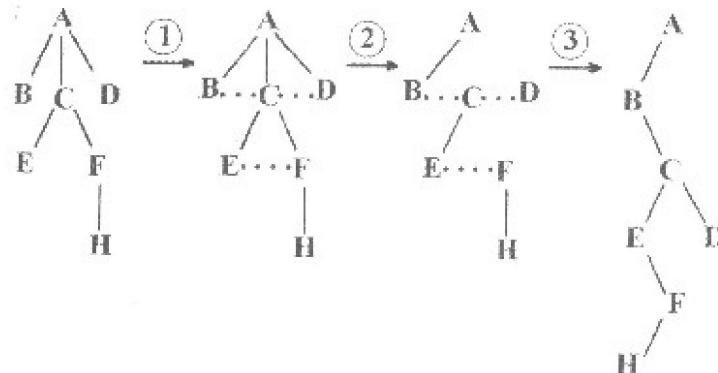
نیز می‌باشد.

عدد ۵ بدست آمده در بالا همین طور

تعداد حالت‌های مختلف ضرب 4 ماتریس

تبديل درخت عادی (عمومی) به درخت دودوئی:

- ۱- تمام فرزندان هر گره را به هم متصل می‌کنیم.
- ۲- برای هر گره اتصال کلیه فرزندان را به جز اتصال چپ حذف می‌کنیم.
- ۳- گره‌های متصل به هم در سطح افقی را ۴۵ درجه موافق عقربه‌های ساعت حرکت می‌دهیم.

ساخته‌مان داده‌ها


ظاهرآ تمام اتصال‌های افقی فرزند راست می‌شوند و اتصال‌های عمودی فرزند چپ.

نکته: پیمایش مبتنوی (LVR) inorder درخت دودوئی بدست آمده با پیمایش پسوندی LRV Postorder درخت عمومی آن متناظر است.

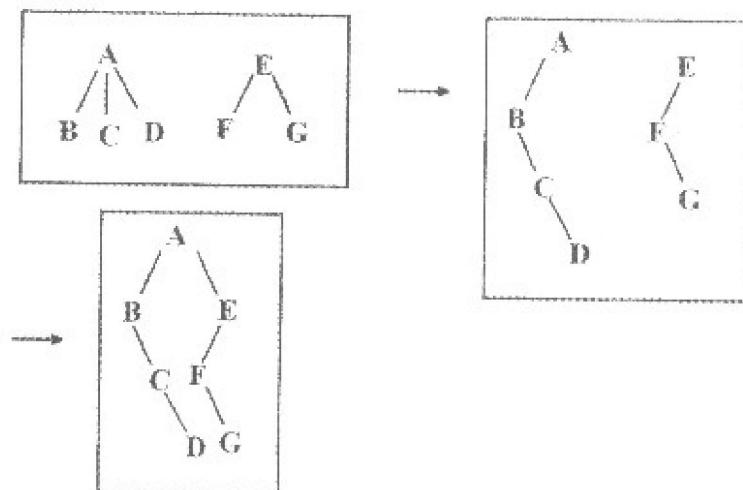
- **جنگل (forest):** جنگل بنا بر تعریف مجموعه‌ای مرتب از صفر یا چند درخت متمایز است. به عبارت دیگر جنگل مجموعه‌ای از $n \geq 0$ درخت مجرد است.

اگر ریشه یک درخت را حذف کیم آن‌گاه دارای جنگل با درخت‌هایی به تعداد فرزندان ریشه است.

▪ تبدیل جنگل به درخت دودوئی:

ابتدا نمایش دودوئی هر یک از درختان جنگل را بدست آورده سیس از جب به راست ریشه هر درخت را به عنوان فرزند راست درخت سمت

چپ در نظر می‌گیریم:

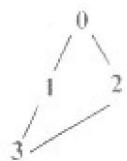


نکته مهم: یکی از کاربردهای درخت نمایش مجموعه‌ها است.

مثال مهم: کدام گزینه درخت است.

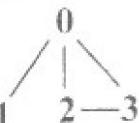
گزینه ۳ صحیح می‌باشد.

$$1) \quad \{\langle 0, 1 \rangle, \langle 0, 2 \rangle, \langle 1, 3 \rangle, \langle 3, 2 \rangle\} \rightarrow$$



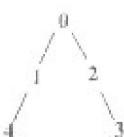
(گراف) حلقه دارد

$$2) \quad \{\langle 0, 1 \rangle, \langle 0, 2 \rangle, \langle 0, 3 \rangle, \langle 2, 3 \rangle\} \rightarrow$$



(گراف) حلقه دارد

$$3) \quad \{\langle 0, 1 \rangle, \langle 0, 2 \rangle, \langle 2, 3 \rangle, \langle 1, 4 \rangle\} \rightarrow$$



(درخت) حلقه ندارد

پیچیدگان (4)

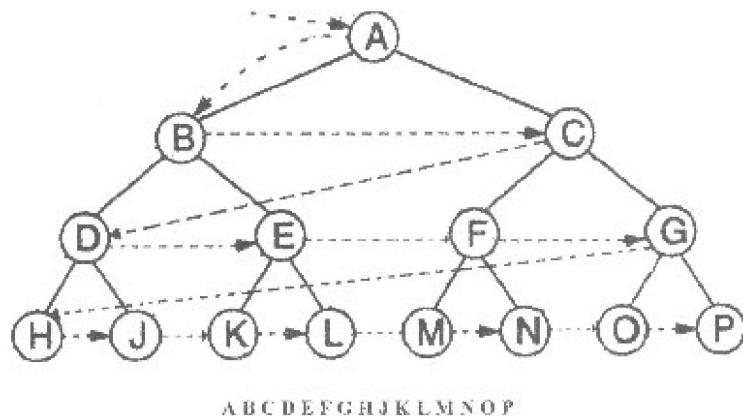
پیماش درخت‌ها دودوئی

در پیماش درخت دودوئی می‌خواهیم به هر گره فقط یکبار دستیابی داشته باشیم:

(۱) پیماش درختان در دو حالت عمقی و سطحی انجام می‌شود.

(۲) در پیماش سطحی (عرضی، ردیفی، level order) پیماش را از ریشه آغاز کرده و از سطح دروم به بعد در هر سطح، گره‌های از سمت چپ به راست روبت کرده و به سطح بعدی می‌رویم تا بین که سطح آخر نیز دیده شود.

مثال:



(۳) در پیماش عمقی با توجه به آن که پیماش از ریشه آغاز می‌گردد، برای هر گره زیر بالا به پائین سه حالت L (چپ) یا R (راست) یا V (مقدار Data) وجود دارد.

(۴) با توجه به سه حالت بیان شده، 6 حالت پیماش مختلف به وجود می‌آید.

LVR	→ inorder	بیانلندی	اول زیر درخت چپ، دوم ریشه، سوم زیر درخت راست
LRV	→ postorder	پسوندی	اول زیر درخت چپ، دوم زیر درخت راست، سوم ریشه
VLR	→ preorder	پیشوندی	اول ریشه، دوم زیر درخت چپ، سوم زیر درخت راست
	VRL		
	RVL		
	RLV		

الگوریتم پیمایش درخت:**روش پیمایش preorder:**

در این روش، درخت دودویی غیرخالی را به صورت زیر ملاقات کنید:

۱- ریشه درخت را ملاقات کنید.

۲- اگر زیر درخت چپ خالی نیست، آن را به روش preorder پیمایش کنید.

۳- اگر زیر درخت راست خالی نیست، آن را به روش preorder پیمایش کنید.

پیمایش preorder به این صورت است: از ریشه شروع کرده آن را ملاقات می کنیم، سپس به سمت چپ حرکت کرده اطلاعات این گره را می نویسیم و این روند را تا رسیدن به منتهی الیه سمت چپ ادامه می دهیم. پس از رسیدن به آخرین گره سمت چپ و ملاقات آن، به سمت راست حرکت می کنیم (چنانچه حرکت به سمت راست ممکن نباشد به گره بالاتر می رویم) و زیر درخت سمت راست آن را ملاقات می کنیم و این روند را برای تمام گره های درخت ادامه می دهیم. شکل ۱ شیوه پیمایش preorder یک درخت دودویی را نشان می دهد. تابع

() preorder که در زیر آمده است، درخت دودویی را به روش preorder (NLR , DLR , VLR) پیمایش می کند:

Procedure preorder (t:treepointer);

begin

if (t<> nil) then

begin

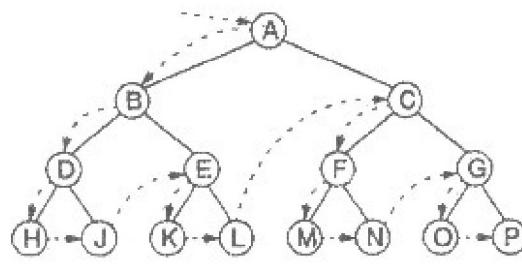
write (data (t));

preorder (Lchild(t));

preorder (Rchild (t));

end;

end;





شکل ۱: شیوه پیمایش preorder درخت دودویی

روش پیمایش postorder

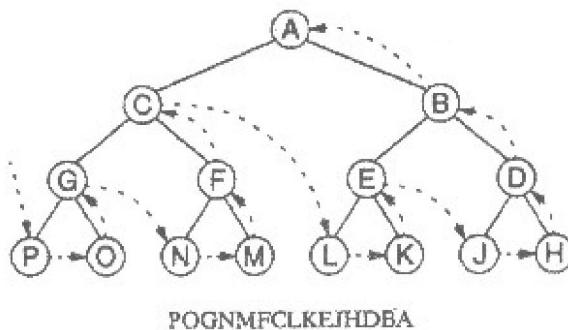
برای پیمایش postorder یک درخت دودویی غیرخالی، الگوریتم زیر را اجرا کنید:

- ۱- اگر زیر درخت چپ خالی نیست، آن را به روش postorder ملاقات کنید.
- ۲- اگر زیر درخت راست خالی نیست، آن را به روش postorder ملاقات کنید.
- ۳- ریشه درخت را ملاقات کنید.

پیمایش postorder به این صورت است: از ریشه شروع می‌کنیم و به طرف چپ حرکت می‌کنیم تا به منتهی ابه سمت چپ برسیم (یعنی به گره‌ای برسیم که فیلد Left آن نهی است). از این گره شروع کرد، تا جایی که ممکن است به سمت راست حرکت می‌کنیم. چنانچه حرکت به سمت راست ممکن نباشد، محتویات این گره را می‌نویسیم و به گره بالایی می‌رویم و با این گره مانند ریشه رفتار می‌کنیم این روند را برای ملاقات تمام گره‌های درخت آدمه می‌دهیم. شکل ۲ شیوه پیمایش postorder یک درخت دودویی را نشان می‌دهد.

تابع () postorder که در زیر آمده است، درخت را به شیوه postorder (پسوندی، LRV) پیمایش می‌کند:

```
Procedure postorder (t:treepointer):
begin
  if (t< > nil) then
    begin
      postorder (Lchild(t));
      postorder (Rchild(t));
      write (data (t));
    end;
  end;
```



شکل ۲: شیوه پیمایش postorder درخت دودویی

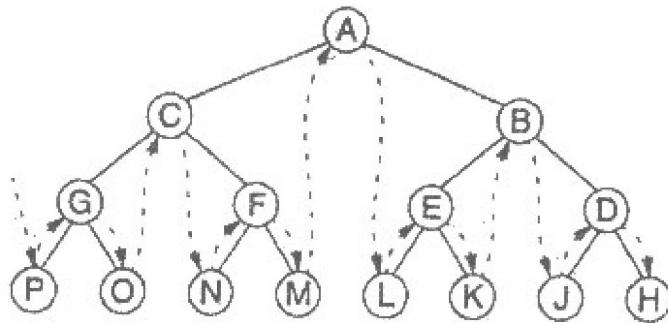
روش پیمایش inorder

برای پیمایش inorder یک درخت دودویی غیرخالی مراحل زیر را انجام دهید:

- ۱- اگر زیر درخت چپ خالی نیست، آن را به روش inorder پیمایش کنید.
- ۲- ریشه درخت را ملاقات کنید.
- ۳- اگر زیر درخت راست خالی نیست، آن را به روش inorder پیمایش کنید.

پیمایش inorder به این صورت است: از ریشه شروع کرده تا جایی که ممکن است به سمت چپ حرکت می‌کیم، با رسیدن به آخرین گره سمت چپ (گره‌ای که فلد Left آن نباید است)، محتویات آن گره را نویسیم و سپس به سمت راست حرکت می‌کنیم و با آن مثل گره ریشه برخورد می‌کنیم و به منتهی این سمت چپ می‌رویم و آن گره را ملاقات می‌کنیم. اگر در گره‌ای حرکت به سمت راست ممکن نباشد، بکه گره به سمت بالا حرکت کرده آن را ملاقات می‌کنیم. این روند را تا ملاقات کردن کلیه گره‌های درخت ادامه می‌دهیم. شکل ۳ شیوه پیمایش درخت دودویی را به روش inorder () که در زیر آمده است درخت را به روش inorder (میانوندی، LVR) پیمایش می‌کند:

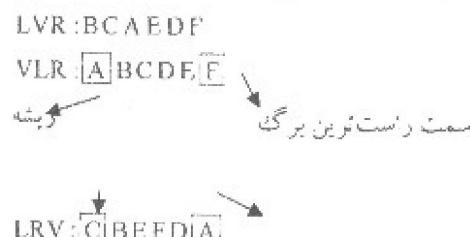
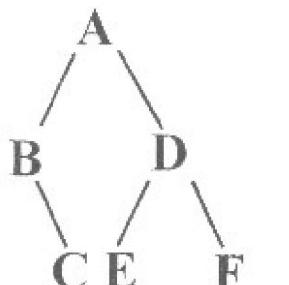
```
Procedure inorder (t : treepointer );
begin
  if (t < > nil) then
    begin
      inorder (Lchild (t));
      write (data (t));
      inorder (Rchild (t));
    end;
end;
```



PGOCNFMALEKBJDH

شکل ۳: شیوه پیمایش درخت دودویی به روش inorder

در هر سه پیمایش اول از چپ شروع می‌کنیم (یعنی جهت شروع چپ است)



ریشه سمت چپ ترین برگ

۷۴۶

لکته ۱: ترتیب دیدن برگ‌ها در هر سه پیمایش بگسان است.

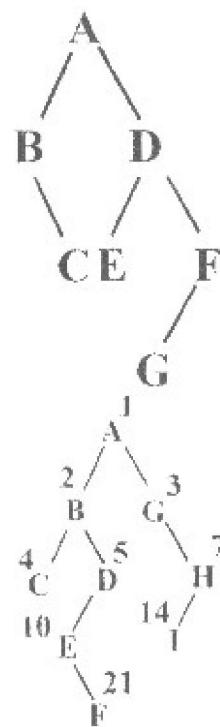
نکته ۲: با داشتن (1) و (2) می‌توان به درخت واحد رسید.

نکته ۳: با داشتن preorder و postorder ممکن است به درخت واحدی ترسیم.

مگر آن‌که درخت پر باشد و همه گره‌ها ۲ فرزندی باشد. چون وضعیت چیزی با راست بودن گره‌های تک فرزندی فقط با inorder مشخص می‌شود.

مثال: مطلوبست نمایش هر سه پیمایش درختان زیر:

Left	Right	info
1	2	A
2	0	B
3	5	D
4	0	C
5	0	E
6	7	F
7	0	G



LVR: B C A F D G E
VLR: A B C D E F G
LRV: C B E G F D A

LVR: C B E F D A G I H
VLR: A B C D L F G H I
LRV: C F L D B I H G A

1	2	3	4	5	6	7	8	9	10	14	21
A	B	G	C	D		H			E	...	I

نکته ۴: متأهلده می‌شود که در پیمایش LRV ابتدا برگ‌ها زیر درخت چپ دیده می‌شود سپس به سمت بالا و سپس برگ‌های زیر درخت راست سپس بالا و در آخر ریشه درخت دیده می‌شود.

نکته مهم:

RVL = معکوس LVR (inorder)

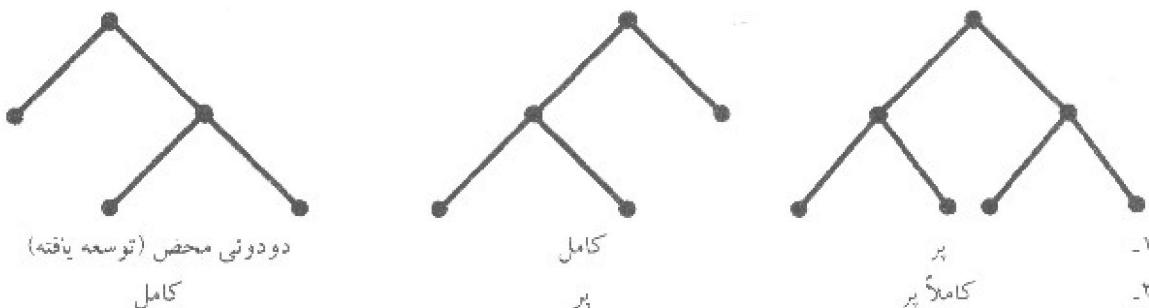
RLV = معکوس VLR (preorder)

VRL = معکوس LRV (postorder)

پس با سه پیمایش inorder و postorder و preorder می‌توانیم با معکوس کردن آن‌ها پیمایش‌های راست را بدست آوریم.

نکته بسیار مهم:

در بعضی مراجع و تست‌های کنکور تعاریف مختلفی از درخت‌های دودویی کامل پر و مخصوص وجود دارد به شرح زیر:



تمام بحث‌های این مرجع براساس مورد ۱ می‌باشد.

بررسی برابری دو درخت دودویی

اگر دو درخت دودویی ساختاری نظیر هم داشته و اطلاعات موجود در گره‌های نظریشان با هم برابر باشد، آن دو درخت برابر تلقی می‌شوند.
منظور از ساختار نظیر این است که هر انشعاب از درخت اول با انشعابی در درخت دوم مطابقت داشته باشد.
الگوریتم مطرح شده در این جا، درخت را به شیوه پیشوندی پیمایش می‌کند و توان از هر روش دیگری استفاده کرد.

```
function equal (t1, t2 : tree pointer) : boolean;
begin
  equal := FALSE;
  if (t1 = nil) and (t2 = nil) then equal := TRUE
  else
    if (t1 < > nil) and (t2 < > nil) then
      if data (t1) = data (t2) then
        if equal (Lchild (t1), Lchild (t2)) then
          equal := equal (Rchild (t1), Rchild (t2));
end;
```

درخت نخی دودویی

در یک درخت دودویی با n گره، $2n$ اشاره گر وجود دارد که از این تعداد $1 - n$ اشاره گر مورد استفاده قرار گرفته‌اند و $1 + n$ اشاره گر مقدار nil دارند. با استفاده از این اشاره گرهای بدون استفاده می‌توان به عناصر قبلی یا بعدی در یک پیمایش خاص اشاره نمود و در نتیجه در هنگام پیمایش به نحو سریع‌تری درخت را پیمایش کرد. درختی که اشاره گرهای بدون استفاده آن بین صورت مورد استفاده قرار گرفته است، درخت نخی با درخت نخ کشی شده نام دارد.

البته با وجود این مزیت، اکنون باید اشاره گرهای واقعی از اشاره گرهای نخی به لحاظ متمایز شوند و بدین منظور باید در هر گره بخشی برای مشخص کردن نوع اشاره گر موجود باشد.

معمولًا اشاره‌گر سمت چپ (Lchild) در صورتی که بلا استفاده باشد، جهت اشاره به عنصری استفاده می‌شود که در یک پیمایش مورد نظر، قبل از این عنصر قرار دارد و اشاره‌گر سمت راست (Rchild) در صورت بدون استفاده بودن برای اشاره به عنصر بعدی در پیمایش مورد نظر، مورد استفاده قرار می‌گیرد.

برای امکان تشخیص اشاره‌گرهای واقعی از اشاره‌گرهای نخی دو فیلد منطقی به هر گره درخت افزوده می‌شود که Ltag و Rtag نامیده می‌شوند. ساختار گره در چنین درختی به صورت زیر خواهد بود:

Ltag	Lchild	Data	Rchild	Rtag
------	--------	------	--------	------

ساختار گره در درخت دودوبی نخی.

اگر Ltag = ۱ باشد Lchild دارای یک اشاره‌گر نخی است و در غیر این صورت اشاره‌گر عادی به فرزند چپ است و به همین صورت اگر Rtag = ۱ باشد Rchild از نوع نخ و در غیر این صورت (Rtag = 0) اشاره‌گر عادی به فرزند راست است.

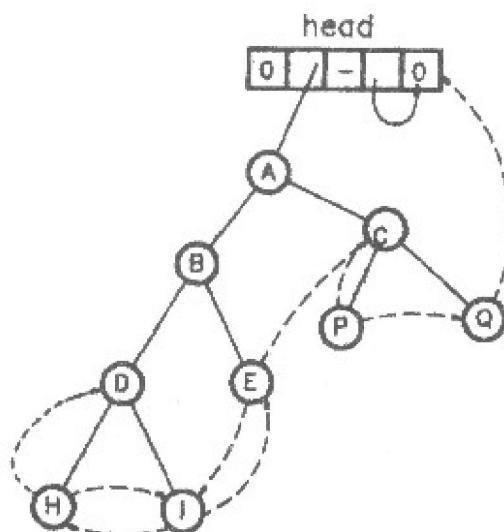
تشکیل درخت نخی براساس پیمایش پیشوندی

در این حالت برای بیجاد نخها از اتصالات نهی به صورت زیر استفاده می‌شود:

Rchild (p): به عنصری که در پیمایش پیشوندی بعد از p می‌آید، اشاره می‌کند.

Lchild (p): به عنصری که در پیمایش پیشوندی قبل از p می‌آید، اشاره می‌نماید.

preorder : A B D H I E C P Q



درخت نخی در پیمایش پیشوندی

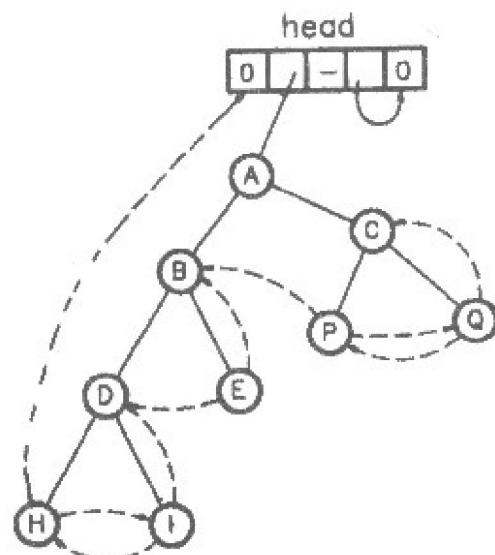
تشکیل درخت نخی براساس پیمایش پسوندی

در این حالت نیز از اتصالات نهی به صورت زیر استفاده می‌شود:

Rchild (p): به عنصری که در پیمایش پسوندی بعد از p می‌آید، اشاره می‌کند.

Lchild (p): به عنصری که در پیمایش پسوندی قبل از p می‌آید، اشاره می‌نماید.

postorder: H I D E B P Q C A



درخت نگی در پیماش پسولدی

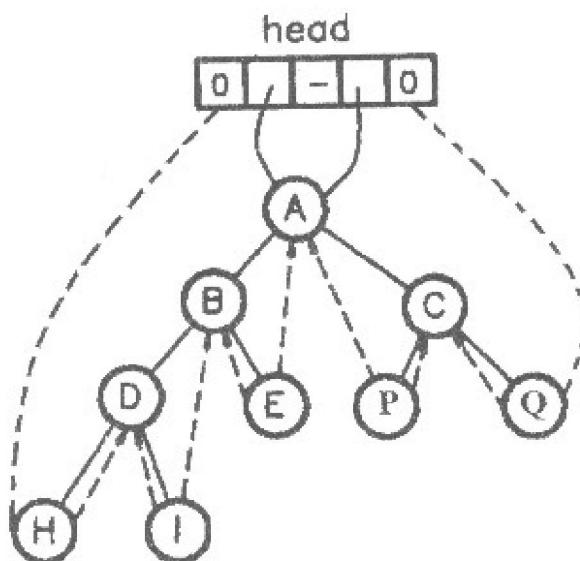
تشکیل درخت نخی براساس سماش میانوندی

در این حالت برای ایجاد نسخه‌ای از اتصالات نهی به صورت زیر استفاده می‌شود:

Rchild (p) به عکس‌بری که در پیش‌بینی‌نده بعد از ۲۰ می‌آید، اشاره می‌کند.

(p) child به عنصری که در بحث‌باز، میانه‌ندهٔ فنا از پسر آید، این وصت نباشد.

in order: H D I B E A P C Q



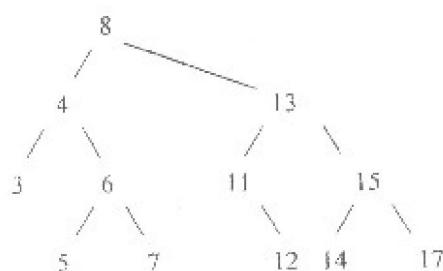
(binary search tree) \longleftrightarrow (BST)

درخت جستجوی دودوئی، درخت دودوئی است که در آن:

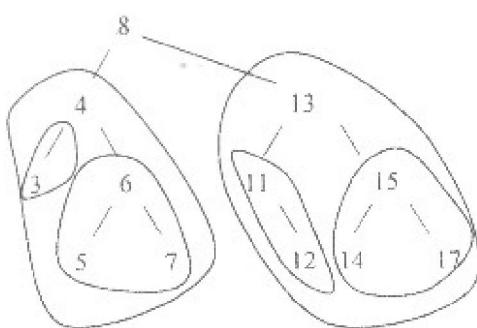
۱- مقدار هر گره از کنیه مقادیر گره‌های زیر درخت چپ بزرگتر و از کلیه مقادیر گره‌های راست کوچکتر باشد.

با

۲- کلیه مقادیر گره‌های زیر درخت چپ هر گره از کلیه مقادیر گره‌های زیر درخت راست هر گره کوچکتر باشد.



نکته: دیده می‌شود که مقادیر زیر درخت چپ گره 8 از زیر درخت راست آن کوچکتر و همین امر برای گره 13 و گره 4 و ریشه همیه زیر درختان صادق است.



مشخصات مهم در درخت BST

۱- هیچ عنصر تکراری در درخت وجود ندارد. (یکی از کاربردهای BST حذف عناصر تکراری است)

۲- پیمایش (LVR) inorder درخت BST عناصر را به صورت مرتب شده صعودی (sort) نشان می‌دهد. (یکی از کاربردهای BST مرتب سازی عناصر است)

پیمایش inorder در درخت BST شکل بالا به صورت زیر است:

3 , 4 , 5 , 6 , 7 , 8 , 11 , 12 , 13 , 14 , 15 , 17

۳- با شروع از ریشه درخت و حرکت به سمت چپ تا رسیدن به گره‌ای که سمت چپ آن گره تهی است به گردای می‌رسیم که مقدار آن در درخت می‌بیم است.

$p = \text{root}$

while ($p^.Left \neq \text{null}$)

$p = p^.Left;$

مقدار می‌بیم (در مثال بالا 3) \rightarrow

۴- با شروع از ریشه درخت و حرکت به سمت راست تا رسیدن به گره‌ای که سمت راست آن تهی (null) است به گره‌ای می‌رسیم که مقدار آن در درخت مذکوریم است.

$p = \text{root}$

while ($p^.Right \neq \text{null}$)

$p = p^.Right$

write ($p^.data$); → (مثال بالا ۱۷)

۵- همه‌ترین عمل در درخت BST جستجو می‌باشد که از ریشه آغاز شده و حداکثر تا عمق درخت ادامه می‌باید به عبارت بهتر:

الف) حداقل مقایسه ۱ مقایسه خواهد بود در صورتیکه عنصر جستجو شونده ریشه باشد. (در درخت بالا ۸)

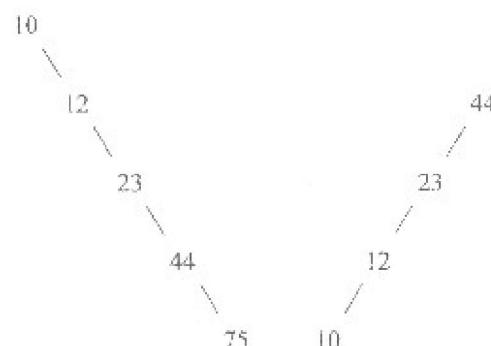
ب) حداکثر مقایسه به اندازه عمق (تعداد سطوح) درخت خواهد بود در صورتیکه عنصر جستجو شونده بکمی از عنصر گردها در پائین‌ترین سطح باشد (در درخت بالا ۱۷, ۱۴, ۷, ۵).

۶- در صورتیکه n عمق درخت باشد زمان جستجو (h) خواهد بود که در حالت متوسط $h = \log_2 n$ و زمان جستجو از مرتبه $O(\log_2 n)$ خواهد بود.

۷- در یک درخت BST با n گره با عمق k زمان جستجو (k) خواهد بود.

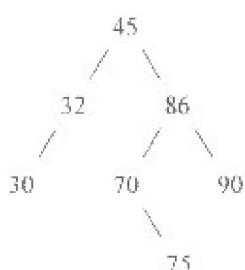
۸- در یک درخت BST با n گره حداقل مقایسه n خواهد بود زمانیکه درخت به صورت اریب به راست یا چپ باشد.

در این درخت با ۱۰ گره، گره ۱۰ در این درخت با ۵ گره، گره ۷۵ در این درخت با ۵ گره، گره ۱۰ با ۵ مقایسه بدست می‌آید.



۹- در عین حال حداقل مقایسه زمانی؛ انفاق می‌افتد که عنصر جستجو شونده ریشه باشد و با ۱ مقایسه عنصر پیدا شود.

مثال: عنصر 75 در درخت BST زیر با چند مقایسه پیدا می‌شود.



جستجو از ریشه آغاز شده و به ترتیب با 4 مقایسه گره 75 پافت می‌شود: 45 راست → 86 چپ → 70 راست → 75.

نتیجه‌گیری:

- در یک درخت BST به n گره با عمق h زمان جستجو ($O(h)$) که متوسط این زمان وضعیت است که $h = \log_2 n$ باشد و زمان $(\log n)$ خواهد بود.
 - در یک درخت BST به صورت اریب به n گره و عمق h حداقل مقابله $\lceil h \rceil$ خواهد بود؛ که بدترین وضعیت است
- ۹- تعداد مقایسه برای رسیدن به این نتیجه که عنصری در درخت BST وجود ندارد حداقل به عمق درخت خواهد بود و زمان آن $O(h)$ (عمق درخت) خواهد بود. و بدترین وضعیت زمانی است که درخت اریب باشد در این حالت حداقل مقابله به تعداد گرهای درخت (n) خواهد بود.

مثال: با چند مقایسه مشخص می‌شود عنصر 35 در درخت BST مثال بالا وجود ندارد. جستجو از ریشه آغاز شده و با 2 مقایسه زیر مشخص می‌شود گره 35 وجود ندارد: 45 \leftarrow 32 \leftarrow راست \leftarrow تهی

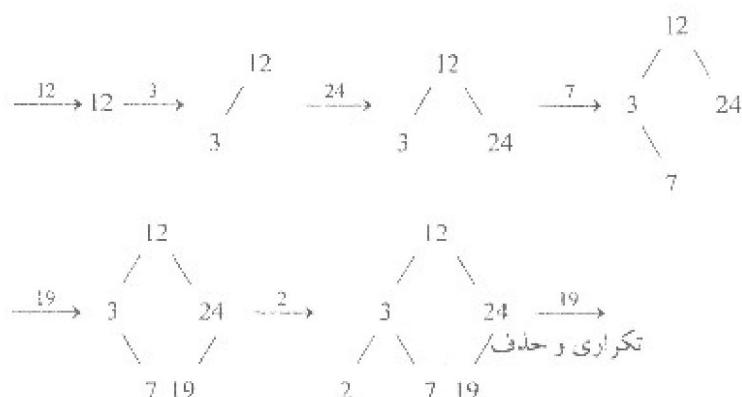
درج در گره BST

برای عمل درج ابتدا عمل جستجو برای گره درج شونده باید انجام شود تا مطمئن شویم گره مورد نظر در درخت BST وجود نداشته باشد. (در BST عنصر تکراری وجود ندارد) در این وضعیت 2 حالت پیش می‌آید.

- ۱- عنصر درج شونده در درخت پیدا شود در این صورت از لیست ورودی حذف شده و دیگر وارد درخت نمی‌شود.
- ۲- عنصر درج شونده در درخت وجود ندارد. در این صورت بعد از جستجو به محض رسیدن به بن بست (یعنی محلی که دیگر نمی‌توانیم به سمت چپ یا راست برویم) عنصر درج شونده را در همان محل درج می‌کیم.

۳- ممکن است درج هر گره جستجو مقابله از ریشه آغاز شده و سعی داریم درخت، حالت BST بودن خود را حفظ کند.

مثال: مطلوب است درج آرایه زیر در یک درخت BST تهی: 12, 3, 24, 7, 19, 2, 19



نحوه عملیات:

۱: بدون مقایسه در ریشه درخت

۲: مقایسه با 12 و سمت چپ 12

۳: مقایسه با 12 و سمت راست 12

۴: مقایسه به ترتیب با 12, 3 و سمت راست 3

۲۴ مقایسه به ترتیب با ۱۲، ۲۴ و قرقر گرفتن در سمت چپ ۱۹

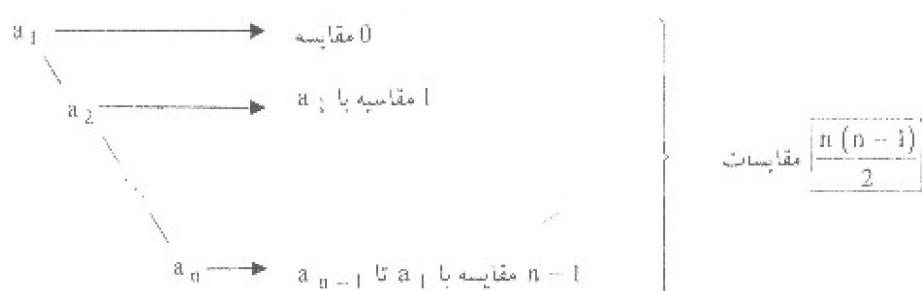
۲۲ مقایسه به ترتیب با ۱۲، ۳ و قرقر گرفتن در سمت چپ ۳

۱۹: ۳ مقایسه به ترتیب ۱۲، ۲۴، ۱۹ و به علت تکراری بودن از لیست ورودی حذف می‌شود.

بدترین وضعیت:

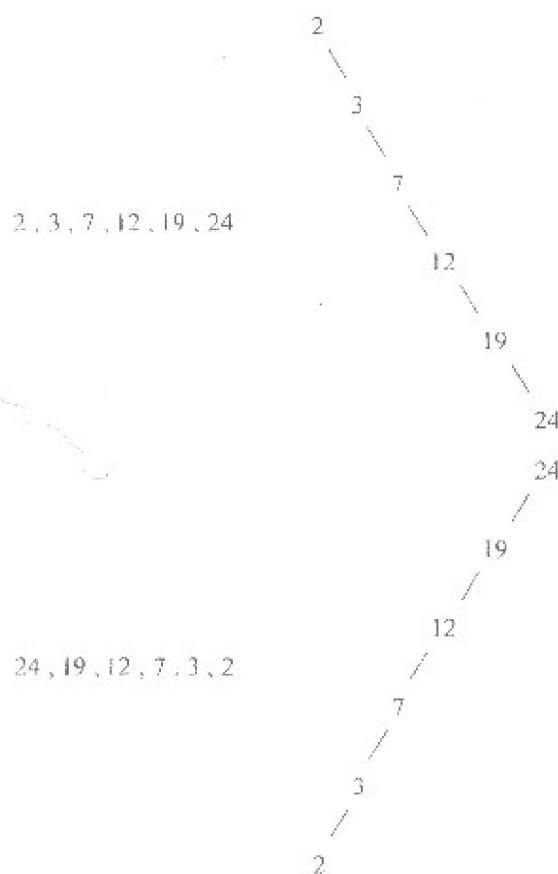
بدترین حالت در درخت BST ورود عناصر به صورت صعودی یا نزولی بوده که باعث می‌شود درخت به صورت اربیبه راست یا چپ ایجاد شود.

فرض کنید $a_1 < a_2 < \dots < a_n$ در نتیجه اگر داده‌ها به صورت a_1, a_2, \dots, a_n وارد شوند.



در این حالت تعداد مقایسات برای درج n داده می‌شود.

مثال:



لکته مهم: عمق درخت BST همیشه بستگی به نحوه ورود داده‌ها دارد.

بهترین وضعیت:

زمانی که داده‌ها به صورت نامرتب وارد شوند، در این صورت درخت دارای ارتفاع متوازن $O(\log n)$ خواهد بود.

زمان درج در BST:

- ۱- جستجو برای عنصر درج شونده تا نکراری نباشد با زمان $O(h)$
- ۲- درج گره مورد نظر در صورت نکراری نبودن در محل بن بست با زمان $O(1)$

برای درج هر عنصر در BST دو عمل انجام می‌شود:

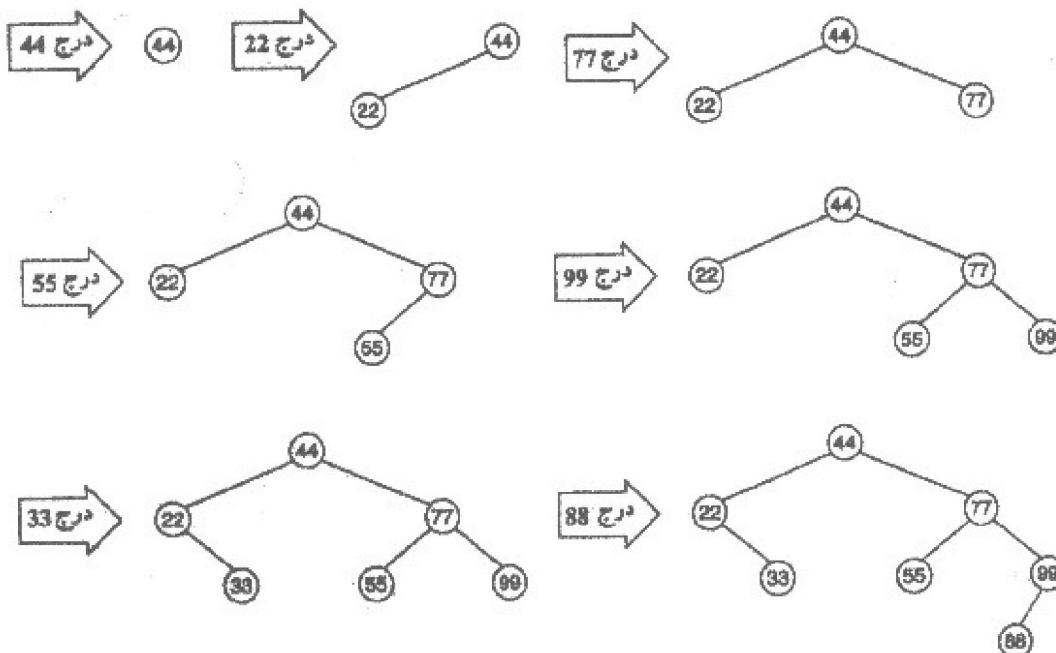
در مجموع زمان درج هر گره $O(h)$ خواهد بود.

- ۱- در حالت متوسط همچنین در بهترین حالت یعنی زمانیکه داده‌ها نامرتب وارد شود ارتفاع $O(\log n)$ و درنتیجه زمان درج هر داده $O(\log n)$ که دو حالت وجود دارد
- ۲- در بدترین وضعیت زمانیکه داده‌ها مرتب وارد شود درخت از بپ به راست با چپ شده و زمان درج هر داده $O(n)$ خواهد بود.

در حالت متوسط و (بهترین حالت) $O(n \log n)$

در بدترین حالت $O(n^2)$

درنتیجه زمان برای درج n گره



حذف گرهای از BST

برای حذف یک گره داخله از درخت BST ابتدا باید عمل جستجو ت Jamie گیرد. که نیاز به زمان $O(h)$ (h عمق درخت) خواهد داشت در این

وضعیت ۲ حالت پیش می‌آید:

(الف) گره مورد نظر برای حذف از درخت BST وجود ندارد در این حالت هیچ عملی انجام نمی‌شود.

(ب) گره مورد نظر برای حذف از درخت BST وجود داشته که در این حالت ۳ وضعیت وجود خواهد داشت که به شرح زیر آن را بررسی می‌کنیم.

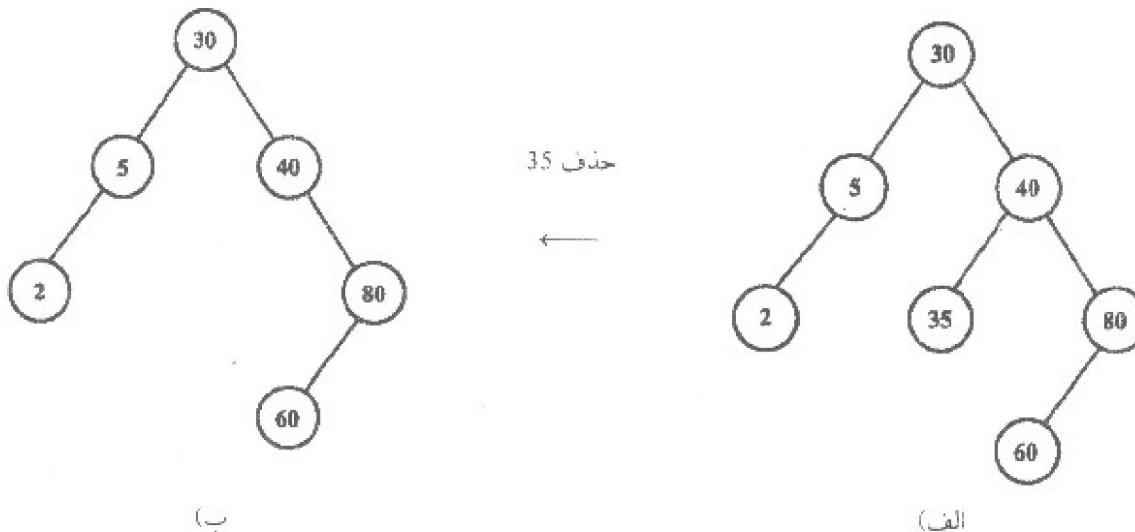
برای حذف گردهای مثل x از درخت BST بعد از آن که توسط روش جستجو موقعیت آن بدست آمد. سه حالت را در نظر می‌گیریم:

۱- x برگ است.

۲- x بک فرزند دارد.

۳- x دو فرزند دارد.

حالات اول بسیار ساده است. اشاره گر مناسبی از گره والد x را نهی می‌کنیم. اگر x فرزند چپ باشد، اشاره گر چپ والد و گرنه اشاره گر راست والد را نهی می‌کنیم. به عنوان مثال، در شکل ۱-الف برای حذف گره برگ حاوی 35، باید فیلد فرزند چپ والد این گره 40 را برابر با تهی فرار دهیم تا شکل ۱-ب به دست آید.



شکل ۱: حذف گره برگ از BST

حذف گره در حالت دوم که x فقط یک فرزند دارد نیز ساده است. در این حالت باید کاری کنیم که اشاره گر مناسبی از گره والد x به فرزند x اشاره نماید. به عنوان مثال، برای حذف گره حاوی 40 از شکل ۲-الف، باید کاری کنیم که اشاره گر راست گره والد (30) به فرزند گره 40 (یعنی گره 80) اشاره نماید (شکل ۲-ب). سهی گره 40 را به مخزن حافظه برمی‌گردانیم.



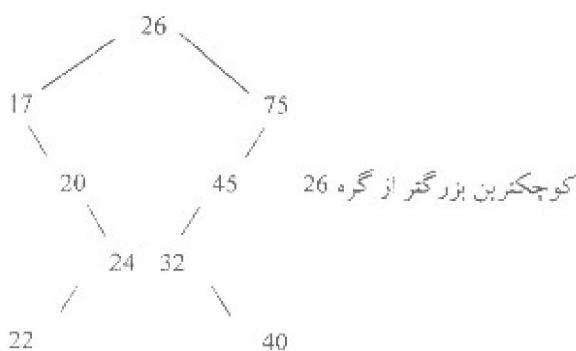
شکل ۲: حذف گره بک فورندی از BST

در حالت سوم، که در آن x دو فرزند دارد، گره‌ای را پیدا می‌کنیم که در پیمایش inorder بعد یا قبل از x قرار می‌گیرد. این گره را y نام‌گذاری می‌کنیم. وقت داشته باشید که گره y با این ویژگی، عنصر بعد از x فاقد فرزند چپ و عنصر قبل از x فاقد فرزند راست است. ابتدا گره y را با استفاده از حالت‌های ۱ یا ۲ از درخت حذف می‌کنیم و سپس گره y را به جای گره x در درخت قرار می‌دهیم. پیدا کردن عنصر بعدی با قبلی برای هر گره در پیمایش inorder به شرح زیر خواهد بود.

الف) پیدا کردن موقعیت گره بعد از گره p در پیمایش inorder (کوچکترین بزرگتر بعد از p)

$T = p^.Right;$
 $\text{while } (T^.Left \neq \text{null}) \quad \left. \begin{array}{l} \text{ابنای سمت راست } p \text{ می‌رویم (بزرگتر از } p \text{) سپس برای بدست آوردن کوچکترین بزرگتر از } p \text{ نا انتهای} \\ \text{چپ می‌رویم.} \end{array} \right\}$

۳) گره بعد از p (کوچکترین بزرگتر از p) در پیمایش inorder مطمئناً فرزند چپ ندارد چون اگر داشت کوچکتر از این گره بود.

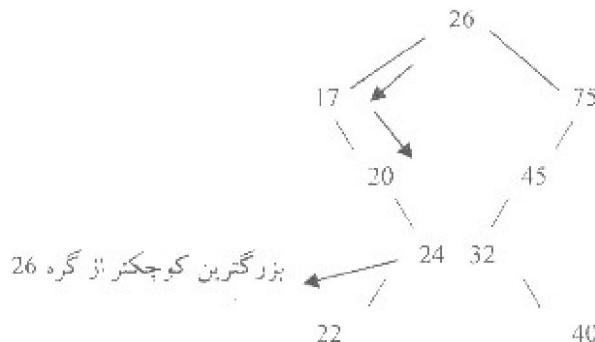


ب) پیدا کردن موقعیت گره قبل از گره p در پیمایش inorder (بزرگترین کوچکتر قبل از p)

$T = p^.Left;$
 $\text{while } (T^.Right \neq \text{null}) \quad \left. \begin{array}{l} \text{ابنای سمت چپ } p \text{ می‌رویم (کوچکتر از } p \text{) سپس برای بدست آوردن بزرگترین کوچکتر از } p \text{ نا انتهای} \\ \text{راست می‌رویم.} \end{array} \right\}$

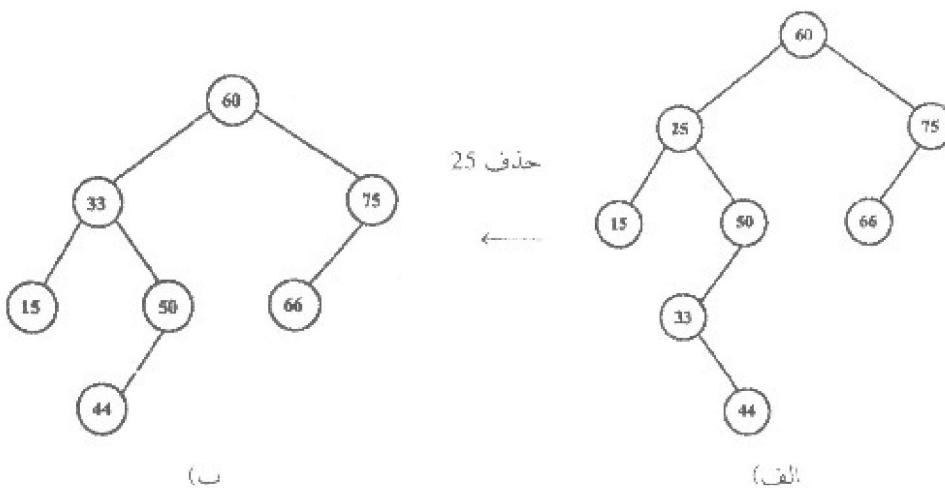
۴) گره قبل از p (بزرگترین کوچکتر از p) در پیمایش inorder مطمئناً فرزند راست ندارد چون اگر داشت بزرگتر از این گره بود.

ساختمان داده‌ها



پیمایش inorder درخت 75، 20، 22، 24، 26، 32، 40، 45 در نظر گرفته می‌شود که با توجه به حالت‌های (الف و ب) گره بعد و قبل از 26 بدست می‌آید.

به عنوان مثال، درخت شکل ۳ را در نظر بگیرید. می‌خواهیم گره حاوی 25 را از درخت شکل ۳ - الف حذف کیم تا درخت شکل ۳ - ب به دست آید. توجه دارید که اگر این درخت را به روش inorder پیمایش کنید، 33 بعد از 25 قرار می‌گیرد. بنابراین، 33 گره بعدی 25 در پیمایش inorder است. در نتیجه، باید گره 33 را حذف کرده گره 44 را به عنوان فرزند چپ گره 50 که والد گره 33 است وصل کنیم (حالت ۱.2 سهیم گره 33 را به حای گره 25 قرار دهیم تا شکل ۳ - ب به دست آید).



شکل ۳: حذف گره دو فرزندی از BST

زمان حذف هر گره از BST

برای حذف هر گره در BST دو عمل به صورت:

۱- جستجو به دنبال عنصر حذف شونده با زمان $O(h)$

۲- حذف گره مورد نظر $O(1)$

بنابراین زمان حذف گره $O(h)$ خواهد بود.

در حالت متوسط و (بهترین حالت) $O(n \log n)$

در بدترین حالت $O(n^2)$

در نتیجه زمان برای حذف n گره

کاربردهای درخت جستجوی دودوئی (BST) (مهنم)

۱- جستجو به دنبال داده‌ها

۲- حذف داده‌های تکراری از لیست‌ها

۳- مرتب‌سازی (sort) لیست‌ها با پیمانش (LVR) inorder

۱- جستجو به دنبال داده‌ها

این عمل از ریشه آغاز شده و با مقایسه با هر گره در صورتیکه گره جستجو شونده بزرگتر باشد به سمت راست و در صورتیکه کوچکتر باشد به سمت چپ می‌رویم در نهایت با گره مورد نظر پیدا می‌شود یا به null می‌رسیم (گره وجود ندارد). زمان جستجو نیز بستگی به ارتفاع درخت دارد که $O(h)$ خواهد بود و در حالت متوسط $O(\log n)$ می‌باشد.

دو الگوریتم زیر به صورت بازگشتی و غیربازگشتی عمل جستجو را انجام می‌دهند.

غیربازگشتی

بازگشتی

```

function find (t : tree pointer ; x : integer ) : boolean;
var
p , q : tree pointer ;
flag : boolean;
begin
  p := t; flag := False;
while (p <> nil) and (not flag) do
  begin
    if (p^.data = x) then flag := True
    else if (x < p^.data) then
      begin
        q := p;
        p := p^.Lchild;
      end;
    else if (x > p^.data) then
      begin
        q := p;
        p := p^.Rchild;
      end;
    end;
  find := flag;
end;
```

```

function find (t : tree pointer ; x : integer ) : boolean;
var
flag : boolean;
begin
  flag := False;
  if (t <> nil) then begin
    if t^.data = x then flag := True
    else if x < t^.data then
      flag := find (t^.Lchild , x)
    else if x > t^.data then
      flag := find (t^.Rchild , x);
  end;
  find := flag;
end;
```

۲- حذف داده‌های تکراری لیست‌ها

برای آن که بخواهیم در لیست داده‌های a_1, a_2, \dots, a_n عناصر تکراری را هنگام ورود داده‌ها حذف کنیم روش‌های مختلفی وجود دارد. الگوریتم عمومی: در یک الگوریتم ساده هنگام ورود داده a_k با $(k-1)$ مقایسه با داده‌های a_1, a_2, \dots, a_{k-1} تکراری بردن داده a_k مشخص می‌شود. در نتیجه تعداد مقایسات برای a_1, a_2, \dots, a_n از رابطه زیر بدست می‌آید.

$$a_1, a_2, \dots, a_n$$

تعداد مقایسات

$$0 + 1 + \dots + (n-1) = \frac{n(n-1)}{2} = O(n^2)$$

الگوریتم درخت BST

با استفاده از داده‌های a_1, a_2, \dots, a_n یک درخت BST ایجاد می‌کنیم در این حالت هنگام درج a_k اگر در درخت تکراری باشد از نیست حذف می‌شود. در این وضعیت به طور متوسط زمان درج هر داده $O(\log n)$ و برای درج n داده $O(n \log n)$ نیاز است که زمان آن از الگوریتم عمومی کمتر خواهد بود.

بهینه بندی: وضعیت زمانی خواهد بود که داده‌ها به صورت مرتب وارد شود در این حالت مرتبه زمانی $O(n^2)$ خواهد بود و تعداد مقایسات

$$\text{در صورتیکه هیچ داده تکراری نداشته باشیم } \frac{n(n-1)}{2} \text{ خواهد بود.}$$

مثال: مطلوبست تعداد مقایسات برای حذف داده‌های تکراری آرایه زیر به دو روش الگوریتم عمومی و درخت BST

$10, 20, 40, 20, 50, 40$

الگوریتم عمومی

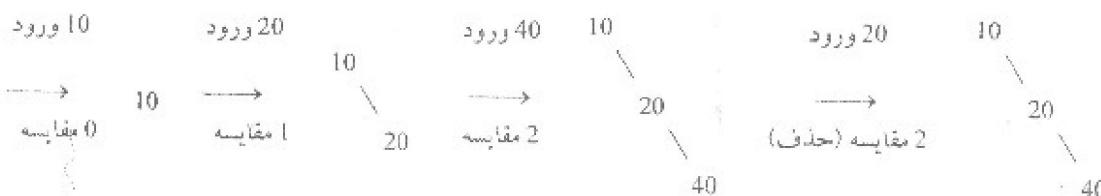
مقایسات	10	20	40	20	50	40
	0	1	2	3	4	5

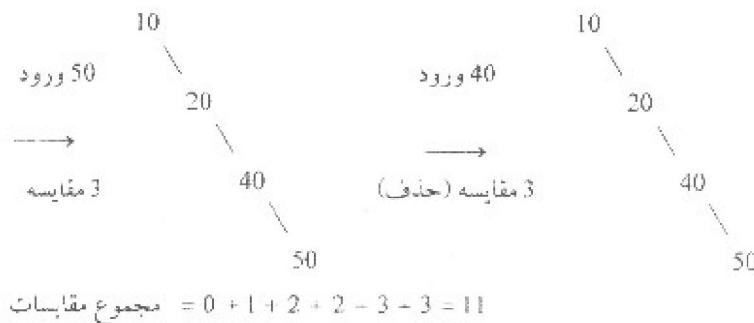
لذا نتیجه با توجه به آن که $n = 6$ است در نتیجه از رابطه $\frac{n(n-1)}{2}$ نیز تعداد مقایسات در این روش بدست می‌آید.

$$\frac{6(6-1)}{2} = 15$$

الگوریتم درخت BST

درخت BST را از داده‌های فوق تشکیل می‌دهیم.





که تعداد مقایسه‌ات کمتر از الگوریتم عمومی است.

۳- مرتب سازی داده‌ها

- الف) درج آرایه در درخت BST تهی
 ب) پیماش inorder (LVR) درخت BST حاصل از مرحله (الف)

} در صورتیکه آرایه n ثانی را داشته باشیم می‌توانیم با آرایه را sort کنیم.

کمترین زمان در حالت متوسط و بهترین وضعیت $O(n \log n)$

کمترین زمان در بدترین وضعیت (درخت اریب) $O(n^2)$

پیاده سازی BST

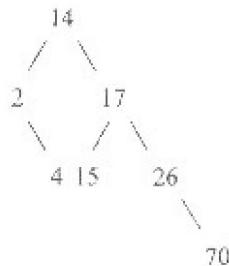
با توجه به درج عناصر به صورت صعودی و نزولی و بوجود آمدن درخت BST به صورت اریب استفاده از سایش BST به صورت آرایه اتفاق حافظه زیادی خواهد داشت در این حالت بهتر است از نمایش پیوندی استفاده کنیم.

پیماش Level order (سطح توپی) یک درخت BST:

در صورتیکه پیماش سطح ترتیب یک درخت BST را داشته باشیم به راحتی می‌توانیم به درخت پکتائی BST برسیم.
 مثال: فرض کنیم پیماش سطح ترتیب یک درخت BST به صورت:

14 , 2 , 17 , 4 , 15 , 26 , 70

باشد به راحتی می‌توان درخت آن را بدست آورد.



ابدا 14 را در ریشه فرار می‌دهیم سپس گره‌های بعدی را از سطح دوم به بعد از چپ به راست با ریشه هر زیر درخت مقایسه و در سمت چپ یا راست ریشه زیر درخت فرار می‌دهیم.

مثال: در صورتیکه اعداد ۱ تا ۱۰۰۰ در یک درخت BST درج شده باشند کدام وضعیت‌های جستجو نمی‌تواند درست باشد؟

۱۲۵ , ۲۴۰ , ۱۸۰ , ۱۹۵ , ۱۷۶ (۱)

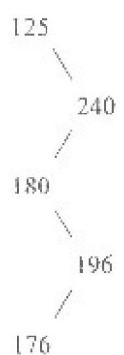
۱۰ , ۱۵ , ۲۵ , ۳۵ , ۷۰ (۱)

۱۷۰ , ۱۴۵ , ۱۳۲ , ۱۴۰ , ۱۴۴ (۲)

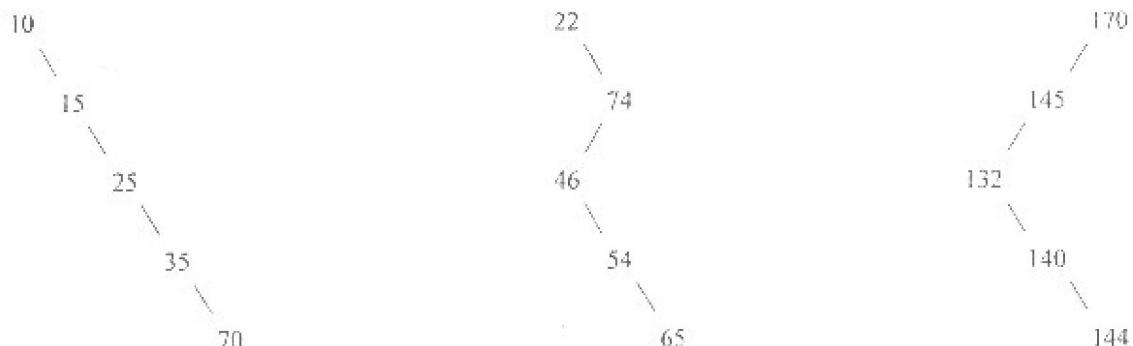
۲۲ , ۷۴ , ۴۶ , ۵۴ , ۶۵ (۲)

گزینه ۲ صحیح می‌باشد.

با توجه به وضعیت سمت راست ۱۸۰ باید از ۱۸۰ بزرگتر باشد ولی ۱۷۶ از ۱۸۰ کوچکتر است و در عین حال سمت راست قرار دارد که این حالت صحیح نیست.

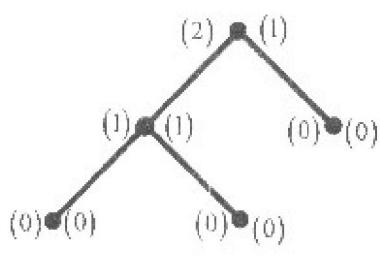


گزینه‌های دیگر همه حالت BST بودن خود را حفظ کرده‌اند.

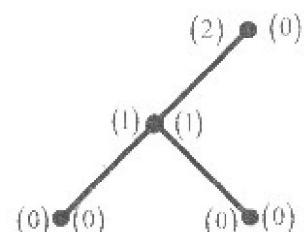


درخت با ارتفاع متوازن

درختی که در آن اختلاف ارتفاع در زیر درخت چپ و راست هر گره حداقل ۱ باشد، درخت با ارتفاع متوازن نامیده می‌شود.



الف) درخت با ارتفاع متوازن است.



ب) درخت با ارتفاع متوازن نیست.



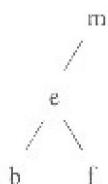
در شکل ب اختلاف دو زیر ارتفاع چپ و راست گره زیست، ۲ است و در نتیجه درخت با ارتفاع متوازن نیست.

مثال آوری:

درخت متوازن به درختی گفته می شود که اختلاف سطح برگ های آن حداقل ۱ باشد.

کلیه هر درخت با ارتفاع متوازن حتماً یک درخت متوازن است. نما عکس این مطلب همیشه درست نیست یعنی هر درخت متوازن لزوماً درخت با ارتفاع متوازن نیست.

مثال ۱:

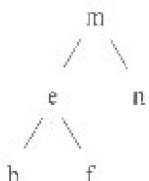


در شکل رو برو

الف) اختلاف سطح برگ های درخت (f, b) صفر بوده و در نتیجه درخت متوازن است.

ب) ارتفاع زیر درخت چپ m ۲ و ارتفاع زیر درخت راست n ۰ و در نتیجه درخت با ارتفاع متوازن نیست.

مثال ۲:



در شکل زو برو

الف) اختلاف سطح برگ های n, f, b حداقل ۱ است بنابراین درخت متوازن است.

ب) اختلاف ارتفاع زیر درخت چپ و راست هر گره حداقل ۱ است. بنابراین درخت با ارتفاع متوازن است.

نکته عهم:

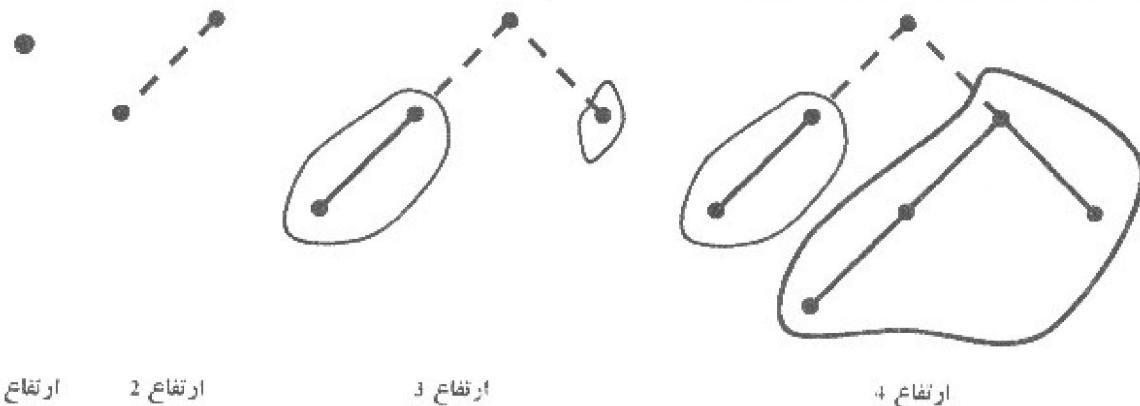
اگر $AVL(h)$ حداقل تعداد گره مورد نیاز برای ساختن یک درخت با ارتفاع متوازن h باشد. رابطه زیر همیشه برقرار است.

$$AVL(h) = AVL(h-1) + AVL(h-2) + 1$$

مثال: در صورتی که حداقل گره های لازم برای ساختن 2 درخت با ارتفاع متوازن 3، 4، 5 به ترتیب 4، 7، 15 باشد حداقل گره های لازم برای ساختن درخت با ارتفاع متوازن 5 گدام است؟

$$\begin{aligned} AVL(3) &= 4 \\ AVL(4) &= 7 \\ AVL(5) &= 15 \end{aligned} \Rightarrow AVL(5) = AVL(4) + AVL(3) + 1$$

شکل‌های درخت با ارتفاع متوازن (حداقل گره)

**درخت AVL**

هر درخت AVL یک درخت جستجوی دودویی (BST) بوده که ارتفاع متوازن دارد.

به درخت با ارتفاع متوازن درختی است که حداقل اختلاف دو زیردرخت چپ و راست هر گره ۱ باشد.

نکات مهم:

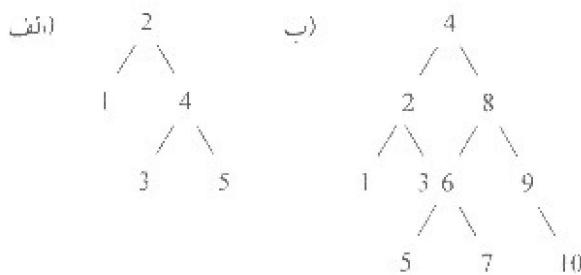
(۱) حداقل ارتفاع درخت AVL با n گره $\lceil \log_2 n \rceil + 1$ (در صورتیکه ریشه در سطح ۰) در نظر گرفته شود خواهد بود.

(۲) زمان عمل درج و حذف و جستجو در AVL بهتر بوده و زمان آن $O(\log_2 n)$ خواهد بود.

پادآوری:

زمان درج و حذف و جستجو در AVL از BST $O(h)$ بود که h عمق درخت بوده در بدترین حالت زمانیکه BST یک درخت با اریب با گره بود زمان حذف و درج $O(n)$ بود.

مثال: درخت‌های زیر نمونه‌هایی از درخت AVL هستند.



دیده می‌شود که درخت‌های (الف) و (ب) در عین حال که BST هستند دارای ارتفاع متوازن نیز هستند.

نتیجه‌گیری:

بین درخت‌های AVL، BST، AVL درخت AVL برای عملیات جستجو، درج و حذف بهتر است. چون ارتفاع آن متوازن بوده و در نتیجه هر عمل

درج یا حذف یا جستجو حداقل با زمان $O(\log_2 n)$ انجام می‌شود.



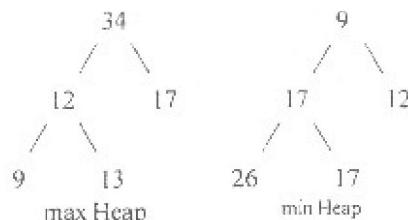
■ درخت Heap (نیمه مرتب - هرم)

درختی که مقدار کلید هر گره آن بزرگتر یا مساوی فرزندانش باشد، maxtree

درختی که مقدار کلید هر گره آن کوچکتر یا مساوی فرزندانش باشد، Mintree

درخت دودونی کاملی که maxtree نیز باشد، maxHeap

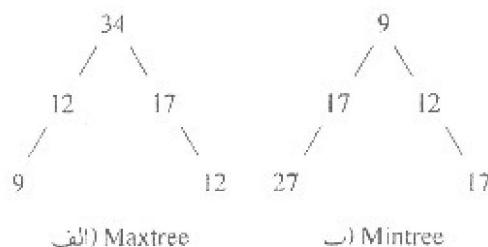
درخت دودونی کاملی که mintree نیز باشد، minHeap



مثال:

درخت (الف) maxtree است اما چون کامل نیست maxHeap نیست.

درخت (ب) mintree است اما چون کامل نیست minHeap نیست.



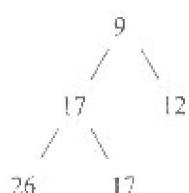
محاسبه گره ماکزیمم و می نیمم در Heap

درخت minHeap

۱- عنصر می نیمم در ریشه وجود دارد که بدون مقایسه بدست می آید.

۲- عنصر ماکزیمم در برگها وجود دارد و چون درخت کامل است تعداد برگها $\left\lfloor \frac{n+1}{2} \right\rfloor$ و تعداد مقایسات برای محاسبه عنصر ماکزیمم خواهد بود.

$$\left\lfloor \frac{n+1}{2} \right\rfloor - 1$$



در این درخت minHeap، برای بدست آوردن عنصر ماکریسم با ۲ مقایسه بین برگ‌ها به نتیجه می‌رسیم.

$$\text{تعداد گره‌ها} = n = 5$$

$$\text{تعداد برگ‌ها} = \frac{n+1}{2} = 3$$

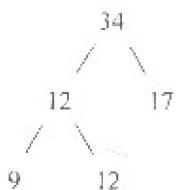
$$\text{تعداد برگ‌ها} = \text{تعداد مقایسات برای محاسبه ماکریسم} - 1 = 2$$

درخت max Heap

۱- عنصر ماکریسم در ریشه وجود دارد که بدون مقایسه بدست می‌آید.

۲- عنصر می‌نیسم در برگ‌ها وجود دارد و چون درخت کامل است تعداد برگ‌ها $\left\lfloor \frac{n+1}{2} \right\rfloor$ و تعداد مقایسات برای محاسبه می‌نیسم

$$\left\lfloor \frac{n+1}{2} \right\rfloor - 1 \quad \text{خواهد بود.}$$



در این درخت max Heap برای بدست آوردن عنصر می‌نیسم با ۲ مقایسه بین برگ‌ها به نتیجه می‌رسیم.

$$\text{تعداد گره‌ها} = n = 5$$

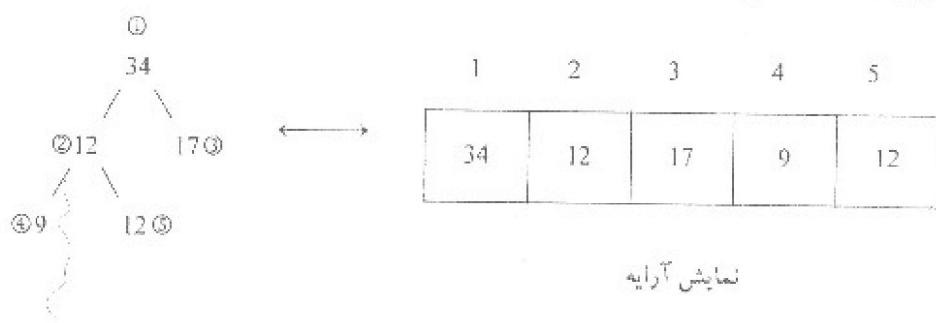
$$\text{تعداد برگ‌ها} = \frac{n+1}{2} = 3$$

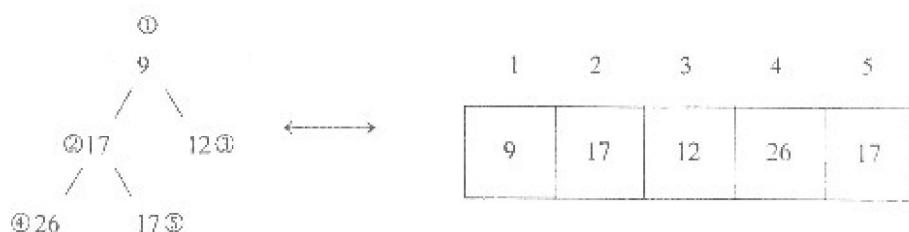
$$\text{تعداد برگ‌ها} = \text{تعداد مقایسات برای محاسبه می‌نیسم} - 1 = 2$$

نمایش Heap در حافظه

۱- برای نمایش mintree و maxtree چون لزوماً درختان کاملی نیستند از تماش پیوندی استفاده می‌شود. چون در صورت استفاده از آرایه اتلاف حافظه خواهد داشت.

۲- برای نمایش minHeap و maxHeap چون درختان کاملی هستند از آرایه‌ها استفاده کیم که در این حالت اتلاف حافظه‌ای نخواهیم داشت.





نمایش آرایه

بادآوری:

در شروع بحث درخت‌ها بیان‌سازی درخت به صورت آرایه نشان داده شد. که شماره‌گذاری از ۱ با گره ریشه آغاز می‌شد و تمام گره‌های دیگر از سطح بعدی از چپ به راست شماره‌گذاری می‌شدند.

درج و حذف در درخت Heap

درج گره در Heap

۱- گره جدید را در آخرین سطح در اولین موقعیت از چپ به راست قرار می‌دهیم.

(الف) در maxHeap: تا جایکه از والدش (parent) بزرگتر باشد باید با

آن جایجا شود (حداکثر تاریخه)

(ب) در minHeap: تا جایکه از والدش (parent) کوچکتر باشد باید با

آن جایجا شود (حداکثر تاریخه)

۲- گره درج شده

:Heap برای درج هر گره در

زمان درج گره در Heap $\left(O(\log_2 n) \right)$

عمل درج گره با نوچه به آن که گره درج شونده در آخرین سطح ممکن است با تمام والدهای خود تاریخه جایجا شود زمان $O(h)$ خواهد داشت (h ارتفاع درخت Heap) و جون این درخت کامل است $(h = \log_2 n)$ در نتیجه زمان درج گره در Heap در $O(\log_2 n)$ خواهد بود. الگوریتم درج: با نوچه به آن که برای نگهداری Heap از آرایه استفاده می‌کنیم در الگوریتم زیر در آرایه $H[1..n]$ که n گره Heap وجود دارد عمل درج عنصر x در maxHeap انجام می‌شود.

```

procedure insertHeap (H : array Heap; n : integer; x : integer);
var
  i, j: integer;
begin
  i := n + 1; j := i div 2;
  while (j > 0) and (H[j] < x) do
    begin
      H[i] := H[j];
      i := j;
      j := i div 2;
    end;
  H[i] := x;
end;

```

در الگوریتم بالا
از گره جدید x را نشان می‌دهد.
از گره والد x را نشان می‌دهد.

نرمیکه $j > 0$ (به ریشه نرسیده باشیم) و $x[j] < x[i]$ از والد بزرگتر باشد) حلقه while تکرار شده و گره $x[i]$ با والدش $x[j]$ جایجا می‌شود. در هر مرحله برای آن که به گره والد $x[i]$ برسیم از جمله $i = j \text{ div } 2$ استفاده می‌کیم.

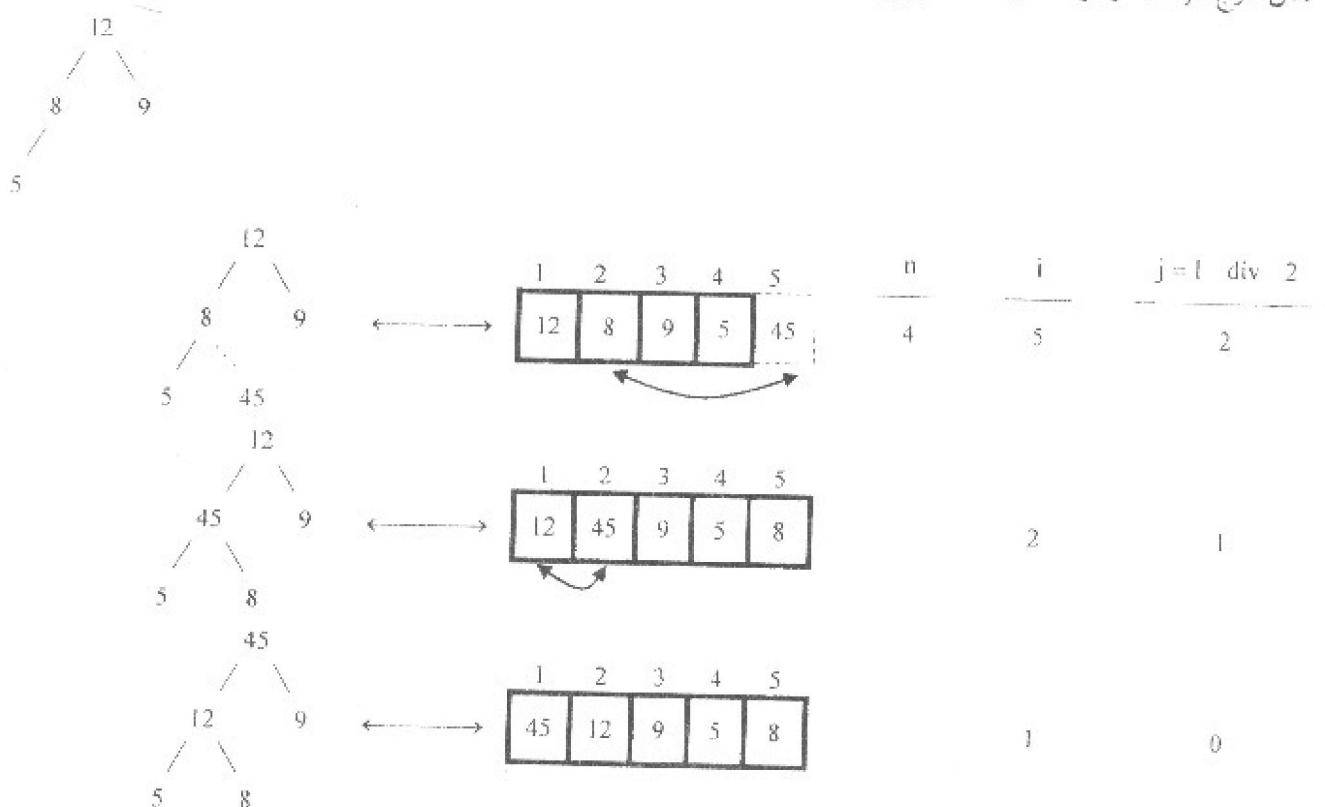
پادآوری:

در بحث نمایش درخت به صورت آرایه بیان شد که

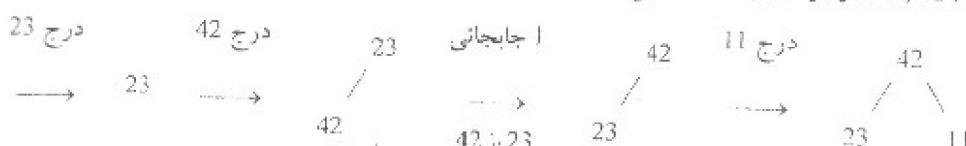


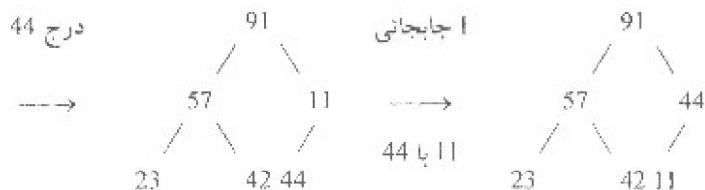
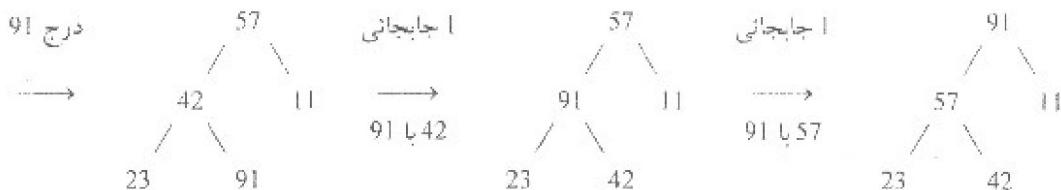
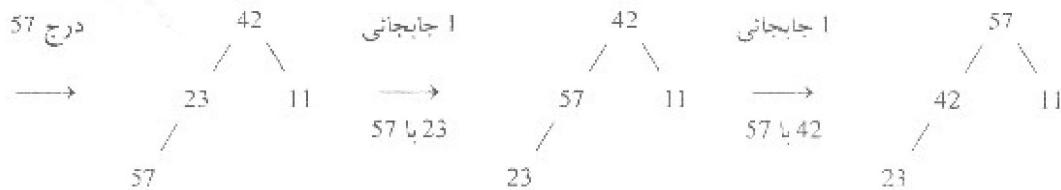
و همچنین والد گره i در برابر $\left\lfloor \frac{i}{2} \right\rfloor$ قرار دارد.

مثال: درج گره 45 در درخت maxHeap را درج



□ درج داده‌های 23, 42, 11, 57, 91, 44 در درخت max Heap تهی:





گره 23 درخت تهی بوده و گره 23 به عنوان اولین گره درج شود.

گره 42 با 1 جایگانی با گره 23 در موقعیت مناسب درخت max Heap قرار می‌گیرد.

گره 11 بدون جایگانی در موقعیت خود باقی می‌ماند.

گره 57 با 2 جایگانی با گره‌های 23، 42 در موقعیت مناسب درخت max Heap قرار می‌گیرد.

گره 91 با 2 جایگانی با گره‌های 42، 57 در موقعیت مناسب درخت max Heap قرار می‌گیرد.

گره 44 با 1 جایگانی با گره 11 در موقعیت مناسب درخت max Heap قرار می‌گیرد.

نکته: هر گاه درخت Heap به صورت آرایه مطرح شد، برای نمایش درخت عنصر اول را ریشه در نظر گرفته و عناصر بعدی گره‌های سطوح را

از جب به راست نشان می‌دهند.

مثال: درخت Heap زیر را در نظر بگیرید.

1	2	3	4	5	6	7
10	15	17	25	27	19	20

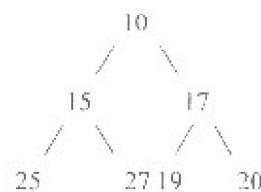
وضعیت درخت Heap چگونه است؟

الف) عنصر اول = ریشه درخت (10) و از همه کوچکتر در نتیجه درخت minHeap است.

ب) نمایش درخت:

ساقتمان داده‌ها

۱	۲	۳	۴	۵	۶	۷
10	15	17	25	27	19	20



حذف گره از Heap

۱- ریشه حذف شده و آن را در خروجی می‌نویسیم.

۲- سمت راست ترین گره آخرین سطح جایگزین ریشه می‌کنیم.

(الف) در `maxHeap`: تا جاییکه از فرزندانش کوچکتر است باید بزرگترین فرزند جایجا شود.

(ب) در `minHeap`: تا جاییکه از فرزندانش بزرگتر است باید کوچکترین فرزند جایجا شود.

۳- گره جایگزین شده

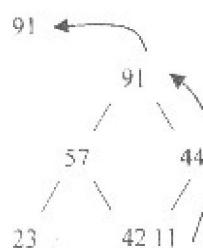
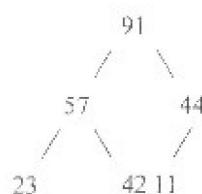
برای حذف هر گره از Heap

زمان حذف گره از Heap

عمل حذف گره باتوجه به آن که گره جایگزین در ریشه حداکثر ممکن است نا آخرین سطح با گره‌های دیگر جایجا شود زمان $O(h)$ خواهد داشت (ا) ارتفاع درخت Heap بوده که چون این درخت کامل است $h = \log_2 n$ در نتیجه زمان حذف گره از heap $O(\log n)$ خواهد بود.

حذف گره‌های درخت max Heap درخت زیر:

(الف) حذف اول (91)

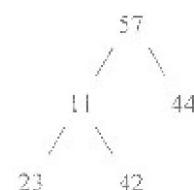


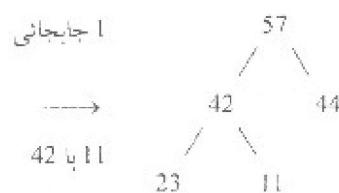
ریشه حذف شده (91) سپس
سمت راست ترین گره آخرین سطح
(11) جایگزین آن می‌شود.



۱- جایگزین

57 با 11





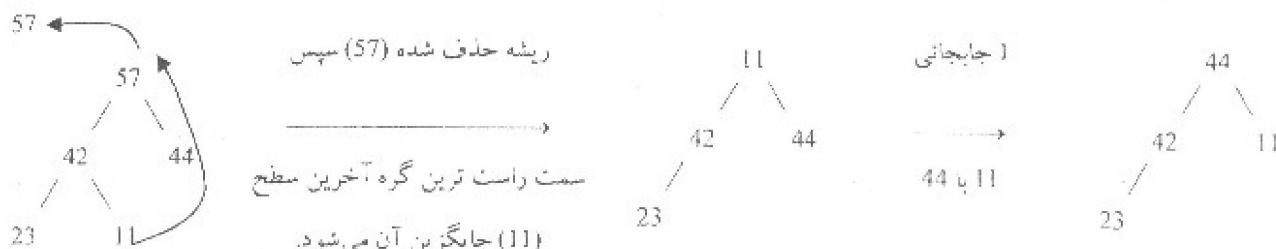
گروه ۹۱ را ۲ جانجانی از درخت حذف می‌شود.

بعد از قرار گرفتن گره 11 به جای 91 برای آن که شرایط max Heap بودن درخت حفظ شود.

^{۲۳} فرزندان گره ۱۱ بعنی ۴۴، ۵۷ فرزند پر نگه دارن (۵۷) با گره ۱۱ جایجا می شود سپس گره ۱۱ والد گره های ۴۲، ۲۳ می شود که درین

ا) ف زندان 42 با 11 حجاجا می شود تا max Heap یه دن درخت حفظ شود.

(٥٧) ملخص درس (٢)

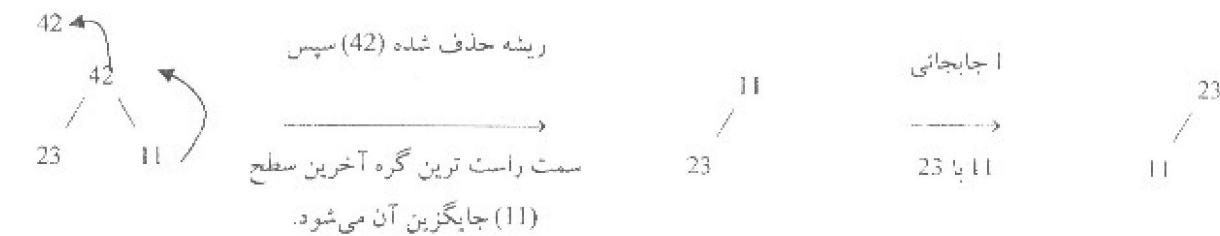


ریشه حذف شده (44) سپس

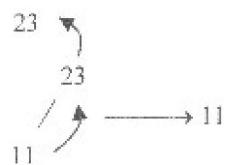
سمت راست ترین گره آخرین سطح

(23) حاگانه آن نمایش دارد

(42) حذف سه م



د) حلف چهارم (23)



(ه) حذف پنجم (11)

**کاربرد درخت Heap**

- ۱- مرتب سازی آرپهها
- ۲- پیاده سازی صفحه اولویت (priority Queue)
- ۳- ادغام نیستهای مرتب

۱- مرتب سازی Heap

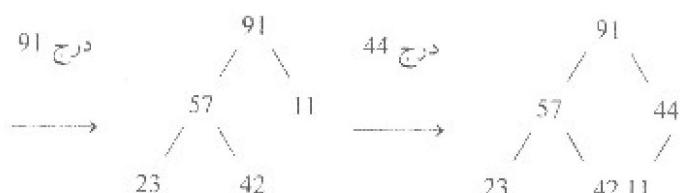
برای مرتب سازی یک آرپه n تایی به روش Heap به ترتیب باید ۲ عمل انجام شود.
 الف) درج تمام داده‌ها در یک درخت Heap تهی (`maxHeap`، `minHeap` مرتب سازی صعودی)
 ب) حذف تمام داده‌ها از ریشه درخت Heap

۲- زمان مرتب سازی Heap

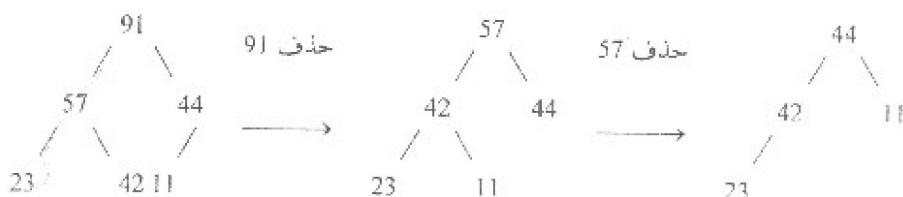
از آنجاییکه زمان درج با حذف در $O(n \log n)$ می‌باشد، در نتیجه برای مرتب سازی یک آرایه n تایی زمان مورد نظر $O(n \log n)$ خواهد بود.

مثال: مرتب سازی داده‌های 23, 42, 11, 57, 91, 44 به سیویه درخت Heap:

قسمت اول: درج داده‌ها در Heap



قسمت دوم: حذف داده‌ها از Heap





۲- صفت اولویت

تعریف: صفت اولویت به مجموعه‌ای از عناصر گفته می‌شود که در آن به هر عنصر یک اولویت داده می‌شود. و ترتیب حذف و پردازش داده‌ها بر طبق دو قاعده زیر است:

- ۱- عنصری که دارای اولویت بیشتر است قبل از بقیه عناصر حذف شده و پردازش می‌شود.
- ۲- دو عنصری که دارای اولویت یکسان هستند با توجه به ترتیبی که به صفت وارد شده‌اند پردازش می‌شود.

یادآوری:

در هر صفت اولویت دو عمل حذف و درج عنصر انجام می‌شود.

پیاده سازی صفت اولویت: به طور کلی سه روش برای پیاده سازی صفت اولویت وجود دارد:

- ۱- آرایه‌ها
- ۲- لیست پیوندی
- ۳- درخت Heap

که درخت Heap از بقیه ساختارها بهتر است چون زمان کمتری برای درج و حذف نیاز دارد.

زمان‌های حذف و درج در این ساختارها به شرح زیر هستند:

ساختار	زمان درج	زمان حذف
آرایه نامرتب	$O(1)$	$O(n)$
آرایه مرتب	$O(n)$	$O(1)$
لیست نامرتب	$O(1)$	$O(n)$
لیست مرتب	$O(n)$	$O(1)$
✓ درخت Heap	$O(\log_2 n)$	$O(\log_2 n)$

۱- درج: عنصر به راحتی در ابتدای لیست با در انتهای آرایه درج می‌شود. $O(1)$

۲- حذف: برای پیدا کردن پیشتر اولویت باید کل عناصر جستجو شوند و در آرایه‌ها نیاز به شیفت

داریم. $O(n)$

در آرایه‌ها و لیست‌ها نامرتب:

۱- درج: برای درج عنصر باید باتوجه به اولویت عنصر موقعيت آن را بین داده‌های پیدا کنیم و در

آرایه نیز نیاز به شیفت داریم $O(n)$

۲- حذف: عنصر با اولویت بالا را در آرایه‌ها از انتهای در لیست‌ها از ابتدا با زمان $O(1)$ حذف

می‌کنیم.

در آرایه‌ها و لیست‌های مرتب:

۱- درج: با زمان $O(\log_2 n)$ انجام می‌شود.

در درخت Heap

۲- حذف: با زمان $O(\log_2 n)$ انجام می‌شود.

نمای دیده می‌شود که ساختار Heap در دو عمل حذف و درج زمان کمتری نسبت به دو ساختار آرایه و لیست پیوندی نیاز دارد.

۱- ادغام لیست‌ها

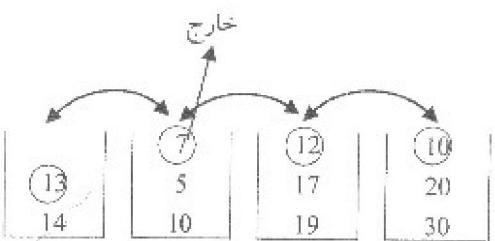
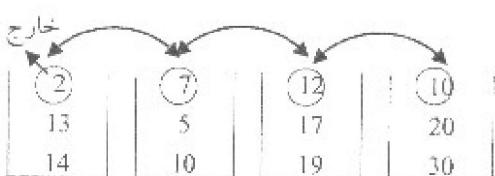
فرض کنید K آرایه مرتب (صعودی) داریم که تعداد کل خانه‌های K آرایه n می‌باشد. حال می‌خواهیم این K آرایه را ادغام کرده و در یک آرایه به صورت مرتب نگهداری کنیم.

(الف) روش عمومی:

۱- عضو اول همه آرایه‌ها را باهم مقایسه کرده و با $1 - K$ مقایسه، عضو می‌نیعم را خارج می‌کنیم.

۲- در این حالت آرایه‌ای که عضو می‌نیعم از آن خارج شده عناصرش را یک واحد به سمت بالا حرکت می‌دهد (رانش - RUN).

۳- عمل ۱ را دوباره انجام می‌دهیم (برای $1 - n$ عضو دیگر)



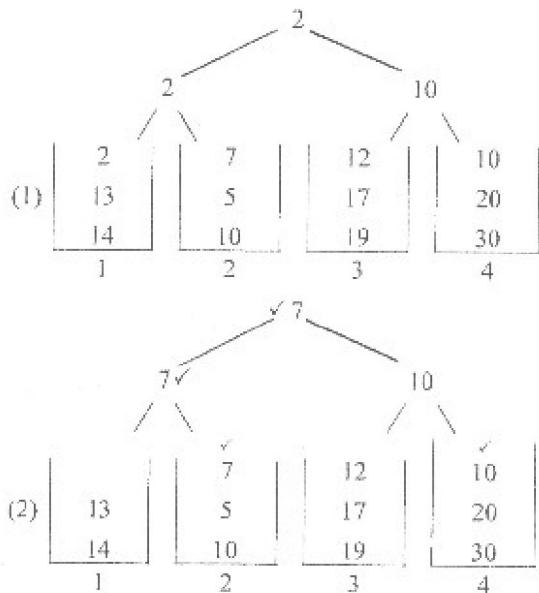
زمان اجرا: برای خروج هر عضو $(1 - k)$ مقایسه انجم می شود در تبجه برای n عضو آرایه $(1 - n)$ مقایسه انجم می شود و زمان ادغام $O(n \log k)$ می باشد.

ب) روش درخت انتخابی

تعریف: هر درخت انتخابی درخت دودوئی کاملی است که در آن هر گره درخت از فرزندانش کوچکتر با مساوی است و عنصر می نیم در ریشه درخت است. (\minHeap)

ادغام آرایه ها با درخت انتخابی

برای ادغام آرایه ها عنصر اول همه آرایه ها را به عنوان برگ های درخت انتخابی در نظر گرفته و درخت انتخابی را درست می کنیم (برنده (کوچکتر) - بازنده (بزرگتر))، به این صورت که بین هر دو برگ کنار هم گره ای را به عنوان والد انتخاب می کنیم (برنده) که کوچکتر باشد. این عمل را آن قدر تکرار می کنیم تا این که به ریشه بررسیم و کوچکترین عنصر در ریشه قرار گیرد. با مقایسه 2 با 2 برند و با مقایسه 10 با 12 برند می شود و در ریشه قرار می گیرد.



روش کار:

- ۱- ریشه درخت انتخابی را حذف کرده و در آرایه نهانی قرار می دهیم (عنصر می نیم) (گره 2 در شکل ۱).
- ۲- برای تجدید ساختار در آرایه ای که عنصر می نیم در مرحله اول از آن انتخاب شده يك واحد به بالا حرکت کرده (در آرایه اول در این مثال) و با مقایسات مناسب (علامت ✓) دوباره درخت انتخابی را ایجاد می کنیم (شکل ۲).
- ۳- عملیات ۱، ۲ را تا زمانی که کلیه عناصر آرایه به صورت مرتب خارج شوند نکار می کنیم.

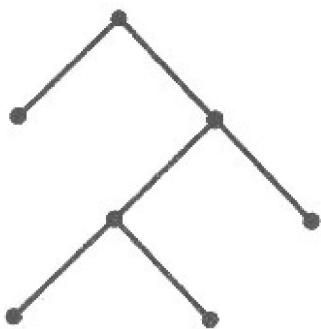
زمان اجرا: برای هر تجدید ساختار به اندازه ارتفاع درخت زمان باز داریم $O(\log_2 k)$ و برای کل عناصر آرایه n عنصر) زمان نهانی $O(n \log k)$ می باشد.

نتیجه گیری:

دیده می شود که روش تبخیر با استفاده از درخت minHeap از نظر زمانی مفروض به صرفه تر از روش عمومی می باشد چون $O(n \log_2 k) < O(nk)$ می باشد.

درخت توسعه یافته (دودوئی محض):

تفصیل: به درختی دودوئی که در آن هر گره با 2 فرزند داشته باشد یا 0 فرزند داشته باشد.



گره های خارجی و داخلی

در درخت دودوئی محض (توسعه یافته) به گره های 2 فرزندی گره های داخلی و به گره های 0 فرزندی گره خارجی می گویند.



لکته:

$$\text{اگر } \begin{cases} \text{تعداد گره های خارجی } N_E = \\ \text{تعداد گره های داخلی } N_I = \end{cases} \quad \begin{aligned} N_E &= N_F + 1 && \text{آن گاه} \\ & \text{به عبارت بeter: } 1 + \text{تعداد گره های 2 فرزندی} && \end{aligned}$$

در مثلا باید $N_E = 4$ و $N_I = 5$ باشد.

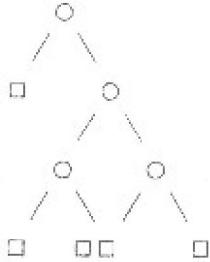
طول مسیر خارجی (L_E) - طول مسیر داخلی (L_I)

بنابر تعریف:

L_E = مجموع تمام طول مسیرها که طول هر مسیر برابر تعداد یالها در مسیر ریشه تا یک گره خارجی می باشد.

L_I = مجموع تمام طول مسیرها که طول هر مسیر برابر تعداد یالها در مسیر ریشه تا یک گره داخلی می باشد.

در درخت بلا



$$L_I = \frac{0}{m} + \frac{1}{n} + \frac{2}{o} + \frac{2}{p} = 5$$

$$L_E = \frac{1}{A} + \frac{3}{B} + \frac{3}{C} + \frac{3}{D} + \frac{3}{E} = 13$$

نکته مهم:

$$L_F = L_I + 2n$$

 در درخت بلا $n = 4$ گره داخلی وجود دارد. در نتیجه

$$L_E = L_I + 2n = 5 + 2 \times 4 = 13$$

طول مسیر وزن در گره‌های خارجی

اگر به هر گره خارجی یک وزن بدheim (w_i)، طول مسیر وزن گره‌های خارجی به صورت زیر قابل محاسبه است:

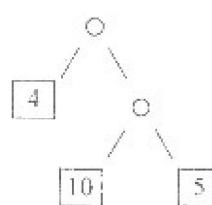
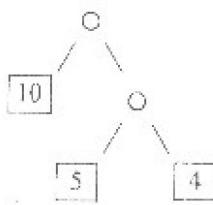
در صورتیکه m گره خارجی داشته باشیم:

$$p = w_1 L_1 + w_2 L_2 + \dots + w_m L_m$$

L_i = طول مسیر یک گره خارجی

w_i = وزن یک گره خارجی

نکته مهم: پیاده‌سازی مختلف گره‌های خارجی در قالب درخت دودوئی مخصوص وزن‌های متفاوتی را از نظر طول مسیر وزن تولید می‌کند.



$$p_1 = 1 \times 10 + 2 \times 5 + 3 \times 4 = 28$$

$$p_2 = 1 \times 4 + 2 \times 10 + 2 \times 5 = 34$$

در دو درخت بلا بین که سه گره خارجی با وزن 4، 5، 10 وجود دارد اما طول مسیر وزن‌ها متفاوت است. دیده می‌شود که بهتر است ابتدا

گره‌های خارجی با وزن کمتر را برای پانین ترین سطح انتخاب کنیم تا در مجموع وزن کمتری داشته باشیم.

الگوریتم هافمن (تعیین درخت با حداقل طول مسیر وزن)

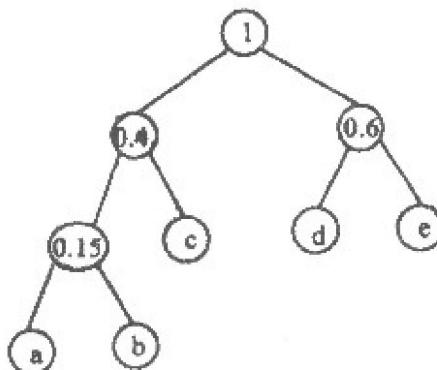
در این الگوریتم برای ایجاد درخت با حداقل طول مسیر وزن در هر مرحله دو درختی را که ریشه می‌نیم دارند را انتخاب می‌کنیم (در شروع

کار از گره‌های خارجی استفاده می‌کنیم).

مثال: حروف e, b, c, d, e با جدول فراوانی زیر داده شده است:

حروف	a	b	c	d	e
فراوانی	0.05	0.1	0.25	0.28	0.32

درخت هافمن وابسته به این حروف به قرار زیر است:



مرحله اول: a, b می‌نیمم بوده و انتخاب می‌شود و ریشه با مقدار 0.15 ایجاد می‌کنند.

مرحله دوم: در بین اریشه (a, b) و c را که می‌نیمم هستند انتخاب می‌کنیم که ریشه ادغام آنها 0.4 می‌شود.

مرحله سوم: در بین اریشه (c, (a, b)) می‌شود.

مرحله چهارم: به ریشه درخت می‌رسیم.

۵. ربرد الگوریتم هافمن در کدگذاری

یکی از کاربردهای مهم درخت می‌نیمم ایجاد شده به روش هافمن فشرده سازی داده‌ها با توجه تکرار آنها می‌باشد، در این حالت برای هر داده کدی انتخاب می‌شود.

فرض کنید می‌خواهیم n عنصر اطلاعاتی A_1, A_2, ..., A_n را به وسیله رشته‌هایی از بیت‌های 0 یا 1 کدگذاری کنیم، در این حالت برای هر عنصر 0 یک کد در نظر گرفته می‌شود، که این کد با توجه به قواعدی ایجاد می‌شود.

نکته مهم:

برای کدگذاری n عنصر اطلاعاتی حداقل به ۳ بیت نیاز داریم که از رابطه زیر بدست می‌آید.

$$2^{r-1} < n \leq 2^r$$

مثال ۲: برای کدگذاری 5 کاراکتر حداقل به چند بیت نیاز داریم

$$4 = 2^{3-1} < 5 \leq 2^3 = 8 \Rightarrow r = 3$$

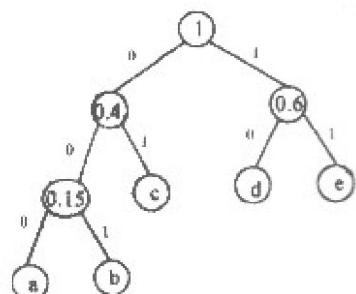
حداقل به ۳ بیت نیاز داریم.



روش کار:

- ۱- درخت هافمن با حداقل طول مسیر وزن را تشکیل می‌دهیم.
- ۲- از ریشه شروع کرده به تمام اتصال‌های چپ ۰ و تمام اتصال‌های راست ۱ می‌دهیم.
- ۳- برای پیدا کردن کد هر عضو از ریشه تا آن عضو ۰ و ۱‌ها را پشت سرهم می‌نویسیم.

مثال ۳: در درخت مثلث ۱ در صورت کدگذاری خواهیم داشت:



در مثال بالا برای ۵ کرکتر با ۳ بیت عملیات کدگذاری انجام شده است.

تفکه مهم:

دیده می‌شود که عناصری که تکریز کمتری داشته‌اند کد آن‌ها دارای پیشوند یکسانی بوده و فقط در قسمت انتهائی با هم متفاوت هستند.

$$a = \underbrace{0 \ 0} \ 0$$

$$b = \underbrace{0 \ 0} \ 1$$

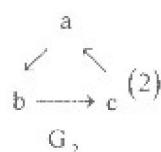
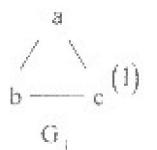
گراف

تعریف:

هر گراف G شامل دو مجموعه V و E است: $V =$ مجموعه محدود و احتمالاً تهی از نماینده $E =$ مجموعه محدود و غیرتهی از زووسهر گراف G به صورت (V, E) نمایش داده می‌شود.

نتیجه ۱: هر گراف حداقل یک رأس دارد و نمی‌تواند کاملاً تهی باشد.

نتیجه ۲: در گراف‌ها دو وضعیت جهت دار و بدون جهت وجود دارد که:



$$V(G_1) = \{a, b, c\}$$

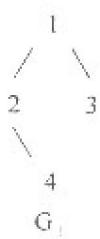
$$V(G_2) = \{a, b, c\}$$

$$E(G_1) = \{(a, b), (a, c), (b, c)\} \quad E(G_2) = \{(a, b), (b, c), (c, a)\}$$

گراف دست کید که در گراف جهت دار G_1 $(v_1, v_2) = (v_2, v_1)$ اما در گراف بدون جهت G_2 $(v_1, v_2) \neq (v_2, v_1)$ است.

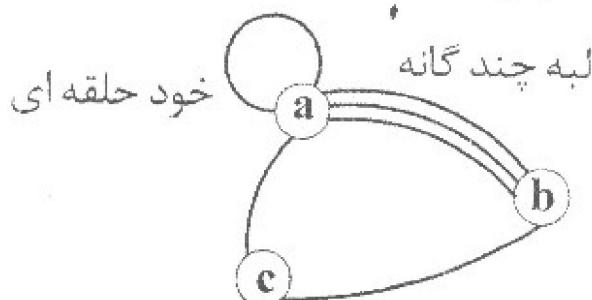
هر درخت حالت خاص از یک گراف است که سیکل یا دور ندارد. (هر گراف، درخت نیست اما هر درخت گراف است)

مثال:

گراف G_3 درخت نیست.گراف G_4 به علت داشتن دور یا سیکل درخت نیست.

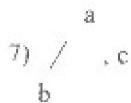
گراف چندگانه (multi graph): گرافی که دارای نمایندهای چندگانه باشد.

گراف خود حلقه‌ای با حلقه باز خودردی: گرفتی که در آن نماینده‌ای از یک رأس به خودش وجود داشته باشد.



زیر گراف: هر زیر مجموعه از V (رنویس) و E (ابهای) به عنوان زیر گراف های G می توانند در نظر گرفته شوند.

مثال ۱:



۱ - گراف زیر را در نظر بگیرید. کدام گزینه، زیر گراف گراف فوق است؟ (آزاد ۸۲)



۴) هر سه گزینه

گزینه ۴ صحیح می باشد.

□ همسایگی (مجاور بودن) (adjacent)

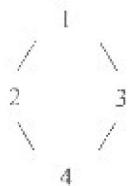
دو گره U ، V را در یک گراف مجاور (همسایه) گویند اگر گاه لبه مستقیمی بین U ، V وجود داشته باشد. به عبارت بهتر:

(الف) در گراف بدون جهت: دو گره U ، V به شرطی مجاور (همسایه) هستند که لبه (یا) (V, U) وجود داشته باشد.

(ب) در گراف جهت دار: هر گاه لبه (V, U) وجود داشته باشد می گویند V مجاور از رأس U است. یعنی یال مستقیمی از U به V وجود دارد.

به عبارت بهتر U مجاور به رأس V است.

مثال ۱: در گراف بدون جهت رویرو:



(الف) گره های مجاور (همسایه) گره ۱: گره های ۲، ۳، ۴

(ب) گره های مجاور (همسایه) گره ۲: گره های ۱، ۳، ۴

(ج) گره های مجاور (همسایه) گره ۳: گره های ۱، ۲، ۴

(د) گره های مجاور (همسایه) گره ۴: گره های ۲، ۳

در عین حال گره ۴ مجاور گره ۱ نیست، گره ۲ مجاور ۳ نیست.

مثال ۲: در گراف جهت دار رویرو:



$$E(G) = \{\langle 1, 2 \rangle, \langle 2, 1 \rangle, \langle 2, 3 \rangle\}$$

$$N(G) = \{1, 2, 3\}$$

(الف) گره ۲ مجاور به رأس ۱، ۳ است.

(ب) گره ۱ مجاور به رأس ۲ است.

(ج) گره ۳ مجاور به رأس ۲ نیست چون یال $\langle 3, 2 \rangle$ وجود ندارد و فقط مجاور از رأس ۲ است چون از ۲ به ۳ یال وجود دارد. $\langle 2, 3 \rangle$

(د) گره ۳ نه مجاور به رأس ۱ است یعنی یال $\langle 1, 3 \rangle$ وجود ندارد و نه مجاور از رأس ۱ است یعنی یال $\langle 3, 1 \rangle$ وجود ندارد.

□ صراف ساده: گرافی که فقط خود حلقه‌ای و لبه‌های چندگانه باشد.

صرف کامل: گرافی که دارای حداقل لبه‌ها باشد. به عبارت بهتر:

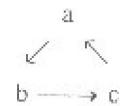
بین هر زوج گره دلخواه آن لبه مستقیمی وجود داشته باشد.

با

هر گره نامجاور هر گره دیگر V در گراف باشد.

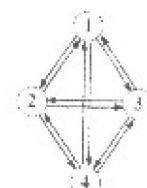
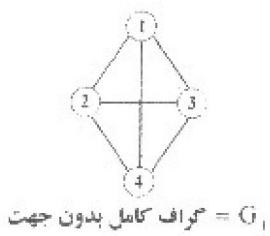


مثال: گراف‌های غیر کامل



گراف غیرکامل جهت‌دار

مثال: گراف‌های کامل


 گراف کامل جهت‌دار $= G_2$

حداکثر تعداد لبه‌ها برای یک گراف بدون جهت با n رأس $\frac{n(n-1)}{2}$

حداکثر تعداد لبه‌ها برای یک گراف جهت‌دار با n رأس $n(n-1)$

فرمول‌های فوق در حقیقت تعداد لبه‌های یک گراف کامل بدون جهت $\frac{n(n-1)}{2}$ و جهت‌دار $n(n-1)$ هستند.

مثال ۱: در گراف بدون جهت کامل G_1 با $4 = n$ رأس با نوچه به رابطه $\frac{n(n-1)}{2}$ به تعداد زیر یافته، وجود دارد:

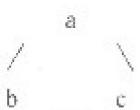
$$n = 4 \rightarrow \frac{4(4-1)}{2} = 6$$

مثال ۲: در گراف جهت‌دار کامل G_2 با $4 = n$ رأس با نوچه به رابطه $n(n-1)$ به تعداد زیر یافته، وجود دارد:

$$n = 4 \rightarrow 4(4-1) = 12$$

مسیر: هر مجموعه از لبه‌های پشت سرهم که دو رأس را به هم متصل می‌کنند مسیر نامیده می‌شود.

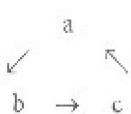
مثال ۱:



در گراف روی رو

a: مسیری از رأس a به رأس c

b: مسیری از رأس b به رأس c



مثال ۲: در گراف روی رو

a: مسیری از a به b

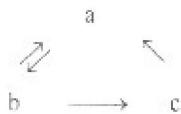
c: مسیری از c به a

در عین حال $b \rightarrow a$ یا $c \rightarrow a$ مسیر نیستند چون از c به b و از b به a لبه مستقیم نداریم.

طول مسیر: تعداد نمایه‌های موجود در مسیر را طول مسیر می‌گویند.

مسیر ساده: مسیری است که همه رئوس آن به جز احتمالاً اول و آخر، متفاوت است.

حلقه یا سیکل: مسیر ساده‌ای که اولین و آخرین رأس آن با هم برابر است.



مثال: در گراف جهت دار رویرو

سیکل	$a \rightarrow b \rightarrow c \rightarrow a$
سیکل	$c \rightarrow a \rightarrow b \rightarrow c$
سیکل	$b \rightarrow a \rightarrow b$

مسیرهای ساده به طول 2

سیکل	$a \rightarrow b \rightarrow c \rightarrow a$
سیکل	$c \rightarrow a \rightarrow b \rightarrow c$
سیکل	$b \rightarrow c \rightarrow a \rightarrow b$

مسیرهای ساده به طول 3

مثال: در گراف غیر جهت دار رویرو



مثال: در گراف غیر جهت دار رویرو

(سیکل)	$a \rightarrow b \rightarrow c \rightarrow a$
(سیکل)	$c \rightarrow a \rightarrow b \rightarrow c$
(سیکل)	$b \rightarrow a \rightarrow b$
(سیکل)	$a \rightarrow c \rightarrow a$
(سیکل)	$a \rightarrow b \rightarrow a$
(سیکل)	$c \rightarrow a \rightarrow c$
(سیکل)	$c \rightarrow b \rightarrow c$
	$b \rightarrow c \rightarrow a$
	$b \rightarrow a \rightarrow c$
	$c \rightarrow b \rightarrow a$
	$a \rightarrow c \rightarrow b$

مسیرهای ساده به طول 2

(سبک)	$a \rightarrow b \rightarrow c \rightarrow a$	سیرهای ساده به طول ۳
(سبک)	$b \rightarrow c \rightarrow a \rightarrow b$	
(سبک)	$c \rightarrow a \rightarrow b \rightarrow c$	

سیرهای ساده به طول ۳

ساده نیستند چون نکرار گره‌ها در وسط سیر دیده می‌شود.

- اگر a, b دو گره در یک گراف بدون جهت G باشند و اگر دو سیر P_1, P_2 از a به b وجود داشته باشد، آنگاه: (دولتی ۸۲)

(۱) a, b مجاورند.
نمی‌تواند یک گراف باشد.

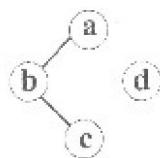
(۲) G دارای چرخه است.
اجتناب به جهت سیر داریم.

زیره ۳ صحیح می‌باشد.

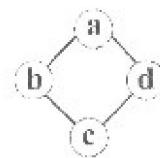
نه ۳: در گراف بدون جهت دور اس دلخواه v_1, v_2, \dots, v_n را متصل می‌گوئیم، اگر مسیری از v_1 به v_2 و یا بالعکس وجود داشته باشد.

نه ۴: گراف بدون جهت را متصل یا همبند گوئیم، اگر برای هر دور اس v_1, v_2, \dots, v_n مسیری از v_1 به v_2 و یا بالعکس وجود داشته باشد.

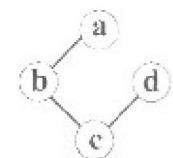
ج: هر درخت یک گراف همبند (متصل) بدون دور یا حلقه است.



گراف غیر متصل درخت نیست.

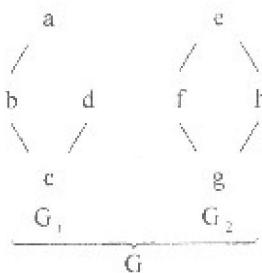


گراف متصل درخت است.



گراف متصل درخت است.

لله اتصال: یک مؤلفه اتصال با به طور ساده‌تر یک مؤلفه در گراف بدون جهت بزرگترین زیرگراف متصل آن گراف است.

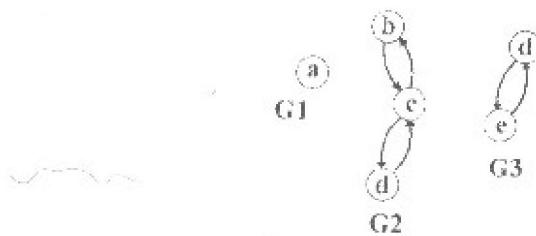


در گراف G رویرو G_1 و G_2 دو مؤلفه اتصال گراف G هستند.

نه ۵: یک گراف جهت دار کاملاً متصل نامیده می‌شود. هر گاه برای هر زوج رأس v_i و v_j مسیری جهت دار از v_i به v_j و همچنین از v_j به v_i وجود داشته باشد.

نه ۶: یک مؤلفه کاملاً متصل بزرگترین زیرگرافی نیست که کاملاً متصل باشد.

سافتمنان داده‌ها



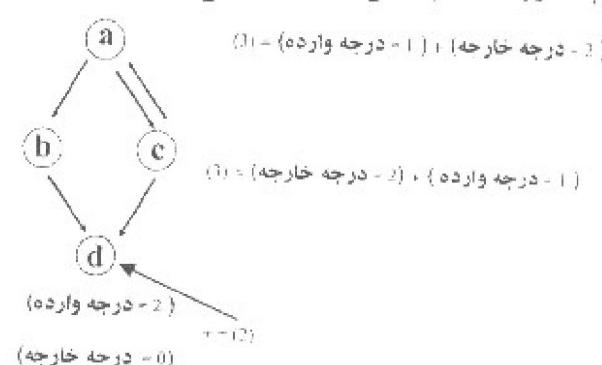
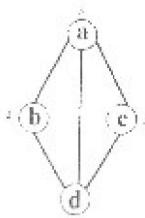
درجه یک رأس در گراف:

۱. درجه یک رأس از گراف بدون جهت تعداد لبه‌ها با بالهای متناظر با آن رأس است.

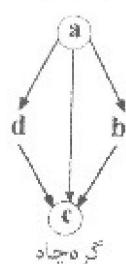
۲. در گراف جهت دار درجه یک رأس برابر است با: درجه وارد + درجه خارج

درجه وارد: تعداد بالهایی که به رأس وارد می‌شود.

درجه خارج: تعداد بالهایی که از رأس خارج می‌شوند.



گره صیغ

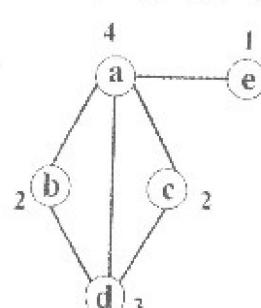
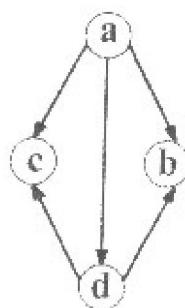


گره چاد: به گره‌ای که درجه خروجی آن در گراف جهت دار ۰ باشد.

گره منبع: به گره‌ای که درجه ورودی آن در گراف جهت دار ۰ باشد.

درجه گراف: درجه هر گراف برابر با بزرگترین درجه گره‌های آن گراف است.

۱ - درجه گراف

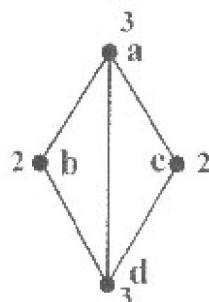


۲ = درجه گراف



نکته: اگر درجه رأسی فرد باشد آن رأس را فرد گویند و اگر زوج باشد آن رأس را زوج گویند.

۳) تعداد رأس‌های فرد یک گراف غیر جهت دار همواره عددی زوج است.

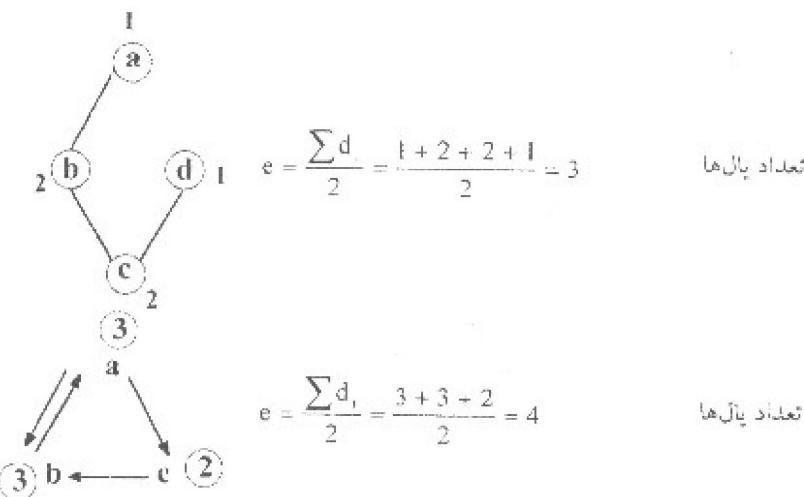


رأس فرد و وجود دارد.

نکته مهم: اگر در گراف دلخواه G (جهت دار یا بدون جهت) درجه تمام رئوس را بدایم آنگاه:

$$\frac{\text{مجموع درجه رئوس}}{2} = \frac{\text{تعداد یال‌ها}}{\text{تعداد یال‌ها}} \quad \text{یا} \quad e = \frac{\sum d_i}{2}$$

مثال ۱:



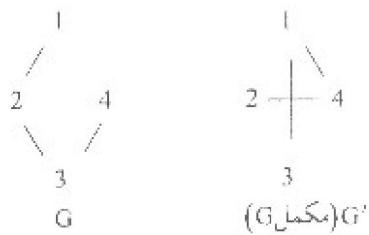
نکته مهم:

گراف مکمل: در صورتی که G یک گراف دلخواه باشد گراف مکمل G' گرافی است که با گراف G هیچ یال مشترکی نداشته ولی رئوس مشترک دارد و مجموع دو گراف یک گراف کامل تشکیل می‌دهند در این حالت

تعداد رئوس

$$\underbrace{G'}_{\text{تعداد یال‌ها}} + \underbrace{G}_{\text{تعداد یال‌ها}} = \frac{n(n-1)}{2} \quad \text{پیش فرض گراف بدون جهت}$$

مثال:



در این حالت:

$$\text{تعداد یال‌های } G = 3 + \text{تعداد یال‌های } G' = 6$$

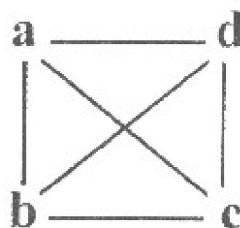
یعنی گراف کامل با 4 رأس $\frac{n(n-1)}{2}$ یا $\frac{4 \times 3}{2} = 6$ دارد.

مثال ۲: اگر G یک گراف ساده از درجه n باشد. در صورتی که گراف G دارای 56 یال باشد و مجموع درجات رئوس مکمل G برابر باشد مقدار n کدام است؟

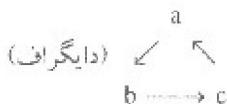
$$\sum d = 56 = \frac{n(n-1)}{2} \Rightarrow \frac{160}{2} + 56 = \frac{n(n-1)}{2} \Rightarrow n = 17$$

تعداد رئوس گراف $n = 17$ است اما در مسئله n درجه گراف G خواسته شده و چون در یک گراف کامل با n رأس درجه گراف $n - 1$ است پس جواب مسئله ۱۶ است.

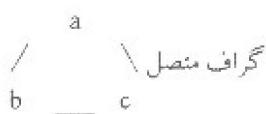
نکته: در یک گراف کامل با n رأس درجه گراف $n - 1$ است.



گراف کامل با 4 رأس از درجه 3



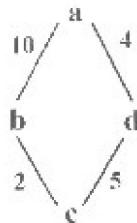
جمع به گراف جهت دار، دایگراف (digraph) نیز گفته می شود.



گراف G منصل است اگر و فقط اگر یک سیر ساده بین هر دو گره آن وجود داشته باشد.

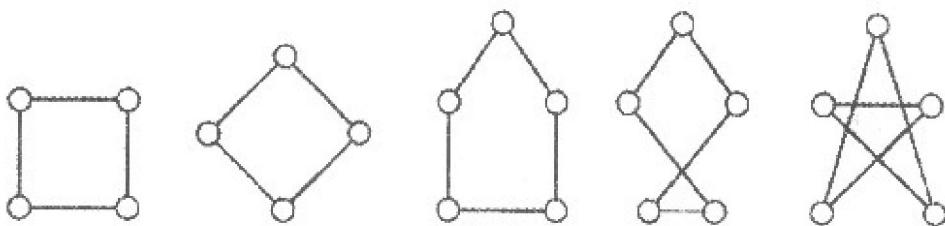
گراف G بر چسب دار یا وزن دار یا شماره دار می گویند اگر اطلاعاتی به یال های آن نسبت داده شود.

اگر اطلاع نسبت داده شده به یال هایک عدد غیر منفی باشد به آن وزن یا هزینه یال های می گویند.



۰ گراف بروجسبدار با وزن دار

تکته مهم: اگر در گرافی درجه تمام رأس‌ها دقیقاً ۲ باشد، گراف را منظم می‌گوئیم.



نمایش گراف:

به طور کلی روش‌های زیر برای نمایش گراف‌ها استفاده می‌شود:

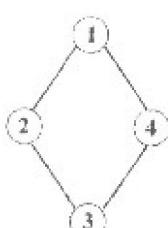
- (۱) ماتریس مجاورتی
- (۲) لیست مجاورتی (همجاوری)
- (۳) لیست مجاورتی محکوس

ماتریس مجاورتی:

فرض کنید $G(V, E)$ یک گراف با n رأس باشد؛ ماتریس مجاورتی گراف G یک آرایه دو بعدی است به صورت $n \times n$ که n^2 خانه دارد. و به صورت زیر یاده‌سازی می‌شود:

(۱) گراف بدون جهت:

در صورتیکه نباید بالی به صورت (v_i, v_j) وجود داشته باشد $A[i][j] = A[j][i] = 1$ است در غیر این صورت $A[i][j] = 0$ است.



	1	2	3	4
1	0	1	0	1
2	1	0	1	0
3	0	1	0	1
4	1	0	1	0

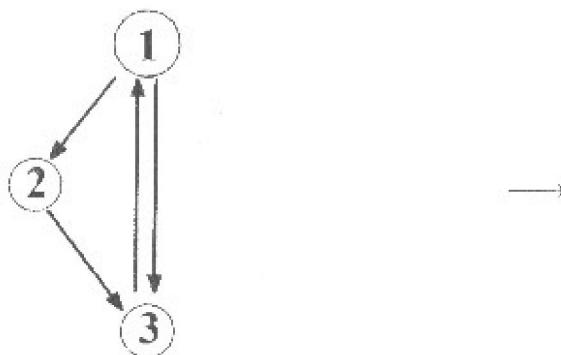
مشخصات:

الف) ماتریس مجاورتی یک گراف غیر جهت دار همیشه متقارن است، بنابراین همیشه می توانیم فقط بالا مثلث یا پائین مثلث را نگهداری کنیم. $\left(\frac{n(n-1)}{2}\right)$ خانه

ب) تعداد اهای ماتریس همیشه ۲ برابر تعداد بال ها است.

ج) مجموع عناصر سطری یا ستونی هر رأس درجه آن رأس را نشان می دهد.

۲- گراف جهت دار:



	1	2	3
1	0	1	1
2	0	0	1
3	1	0	0

الف) ماتریس مجاورتی یک گراف جهت دار همیشه متقارن نبست بنابراین همیشه باید $n \times n$ خانه ماتریس را نگهداری کرد.

ب) تعداد اهای ماتریس همیشه برابر تعداد بال ها است.

ج)

$$\text{مجموع عناصر ستونی هر رأس} = \text{درجه ورودی}$$

$$= \text{درجه رأس}$$

$$\text{مجموع عناصر سطری هر رأس} = \text{درجه خروجی}$$

نکات مهم در مورد ماتریس مجاورتی:

۱) از نقاط قوت ماتریس مجاورتی این است که سرعت عملیات دسترسی به این که آیا دو گره ۱ و ۰ مجاور هم هستند یا نه (یا مستقیم از آیه ز

هست با خبر) از مرتبه (۱) ۰ است.

۲) اغلب الگوریتم ها با استفاده از ماتریس مجاورتی با زمان (n^2) بدست می آیند.

۳) برای گراف اسپارس یعنی گرافی که دارای تعداد کمی نبه هست زمان لازم $(\alpha + \beta)n^2$ است.

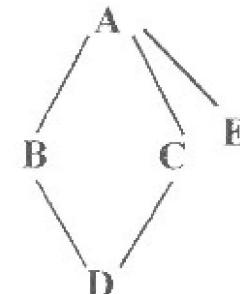
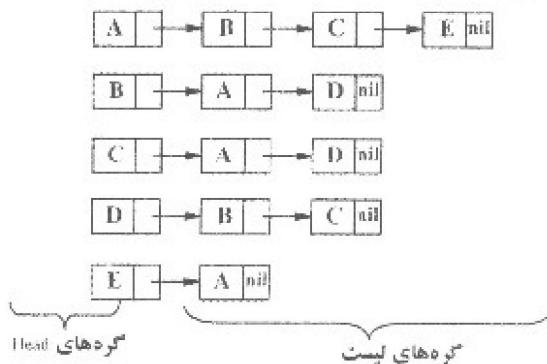
البته به شرطی که $\frac{n^2}{2} < \beta$ باشد.

۴) اگر A ماتریس مجاورتی گراف G باشد. در این صورت در ماتریس A^k تعداد مسیر هایی از آیه j را نشان می دهد که طول k

دارند.

لیست مجاورتی:
۱) گراف بدون جهت:

در این وضعیت برای یک گراف با n رأس و e نبه (بال) برای نمایش لیست مجاورتی به n گره Head و e گره، لیست احتیاج داریم.

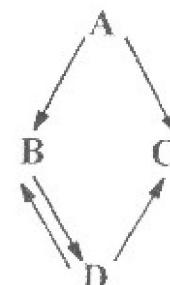
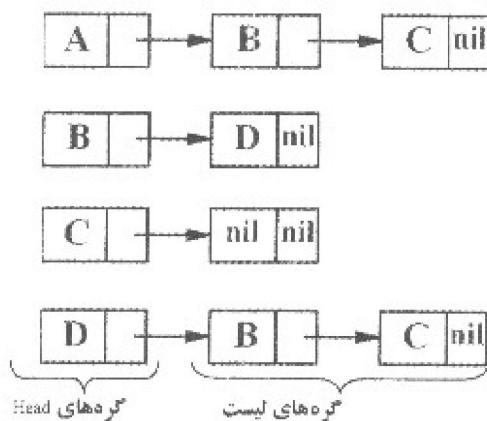


نکته: در شکل فوق درجه هر گره تعداد گره های لیست آن رأس می باشد. همسایه های هر گره را نیز می توانیم به راحتی در این وضعیت بدست

آوریم.

۲) لیست مجاورتی برای گراف جهت دار:

در این وضعیت برای گراف جهت دار با n رأس و e نبه (بال) لیست مجاورتی احتیاج به n گره Head و e گره نیست دارد.



نکات مهم در مورد لیست مجاورتی:

۱) اگر تعداد رئوس گراف G برابر n باشد. تعداد کل نبها در زمان $O(n + e)$ تعیین می شود.

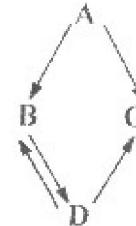
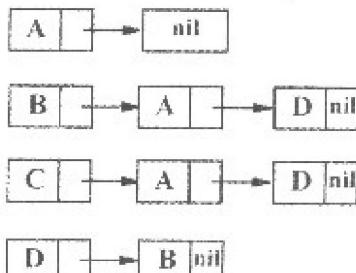
۲) برای یک گراف جهت دار، درجه خروجی هر رأس با شمارش تعداد گره ها در لیست مجاورتی آن گره به دست می آید این معناست که می توانیم تعداد کل خطوط یک گراف جهت دار را در زمان $O(n + e)$ تعیین کنیم.

لیست مجاورتی معکوس

در لیست مجاورتی (در ماتریس جهت دار) به راحتی می توانیم درجه خروجی هر رأس را بدست آوریم که همان تعداد گره های لیست هر گره می باشد. برای پیدا کردن درجه ورودی به راحتی نیز توانیم عمل کنیم. چون باید تمام گره های Head را جستجو کنیم به غیر از گره های

سافتمنان داده‌ها

که می‌خواهیم درجه خروجی آن را بدست آوریم، برای آن که به راحتی بتوانیم این کار را انجام دهیم می‌توانیم از لیست مجاورتی معکوس استفاده کنیم که در این وضعیت باز هم **گره Head** داریم.

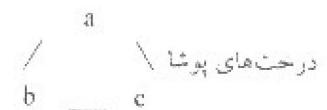


تکه: در لیست مجاورتی معکوس گره‌های لیست، درجه ورودی را نشان می‌دهند و از روی آن می‌توانیم درجه گره خروجی را بدست آوریم ولی به سختی.

درخت پوشش (Spanning tree)

در صورتیکه G یک گراف متصل (همبند) با n رأس، حداقل $n - 1$ لبه دارد. در این صورت درخت پوشش در این گراف درختی است که n رأس گراف را داشته و دقیقاً $n - 1$ لبه (یال) را اختیار می‌کند.

مثال: در گراف کامل روی رو با 4 رأس درخت‌های پوشای ممکن نشان داده شده‌اند که هر کدام 3 لبه دارد.



مثال ۲: در یک گراف کامل با 10 رأس چند یال را کم کنیم تا به درخت پوشش برسیم:

10 (۴)

9 (۳)

36 (۲)

45 (۱)

حل: تعداد یال‌های یک گراف کامل با n رأس (بیش فرض بدون جهت)

$$\frac{n(n-1)}{2} = \frac{n(n-1)}{2} \rightarrow n = 10 \quad \text{تعداد یال‌ها} = \frac{10 \times 9}{2} = 45$$

در یک درخت پوشش از یک گراف با n رأس $n - 1$ یال وجود دارد در نتیجه در گراف با 10 رأس درخت پوشش 9 یال دارد.

بنابراین $45 - 9 = 36$ یال باید کم شود.

پیمایش گراف‌ها

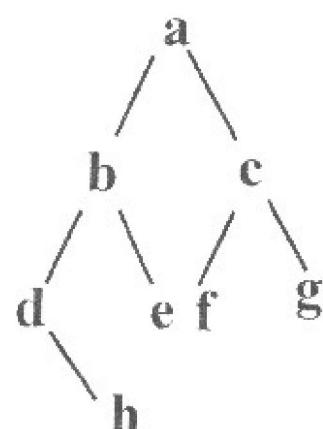
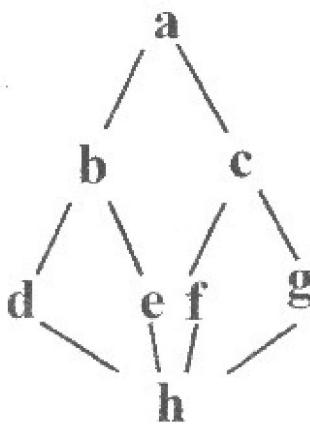
به طور کنی برای پیمایش گراف دو روش وجود دارد که منجر به درخت پوشش spanning tree می‌شود. درخت پوشای یک گراف که n رأس دارد و حاصل از پیمایش گراف می‌باشد حتماً $n - 1$ یال دارد.

(الف) پیمایش bfs:

این پیمایش را پیمایش سطحی (عرضی، ردیفی) می‌نامیم و برای شبیه‌سازی آن از صفت استفاده می‌کنیم. به ترتیب زیر هر گراف دلخواه را می‌توانیم به فرم سطحی یا bfs پیمایش کنیم. ابتدا رأس شروع را در نظر می‌گیریم. سپس همسایه‌های مستقیم آن را ملاقات می‌کنیم. این کار را از سمت چپ به راست انجام می‌دهیم. بعد از این کار از سمت چپ‌ترین همسایه شروع کرده همسایه‌های مستقیم آن را می‌ینیم. دقت کنید اگر همسایه‌ای برای یک گره دیده شود، آن گره برای گره‌های دیگر دیده نمی‌شود. برای آن که حتماً باید درخت پوشا داشته باشیم.

مثال: مطابقت پیمایش bfs گراف زیر از رأس a:

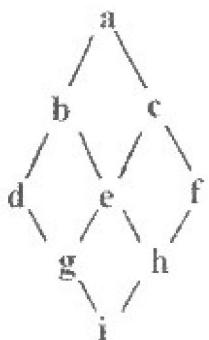
bfs (a): a b c d e f g h



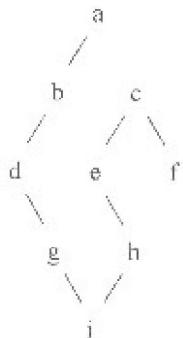
زمان اجرای الگوریتم bfs چنانچه از لیست های مجاور تری استفاده شود $O(n + e)$ می‌باشد.

(ب) پیمایش dfs (depth first search) (عمقی)

در این پیمایش از هر گره که شروع کنیم با استفاده از صفت چپ‌ترین همسایه مستقیم آن در عمق حرکت می‌کنیم و مراحل در یک stack شبیه‌سازی و نگهداری می‌شود. در هر مرحله اگر به بن‌بست برخورد کنیم، یک عنصر را از پشته خارج کرده و به مرحله قبلی بر می‌گردیم تا شاید عنصر دیگری برای دیدن وجود داشته باشد. ادر هر مرحله از پیمایش گره‌ای را انتخاب می‌کنیم که اولاً قبلاً رویت نشده باشد و ثانیاً در بن‌بست همسایه‌های چپ‌ترین باشند اگر به بن‌بست خوردم به مرحله قبل برگشته تا همسایه‌های دیگر آن را بررسی کنیم.

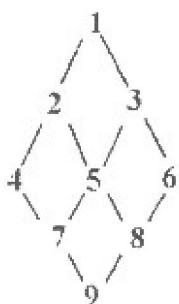
مثال: پیمایش عمقی گراف زیر چیست؟ (مسابقات آموزشکده‌های فنی ۷۹)


حل:

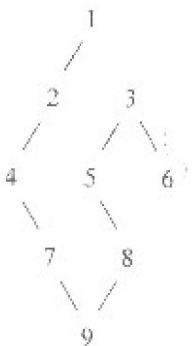


dfs: a, b, d, g, i, h, e, c, f

مثال: پیمایش عمقی گراف زیر کدام است؟ (دولتی ۸۲)



حل:



dfs: 1, 2, 4, 7, 9, 8, 5, 3, 6

پیمایش ۱۰۰ با استفاده از لیست مجاورتی:

۱- از گره رأس ملاقات را شروع می‌کیم.

۲- اوین گره لیست از گره رأس که تا به حال ملاقات نشده است را ملاقات کرده و سپس به لیستی می‌رویم که گره رأس آن گره لیست انتخاب شده است. هر زمان در یک لیست تمام گره‌ها قبل ملاقات شده بوده به مرحله قبل برگشته و گره دیگر را انتخاب می‌کیم.

۳- عنن ۲ را تازماینکه همه گره‌ها ملاقات شوند دنبال می‌کیم.

مثال: لیست مجاورتی مربوط به گراف G به صورت زیر است. پیمایش عمتمی این گراف کدام است:

Adjacent list

A : B, C, D

B : A, D, E

C : A, D, F

D : A, B, C

E : B, F

F : C, E

A B D C F E (۱)

AB E F C A (۲)

A C F D E B (۳)

A D C B E F (۴)

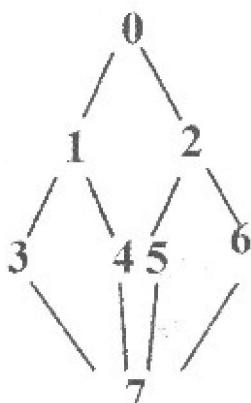
حل: گزینه ۱ صحیح می‌باشد.

از رأس A شروع می‌کنیم، و داخل لیست آن B را ملاقات کرده به رأس B می‌رویم چون A قبلًا ملاقات شده D را ملاقات کرده به رأس D می‌رویم A, B قبلًا ملاقات شده‌اند پس C را ملاقات کرده به رأس C می‌رویم. A, C قبلًا ملاقات شده‌اند پس F را ملاقات کرده به رأس F می‌رویم C قبلًا ملاقات شده، E را ملاقات کرده به رأس E می‌رویم در این حالت F قبلًا دیده شده‌اند و همین طور گره‌ها ملاقات شده پس

A B D C F E: پیمایش تمام شده است. خروجی:

اگر گراف G با لیست مجاورتی نمایش داده شود، زمان لازم توسط dfs برابر $O(n)$ است از آن‌جا که حلقه for به زمانی برابر $O(n)$ نیاز دارد، کی زمان برای تولید همه گراف‌های متصل $(n \times n)$ است. اگر گراف G توسط ماتریس مجاورتی ارائه شود آن‌گاه زمان لازم برای تعیین گراف‌های متصل برابر $O(n^2)$ می‌باشد.

مثال: پیمایش dfs گراف زیر کدام است؟



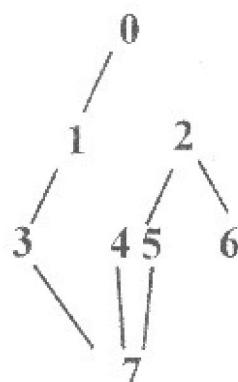
1, 3, 4, 6, 0, 2, 4, 6 (۱)

0, 1, 2, 3, 4, 5, 6, 0, 7 (۲)

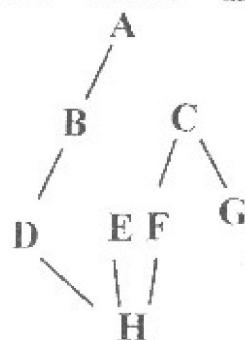
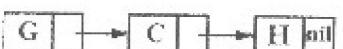
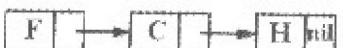
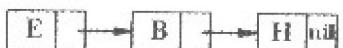
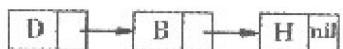
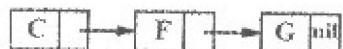
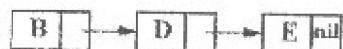
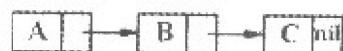
0, 1, 3, 7, 4, 5, 2, 6 (۳)

0, 2, 1, 4, 3, 5, 6, 7 (۴)

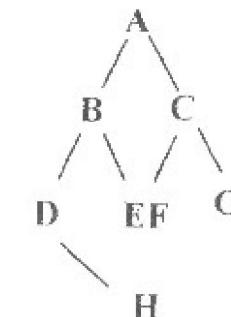
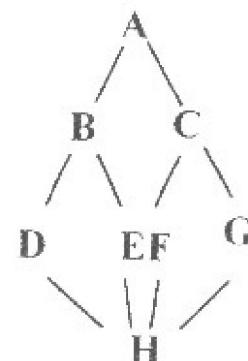
گزینه ۳ صحیح می‌باشد.



تمرین: مصلویست پیمایش سطحی و عمیق گراف زیرا



bfs: A B C D E F G H



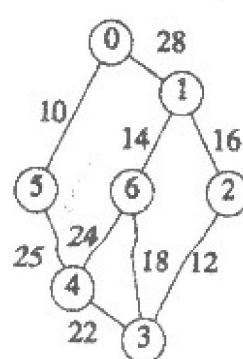
dfs: A B D E F C G

درخت پوشای کم هزینه:

اساساً در گرافی که بیان‌ها وزن داشته باشند به آن گراف وزن‌دار می‌گوییم. در این وضعیت پیمایش گراف وزن‌های متفاوتی را به وجود می‌آورد. برای بدست آوردن درخت پوشایی که هزینه با وزن کمی را داشته باشد می‌بایست از الگوریتم‌های خاصی استفاده کنیم. تمام این الگوریتم‌ها بر دو اصل پیاده‌سازی می‌شود.

۱- همیشه در هر مرحله‌ای یائی را انتخاب می‌کنند که کمترین وزن را با مسیر انتخاب شده تا به حال داشته باشد.

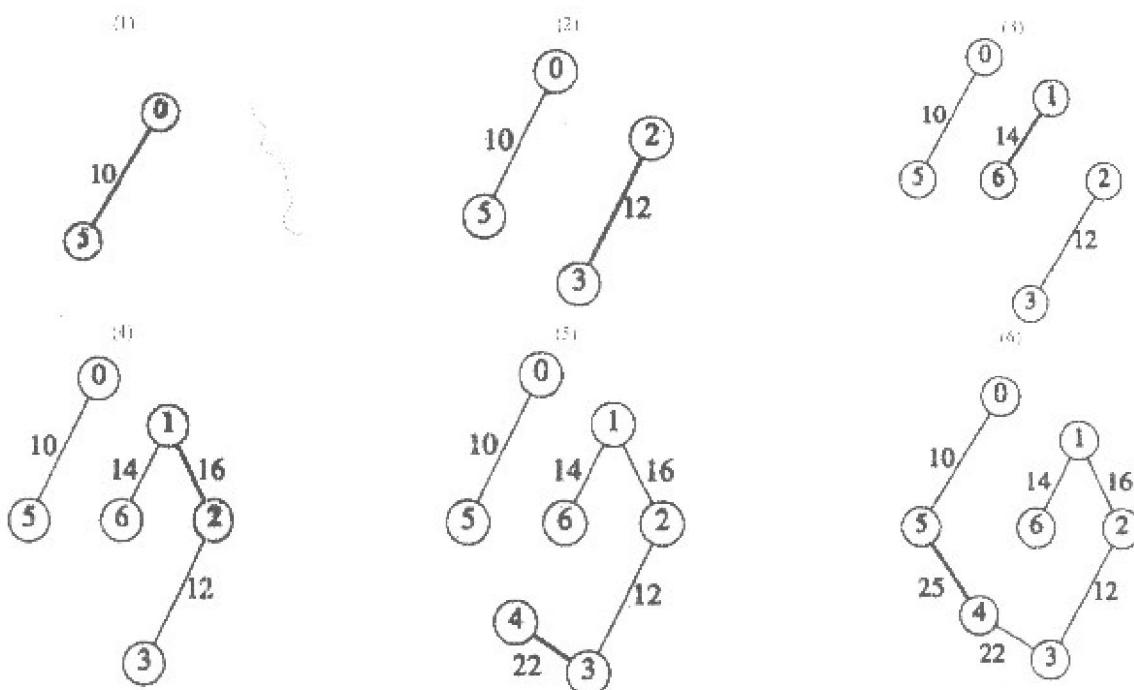
۲- همیشه یائی که انتخاب می‌کنیم نباید با مسیر انتخاب شده تا به حال حلقه ایجاد کند.





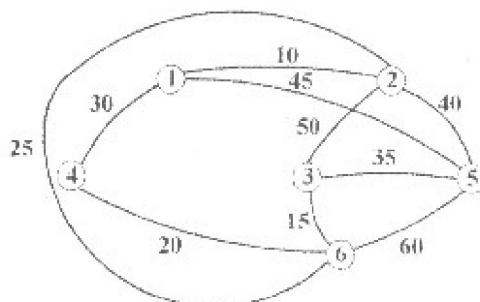
الف) الگوریتم راشل (کراسکال)

در این الگوریتم در هر مرحله یائی را انتخاب می‌کنیم که کمترین وزن را دارد. در این حالت رأس شروع مهم نیست چون ما از یال کوچک شروع می‌کنیم و فقط دقت می‌کنیم یائی که انتخاب می‌کنیم با درخت ایجاد شده تا به حالت خنثه تولید نکند. نکته مهم این است که در هر مرحله از الگوریتم راشل ممکن است یک جنگل داشته باشیم. در مثال زیر هزینه مینیمم گراف را برای درخت پوشاند است آورده‌ایم. (گراف بالا)

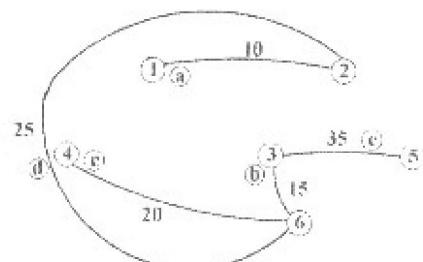


Rashal:

مثال:



حل:

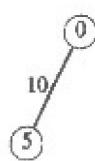


(prim) پرایم:

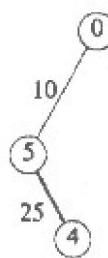
از یک رأس دلخواه شروع کرده و در هر مرحله رأسی را انتخاب می‌کنیم که با یکی از رئوس انتخاب شده تا به حال کمترین وزن را داشته باشد. تا درخت حاصل نیز کمترین وزن را داشته باشد. دقت می‌کنیم رأسی که انتخاب می‌کنیم نباید با مسیر انتخاب شده تا به حال ایجاد حلقه کند.

دقت کنید در هر مرحله از الگوریتم پرایم حتماً یک درخت وجود دارد.

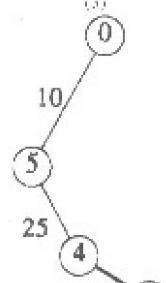
(1)



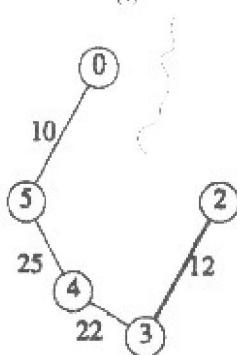
(2)



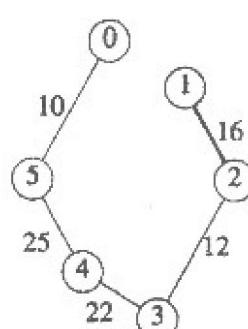
(3)



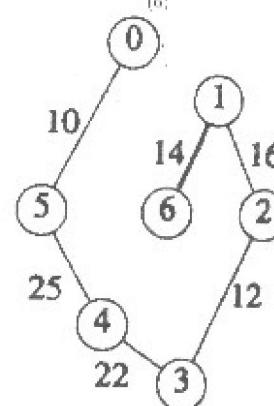
(4)



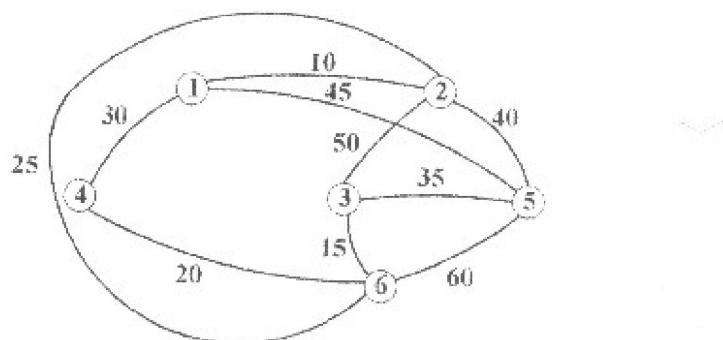
(5)



(6)

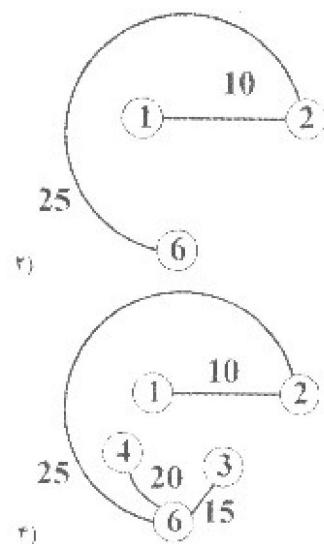
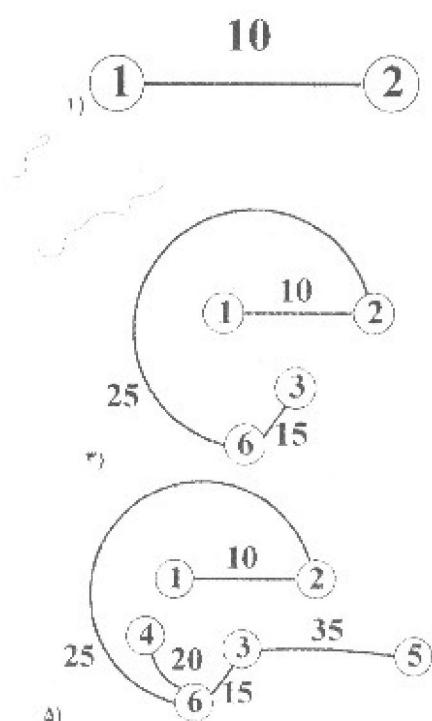


مثال:

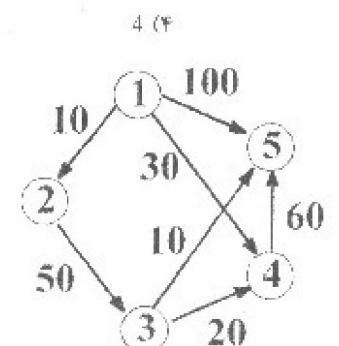


حل:

Prim:



تعریف: در گراف زیر کمترین هزینه از گره ۱ به ۵ دارای مسیری به طول چند است؟



۳ (۲)

۲ (۲)

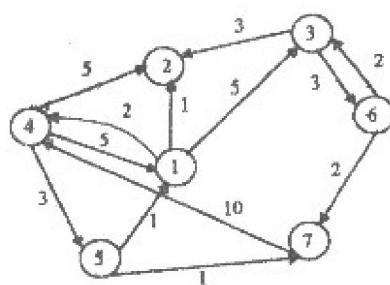
۱ (۱)

$$\begin{array}{c}
 1 \rightarrow 2 \rightarrow 3 \rightarrow 5 \\
 \hline
 70
 \end{array}$$

$$\begin{array}{c}
 1 \rightarrow 4 \rightarrow 5 \\
 \hline
 90
 \end{array}
 \quad
 \begin{array}{c}
 1 \rightarrow 5 \\
 \hline
 100
 \end{array}$$

$$\begin{array}{c}
 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \\
 \hline
 140
 \end{array}$$

۳. مثال: در گراف زیر کمترین هزینه از گره ۱ به گره 7 عبارت است:



8 (۱)

10 (۲)

20 (۳)

6 (۴)

گزینه ۴ صحیح می‌باشد.

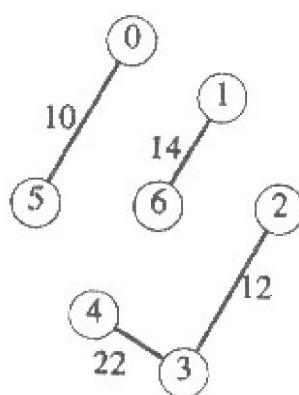
$$① \xrightarrow{3} ④ \xrightarrow{3} ⑤ \xrightarrow{1} ⑦$$

مسیر

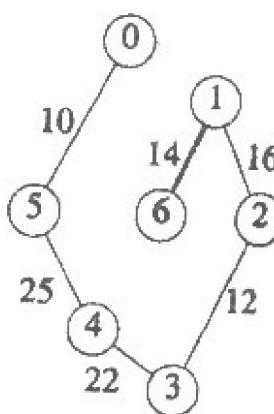
ج) الگوریتم سالین: (sollin)

در این روش در مرحله اول یک جنگل مشکل از تمام رأس‌های گراف تشکل می‌دهیم به طوریکه هر یکی از آن می‌بایست باشد. سپس با انتخاب یال‌هایی با هزینه می‌بایست مسیری می‌کنیم جنگل را به درختی با هزینه می‌بایست تبدیل کنیم.

(۱)



(۲)



زمان الگوریتم‌ها

۱- حلقه‌ها

برای محاسبه زمان حلقه‌ها، تعداد تکرار آن‌ها را بحسب آورده پس از روی آن زمان الگوریتم را محاسبه می‌کنیم.

مثال ۱

$$\begin{aligned} & \text{for } i := 1 \text{ to } n \text{ do} \\ & \quad \text{for } j := 1 \text{ to } n \text{ do} \quad (\text{نکرار } n \times n = O(n^2)) \\ & \quad \quad \dots \end{aligned}$$

مثال ۲

for $i := 1$ to n do	i	j	نکرار
for $j := i + 1$ to n do	1	2 .. n	$n - 1$
...	2	3 .. n	$n - 2$
	:	:	:
	n	$n + 1 .. n$	0
			$\frac{n(n-1)}{2} = O(n^2)$

مثال ۳

for $i := 1$ to n do	i	j	k	نکرار
for $j := 1$ to $n - i$ do	1	1 .. $n - 1$	n	$(n-1) n$
for $k := 1$ to n do	2	1 .. $n - 2$	n	$(n-2) n$
...	:	:	:	:
	n	1 .. $n - n$	n	$(0) n$
				$O(n^3) = \left(\frac{n(n-1)}{2}\right) n$

مثال ۴

for $i := 1$ to k do	i	m	نکرار
begin	1	1 .. m	m
$j := 1$	2	1 .. m	m
while $j < m$ do	:	:	:
$j := j + 1;$	k	1 .. m	m
end;			$k m = O(k m)$

$$O(\log_2 n)$$

در دو وضعیت الگوریتم‌ها درای این مرتبه زمانی خواهد بود

۱- متغیر دلخواهی مانند A داتیمی بر مقدار ثابتی (k) تقسیم شود تا به حد پائینی برسد.

۲- متغیر دلخواهی مانند A داتیمی در مقدار ثابتی (k) ضرب شود تا به مقدار 0 برسد.



مثال ۱: فرض کنید $n = 8$ مطلوبست تعداد تکرار حلقه زیر:

```
j := 1;
while j <= n do
```

begin

 j := j * 2;

end;

j	تکرار
1	1
$1 \times 2 = 2^1$	2
$2^1 \times 2 = 2^2$	3
$2^2 \times 2 = 2^3$	4
$2^3 \times 2 = 2^4$	خروج

= 4 = $\log_2 8 + 1$

= $\log_2 n$

در حالت کلی در مثال بالا زمان $O(\log_2 n)$ خواهد بود.

نکته: در مثال بالا اگر شرط while به صورت $n < j$ باشد، تعداد تکرار $\log_2 n$ خواهد بود ولی زمان بازهم $O(\log_2 n)$ می‌باشد.

مثال ۲: تکرار حلقه زیر دارای چه مرتبه‌ای می‌باشد. (کنکورد ۸۳)

```
i = 1;
while (i <= n)
{
    j = i;
    while (j <= n)
        {
            j = j * 2;
        }
    i = i + 1;
}
```

n	تکرار
$8 = 2^3$	1
$4 = 2^2$	2
$2 = 2^1$	3
$1 = 2^0$	4
0	خروج

مثال ۳: برای $n = 8$ مطلوبست تعداد تکرار حلقه زیر:

```
i = n;
while i >= 1 do
```

 i = i / 2;

= 4 = $\log_2 8 + 1$

= $\log_2 n$

در حالت کلی زمان الگوریتم بالا $O(\log_2 n)$ خواهد بود.

نکته: در مثال بالا اگر شرط while به صورت $i > n$ باشد، آن‌گاه تعداد تکرار $\log_2 n$ خواهد بود ولی زمان بازهم $O(\log_2 n)$ می‌باشد.

۳- مرتبه زمانی در توابع بازگشته

در این توابع برای محاسبه مرتبه زمانی تعداد فرآخوانی‌ها با تعداد عملیات اصلی انجام شده نشان دهنده مرتبه زمانی خواهد بود که با محاسبه

برای چند مقدار نوبه می‌توان مرتبه زمانی بدست آورد.

مثال ۱:

$$T(n) = \begin{cases} T(n-2) * T(n-2) & n > 1 \\ 1 & n = 1 \end{cases}$$

 $T(4)$
 $n = 4 \rightarrow \text{تعداد فراخوانی} = 3$
 $T(8)$
 $n = 8 \rightarrow \text{تعداد فراخوانی} = 7$
 $T(2) * T(2)$
 $T(4) * T(4)$
 $T(1) * T(1) * T(1) * T(1)$

دیده می شود که تعداد فراخوانی برای هر $T(n)$ از رابطه $1 - 2^{n-2}$ بدست می آید. بنابراین مرتبه زمانی $O\left(2^{\frac{n}{2}}\right)$ خواهد بود.

مثال ۲:

$$T(n) = \begin{cases} T(n-1) * T(n-1) & n > 1 \\ 1 & n = 1 \end{cases}$$

 $n = 3 \rightarrow \text{تعداد فراخوانی} = 3$
 $T(3)$
 $T(2) * T(2)$
 $T(1) * T(1) * T(1)$
 $n = 4 \rightarrow \text{تعداد فراخوانی} = 7$
 $T(4)$
 $T(3) * T(3)$

دیده می شود که تعداد فراخوانی برای هر $T(n)$ از رابطه $1 - 2^{n-1}$ بدست می آید بنابراین مرتبه زمانی $O\left(2^n\right)$ می باشد.

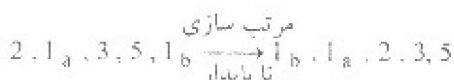
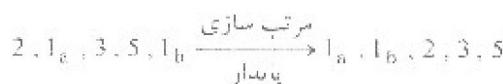
مرتب سازی (sort)

✓ عملیات مرتب سازی برای هر آرایه از دو عمل مقایسه و تعویض تشکیل می شود.

نکته ۱: فرمول از عملیات مرتب سازی که مقایسه براساس آن انجام می شود کلید نامبده می شود.

نکته ۲: الگوریتم های مرتب سازی انواع مختلفی دارند.

- (۱) پایدار (معادل) (stable): الگوریتمی که ترتیب عناصر با مقدار مساوی را بعد از مرتب سازی حفظ کند.
 (۲) ناپایدار (نامعادل) (non stable): الگوریتمی که ترتیب عناصر با مقدار مساوی را بعد از مرتب سازی حفظ نکند.



- (۱) درجا (inplace): استفاده از فضای کمکی ثابت برای مرتب سازی بدون وابستگی به تعداد عناصر.
 (۲) برون از جا (Out place): استفاده از فضای کمکی متغیر برای مرتب سازی که وابسته به تعداد عناصر آرایه است.

(۱) مرتب کننده داخلی (Internal): عناصر مرتب شونده همگی در حافظه اصلی بوده و مقایسه مهم ترین عمل در این الگوریتمها است و تعیین کننده زمان اجرا مقایسه است.

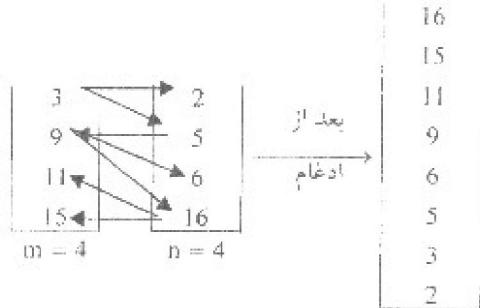
(۲) مرتب کننده خارجی (External): عناصر مرتب شونده همگی در حافظه اصلی نبوده و دسترسی به عناصر مهم ترین عمل بوده و تعیین کننده زمان اجرا است.

نکته ۳: الگوریتم های مرتب سازی معمولاً از ۲ حلقه تو در تو تشکیل می شوند.

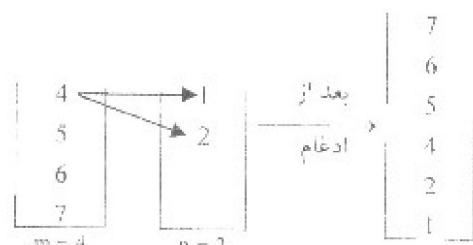
نکته ۴: اگر بخواهیم 2 آرایه مرتب شده را به فرم $m \times n$ و $n \times 1$ بعد از ادغام بازهم مرتب بینیم:

ب) حداقل مقایسه ۱

الف) حداقل مقایسه $\min(m, n)$



با ۷ مقایسه



با ۲ مقایسه

تکنیک زمان‌های مختلف الگوریتم‌های مرتب‌سازی به شرح زیر است:

	توضیحات	بهترین حالت	متوسط	
۱- سریع	نامتعادل و درجا	$O(n^2)$	$O(n \log n)$	(Quick sort)
۲- انتخابی	نامتعادل و درجا	$O(n^2)$	$O(n^2)$	(Selection sort)
۳- حبابی	متعادل و درجا	$O(n^2)$	$O(n^2)$	(Bubble sort)
۴- درجی	متعادل و درجا	$O(n^2)$	$O(n^2)$	(Insertion sort)
۵- Heap	نامتعادل و درجا	$O(n \log n)$	$O(n \log n)$	
۶- ادغامی	متعادل و غیر درجا	$O(n \log n)$	$O(n \log n)$	(Merge)
۷- درختی	غیر درجا	$O(n^2)$	$O(n \log n)$	(BST)

☞ همه الگوریتم‌های sort زمانی بین $O(n^2)$, $O(n \log n)$, $O(n)$ را اختیار می‌کنند.

چون عمل مهم در این الگوریتم‌ها که از نوع مرتب کننده داخلی هستند، مقایسه کردن است زمان‌های بالا با توجه به وضعیت مقایسه‌ای می‌باشد.

مثال: کدامیک از روش‌های مرتب‌سازی ذیل ترتیب اولیه رکوردهای هم کنید را حفظ می‌کنید؟ (کنکور ۸۳ آزاد)

Heap (۴) ✓ insertion (۴) Quick (۴) selection (۱)

مرتب‌سازی حبابی (Bubble sort) (متعادل و درجا)

در این مرتب‌سازی آرایه در چندین گذر (pass) عمل می‌کند، در هر گذر هر عنصر با عنصر بعدی مقایسه شده و در صورت نیاز با هم جایجا می‌شوند.

15 \longleftrightarrow 19 14 11

15 19 \longleftrightarrow 14 11

15 14 19 \longleftrightarrow 11

15 14 11 19

انتهای گذر اول بزرگترین عنصر در انتهای لیست

☞ در یک آرایه n ‌تایی به طور کلی در روش حبابی $1 - n$ گذر وجود خواهد داشت. و در هر گذر عدد مقایسه‌ها بکمتر از گذرهای قبلی است چون در هر گذر یک داده در مکان اصلی خودش قرار می‌گیرد.



(حداقل) بهترین (حداکثر)

$$\text{مقایسه} = \frac{n(n-1)}{2} = O(n^2) \quad \text{متوسط} = O(n^2) \quad \text{مقایسه} = O(n)$$

$$\text{جامیجانی} = \frac{n(n-1)}{2} = O(n^2) \quad \text{جامیجانی} = O(1) = 0$$

ساختمان دادهها

نکته متوسط تعداد جابجایی با مقایسه $\frac{n(n-1)}{2}$ است. (حداکثر مقایسه یا جابجایی)

$n - 1$ مقایسه (در گذر اول)

جابجایی

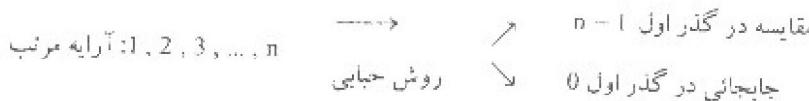
بهرین حالت = لیست مرتب باشد.

$\frac{n(n-1)}{2}$ مقایسه در $n - 1$ گذر

$\frac{n(n-1)}{2}$ جابجایی

بدترین حالت = لیست مرتب معکوس باشد.

پیشترین حالت: (لیست مرتب)



در صورتیکه از الگوریتم || استفاده کیم در انتهای گذر اول با مقایسه دو به دو در n داده ($n - 1$ مقایسه) در آرایه مرتب چون هیچ

$n - 1$ (مقایسه) و 0 (جابجایی)

داریم.

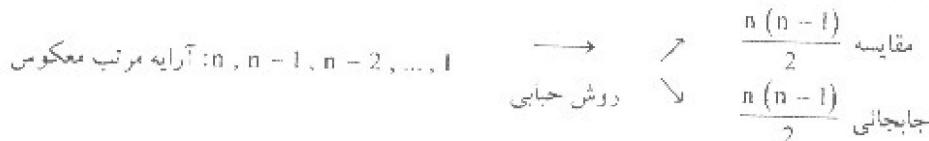
جابجایی انجام نمی شود. نیازی به رفتن به گذر بعدی نداریم در نتیجه:

در صورتیکه از الگوریتم استفاده کیم در انتهای گذر اول در آرایه مرتب با آن که هیچ جابجایی انجام نشده ولی چون مانند الگوریتم || متغیر flag

برای تشخیص وجود نثارد گذرهای بعدی نیز انجام می شود گرچه هیچ جابجایی نداریم. در نتیجه:

$\frac{n(n-1)}{2}$ (مقایسه) و 0 (جابجایی)

بدترین حالت: (لیست نامرتب - مرتب معکوس)



مثال:

آرایه: $n, n - 1, \dots, 2, 1$

مقایسه	جابجایی	
$n - 1$	$n - 1$	گذر اول: $n - 1, n - 2, \dots, 2, 1, n$
$n - 2$	$n - 2$	گذر دوم: $n - 2, n - 3, \dots, 2, 1, n - 1, n$
\vdots	\vdots	\vdots
1	1	گذر $n - 1$ م: $1, 2, \dots, n - 1, n$
$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	

الگوریتم

(II)

```

flag = true; i := 1;
while (i <= n) And (flag = true) do
begin
    flag := false;
    for j := 1 to n - i do
        if A[j] > A[j + 1] then
begin temp := A[j]; A[j] := A[j + 1]; A[j + 1] := temp; end;
    i := i + 1;
end;
    
```

(I)

```

for i := 1 to n - 1 do
    for j := 1 to n - i do
        if A[j] > A[j + 1] then
begin
    temp := A[j];
    A[j] := A[j + 1];
    A[j + 1] := temp;
end;
    
```

مرتب سازی درجی (insertion sort) (متداول و درجا)

در این مرتب سازی آرایه در چندین گذرا (pass) عمل می کند، در هر گذرا به ترتیب زیر عمل می کنیم:

گذرا 1: عنصر $A[1]$ به طور طبیعی مرتب است. (0 مقایسه)

گذرا 2: عنصر $A[2]$ با $A[1]$ مقایسه قبل با بعد از $A[1]$ قرار می گیرد. (1 مقایسه) (حداقل 0 جابجایی و حداکثر 1 جابجایی)

گذرا 3: عنصر $A[3]$ با مقایسه طوری بین $A[1]$ و $A[2]$ قرار می گیرد که آرایه مرتب باشد. (حداکثر 2 مقایسه و حداقل 1 مقایسه) (حداکثر 2 جابجایی و حداقل 0 جابجایی)

گذرا n : عنصر $A[n]$ در میان عناصر $A[n-1], \dots, A[1]$ طوری قرار می گیرد که آرایه مرتب باشد. (حداکثر $1 - n$ مقایسه و حداقل 1 مقایسه) (حداکثر $1 - n$ جابجایی و حداقل 0 جابجایی)



مثال: آرایه روبرو به صورت درجی مرتب شده است:

12, 14, 5, 9, 7

گذرا اول: 12

ورود 12

ورود 14

گذرا دوم: 12, 14

14 با 12 مقایسه

ورود 5

5 با 14 و 12 مقایسه

گذرا سوم: 5, 12, 14

ورود 7

5 با 12, 14 مقایسه

گذرا سوم: 5, 7, 12, 14

ورود 9

7 با 14, 12, 9 مقایسه

گذرا چهارم: 5, 7, 9, 12, 14

ساختمان دادهها



برای اینجا

بدترین (حداکثر)

متوسط

بهترین (حداقل)

$$\frac{n(n-1)}{2} = O(n^2)$$

$$O(n^2)$$

$$n-1 = O(n)$$

مقایسه

$$\frac{n(n-1)}{2} = O(n^2)$$

$$O(n^2)$$

$$0 = O(1)$$

جابجایی

کلیه حد اکثر مقایسه و جابجایی $\frac{n(n-1)}{2}$ است.

$$0 = O(1)$$

کلیه بهترین حالت = نیست کاملاً مرتب

$$O(n) = \text{مقایسه}$$

$$\left. \begin{array}{c} O(n^2) \text{ جابجایی} \\ O(n^2) \text{ مقایسه} \end{array} \right\} \frac{n(n-1)}{2}$$

بدترین حالت = نیست کاملاً مرتب معکوس

بهترین حالت: نیست کاملاً مرتب

1, 2, 3, ..., n

	جابجایی	مقایسه
1: گذراول	+ 0	+ 0
1, 2: گذر دوم	+ 0	+ 1
1, 2, 3: گذر سوم	+ 0	+ 1
1, 2, ..., n: گذر nام	+ 0	+ 1
	0	n - 1

بدترین حالت: نیست مرتب معکوس

n, n - 1, ..., 2, 1

	جابجایی	مقایسه
n: گذراول	0	0
n - 1, n: گذر دوم	+ 1	+ 1
n - 2, n - 1, n: گذر سوم	+ 2	+ 2
⋮	⋮	⋮
n - 1, 2, ..., n: گذر nام	n - 1	n - 1

n - 1 مقایسه و با یک جابجایی قبل از آن درج می شود.

n - 2 با n - 1 و n مقایسه و با 2 جابجایی قبل از آنها درج می شود.

با 2 تا n - 1 مقایسه و با 1 - n جابجایی قبل از آنها درج می شود.

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

این روش برای آرایه با تعداد عناصر کمتر با مساوی ۲۰ سریعترین روش است.

الگوریتم (II)

```

for j ← 2 to n do
    key ← A[j]
    Insert A[j] Into the sorted sequence A[1..j-1]
    i ← j - 1
    while i > 0 and A[i] > key
        do A[i+1] ← A[i]
    i = i - 1
    A[i+1] ← key

```

الگوریتم (III)

```

for i := 2 to n do
begin
    j := i;
    while A[j] < A[j-1] do
begin
    T := A[j-1];
    A[j-1] := A[j];
    A[j] := T;
    j := j - 1;
end;
end;

```

الگوریتم مرتب سازی انتخابی (Selection sort) (نامتعادل و درجا)

در این روش با شروع از بتدای آرایه در هر مرحله (گذر = pass) از مرتب سازی با مقایسه اولین داده با بقیه داده‌ها هنگام رسیدن به انتهای آرایه محل درست یک عنصر (بزرگترین یا کوچکترین) پیدا شده می‌پس در صورت نیاز با یک جابجایی آن را در محل درست خودش فرار می‌دهیم در این وضعیت برای یک آرایه n تانی به $1 - n$ گذر نیاز داریم چون در انتهای گذر $1 - n$ داده آخر نیز در مکان واقعی خودش قرار گرفته است.

10 15 20 7 14

انتهای گذر اول

انتهای گذر دوم

انتهای گذر سوم

انتهای گذر چهارم



نکته مهم:

۱- بهترین حالت و بدترین حالت برای این روش مفهومی ندارد چرا که اگر آرایه مرتب باشد الگوریتم به صورت زیر عمل می‌کند.

$$O(n^2) \quad \left\{ \begin{array}{l} \text{مقایسات: } \frac{n(n-1)}{2} \\ \text{جایگانی: 0} \end{array} \right\} \quad ۱- آرایه مرتب$$

$$O(n^2) \quad \left\{ \begin{array}{l} \text{مقایسات: } \frac{n(n-1)}{2} \\ \text{جایگانی: حداقل ۱} \end{array} \right\} \quad ۲- آرایه نامرتب$$

در هر دو وضعیت مقایسات باید انجام شود و در انتهای گذرا اول امکان تشخیص مرتب بودن آرایه وجود ندارد چون فقط مقایسات برای یک عنصر انجام می‌شود.

```

for i := n downto 1 do
    begin
        max := x[i];
        index := i;
    for j := 1 to i do
        if (x[j] > max) then
            begin
                max := x[j];
                index := j;
            end;
        x[index] := x[i];
        x[i] := max;
    end;

```

آرایه: 10, 12, 13, 1, 5
 انتهای گذرا اول: 10, 12, 5, 1, 13
 انتهای گذرا دوم: 10, 1, 5, 12, 13
 انتهای گذرا سوم: 5, 1, 10, 12, 13
 انتهای گذرا چهارم: 1, 5, 10, 12, 13
 انتهای sort

در گذرا اول: $x[1]$ در نظر گرفته شده و با مقایسه با $x[2]$ $x[n]$ در انتهای گذرا بزرگترین مقدار در x قرار گرفته و محل آن در $x[index]$ قرار می‌گیرد. سپس در صورت نیاز با یک جایگانی مقدار $x[index]$ (ماکریم) با مقدار محل واقعی آن $x[n]$ عرض می‌شود. در گذرا دوم: $x[1]$ در نظر گرفته شده و با مقایسه با $x[3]$ $x[n-1]$ در انتهای گذرا بزرگترین مقدار در داده‌های باقیمانده در x قرار گرفته و محل آن در $x[index]$ ذخیره می‌شود سپس در صورت نیاز با یک جایگانی مقدار $x[index]$ (ماکریم) با مقدار محل واقعی آن $x[n-1]$ عرض می‌شود.

در این الگوریتم در n گذرا یک آرایه تابعی مرتب می‌شود.

$$\frac{n(n-1)}{2}$$

تعداد مقایسات:

کمترین و بدترین حالت با هم فرقی ندارد.

بدترین حالت

متوسط

بهترین حالت

نکته:

$$\frac{n(n-1)}{2} \text{ جمع مقایسه}$$

در گذار اول 1 - n مقایسه

در گذار دوم 2 - n مقایسه

⋮

⋮

نکته مهم:

نحوه در انتهای هر گذار در صورتیکه آرایه نامرتب باشد، جداکر ۱ مقایسه را داریم اما در هر وضعیتی (مرتب بودن یا نبودن آرایه) مقایسات باید انجام شود.

مرتب سازی ادغام (Merge sort) (متعادل و غیردرجا)

این روش مرتب سازی عموماً بر روی عناصری که در فایل ها فرار دارند اجرا می شود. اگرچه می توان این روش را در مورد بردارها نیز به کار برد. در این روش فایل n عضوی به n قسمت به طول یک تقسیم و سپس هر دو قسمت مجاور به صورت مرتب شده با هم ادغام می شوند. در

این حالت $\frac{n}{2}$ فایل به طول ۲، بجای شده است. روند ادغام تا رسیدن به یک فایل به طول n ادامه می یابد. به عنوان مثال مرتب سازی زیر

در روش ادغام به صورت زیر است:

[17][12], [44][12], [23][58], [25][9]

بردار ویده:

[17, 21][12, 44] [23, 58][9, 25]

گذار اول:

[12, 17, 21, 44] [9, 23, 25, 58]

گذار دوم:

[9, 12, 17, 21, 23, 25, 44, 58]

گذار سوم:

در هر مرحله برای ادغام دو قسمت، عناصر اول از دو قسمت در نظر گرفته شده با یکدیگر مقایسه می شوند؛ مقدار کوچکتر در دنباله حاصل فرار می گیرد و عنصر بعدی از دنباله ای که یک عنصر آن استفاده شده جهت مقایسه در نظر گرفته می شود و این روند ادامه می یابد تا دو دنباله در هم ادغام شوند.

نکته: این الگوریتم همواره با بیجیدگی $O(n \log n)$ انجام می شود.

بیجیدگی اجرا در الگوریتم مرتب سازی ادغام.

بدترین حالت	حالت متوسط	بهترین حالت	بیجیدگی اجرا
$O(n \log n)$	$O(n \log n)$	$O(n \log n)$	بیجیدگی اجرا

(غیر درجا)

بهترین	متوسط	بدترین	وضعیت
$O(n \log n)$	$O(n \log n)$	$O(n^2)$	زمان

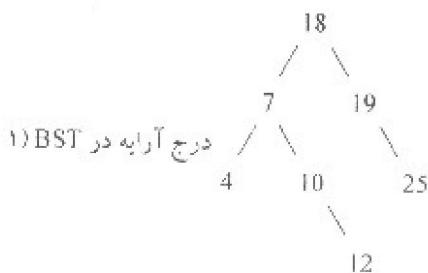
همانطور که در بحث درخت BST مطرح شد، یکی از کاربردهای مهم درخت BST مرتب سازی لیست ها می باشد برای این کار در دو مرحله عملیات انجام می شود.

- ۱- درج عنصر لیست در یک درخت BST نهی. (درج هر داده در حالت متوسط $O(\log n)$ در نتیجه درج n داده $O(n \log n)$ خواهد بود)
- ۲- پیمایش (LVR) inorder درخت BST که باعث نمایش لیست مرتب شده به صورت صعودی می باشد.

پیمایش RVL باعث نمایش لیست مرتب شده به صورت نزولی می باشد.

نکته: دقت می کنیم این روش مرتب سازی روی لیست هایی که داده تکراری دارند، ابتدا داده های تکراری را حذف کرده سپس در درخت نگهداری می کند. در این حالت متعادل بودن روش مرتب سازی مطرح نخواهد بود.

مثال: مرتب سازی آرایه $25, 19, 25, 12, 10, 12, 7, 18$ به روش BST خروجی عمل چیست؟



۱) BST (LVR) inorder: پیمایش $4, 7, 10, 12, 18, 19, 25$

۲) BST RVL: پیمایش $25, 19, 18, 12, 10, 7, 4$

بدترین وضعیت در مرتب سازی BST (Worst Case)

بدترین حالت ورودی برای مرتب سازی BST، مرتب بودن آرایه ورودی (صعودی یا نزولی) است که درخت به صورت اربی ظاهر شده در

این حالت در صورتیکه داده ها تکراری نباشند تعداد مقابسات برای درج $\frac{n(n-1)}{2}$ خواهد بود و زمان درج $O(n^2)$ خواهد بود.

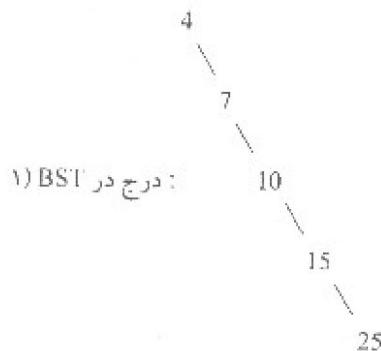
فرض کنید $a_1 < a_2 < a_3 < \dots < a_n$ در نتیجه اگر داده به صورت a_1, a_2, \dots, a_n وارد شوند.



مثال ۱:

آرایه ورودی:

4, 7, 10, 15, 25



مقایسات
 0 1 2 3 4 = 10

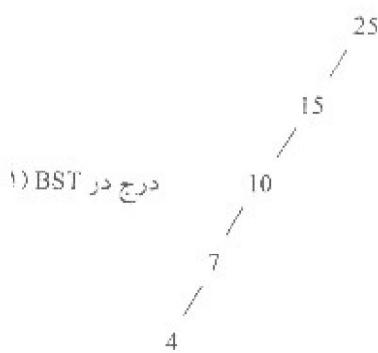
$$n = 5 \rightarrow \text{تعداد مقایسات} = \frac{n(n-1)}{2} = \frac{5 \times 4}{2} = 10$$

2) (LVR) inorder: پیمایش 4, 7, 10, 15, 25

مثال ۲:

آرایه ورودی:

25, 15, 10, 7, 4



مقایسات
 0 1 2 3 4 = 10

$$n = 5 \rightarrow \frac{n(n-1)}{2} = \frac{5 \times 4}{2} = 10$$

Y) (LVR) inorder : پیمایش 4, 7, 10, 15, 25

دیده می‌شود که برای یک آرایه n تابع در وضعیت بالا درخت با n سطح ایجاد شده است که برای درج n داده مقابله

اجام می‌شود در نتیجه زمان درج $O(n^2)$ خواهد بود.

بهترین حالت در مرتب‌سازی BST

بهترین حالت ورودی برای مرتب‌سازی BST، نامرتب بودن آرایه ورودی می‌باشد که باعث می‌شود عمق درخت BST در حالت مترازن

$O(n \log_2 n)$ فرار گرفته و در نتیجه زمان درج $O(n \log_2 n)$ و زمان مرتب‌سازی $O(n \log_2 n)$ خواهد بود.

مرتب‌سازی سریع (نامتعادل و درجا) Quick sort

الگوریتم مرتب‌سازی سریع از نوع الگوریتم‌های تعویضی بوده است که براساس جایجا کردن زیاد عناصر عمل می‌کند در این الگوریتم هر بار عنصری به عنوان pivot یا محور انتخاب می‌شود (بیش فرض داده اول آرایه) در این حالت بقیه داده‌های آرایه به گونه‌ای جایجا می‌شوند که کلیه عناصر کوچکتر از عنصر لولا در یک طرف آن و کلیه عناصر بزرگتر در طرف دیگر آن فرار گیرند. سپس دو آرایه ایجاد شده در دو طرف عنصر لولا (pivot) را به همین ترتیب دوباره مرتب می‌کنیم. (به صورت بازگشتی)

لولا

↓

$a_1 a_2 \dots a_n$

کوچکتر از a_1

بزرگتر از a_1

گذر اول

$\overbrace{a_k \dots a_s}$

(a_1)

$\overbrace{a_1 \dots a_m}$

↓

لولا

↓

لولا

:

روش انجام کار در یک مرحله:

۱- عنصر اول را به طور پیش فرض به عنوان لولا (pivot) در نظر می‌گیریم.

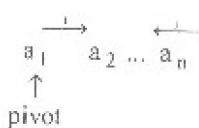
۲- اندیس از با شروع از حد بالا به سمت اولین داده‌ای در آرایه حرکت می‌کند که $A[i] > pivot$ باشد.

اندیس ز با شروع از حد بالا به سمت اولین داده‌ای در آرایه حرکت می‌کند که $A[i] < pivot$ باشد.

الف) $j < i$ در نتیجه $A[j]$ را با $A[i]$ جایجا کرده و به مرحله ۲ بر می‌گردیم.

ب) $j = i$ در نتیجه pivot را با $A[j]$ جایجا کرده گذر مورد نظر تمام شده است.

۳- بعد از مرحله ۲



الگوریتم مرتب سازی سریع به شکل زیر خواهد بود:

```

procedure Quicksort ( var x : arraylist; Left , Right : integer );
var
    i , j , pivot : integer;
begin
    if left < right then
        begin
            i := Left; j := Right + 1; pivot := x [left];
            repeat
                repeat
                    i := i + 1;
                until x [i] >= pivot;
                repeat
                    j := j - 1;
                until x [j] <= pivot;
                if i < j then swap (x [i] , x [j]);
            until i >= j;
            swap (x [Left] , x [j]);
            Quicksort (x , Left , j - 1);
            Quicksort (x , j + 1 , Right);
        end;
    end;

```

مثال: فرض کنیم بخواهیم روش Quicksort را در مورد آرایه زیر به کار ببریم. در اولین مرحله از کار شکل آرایه مطابق کدام گزینه خواهد

بود: (کنکور آزاد ۸۳)

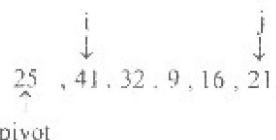
25 , 41 , 32 , 9 , 16 , 21

9 , 16 , 21 , 25 , 32 , 41 (t)

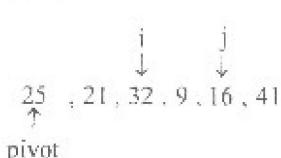
41 , 32 , 25 , 9 , 21 , 16 (t)

21 , 16 , 9 , 25 , 32 , 41 (f)

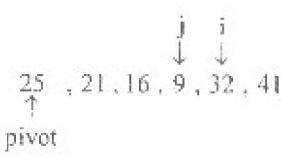
9 , 21 , 16 , 25 , 32 , 41 (t)



چون $j < i$ است $A[i] \leq A[j]$ جایجا می شود.



چون $j < i$ است $A[j] \leq A[i]$ جایجا می شود.



چون $j = i$ است $A[j] \leq A[pivot]$ و مرحله نول تمام می شود.

9 , 21 , 16 , 25 , 32 , 41

انتهای مرحله اول

گزینه ۳ صحیح می‌باشد.

(Quick sort)

بدترین وضعیت برای این الگوریتم زمانی است که داده به صورت مرتب یا مرتب معکوس وارد شوند چون هر بار که عنصر اول محور انتخاب

می‌شود، بقیه داده‌ها با سمت چپ و یا سمت راست آن قرار می‌گیرد.

در این وضعیت

$$1. \text{ تعداد مقایسات} \quad \frac{n(n-1)}{2}$$

$$2. \text{ مرتبه زمانی} \quad O(n^2)$$

مثال: لیست s که از $n = 6$ حرف تشکیل شده به صورت A, B, C, D, E, F می‌باشد مقایسه‌های لازم برای مرتب کردن s با الگوریتم

Quick sort عبارتست از: (کنکور ۸۳)

36 (۲)

(۱) لیست مرتب و مقایسه‌ای نداریم.

10 (۴)

✓ 15 (۳)

به توجه به مرتب بودن لیست بدترین وضعیت را داریم در نتیجه تعداد مقایسات

$$\frac{n(n-1)}{2} = \frac{6 \times 5}{2} = 15$$

(Quicksort)

بهترین وضعیت برای این الگوریتم زمانی است که داده به صورت نامرتب وارد شوند تا در هر بار در دو طرف عنصر نولا به صورت منتعال داده‌ها

توزیع شوند در این وضعیت زمان مرتب سازی $O(n \log n)$ خواهد بود که زمان وضعیت متوسط نیز خواهد بود.مثال: الگوریتم Quick sort یک رشته n تانی را با چه سرعی مرتب می‌نماید؟ (کنکور ۸۳ آزاد)

$$O(n) \quad (2)$$

$$\checkmark O(n \log n) \quad (1)$$

$$\log_2 n \quad (4)$$

$$O(n^2) \quad (3)$$

زمان هر الگوریتم مرتب سازی در حالت عادم بیان (بدترین یا بهترین حالت) مربوط به حالت متوسط آن می‌شود که در این مسئله نیز متنظر

زمان متوسط است که همان $O(n \log n)$ است.

مرتب سازی Heap (نامتعادل و درجا)

برای مرتب سازی یک آرایه n تانی به روش Heap به ترتیب باید ۲ عمل انجام شود.

(الف) درج تمام داده‌ها در یک درخت Heap نهی (maxHeap : minHeap) مرتب سازی صعودی)

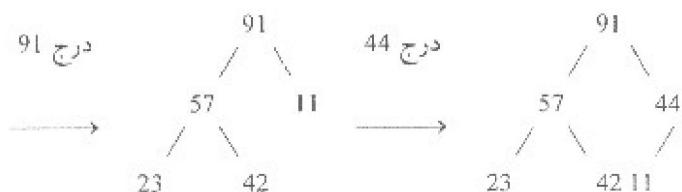
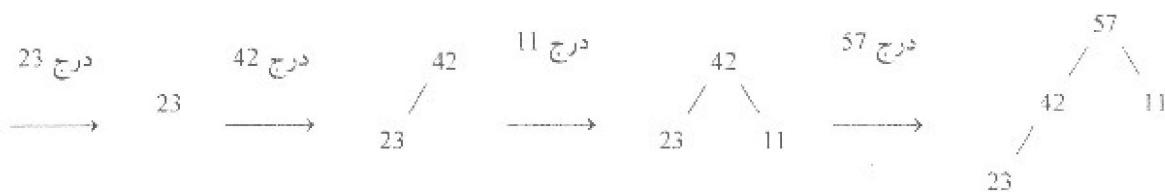
ب) حذف تمام داده‌ها از ریشه درخت Heap

■ زمان مرتب سازی Heap

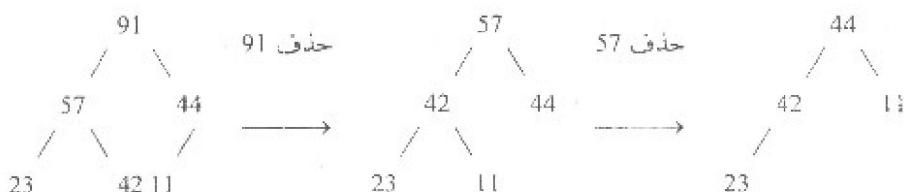
از آنجاییکه زمان درج یا حذف در Heap $O(n \log n)$ می باشد، در نتیجه برای مرتب سازی یک آرایه n تایی زمان مورد نظر $O(n \log n)$ خواهد بود.

مثال: مرتب سازی داده های $44, 11, 57, 91, 42, 23$ به سیولة درخت Heap

قسمت اول: درج داده ها در Heap



قسمت دوم: حذف داده ها از Heap



نهی

مجموعه سوالات کنکوری

پشته و صف

۱- یک پشته خانوی با اعداد از ۱ تا ۶ در ورودی داده شده است، اعمال زیر بر روی پشته قابل انجام هستند:

PUSH: کوچکترین عدد ورودی را بردشته و وارد پشته می کنیم.

POP: عنصر بالای پشته را در خروجی نوشته و سپس آن را حذف می کنیم.

کدامیک از گزینه های زیر را نمی توان با هیچ ترتیبی از اعمال فوق به دست آورد؟ (اعداد را از چپ به راست بخوانید) (کارشناسی ارشد دولتی ۷۴)

2 1 5 3 4 6 (۴)

4 3 2 1 6 5 (۳)

3 2 4 6 5 1 (۲)

1 2 3 5 6 4 (۱)

۲- کدام گزینه درست است؟

(۱) یک پشته و با صف را فقط به کمک خانه های مجاور هم در حافظه می توان پیاده سازی کرد.

(۲) یک روش مناسب برای پیاده سازی 2 پشته آن است که به سمت همدیگر رشد کنند.

(۳) یک پشته همواره به دو اشاره گر برای مشخص کردن بیند و انتهایش نیاز دارد.

(۴) یک صف را تنها به صورت حلقوی می توان پیاده سازی کرد.

۳- کاراکتر های رشته ABC به ترتیب جهت پردازش وارد پشته ای می شوند، کدام خروجی نمی تواند پدید بیابد؟

ACB (۴)

CAB (۳)

CBA (۲)

ABC (۱)

۴- برای محاسبه عبارت $(A + B)^*(A - B)$ ابتدا آن را به postfix تبدیل کرده و از پشته استفاده می کنیم. برای این کار پشته مورد نیاز حداقل چند خانه باید داشته باشد و چند بار عمل push در آن صورت می گیرد؟

(۴) ۵ خانه - ۱۰ بار

(۳) ۴ خانه - 7 بار

(۲) ۵ خانه - 7 بار

(۱) 4 خانه - 10 بار

۵- برای تعیین ترتیب پردازش برنامه های کامپیوتری دسته ای (hatch) که به مرکز محاسبات ارائه می شوند. کدام ساختار مناسب تر است؟ (مسابقات آموزشکده های فنی ۷۷)

(۴) آرایه

(۳) درخت

(۲) پشت

(۱) صف

۶- اگر رشته اعداد 1 , 2 , 3 , 4 , 5 را به ترتیب به یک stack وارد کنیم کدامیک از خروجی های زیر از این stack امکان پذیر خواهد بود (خروجی های پشته را از سمت چپ به راست بخوانید) (مسابقات آموزشکده های فنی - ۷۸) (کارشناسی ارشد - آزاد ۷۱)

5 - 1 - 3 - 2 - 4 (۴)

1 - 3 - 5 - 4 - 2 (۳)

5 - 4 - 3 - 1 - 2 (۲)

2 - 3 - 5 - 1 - 4 (۱)

۷- اگر رشته اعداد 1 , 2 , 3 , 4 , 5 , 7 را به ترتیب از سمت راست وارد یک stack کنیم کدامیک از خروجی های زیر از این stack امکان پذیر نیست. (خروجی ها را از سمت راست بخوانید) (کارشناسی ارشد - آزاد ۷۲)

5 7 3 4 1 (۴)

4 5 3 7 1 (۳)

4 5 7 3 1 (۲)

1 3 4 5 7 (۱)

۷۶ - push , pop کردن به ترتیب به چه معنایست؟ (مسابقات آموزشکده‌ها - ۷۶)

- ۱) برداشتن عنصر بالایی پشته - گذاشتن عنصر جدید روی پشته
 - ۲) خالی کردن پشته - پر کردن پشته
 - ۳) دین عنصر بالایی پشته - برداشتن عنصر بالایی پشته
 - ۴) گذاشتن عنصر در پشته - خواندن عنصر بالایی پشته
- ۹ - نیستی که در آن آخرین عنصر وارد، اولین عنصر حذف شده از آن می‌باشد چه نام دارد؟ (مسابقات آموزشکده‌ها - ۷۶) (مشابه مسابقات آموزشکده‌ها - ۸۰)

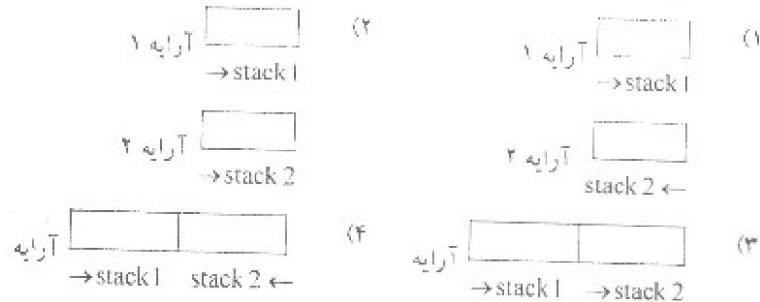
- (۱) پشته (۲) درخت دودونی (۳) صف (۴) لیست پیوندی حلقوی

۱۰ - اگر یک صف با آرایه نمایش داده شود، مشکل حرکت تدریجی صف در آرایه بیش می‌آید برای حل آن باید:

(مسابقات آموزشکده‌ها - ۷۶)

- ۱) از صف حلقوی استفاده می‌کنیم.
- ۲) از لیست پیوندی استفاده می‌کنیم.
- ۳) با تابعی پرشدن آرایه بررسی شود.
- ۴) عناصر داخل صف به ابتدای آرایه انتقال داده شوند.

۱۱ - کدام روش برای نمایش دو پشته در یک برنامه با استفاده از آرایه مناسب‌تر است؟ (مسابقات آموزشکده‌ها - ۷۶)



۱۲ - مناسب‌ترین ساختار داده جهت ثبت آدرس محل بازگشت در موقع فراخوانی زیر برنامه‌ها کدام است؟ (مسابقات آموزشکده‌ها - ۸۰)

- (Array) (Tree) (Drخت) (Stack) (Queue) (صف)

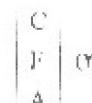
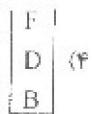
۱۳ - عبارت $\{x + (y - [a + b] * c - [(d + e)])\} / (j - (k - [L - n]))$ را در نظر بگیرید. می‌خواهیم با استفاده از یک پشته بررسی کنیم که آیا پرانتز، کروشه و آکولادها بدرستی تطبیق می‌شوند یا نه. پشته مورد استفاده حداقل بایستی گنجایش چند عنصر را داشته باشد؟ (مسابقات آموزشکده‌ها - ۷۹)

- 27 (۴) 18 (۴) 9 (۲) 4 (۱)

۱۴ - اگر داده ورودی به ترتیب از چپ به راست A , B , C , D , E , F باشد کدام پشته زیر می‌تواند یک حالت ایجاد شده باشد؟



۱۵ - اگر داده ورودی به ترتیب از چپ به راست A , B , C , D , E , F باشد کدام یعنی تواند یک حالت ایجاد شده باشد؟



۱۶ - در push کردن به یک stack

(۱) ابتدا Top یک واحد اضافه می شود و بعد داده وارد پشته می شود.

(۲) ابتدا Top یک واحد کم می شود و بعد داده وارد پشته می شود.

(۳) داده وارد می شود، بعد Top یک واحد اضافه می شود.

(۴) داده وارد می شود، بعد Top یک واحد کم می شود.

۱۷ - کدام عبارت صحیح است؟

(۱) در عمل POP عبارت 0 به معنای Stack Full است.

(۲) در عمل POP عبارت Top = n به معنای Stack Empty است.

(۳) در عمل PUSH عبارت Top = n به معنای Stack Empty است.

(۴) در عمل PUSH عبارت Top = n به معنای Stack Full است.

۱۸ - کدامیک از تعاریف ذیل برای صف (Queue) درست است؟ (مسابقات آموزشکده های فنی ۷۸)

(۱) یک لیست منظم که همه اضافه شدنی ها با حذف شدنی ها از یک طرف انجام می شود.

(۲) یک لیست منظم که همه اضافه شدنی ها با حذف از دو طرف انجام می شود.

(۳) یک لیست منظم که همه اضافه شدنی ها از انتهای لیست و همه حذف شدنی ها از ابتدای لیست صورت می گیرد.

(۴) یک لیست منظم که دارای سه اشاره گر است.

۱۹ - دو صفح در آرایه (n) Q ضروری ذخیره شده اند که سر صفح اولی در ابتداء و سر صفح دومی در انتهای Q قرار دارد، انتهای صفحات به ترتیب

در متغیرهای R1 ، R2 قرار دارند. زیر برنامه زیر چه عملی انجام می دهد؟

```
proc f(i, y);
if i=1 then R1 = R1+1 else R2 = R2-1;
if R1 = R2 then callfull;
case i of
1 : q(R1) = y;
2 : q(R2) = y;
end f;
```

(۲) عنصر لارا به صفح مشخص شده اضافه می کند.

(۱) از بروزرد دو صفح جلوگیری می کند.

(۴) هیچکدام

(۳) هردو

۲۰ - اگر يك صفت در يك آرایه به طول N به صورت حلقوی تعریف شده باشد و $Front$ ابتدای صفت و $Rear$ انتهای صفت را نشان دهد.

می توان گفت:

(۱) اگر $Front \leq Rear$ تعداد عناصر برابر 1 است.

(۲) اگر $Front < Rear$ تعداد عناصر برابر N است.

(۳) اگر $Front - Rear = 1$ نیست.

(۴) اگر $Front - Rear < Front$ تعداد عناصر برابر 0 است.

۲۱ - کدام گزینه در ساختار يك صفت حلقوی n عنصری، بيان گننده خالی و پر بودن صفت می باشد؟ (کارشناسی نابوسته - آزاد ۷۹)

(۱) خالی: $Front = 0$, $rear = n - 1$ و پر $Front = 0$, $Rear = 1$

(۲) خالی: $Front = 0$, $rear = n - 1$ و پر $Front = 0$, $Rear = 0$

(۳) خالی: $Front = n - 1$, $rear = n$ و پر $Front = n$, $Rear = n + 1$

(۴) خالی: $Front = 0$, $rear = n$ و پر $Front = 0$, $Rear = 0$

۲۲ - در ساختمان MAZE زمان اجرای مسیر طراحی شده (راهیابی) چگونه تعیین می شود؟ (کارشناسی نابوسته - آزاد ۷۹)

(۱) توسط اندازه مسیر پر پیچ و خم (۲) توسط طول کل مسیر

(۳) فقط توسط سرعت کامپیوتر (۴) توسط الگوریتم

۲۳ - معادل میانوندی عبارت پسوندی $- / * de - * abc$ چیست؟ (مسابقات آموزشکده ها - ۷۶)

$a * (b + c) - d / e$ (۴) $a - (b / c) * (d + e)$ (۵) $a + b * c / (d - e)$ (۶) $a - (b * d) / (c + e)$ (۷)

۲۴ - اگر $d = 10$, $c = 8$, $b = 4$, $a = 2$ باشد، ارزش عبارت پسوندی $ab * c - da - /$ چیست؟ (مسابقات آموزشکده ها - ۷۶)

2 (F) 1 (۳) -1 (۲) -2 (۱)

۲۵ - معادل postfix عبارت ریاضی infix $x + y / (u - t * w) * y$ (مسابقات آموزشکده ها - ۷۹)

$(w * u - y / x + y * ($ $utw * - xy / + y * ($ $xytw * - / y * - ($ $xytwy * - * / - ($

۲۶ - عبارت پسوندی (Postfix) معادل عبارت ریاضی $a / b - c + d * e - a * c / d$ کدام است؟ (با توجه به اولویت عملگرها) (کارشناسی نابوسته - آزاد ۷۹)

(۷۹) آزاد

$abc / - d + e * a - c * d / ($ $ab / c - d * e + a - c * d / ($ $ab / c - de * ac * - d / ($ $ab / c - de * + ac * d / - ($

۲۷ - عبارت پیشوندی (prefix) معادل عبارت ریاضی $a / b - c + d * e - a * c$ کدام است؟ (با توجه به اولویت عملگرها) (کارشناسی نابوسته

(۷۹) آزاد

$+ - / ab cde * - ac * ($ $- + - / abc * de * ac ($ $/ ab - cd + ea - c * ($ $+ - / abcd * e - - ac * ($

۲۸ - معادل پسوندی $(A - B / (C * D * E))$ کدام است؟ (مسابقات آموزشکده های فنی - ۸۰)

$AB / CDE ^ * - ($ $AB - CD * E ^ / ($ $ABCDE ^ * / - ($ $ABCD * E ^ - / ($

ساخته‌مان داده‌ها

۲۹ - مزایای عبارت پسوندی یا پیشوندی نسبت به عبارت میانوندی چیست؟

- (۱) اولویت‌ها مطرح نیستند.
- (۲) نیاز به پرانتزگذاری ندارد.
- (۳) گزینه ۱ و ۲
- (۴) هیچکدام

$$(A/B/C)^* D + E$$

۳۰ - معادل پسوندی (postfix) عبارت میانوندی (infix) زیر کدام است؟ (مسابقات آموزشکده‌های فنی - ۷۷)

$$ABC//DE^{*+} \quad (۱)$$

$$AB/C/DE^* \quad (۲)$$

$$AB/C^*DE^{*+} \quad (۳)$$

$$ABC//D^*E^+ \quad (۴)$$

۳۱ - معادل پیشوندی (prefix) عبارت میانوندی (infix) زیر چیست؟

$$(A+B)^*(C-D)$$

$$*+AB-CD \quad (۱)$$

$$*AB+-CD \quad (۲)$$

$$*+AB-DC \quad (۳)$$

$$*+ABCD- \quad (۴)$$

۳۲ - معادل پیشوندی عبارت پسوندی زیر چیست؟

$$AB/C^*D-EF/GH^{*+}$$

$$+-AB^{*^*}CD//EF+GH \quad (۱)$$

$$+-*^*ABCD//EF+GH \quad (۲)$$

$$+-*^*ABC^*//EF+GH \quad (۳)$$

$$+-*^*ABCDEF//+GH \quad (۴)$$

۳۳ - معادل پسوندی عبارت پیشوندی زیر چیست؟

$$-A/B^*C^*DE$$

$$A^*BCDE^{*/-} \quad (۱)$$

$$ABCDE^*^{*/-} \quad (۲)$$

$$ABCD^*E^{*/-} \quad (۳)$$

$$ABC^*DE^{*/-} \quad (۴)$$

۳۴ - معادل پسوندی عبارت پیشوندی زیر چیست؟

$$- - * + ABC - DE + FG$$

$$ABC + *DE - -FG^{*^*} \quad (۱)$$

$$AB+C*DE-F-G \quad (۲)$$

$$AB+C*DE - - F + ^* G \quad (۳)$$

$$AB+C*DE - - FG + \quad (۴)$$

۳۵ - در رابطه با پشته‌های چندگانه کدام گزینه نادرست است؟ (۱) [i] انتها و (۲) [i] ابتدای پشته آرا نشان می‌دهد.

(۱) شرط پر بودن پشته آام آن است که $[i] = b[i+1]$ باشد.

(۲) شرط خالی بودن پشته آام آن است که $b[i] = b[i-1]$ باشد.

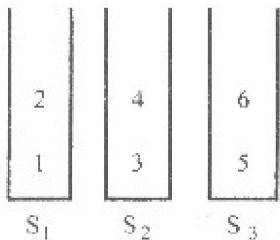
(۳) تعداد خانه‌های کل آرایه و n تعداد پشته‌ها

$$. b[i] = t[i+1] = \frac{m}{n} (i+1) - 1$$

(۴) ممکن است پشته‌ای پر شود در حالیکه پشته‌های دیگر خالی باشند.

۳۶ - سه پشته s1 و s2 و s3 هر یک حاوی دو عدد به شکل زیر می‌باشند:





دو عملگر (j) و $\text{pop}(i)$ به صورت زیر تعریف شده‌اند.

پک قلم از پسته s حذف و به پسته s اضافه می‌کند. $\text{pop}(i)$ یک قلم از پسته s حذف و سپس آن را چاپ می‌کند. برای جذب اعداد ۱ تا ۶ به صورت $\text{poppush}(i,j)$ با استفاده از چندبار مورد استفاده قرار گیرد؟ اولین عددی که چاپ می‌شود عدد ۱، دوین عدد، عدد ۳ و ... می‌باشد. (کارشناسی ارشد - آزاد - ۷۰)

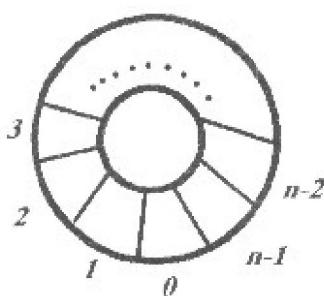
۴ بار

۶ بار

۵ بار

۳ بار

۳۷ - کدامیک از فرمول‌های زیر، N تعداد افلام در یک صفحه دایره‌ای را محاسبه می‌کند؟ متغیر F به خانه‌ای که بلا فاصله قبل از جلوی صفحه فرار دارد، (در جهت عکس عقربه‌های ساعت) اشاره می‌کند و متغیر R به عقب صفحه اشاره می‌نماید.



$$N = R - F \quad (1)$$

$$N = n - (R - F) \quad (2)$$

$$N = \begin{cases} n - (F - R) & \text{if } F > R \\ R - F & \text{if } R > F \end{cases} \quad (3)$$

$$N = \begin{cases} n - (R - F) & \text{if } F < R \\ R - F & \text{if } R > F \end{cases} \quad (4)$$

۳۸ - عبارت زیر را به polish وارونه تبدیل کنید. (کارشناسی ارشد - آزاد - ۷۷)

$$A^*(B - D)/E - F^*(G + H/K)$$

$$ABD - *E/F/GHK/-* - \quad (1)$$

$$A + B ^* DEF / GHK / - ^* - \quad (2)$$

$$- ^* + / KHG / FED ^* B + A \quad (3)$$

$$AB ^* + DEF / G + HK / ^* \quad (4)$$

۳۹ - عبارت postfix زیر مساوی است با: ۱۵ + * ۷ ۳ - / ۲ ۱ ۵ + * + ۷ ۱۲ (کارشناسی ارشد - آزاد - ۷۶)

۱۵ (۲)

۸ (۱)

۰ (۴) هیچکدام

۰ (۳)

۴۰ - عبارت infix زیر معادل با کدامیک از عبارات postfix است؟ (کارشناسی ارشد - آزاد - ۷۶)

$$((A + B)^* D) \uparrow (E - F)$$

ساختمان داده‌ها

ABD + *EF- ↑ (۲)

AB + D * EF- ↑ (۱)

۴) هیچکدام

+AB * D- ↑ EF (۳)

۴۱- می خویند عبارت میانو ندی (infix) زیر را به کمک پنجه به عبارت پسوندی (postfix) تبدیل کنیم:

برای این کار چندبار باره PUSH را برای گذاشتن اجزاء این عبارت در پنجه، فراخوانی کنیم؟ (مسابقات آموزشکده‌های فنی - ۷۷)

7 (۴)

9 (۳)

5 (۲)

15 (۰)

۴۲- پنجه (stack) ساختمان داده‌ای است از نوع: (مسابقات آموزشکده‌های فنی - ۷۸)

۴) هیچکدام

FIFO (۳)

FILO (۲)

IFOF (۱)

۴۳- برای پیاده‌سازی اعداد صحیح بسیار بزرگ (30 رقمی و یا بیشتر) کدام ساختار مناسب‌تر است؟ (به عنوان مثال: 2^x که آن عدد نسبتاً

بزرگی باشد). (مسابقات آموزشکده‌های فنی - ۸۰)

Linked List (۶)

Long integer (۳)

Queue (۲)

Stack (۱)

۴۴- کدامیک از اصطلاحات زیر یک صف یا queue را توصیف می‌نماید: (کارشناسی نایپوسته - آزاد - ۸۰)

۴) هیچکدام

FIFO (۳)

LIFO (۲)

IFOF (۱)

۴۵- کم‌هزینه‌ترین (از نظر تخصیص حافظه) برای این که عناصر یک پنجه (stack) را به پنجه دبگر ۶ بدون این که ترتیب عناصر تغییر

بابد، انتقال دهیم کدام است؟ (کارشناسی نایپوسته - آزاد - ۸۰)

۱) از طریق یک پنجه (stack) اضافی

۲) از طریق یک متغیر

۳) از طریق دو پنجه (stack) اضافی (۴) از طریق چند متغیر

۴۶- کم‌هزینه‌ترین (از نظر تخصیص حافظه) راه برای این که ترتیب عناصر یک stack را بر عکس کنیم کدام است؟ (کارشناسی نایپوسته -

آزاد - ۸۰)

۱) از طریق ۲ پنجه (stack) اضافی (۲) از طریق یک صف اضافی (queue)

۴) هیچکدام

۳) از طریق یک پنجه اضافی و چندین متغیر

۴۷- کدام مورد جزو کاربردهای پنجه نمی‌باشد؟ (کارشناسی نایپوسته - دولتشی - ۸۰)

۱) ارزشیابی عبارات ریاضی پسوندی (postfix)

۲) فراخوانی زیر برنامه‌ها در یک برنامه

۳) مدیریت درخواست‌های چاپ برای چاپگر

۴) مسئله مسیر بر پیج و خم (Maze)



۴۷ - معادل پسوندی (postfix) عبارت ریاضی $a + b \otimes c / (d - a \otimes (f + b) + b)$ کدام است؟ فرض کنید اولویت + و - بیکسان و کمتر از اولویت \otimes و اولویت \otimes کمتر از / باشد. (کارشناسی نایپوسته - دولتی ۸۰)

$$\text{abcdafb} + b \otimes - / \otimes + \quad (۲)$$

$$\text{abdafb} + b \otimes - + / \otimes - \quad (۱)$$

$$\text{abedafb} - \otimes - b + / \otimes + \quad (۴)$$

$$\text{abedafb} + + \otimes - b / \otimes - \quad (۳)$$

۴۸ - در تماش صفت حلقوی به کمک آرایه، چرا از یک خانه استفاده نمی شود؟ (کارشناسی نایپوسته - دولتی ۸۰)

- لذیس خانه مزبور صفر است.

(۲) به عنوان رزرو برای موقعی خاص نگه داشته می شود.

(۳) برای ارتباط خانه آخر با خانه اول آرایه باید از یک خانه استفاده کنیم.

(۴) در صورت استفاده، بروخانی بودن صفت با یکدیگر اشتباہ می شود.

۴۹ - معادل میانوندی (infix) عبارت پیشوندی (prefix) رو برو چیست؟ (کارشناسی نایپوسته - دولتی ۸۰)

$$/ * + abc-ab$$

$$a+b*c/a-b \quad (۴)$$

$$(a+b)*c/(a-b) \quad (۳)$$

$$(a+b)*c/a-b \quad (۲)$$

$$(a-b)*c/a-b \quad (۱)$$

۵۰ - عدد ۱ تا ۶ به ترتیب وارد stack می شوند، کدامیک از گزینه ها را نمی توان در خروجی تماش داد؟

$$2\ 1\ 5\ 3\ 6\ 4 \quad (۴)$$

$$4\ 3\ 2\ 1\ 6\ 5 \quad (۳)$$

$$3\ 2\ 4\ 6\ 5\ 1 \quad (۲)$$

$$1\ 2\ 3\ 5\ 6\ 4 \quad (۱)$$

۵۱ - اگر رشته اعداد ۱ ، ۳ ، ۴ ، ۵ ، ۷ را به ترتیب وارد stack کنیم کدام خروجی درست تحویل دارد؟

$$5\ 7\ 3\ 4\ 1 \quad (۴)$$

$$4\ 5\ 3\ 7\ 1 \quad (۳)$$

$$4\ 5\ 7\ 3\ 1 \quad (۲)$$

$$1\ 3\ 4\ 7\ 5 \quad (۱)$$

۵۲ - کدام گزینه در ساختار یک صفت حلقوی با $n=7$ بیان کننده خالی و پر بودن صفت می باشد؟

(۱) خالی: $R=7$ و $F=0$ و پر: $R=6$ و $F=0$

(۲) خالی: $R=2$ و $F=6$ و پر: $R=6$ و $F=0$

(۳) خالی: $R=0$ و $F=6$ و پر: $R=6$ و $F=7$

(۴) خالی: $R=0$ و $F=7$ و پر: $R=6$ و $F=0$

۵۳ - معادل پسوندی عبارت رو برو چیست؟

$$+ - * ^ ABCD // EF + GH$$

$$AB ^ CD * E - FG / H + / + \quad (۱) \quad A ^ B * CDEF - / G + H / + \quad (۰)$$

$$A ^ BCD * E - F / + GH / + \quad (۴) \quad AB ^ C * D - EF / GH + / + \quad (۰)$$

۵۴ - در کدام عملیات ابتدا خواندن و یا نوشتن صورت می گیرد و سپس اشاره گر تغییر می کند؟

Stack ji pop (۲)

Stack داخل Push (۰)

(۴) نوشتن در صفت حلقوی

(۳) خواندن از صفت حلقوی

۵۶- نتیجه عبارت postfix رو برو جست؟

$8 \ 2 / 5 - 7 \ 3 * +$

8.2 (۴)

20 (۳)

-28 (۲)

28.6 (۱)

۵۷- کدام گزینه معرف حذف کردن از صفح است؟

front := (front +1) mod n (۳)

front := front mod n (۱)

rear := (rear +1) mod n (۴)

rear := rear mod n (۳)

۵۸- اگر a=1 و b=2 و c=3 و d=4 باشد، آنگاه ارزش عبارت پسوندی $ab*c+dc/-$ کدام است؟

4 (۴)

5 (۳)

6 (۲)

7 (۱)

۵۹- یک پشته خانی و یک صفح با متغیرات 3, 2, 1 عدد ابتدای صفح است) داریم، پس از انجام سری دستورات زیر از جب به راست

متوازی صفح چه می شود؟

delq(A), push(A), delq(A), push(A), delq(A), push(A), pop(A), addq(A), pop(A), addq(A)

3, 2 (۴)

2, 1 (۳)

3, 2, 1 (۲)

3, 1, 2 (۱)

۶۰- اگر به جای پشته ای از دو پشته s1, s2 و عملیات push و pop آنها استفاده کنیم، کدامیک از پشته های زیر با داده های ورودی A, B،

C, D, E, F امکان پذیر است؟

(۲) فقط همان پشته هایی که به تهابی با s1 امکان پذیر است.

(۱) همه پشته های ممکن با A, B, C, D, E, F

A, B, C, D, E, F (۴) نصف همه پشته های ممکن با

s1 امکان پذیر است.

۶۱- اگر یک صفح دور با گنجایش n عنصر و اشاره گرهای head و tail داشته باشیم، حداقل چه تعداد عنصر می توان داخل این صفح قرار داد؟

n+2 (۴)

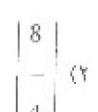
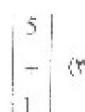
n+1 (۳)

n (۲)

✓ n-1 (۱)

۶۲- در هنگام محاسبه عبارت میانو ندی (3+5)-4 کدامیک از پشته های زیر غیر ممکن است؟

(۵) هر سه امکان پذیرند.



۶۳- سیتم تعداد متغیرهای مباني در محاسبه عبارت جبری postfix که به صورت $ab\mid cd^*\mid a\mid b$ است، برابر است با ... (کارشناسی ارشد - علوم

کامپیوتر) ۷۹

4 (۴)

3 (۳) ✓

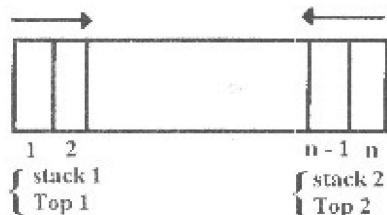
2 (۲)

1 (۱)



۶۴. دو یشه در یک آرایه به اندازه (1..n) پیاده‌سازی شده‌اند. اگر Top یکی به خانه خالی و Top دیگری به خانه پر اشاره کند، شرط پر بودن

بردار کدام گزینه می‌باشد؟



$$\text{Top1} = \text{Top2} = 1$$

$$\text{Top1} = \text{Top2} = n$$

$$\text{Top1} > \text{Top2}$$

هیچ کدام

۶۵. عبارت postfix معادل عبارت $(A + B)^* D + E / (F + A^* D)$ بروز نست با..... (کارشناسی ارشد - علوم کامپیوتر ۸۰)

$$AB + D * EF AD ^* + +$$

$$AB + D * E - F / A + D ^*$$

$$AB + DE + FAD ^* + /$$

$$ABDEFAD - ^* + / + ^*$$

۶۶. عبارت prefix زیر داده شده است: (کارشناسی ارشد - آزاد ۷۹)

$$+ - * \uparrow ABCD / B / F + GH$$

کدامیک از عبارات زیر معادل infix برای این عبارت است؟

$$A^B * (C - D) + \frac{EF}{G} + H \quad (A^B * (C - D) + E(F / (G + H)))$$

$$/ A^B * C - D + E / (F / (G + H)) \quad (A^B * C - D + E / F / G + H)$$

۶۷. جانچه بخواهیم k تا صفر را در یک بردار پیاده‌سازی کنیم برای کدامیک نز مقادیر k هزینه پیاده‌سازی شده همواره (1/Q) خواهد بود؟

$$k \leq \frac{n}{2}$$

$$k \leq n$$

$$k \leq 2$$

$$k=1$$

۶۸. عبارت پیشوندی (prefix) زیر داده شده است: (کارشناسی ارشد - دولتی ۷۷)

$$++a/b - cd/- ab - +c*d5/a - bc$$

معادل پسوندی (postfix) آن کدام است؟

$$ab + cd - / ab - cd - / 5 ^* + ab / c - - -$$

$$abcd - / + ab - cd5 ^* + abc - / - / +$$

$$abcd / - + abc - d5 ^* abc - + / - / +$$

$$ab + cd - ab - - / cd5 ^* + / abc - - /$$

۶۹. عبارت $ABC^*D/+$ در نماد لهستانی وارونه معادل کدام عبارت ریاضی زیر است؟

$$D^*C/A+B$$

$$C^*B/D+A$$

$$C^*D/B+A$$

$$A^*D+B/C$$

۷۰. عبارت prefix معادل عبارت میانوندی زیر کدام است؟

$$A^*B^{\wedge}(C + D^{\wedge}-E^*F)/G - H^*K$$

$$-*^{\wedge}AB C + ^{\wedge}DEF^*/G - ^*HK$$

$$-/*A^{\wedge}B + C^{\wedge}D - EFG^*HK$$

$$^{\wedge\wedge\wedge}ABC + -EF^*/G - HK^*$$

$$-/*^{\wedge}ABC + ^{\wedge}D^{\wedge}-EFG^*HK$$

۷۹- اگر اعداد ۱ ، ۲ ، ۳ به ترتیب فرار داشته باشند (۳ بالاترین) و عملیات (x) به معنای فرار گرفتن بالاترین عنصر پشته در متغیر x و عملیات (x) به معنای ذخیره x در پشته باشد پس از انجام عملیات زیر (از راست به چپ) وضعیت پشته چگونه است؟ (کارشناسی نایپوسته علمی کاربردی ۸۱)

$\text{Pop}(x)$ ، $\text{Push}(z)$ ، $\text{Push}(x)$ ، $\text{Push}(y)$ ، $\text{Pop}(z)$ ، $\text{Push}(x)$ ، $\text{Pop}(y)$ ، $\text{Pop}(x)$

۲ ، ۳ ، ۱ (۴)

۳ ، ۲ ، ۱ (۳)

۱ ، ۲ ، ۳ (۴)

۲ ، ۱ ، ۳ (۱)

۸۰- عناصر صفحه‌های Q_1 و Q_2 از چپ به راست به صورت زیر است (عنصر سمت چپ ابتدای صفحه است) : (کارشناسی ارشد ۷۹ دولتی)

$Q_1 = 10 , 25 , 17 , 41 , 19 , 26 , 75$

$Q_2 = 1 , 5 , 7 , 4 , 9 , 6$

اگر x لاعناصر صفحه باشند؛ پس از اجرای قطعه برنامه زیر :

$\text{make } \text{NULL} (Q_3)$

$i = 0$

$\text{While } (\text{not empty } (Q_1) \text{ and not empty } (Q_2))$

$i = i + 1$

$x = \text{DeleteQ} (Q_1)$

$y = \text{DeleteQ} (Q_2)$

$\text{if } (y = i) \text{ then Add Q } (Q_3 , x)$

end while

محتوی صفحه Q_3 برابر است با :

$Q_3 = 1 , 5 , 7$ (۴)

$Q_3 = 10 , 41 , 26$ (۴)

$Q_3 = 10 , 25 , 17$ (۴)

$Q_3 = 1 , 4 , 6$ (۱)

۸۱- معادل Postfix عبارت Prefix زیر کدام است؟ (مسابقات آموزشکده‌های فنی ۷۸)

$+ + a / b - cd / - ab - + c * de / a - bc$

$ab + cd - ab - + / cde * + / abc - / -$ (۴)

$ab + cd - / ab - cd - / e * + ab / c - - +$ (۱)

$abcd / - + abc - dc * abc - + / - +$ (۴)

$abcd - / - ab - cdc * + abc - - / - +$ (۳)

۸۲- عبارت میانولندی روی رو زای صورت نمادگذاری لهستانی تبدیل کنید؟ (مسابقات آموزشکده‌های فنی - ۸۲)

$(A+B) / (C-D)$

$()/+AB()-CD$ (۴)

$(AB+)/(CD-)$ (۴)

$AB+/CD-$ (۲)

$/+AB-CD$ (۱)

۸۳- هنگامی که به کمک آرایه یک صفحه حلقوی را نمایش می‌دهیم، چرا از یک خانه آرایه استفاده نمی‌شود؟ (کارشناسی نایپوسته -

دولتی ۸۲)

(۲) برای ارتباط خانه آخر با خانه اول

(۱) برای اشتباہ نشدن پروتکلی بودن صفحه

(۴) در زبان برنامه‌نویسی خانه اول آرایه صفر است.

(۳) به عنوان آدرس استفاده می‌شود

ساختمان دادهها

۸۴ - حاصل عبارت زیر چیست؟ (کارشناسی نایپوسته - دولتی ۸۲)

$$6, 2, 3, +, -, 3, 8, 2, /, +, *, 2, \uparrow, 3, +$$

52 (۴)

49 (۳)

25 (۲)

7 (۱)

۸۵ - شکل Postfix عبارت چیست؟ (کدام است؟) $(a + b \uparrow c \uparrow d) * (e + f / d)$ (کارشناسی نایپوسته - دولتی ۸۲)

$$abcd \uparrow \uparrow + efd + * (۲)$$

$$abc \uparrow d \uparrow + efd / + * (۱)$$

$$abcd \uparrow \uparrow + * e / fd + (۳)$$

$$abcd \uparrow \uparrow + e / fd + * (۳)$$

۸۶ - شکل Postfix عبارت زیر چیست؟ (کارشناسی نایپوسته - آزاد ۸۲)

$$A + (B * C - (D / E \uparrow f) * G) * H$$

$$ABCD * EF \uparrow / G * - H * - (۲)$$

$$ABC * DEF \uparrow / G * - H * + (۱)$$

$$ABC * DEF \uparrow / G * - H * + (۴)$$

$$ABCDEF \uparrow G * - H * + (۳)$$

۸۷ - فرض کنید به $N = 6$ stack خانه اختصاص داده ایم؛ در ابتدا Stack خالی است، خروجی برنامه زیر چیست؟ از چپ به راست (کارشناسی نایپوسته - آزاد ۸۲)

موسسه آزاد

۱/. Set A = 2, B = 5

۲/. Push(Stack, A)

Push(Stack, B+10)

Push(Stack, 3)

Push(Stack, A-B)

۳/. Repeat while Top <>0 POP (Stack, item)

write : item

۴/. Return

2,15,3,-3 (۱)

-3,3,15,2 (۱)

2,5,-3 (۴)

2,15,-3,3 (۳)

۸۸ - اگر رشته a, b, c, d, e را به ترتیب از سمت راست وارد Stack کنیم، کدامیک از خروجی های زیر از این Stack امکان پذیر نیست؟

خروچی ها از سمت راست خوانده می شود؟ (کارشناسی نایپوسته - آزاد ۸۲)

d,e,b,c,a (۴)

a,b,c,d,e (۳)

c,d,b,e,a (۲)

c,d,e,b,a (۱)

۸۹ - فرض کنید ساختار صفح را با دو پشتہ پیاده سازی کردی ایم. اگر صفح حاوی n عنصر باشد چند عمل Push برای درج یک عنصر به این

صف لازم است؟ (کارشناسی نایپوسته - علمی کاربردی ۸۲)

n + 1 (۴)

2n + 1 (۳)

2n (۲)

1 (۱)



۹۰ - درجه صورت دستور write در قطعه برنامه زیر اجرا می شود: فرض کنید n عدد خوانده شده متمایز باشد؟ Inqueue (روتین درج در صف)

روتین حذف از صف، Push در پشت و POP حذف از پشت است) (کارشناسی نایوسته - علمی کاربردی ۸۲)

for $i = 1$ To n Do

Begin

Read (x);

Push (x) ; Inqueue (x);

End;

for $i = 1$ To n Do

Begin

$A := POP$; $B := Outqueue$;

if ($A = B$) then write ('found');

End;

(۲) اعداد تزیینی وارد شوند.

(۱) اعداد صعودی وارد شوند.

(۴) اعداد یک دنباله متقارن را تشکیل دهند.

(۳) فرد باشد.

لیست پیوندی

۹۱ - در یک لیست پیوندی حلقوی درستی شرط start'.link = start' تماشگر کدام است؟

(۱) لیست هیچ عنصری ندارد و کاملاً خالی است.

(۲) به آخرین عنصر لیست رسیده ایم.

(۳) لیست فقط یک عنصر (مرابیت) دارد.

(۴) لیست پر شده است.

۹۲ - تابع زیر بر روی یک لیست یک طرفه چه می کند؟

function T(L: pointer);

Var x: pointer;

Begin

$T := 0$;

if $L \neq nil$ then Begin

$x := L$;

while $x \neq nil$ do Begin

$T := T + 1$;

$x := x^.link$;

end;

end;

end;

(۱) تعداد گره های لیست را بر می گرداند.

(۲) اشاره گر به گره آخر را بر می گرداند.

(۳) لیست را معکوس می کند.

(۴) محتويات داده گره ها را با هم جمع می کند.



۹۳ - تابع زیر چه عملی انجام می‌دهد؟ توضیح این که list نشان‌دهنده لیستی از اعداد بوده و منتظور از تابع Head، نابعی است که مقدار اولین عنصر در لیست را برمی‌گرداند و تابع tail لیستی حاوی همه عناصر لیست ورودی به استثنای اولین عنصر را برمی‌گرداند. مکان‌های لیست را از مکان ۱ در نظر بگیرید: (کارشناسی ارشد - دولتی ۷۶)

```
function what(L>List):integer;
begin
    if L=nil then
        what:=(0)
    else
        if Tail(L)≠nil then
            what:=(Head(L)+what(Tail(Tail(L))))
        else
            what:=Head(L)
end;
```

۱) تعداد عناصر لیست را برمی‌گرداند.

۲) مجموع عناصر در مکان‌های فرد لیست ورودی را برمی‌گرداند.

۳) تعداد عناصر در مکان‌های فرد لیست را برمی‌گرداند.

۴) مجموع عناصر لیست ورودی را برمی‌گرداند.

۹۴ - روال زیر چه عملی انجام می‌دهد؟ (کارشناسی ارشد دولتی ۷۶)

```
Procedure f(x,y:ptr; var z:ptr);
var p:ptr;
begin
    P:=x; Z:=x;
    while p↑.Link<>nil do
        p:=p↑.Link;
        p↑.Link:=y;
end;
```

۱) دولیست پیوندی را به هم وصل (Concat) می‌کند.

۲) دو لیست حلقوی (Circular) را به هم وصل می‌کند.

۳) دو لیست پیوندی که حداقل یکی تر آنها غیرنهی می‌باشد را به هم وصل می‌کند.

۴) دو لیست حلقوی غیرنهی را به هم وصل می‌کند.

۹۵ - اگر P گرهی از یک لیست دو پیوندی (Doubly Linked List) با پیوندهای llink و rlink باشد، کدام گزینه همواره صحیح است؟

$$P = P \uparrow \text{llink} \uparrow \text{rlink} \quad (1)$$

$$P = P \uparrow \text{rlink} \uparrow \text{llink} \quad (2)$$

۴) هر دو

✓ ۳) هیچکدام

۹۶ - اگر X آدرس شروع یک لیست پیوندی زنجیری باشد؟ روال زیر چه عملی انجام می‌دهد؟

Proc S(x)

```
Define P,Q,r:pointer;
p=x;q=nil;
while (p<>nil) do
    r=q; q=p;
    p=p↑ link;
    q↑ link = r;
end;
x=q;
end;
```

۱) لیست را به ترتیب صعودی مرتب می‌کند.

۲) آخرین گره لیست را بازگشت می‌دهد.

۳) لیست را倒 وارون می‌کند.

۴) لیست را به ترتیب نزولی مرتب می‌کند.

۹۷ - مزبت لیست پیوندی نسبت به آرایه چیست؟ (مسابقات آموزشکده‌های فنی - ۸۱)

(۱) مصرف حافظه کمتر

(۲) ساده‌تر بودن عملیات حذف و درج

(۳) سریعتر بودن عمل پیمایش

(۴) سریعتر بودن عمل جستجو

برای حذف کردن یک عنصر از یک لیست پیوندی یک طرفه n عضوی چند دستور انتساب (جاگزینی) و چند دستور جابجایی لازم (۹۸)

است؟ (مسابقات آموزشکده‌های فنی - ۷۶)

(۱) یک دستور انتساب و هیچ جابجایی

(۲) $\frac{n}{2}$ دستور انتساب و جابجایی

برای امکان وجود پیمایش یک لیست در دو جهت از کدام مورد استفاده می‌شود؟ (مسابقات آموزشکده‌های فنی - ۷۶)

(۱) لیست دور نک پیوندی

(۲) لیست دور با گره آغازین

(۳) دوزار با گره آغازین

(۴) لیست پیوندی دو گانه

۱۰۰ - برای حذف کردن یک عنصر از یک لیست پیوندی دو طرفه چند آدرس باید جایگزین شود؟ (مسابقات آموزشکده‌های فنی - ۷۶)

(کارشناسی ناپیوسته - دولتی - ۸۰)

۱ (۴)

۲ (۳)

۳ (۲)

۴ (۱)

۱۰۱ - برای حذف کردن یک عنصر از یک لیست یک طرفه چند آدرس باید جایگزین شود؟

۱ (۴)

۲ (۳)

۳ (۲)

۴ (۱)

۱۰۲ - برای اضافه کردن یک گره به یک لیست پیوندی دو طرفه چند جایگزینی لازم است؟ (کارشناسی ناپیوسته - علمی کاربردی - ۸۱)

۱ (۴)

۲ (۳)

۳ (۲)

۴ (۱)

۱۰۳ - برای حذف کردن یک عنصر از یک لیست پیوندی یک طرفه n عضوی، چند دستور پیمایش و چند دستور جایگزینی لازم است؟

(۱) یک دستور پیمایش و یک دستور جایگزینی

(۲) یک دستور پیمایش و n دستور جایگزینی

(۳) $\frac{n}{2}$ دستور پیمایش و ۱ دستور جایگزینی

برای تعریف صحیح نوع لیست پیوندی، از کدام دستورات می‌توان استفاده کرد؟ (کارشناسی ناپیوسته - آزاد - ۷۹)

`typedef struct node *list; typedef struct node{char ch;list link;};` (۱)

`typedef struct node list; struct node{char ch;list link;};` (۲)

`struct node *list struct node{char ch;list link;};` (۳)

`struct node{char ch;struct node link;};` (۴)

۱۰۵ - عبارت رو برو را در نظر بگیرید. صدق این عبارت کدام گزینه صحیح است؟ (کارشناسی نایبیوسته - آزاد ۷۹) $a \rightarrow b, c \rightarrow d$

- (۱) a ساختاری از نوع اشاره گر دارد.
- (۲) b و c هر دو از نوع اشاره گرند.
- (۳) c ساختاری از نوع اشاره گر دارد.
- (۴) از طریق دستور c.d می‌توان به عنصر a دسترسی داشت.

۱۰۶ - تابع زیر چه عملی انجام می‌دهد؟ (با فرض این که نوع list اشاره گر است) (کارشناسی نایبیوسته - آزاد ۷۹)

```
list x(list L)
{ list m,t;
  m=NULL;
  while(L) {
    t=m; m=L;
    L=L->link;
    m->link=t;
  }
  return m;
}
```

- (۱) محن دو عنصر در لیست را جایه‌جا می‌کند.
- (۲) لیست پیوندی L را معکوس می‌کند.
- (۳) عنصری را از لیست L جایه‌جا می‌کند.
- (۴) لیست L را مرور می‌کند.

۱۰۷ - مزیت لیست یک طرفه دور (چرخشی) نسبت به لیست غیردور یک طرقه چیست؟ (مسابقات آموزشکده‌های فنی - ۷۹)

- (۱) پیمایش لیست را سریعتر می‌کند.
- (۲) فیلد null در این لیست وجود ندارد.

(۳) بدون مصرف حافظه اضافی، حرکت به گره قبلی را میسر می‌سازد.

(۴) با مصرف حافظه‌ای به اندازه یک فیلد پیوند، حرکت به گره قبلی را امکان‌پذیر می‌کند.

۱۰۸ - اگر first اشاره گر به ابتدای یک لیست یک طرفه باشد. اثر اجرای قطعه دستورات زیر چیست؟ (مسابقات آموزشکده‌های فنی - ۷۹)

(مشابه کارشناسی نایبیوسته - دولتی ۸۰) (مشابه مسابقات آموزشکده‌های فنی ۷۷ و ۸۱)

```
p:=first;
q:=nil;
while p<>nil do
begin
  r:=q;
  q:=p;
  p:=p^.link;
  q^.link=r;
end;
first := q;
```

(۱) لیست را 80% می‌نماید.

(۲) لیست را معکوس می‌نماید.

(۳) لیست یک طرفه را به لیست دور تبدیل می‌کند.

(۴) لیست را با سه پوینتر پیمایش کرده و first را به گره آخر اشاره می‌دهد.

۱۰۹ - روال زیر یک گره با فیلد داده‌ای ۱۰ را به بعد از گره‌ای که آدرس آن در x می‌باشد درج می‌کند. اگر جای دستورات (۱) و (۲) عوض شود کدام گزینه رخ می‌دهد؟ (مسابقات آموزشکده‌های فنی - ۷۹) (توضیح: first به ابتدای یک لیست پیوندی اشاره می‌کند.)

```
procedure insert(var first: pointer; x: pointer);
var
  t: pointer;
begin
  new(t);
  t^.data:=10;
  if first=nil then begin
    first:=t;
    t^.link:=nil;
  end
  else begin
    (1)→ t^.link:=x^.link;
    (2)→ x^.link:=t;
  end
end;
```

(۱) $t = \text{nil}$ می‌شود.

(۲) $\text{first} = \text{nil}$ می‌شود.

(۳) خانه‌های که x به آن اشاره می‌کند گم می‌شود.

(۴) قسمتی از لیستی که first به آن اشاره می‌کند گم می‌شود.

۱۱۰ - زیر روال زیر چه عملی انجام می‌دهد؟ (مسابقات آموزشکده‌های فنی - ۷۹)

```
procedure f(x,y: pointer; var z: pointer);
var
  p: pointer;
begin
  p:=x;z:=x;
  while p^.link<>nil do
    p:=p^.link;
  p^.link:=y;
end;
```

(۱) دو لیست پیوندی y و x را به همین ترتیب به هم وصل می‌کند و لیست پیوندی z را بوجود می‌آورد.

✓ (۲) دو لیست پیوندی z و x را به همین ترتیب به هم وصل کرده و لیست پیوندی z را درست می‌کند.

(۳) دو لیست دایره‌ای X و Y را به همین ترتیب وصل می‌کند و لیست z را بوجود می‌آورد.

(۴) z را فقط معادل لیست پیوندی x در نظر می‌گیرد.

۱۱۱ - مهمترین مزیت لیست پیوندی نسبت به آرایه در تماش بک لیست چیست؟ (مسابقات آموزشکده‌های فنی - ۸۰ و ۷۹)

(۱) مصرف حافظه کمتر

(۲) جستجوی عنصری از لیست

(۳) پیمایش (Traverse) لیست

✓ (۴) سادگی عمل حذف و درج عنصر از لیست

۱۱۲ - مزت لیست پیوندی نسبت به آرایه در نمایش یک لیست کدام است؟ (کارشناسی ناپیوسته - دولتی ۸۰)

(۱) پیماش لیست

(۲) جستجوی عنصری از لیست

(۳) سادگی عمل حذف و درج عنصر از لیست

(۴) مصرف حافظه کمتر

۱۱۳ - کدام گریته جزو معابد روش پیوندی (لیستهای پیوندی) است؟ (کارشناسی ناپیوسته - دولتی ۸۰)

(۱) اتصاف حافظه ✓

(۲) بالا بردن سرعت دسترسی به گره‌ها

(۳) پیچیدگی نجام عمل حذف یا درج

(۴) سادگی دسترسی به گره‌ها

۱۱۴ - کدام یک از عملیات زیر در لیست دو طرفه، ساده‌تر از لیست یک طرفه نجام می‌شود؟ (کارشناسی ناپیوسته - دولتی ۸۰)

(۱) حذف گره با آدرس p از لیست ✓

(۲) پیماش لیست از 'بتد' تا 'انتها'

(۳) درج گره در انتهای لیست

(۴) درج گره در ابتدای لیست

۱۱۵ - روش جستجو در یک لیست پیوندی یک طرفه با ۱۰ گره که داده‌های آن به ترتیب صعودی مرتب شده‌اند چیست؟ (کارشناسی ناپیوسته - دولتی ۸۰) (مشابه کارشناسی ناپیوسته - آزاد ۸۱)

(۱) ترتیبی ✓

(۲) دودوئی

(۳) ترتیبی یا دودوئی

۱۱۶ - کار قطعه برنامه زیر چیست؟ (start به ابتدای لیست اشاره می‌نماید) (کارشناسی ناپیوسته - دولتی ۸۰)

`new(p)`

(۱) درج در تنهای لیست

`p^.data:=data;`

(۲) درج در صفت پیوندی

`P^.link:=start;`

(۳) درج در پشه پیوندی

`start:=p;`

(۴) درج در ابتدای یک لیست غیرنهایی

۱۱۷ - شرط پایان در حلقه Repeat until فلکه برنامه پاسکال زیر که به منظور پیماش یک لیست یک طرفه دور بدون گره آغازین

(نوشته شده است چیست؟ (کارشناسی ناپیوسته - دولتی ۸۰) Headernode)

`p:=start;`

P <> start (۱)

`if p <> nil then`

✓ p:=start (۲)

`Repeat`

p^.link:=start (۳)

`writeln(p^.data);`

p^.link <> start (۴)

`p:=p^.link;`

`until` شرط

۱۱۸ - زیربرنامه ۸ بر روی یک لیست پیوندی یک طرفه کدام عمل را انجام می‌دهد؟ (کارشناسی نایپرسه - دولتی - ۸۰)

```

Procedure g(Var start:Nodeptr);
Var
    p,q,r:Nodeptr;
Begin
    p:=start;
    q:=nil;
    while p <> nil Do
        Begin
            r:=q;
            q:=p;
            p:=p^.link;
            q^.link:=r;
        end;
        start:=q;
    End;
    
```

(۱) پیماش لیست
 (۲) حذف کردن لیست
 (۳) معکوس کردن لیست
 (۴) مرتب کردن لیست

۱۱۹ - کدامیک از زیر روال‌ها عنصر جدید را بعد از عنصر p درج می‌کند؟ (مسابقات آموزشکده‌های فنی - ۷۶)

```

Procedure insert list(Var x,p,head:pointer );
Var
    y:pointer ;
Begin
    y:=head;
    while y <>p do
        y:=y^.link; x^.link:=y^.link; y^.link:=x;
    end;
    
```

```

Procedure insert list(Var x,p,head:pointer );
Var
    y,z:pointer ;
Begin
    y:=head; z:=head;
    while y <>p do begin
        z:=y; y:=y^.link
    end;
    x^.link:=z^.link; z^.link:=x;
    end;
    
```

Procedure insert list(Var x,p:pointer);

```

Begin
    p^.link:=x;
    x^.link:=p^.link;
end;
    
```

Procedure insert list(x,p:pointer);

```

Begin
    x^.link:=p^.link;
    p^.link:=x;
end;
    
```

۱۲۰ - کدام گزینه درباره لیست پیوندی دوطرفه درست می‌باشد؟

(۱) حرکت از ابتداء انتها و بالعکس به مادگی امکان‌پذیر است.

(۲) برای حذف گرهی که آدرس آن را داریم نیاز به پیماش لیست وجود ندارد.

(۳) درج پس از گره دلخواه x در آن سریعتر از لیست یک طرفه است.

ساختهای داده‌ها

۱۲۱ - یک لیست خطی یک طرفه با دو اشاره‌گر F و R که به ترتیب به عنصر اول و آخر لیست اشاره می‌کنند پیاده‌سازی شده است. هرینه کدامیک از اعمال زیر را بسته به تعداد عناصر لیست است؟ (کارشناسی آزاد - دولتی ۸۰)

۱) حذف اولین عنصر

۲) حذف آخرین عنصر

۳) درج یک عنصر در انتهای لیست

۴) درج یک عنصر در ابتدای لیست

۱۲۲ - کدامیک از گزینه‌های زیر لیست دو طرفه را از حالت A به حالت B تغییر می‌دهد؟ (کارشناسی نایپوسه آزاد ۸۰)

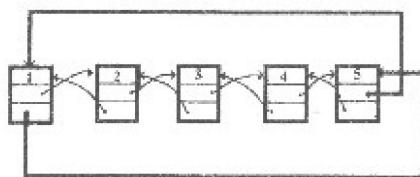
لیست A

`list → next → next → next = list → prev` (۱)

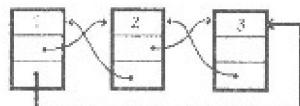
`list → next → next → next → next = list → next` (۲)

`list → prev = list → next → next` (۳)

`list → next → prev → next = list → next → next` (۴)



لیست B



۱۲۳ - در لیست پیوندی حلقوی اغلب به جای ذخیره آدرس ابتدای لیست آدرس انتهای لیست ذخیره می‌شود. مزیت این کار در چیست؟

۱) زمان حذف یک گره از اول لیست از مرتبه (n⁰) می‌شود.

۲) زمان درج یک گره قبل از گره ابتدایی ساده می‌شود.

۳) زمان اضافه کردن و یا حذف یک گره از اول لیست از مرتبه (n⁰) می‌شود.

۴) هر سه گزینه

۱۲۴ - در یک صیف که به صورت یک لیست پیوندی ساخته شده است حذف یک عنصر مطابق با کدام انتخاب زیر انجام می‌گردد؟

(کارشناسی نایپوسه - آزاد ۸۰)

۱) حذف عنصر از انتهای لیست انجام گرفته و ارزش ذخیره شده در آن باز گردانده می‌گردد.

۲) حذف عنصر از انتهای لیست انجام گرفته و ارزش باز گردانده نمی‌شود.

۳) حذف عنصر از ابتدای لیست انجام گرفته و هیچ ارزشی نیز باز گردانده نمی‌شود.

۴) هیچ یک از حالات بالا درست نیست.

۱۲۵ - یک لیست پیوندی را چگونه می‌توانیم بیمایش کنیم؟ (کارشناسی نایپوسته - آزاد ۸۰)

۱) از انتهای ابتدای لیست.

۲) از طریق متغیر Tail در ساختهای داده‌های لیست پیوندی.

۳) از طریق متغیر Head و از ابتدای لیست به انتهای آن.

۴) همه حالات گفته شده در بالا.

۱۲۶ - در کدامیک از انتخاب‌های زیر یک لیست پیوندی خالی است؟ (کارشناسی نایپوسته - آزاد ۸۰)

۱) متغیرهای Head، Tail برابر باشند و مقدار آن‌ها چیزی به غیر NULL باشد.

۲) Head مقدار NULL داشته باشد.

۳) Tail مقدار NULL داشته باشد.

۴) متغیرهای Head و Tail برابر باشند و مقدار آن‌ها NULL باشد.

۱۲۷ - در لیست پیوندی دوطرفه (double link list) امکان بیمایش لیست چگونه است؟ (کارشناسی نایپوسته - آزاد ۸۰)

۱) از طرف Head به طرف Tail

۲) از طرف Tail به طرف Head

۳) هیچ‌کدام

۴) انتخاب ۱ و ۲

۱۲۸ - برای اضافه نمودن یک گره (node) در یک لیست پیوندی در ابتدای لیست در کدامیک از انتخاب‌های زیر ارزش متغیر Head به کار برده می‌شود؟ (کارشناسی نایپوسته - آزاد ۸۰)

۱) در ارزش‌دهی متغیر Tail

۲) در ارزش‌دهی قسمت اشاره گر در گره جدید

۳) در ارزش‌دهی استفاده تمی گردد.

۴) در ارزش‌دهی آخرین گره (node) در لیست پیوندی

۱۲۹ - کدامیک از کدهای زیر دو گره بعد از گره x را از یک لیست پیوندی خطا حذف می‌کند؟

$x := \text{link}(\text{link}(x));$ (۱)

$\text{link}(x) := \text{link}(\text{link}(x));$ (۱)

$\text{link}(x) := \text{link}(\text{link}(\text{link}(x)));$ (۴)

$\text{link}(\text{link}(x)) := x;$ (۴)

۱۳۰ - صف اولویت دار (Priority Queue) شامل کدامیک از اعمال زیر می‌شود؟

۱) جستجو، درج، حذف کوچکترین عضو

۲) درج، حذف

۳) جستجو، درج، حذف

۴) درج، حذف

ساختمان دادهها

۱۳۱ - تابع head در برنامه زیر مصرف برگرداندن اولین عنصر لیست و تابع tail مصرف ماقبی list به غیر از اولین عنصر را می‌رساند و همچنین تابع (x, y) به معنی ترکیب دو لیست x و y به صورت Cons(x, y) باشد زیرا برنامه زیر چه عملی انجام می‌دهد؟

```

Fun (s:list)
begin
  if (S=nil) then return(nil);
  else return(cons(fun(tail(s)),head(s)));
end.

```

- ۱) معکوس سازی ترتیب عناصر list
- ۲) جابجایی عنصر اول و آخر لیست
- ۳) عمل خاصی انجام نمی‌دهد.
- ۴) حذف عناصر اول و آخر لیست

۱۳۲ - بردازه آنچه عملی روی لیست یک طرفه انجام می‌دهد؟ (کارشناسی نایپوسته - دولتشی ۸۱)

```

procedure f(Var start:Nodeptr);
Begin
  if start<>nil then
    if start^.link = nil then
      begin
        Dispose(start);
        start:=nil;
      End
    Else
      f(start^.link)
  End;

```

- ۱) حذف اولین گره
- ۲) حذف دومین گره
- ۳) حذف آخرین گره
- ۴) حذف گره ماقبل

۱۳۳ - کدام گزینه صحیح است؟ (کارشناسی نایپوسته - دولتشی ۸۱)

- ۱) اتلاف حافظه و ایستا بودن لیست‌های پیوندی از معایب آن محسوب می‌شود.
- ۲) اتلاف حافظه در آرایه به مرتب از لیست‌های پیوندی بیشتر است.
- ۳) دسترسی به عناصر لیست پیوندی نسبت به آرایه سریع‌تر انجام می‌شود.
- ۴) زمان دسترسی به عنصر اول و عنصر آخر آرایه با پکدیگر یکسان است.

۱۳۴ - کدام گزینه از مزایای صفت پیوندی نسبت به صفت ایجاد شده در آرایه است؟ (کارشناسی نایپوسته - آزاد ۸۱)

- ۱) اختصاص حافظه به صورت پویا و مناسب با داده‌ها
- ۲) سرعت زیاد در دسترسی به داده سرافصل
- ۳) امکان دسترسی به داده سرو و انتهای صفت
- ۴) جلوگیری از اتلاف حافظه

۱۳۵ - مزیت نیست یک طرفه دوار نسبت به لیست یک طرفه غیردوار کدام است؟ (کارشناسی نایپوسته - آزاد ۸۱)

- ۱) اتلاف حافظه کمتری دارد.
- ۲) سرعت دسترسی به گره وسط بیشتر است.
- ۳) امکان دسترسی به گره قبل وجود دارد.
- ۴) مزیتی ندارد.



۱۳۶ - کدام گزینه نادرست است؟ (کارشناسی نایپوسته - آزاد ۸۱)

۱) اندلaf حافظه در لیست دو طرفه بیشتر از لیست یک طرفه است.

۲) لیستی که درج و حذف در بتدای آن صورت می‌گیرد پشته است.

۳) لیستی که درج از یک طرف و حذف از طرف دیگر آن صورت می‌گیرد صرف است.

۴) پیماش لیست دو طرفه سریعتر از لیست یک طرفه انجام می‌شود. ✓

۱۳۷ - یک لیست پیوندی یک طرفه در آرایه‌های موازی start و link پیاده‌سازی شده است و به شروع لیست اشاره می‌کند. قطعه برنامه

زیر چه کاری روی لیست انجام می‌دهد؟ (کارشناسی نایپوسته علمی کاربردی ۸۱)

۱) داده‌های هر گره را با گره بعدی عوض می‌کند به جز گره آخر که تعویض نمی‌کند.

۲) به داده‌های هر گره یک واحد اضافه می‌کند.

۳) به داده‌های هر گره به جز گره آخر یک واحد اضافه می‌کند.

۴) داده‌های هر گره را با گره ماقبل عوض می‌کند.

۱۳۸ - در لیست پیوندی یک طرفه کدامیک از الگوریتم‌های زیر را نمی‌توان مورد استفاده قرار داد؟ (کارشناسی نایپوسته - علمی کاربردی ۸۱)

۱) الگوریتم بازگشتنی

۲) جستجوی خطي

۳) جستجوی دودویی

۴) مرتب‌سازی حبابی

درخت

۱۳۹ - در یک درخت دودوئی کامل با n گره، برای هر گره با اندیس i داریم: (آزاد ۷۹)

(۱) اگر $i < 1$ باشد، آریشه است و پدری نخواهد داشت.

(۲) اگر $n > i > 1$ باشد آن گاه آفرزند راست ندارد.

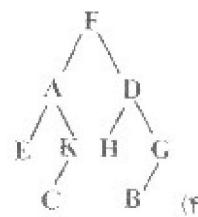
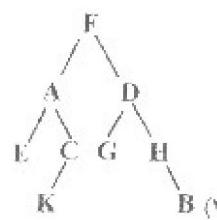
(۳) اگر $1 < i < n$ باشد آن گاه پدر ادر $[i/2]$ است.

(۴) اگر $n > i > 1$ باشد آن گاه آفرزند چپ ندارد.

۱۴۰ - درخت دودوئی که پیماشنهای LVR و VLR آن به صورت زیر است گدام است؟ (دولتی ۸۰)

LVR : E, A, C, K, F, H, D, B, G

VLR : F, A, E, K, C, D, H, G, B



۱۴۱ - کار تابع H به روی یک درخت دودوئی جیست؟ (دولتی ۸۰)

```

procedure h (root: Treeptr; var i: integer);
var
  x, y: Integer;
begin
  if root = nil then
    i := 0
  else begin
    h (root^.Left, x);
    h (root^.Right, y);
    if x > y then
      i := x + 1
    else
      i := y + 1
  end
end;
  
```

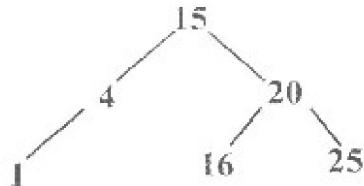
(۱) شمارش تعداد گرههای درخت

(۲) شمارش تعداد گرههای سطح آخر

(۳) شمارش تعداد سطوح درخت

(۴) شمارش تعداد شاخههای درخت

۱۴۲ - کدامیک از انتخاب‌های زیر درخت روبرو را به صورت Inorder پیمایش نموده است؟ (آزاد ۸۰)



۱, ۴, ۱۵, ۱۶, ۲۰, ۲۵ (۱)

۲۵, ۲۰, ۱۶, ۱۵, ۴, ۱ (۲)

۱۵, ۱, ۴, ۲۰, ۱۶, ۲۵ (۳)

۴, ۱, ۱۵, ۱۶, ۲۰, ۲۵ (۴)

۱۴۳ - چند روش پیمایش در درخت‌های دودوئی به صورت depth-first وجود دارد؟ (آزاد ۹۰)

۱۶ (۴)

۶ (۳)

۸ (۲)

۴ (۱)

۱۴۴ - در روش Preorder کدامیک از انتخاب‌های درست است؟ (آزاد ۸۰)

۱) گره را ویزیت کنیم بعد زیر درخت راست را پیمایش کنیم سپس زیر درخت چپ را پیمایش کنیم.

۲) گره را ویزیت کنیم بعد زیر درخت چپ را پیمایش کنیم سپس زیر درخت راست را پیمایش کنیم.

۳) زیر درخت چپ را پیمایش کنیم بعد زیر درخت راست را پیمایش کنیم سپس گره را ویزیت کنیم.

۴) زیر درخت راست را پیمایش کنیم بعد زیر درخت چپ را پیمایش کنیم سپس گره را ویزیت کنیم.

۱۴۵ - عمق بک درخت کامل با ۱۰۰۰ گره کدام نست؟ (دولتی ۸۱)

1000 (۴)

11 (۳)

10 (۲)

9 (۱)

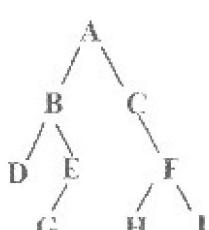
۱۴۶ - پیمایش RLV درخت روبرو کدام است؟ (دولتی ۸۱)

A,B,D,E,G,C,F,H,I (۱)

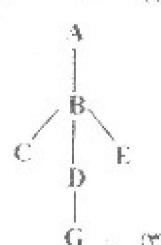
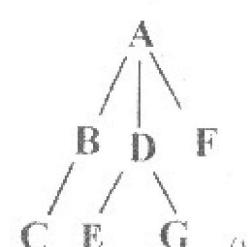
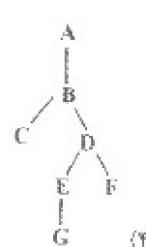
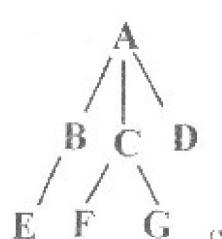
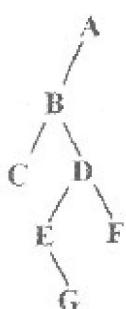
A,C,B,F,E,D,I,H,G (۲)

I,H,F,C,E,G,B,D,A (۳)

I,H,F,C,G,E,D,B,A (۴)



۱۴۷ - کدام درخت با استفاده از الگوریتم فرزند چپ - برادر راست به درخت دودوئی شکل روبرو تبدیل می‌شود؟ (دولتی ۸۱)



۱۴۸ - درخت دودوئی بر کدام است؟ (دولتی ۸۱)

(۱) درخت دودوئی به عمق h که دارای 2^{h-1} گره است.(۲) درخت دودوئی به عمق h که دارای 2^{h-1} گره است.

(۳) درخت دودوئی که حداقل گره‌های سطح آخر آن پر باشد.

(۴) درخت دودوئی که فقط گره‌های سطح آخر آن پر باشد.

۱۴۹ - در کدام پیماش می‌توان با استفاده از دستور Dispose به جای دستور Writeln این دستور را حذف کرد؟ (دولتی ۸۱)

(A) LRV (۱)

LVR (۲)

VLR (۳)

(۴) هر سه گزینه

۱۵۰ - یک درخت دودوئی در سه آرایه موازی به نام‌های Right , left , info پیاده‌سازی شده بطوری که info به اطلاعات مرگره و left

شاره گر به فرزندان راست و چپ آن می‌باشد و Root به ریشه درخت اشاره می‌کند. زیربرنامه بازگشتنی زیر جه کاری روی درخت

فوق انجام می‌دهد؟ (علمی کاربردی ۸۱)

A (left , Right , Root , s)

if Root = NULL then S := 0 & Return

call A (left, Right , left [Root], s1)

call A (left , Right , Right [Root] , s2)

S := S1 + S2 ||

Return

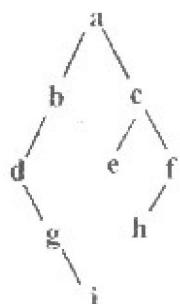
(۱) تعداد برگ‌های درخت را پیدا می‌کند.

(۲) عمق درخت را پیدا می‌کند.

(۳) سطح درخت را پیدا می‌کند.

(۴) تعداد گره‌های درخت را پیدا می‌کند.

۱۵۱ - کدام گزینه پیماش Postorder درخت روبروست؟ (علمی کاربردی ۸۱)



dgibehfca (۱)

igdbbehfca (۲)

dgibaechf (۳)

abdgjicefh (۴)



۱۵۲ - روش پیمایش درخت دودوئی (Binary tree) زیر را که به صورت بازگشتی تعریف می‌شود درنظر بگیرید:

(۱) ریشه را ملاقات (Visit) می‌کنیم.

(۲) درخت فرعی سمت راست را پیمایش می‌کنیم.

(۳) درخت فرعی سمت چپ را پیمایش می‌کنیم.

بین گره‌های ملاقات شده توسط روش فوق الذکر و postorder , Inorder , preorder چه رابطه‌ای وجود دارد؟ (۷۶)

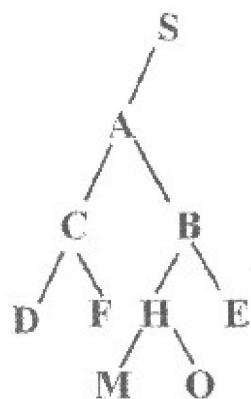
(۱) گره‌های ملاقات شده معکوس گره‌های ملاقات شده توسط preorder است.

(۲) گره‌های ملاقات شده معکوس گره‌های ملاقات شده توسط Inorder است.

(۳) گره‌های ملاقات شده معکوس گره‌های ملاقات شده توسط postorder است.

(۴) هیچ ربطی بین گره‌های ملاقات شده و گره‌های ملاقات شده توسط روشن‌های کلاسیک فوق نیست.

۱۵۳ - نتیجه پیمایش درخت زیر به روش LNR چیست؟ (۸۰ و ۷۹)



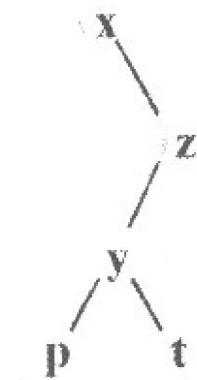
DCFAMHOBES (۰)

DECMAHOBES (۱)

SACDFBHMOE (۲)

SABELHOMCFD (۳)

۱۵۴ - پیاده‌سازی درخت زیر با آرایه به کدام صورت صحیح است؟ (۷۹)



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x		z			y						p	t		

(۱)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x	z	y	p	t										

(۲)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		1	p					y			x		z	

(۳)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x		z	y	p	t									

(۴)



۱۵۵ - کدام گزینه نادرست است؟ (۷۶)

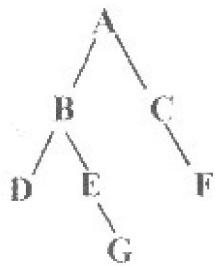
(۱) بیشترین تعداد گره‌ها در یک درخت دودوئی به عمق $2^k - 1, k \geq 1$ است.

(۲) بیشترین تعداد گره‌های سطح آم یک درخت دودوئی 2^{k-1} است ($k > 1$).

(۳) در هیچ درخت عادی گره صفر وجود ندارد.

(۴) در هیچ درخت دودوئی گره صفر وجود ندارد.

۱۵۶ - پیماش ABDEGCF کدامیک از گزینه‌های زیر است؟ (۷۶)



LRD (۱)

DLR (۲)

LDR (۳)

RDL (۴)

۱۵۷ - در یک درخت دودوئی کامل با ۵ سطح حداقل چند گره وجود دارد؟ (سطح ریشه برابر یک فرض شود) (۸۰)

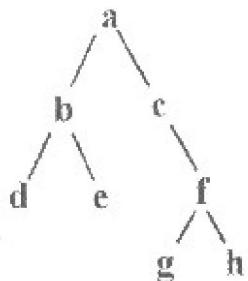
32 (۱)

31 (۲)

16 (۳)

15 (۴)

۱۵۸ - پیماش درخت زیر به روشن postorder کدام است؟ (۷۷)



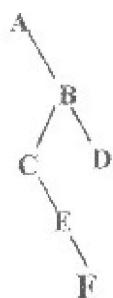
dhcaegfh (۱)

aefhgbed (۲)

abdecfgh (۳)

debghfcda (۴)

۱۵۹ - پیماش ABCEFD کدام یک از گزینه‌های زیر است؟ (۷۷)



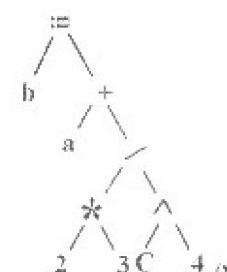
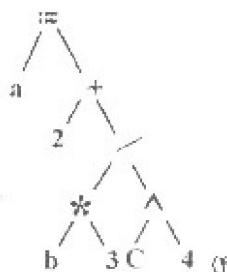
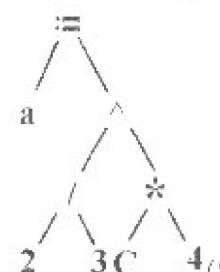
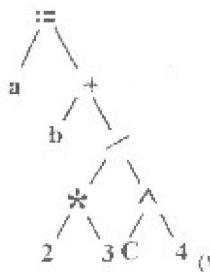
LDR (۱)

DLR (۲)

DRL (۳)

RDL (۴)

۱۶۰ - عبارت $a+b/*23^c4$ پیماش preorder کدامیک از درخت‌های زیر است؟ (۷۸)



۱۶۱ - درخت دودویی T به صورت آرایه‌ای نمایش داده شده است. پیماش inorder (LDR) درخت کدام است؟ (۷۸)

a	b	c		d					e
1	2	3	4	5	6	7	8	9	10

baecd (۴)

bade... (۳)

bdaec (۲)

bedac (۱)

۱۶۲ - در نمایش و ارزیابی عبارت‌های ریاضی با استفاده از درخت‌های دودویی کدام یک از روش‌های طی کردن درخت فرم طبیعی عبارت را بدست می‌دهند؟ (۷۸)

inorder LVR (۴)

family order (۳)

preorder (۲)

postorder (۱)

۱۶۳ - درجه یک گره (node) در درخت شامل چیست؟ (۷۸)

(۱) تعداد گره‌ها (۴)

(۲) تعداد ریشه‌ها (۳)

(۳) تعداد اشواب‌ها (۲)

(۴) تعداد فیده‌ها (۱)

۱۶۴ - کدام گزینه نادرست است؟ (۷۷)

(۱) در هر درخت تعداد ریال‌ها یکی کمتر از تعداد رأس‌هاست.

(۲) یک درخت، یک گراف همبند بدون دور است.

(۳) در هر درخت هر دو رأس با یک مسیر منحصر به فرد به هم متصل هستند.

(۴) در هر درخت تعداد ریال‌ها یکی بیشتر از تعداد رأس‌هاست.

۱۶۵ - بیشترین تعداد گره‌ها در یک درخت دودویی (Binary tree) با عمق h برابر کدام است؟ (۷۷)

2^{h-1} (۴)

$2^h - 1$ (۳)

$2^h + 1$ (۲)

2^h (۱)

(۱۶۶) - پردازه زیر را برای پیمایش درخت دودویی نوشت‌ایم:

```

procedure trav (r , tree);
begin
  if r<> nil then begin
    trav(r^.left );
    writeln(r^.info );
    trav(r^.right );
  end;
end;

```

کدامیک از عبارت‌های زیر درست است؟

(۱) اگر جای سطوحای ۴ ، ۵ پردازه را عوض کنیم پردازه برای پیمایش پس ترتیبی به کار می‌رود.

(۲) این پردازه برای پیمایش پس ترتیبی (Postorder) به کار می‌رود.

(۳) این پردازه برای پیمایش میان ترتیبی (inorder) به کار می‌رود.

(۴) اگر جای سطوحای ۵ ، ۶ پردازه را عوض کنیم پردازه برای پیمایش پیش ترتیبی (preorder) به کار می‌رود.

(۱۶۷) - کدام گزینه نادرست است؟ (۷۶)

(۱) تعداد زیردرخت‌های یک گره درجه آن گره نامیده می‌شود.

(۲) تعداد زیردرخت‌های یک گره درجه آن درخت نامیده می‌شود.

(۳) فرزندان یک گره، گره‌های هم‌زاد با هم نامیده می‌شوند.

(۴) یک جنگل شامل n درخت مجزا است ($n \geq 0$)

(۱۶۸) - به یک درخت دودویی T یک درخت توسعه‌یافته گفته می‌شود اگر :

(۱) دارای دو گره ریشه باشد.

(۲) دارای ۴ گره ریشه باشد.

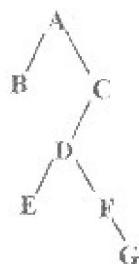
(۳) هر گره آن دارای صفر یا دو فرزند باشد.

(۴) هر گره آن دارای ۴ فرزند باشد.

(۱۶۹) - به فرض داشتن یک درخت دودویی کامل با N گره، برای هر گره با اندیس i به گونه‌ای که $N \leq i \leq 1$ باشد کدام گزینه نادرست است؟

(۸۱)

(۱) اگر $i \neq n$ آنگاه فرزند ادر ۲/۱ است.(۲) اگر $N \leq 2i$ باشد آنگاه فرزند چپ i در $2i$ است.(۳) اگر $N > 2i$ باشد آنگاه فرزند چپ ندارد.(۴) اگر $N \leq 2i+1$ باشد آنگاه فرزند راست i در $2i+1$ است.



۱۷۰ - پیمایش ABCDEFG کدامیک از گزینه‌های زیر است؟ (۸۱)

(NLR) DLR (۱)

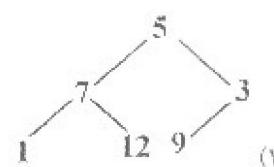
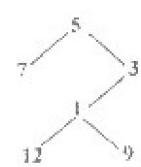
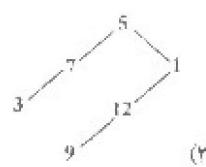
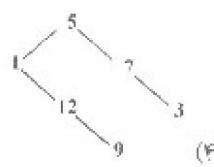
(LNR) LDR (۲)

(LRN) LRD (۳)

(RNL) RDL (۴)

۱۷۱ - اگر نمایش ترتیبی ذخیره‌سازی یک درخت دودویی به شکل زیر باشد، آن درخت کدام است؟ (مسابقات آموزشکده‌های فنی ۸۲)

5	7	3	4	5	6	7	8	9	10	11	12	13	14
1	2	3	4	5	6	7	8	9	10	11	12	13	14



۱۷۲ - اگر آدرس گره درخت دودویی به وسیله ۱ مشخص شود، الگوریتم بازنگشتنی زیر نشان‌دهنده کدام پیمایش است؟ LPTR اشاره گر چپ و RPTR اشاره گر سمت راست هستند. (دولتی ۸۲)

Recursive (t)

```

IF t = NULL then 'empty tree' return
IF LPTR (t) ≠ NULL then Recursive (LPTR (t))
IF RPTR (t) ≠ NULL then Recursive (RPTR(t))
Write (DATA(t))
Return
  
```

postorder (۴)

preorder (۲)

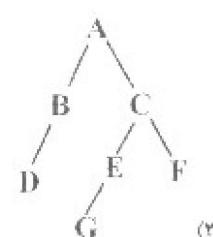
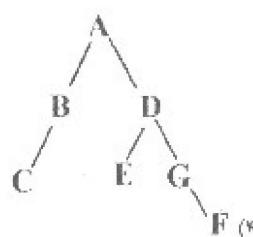
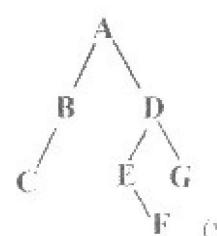
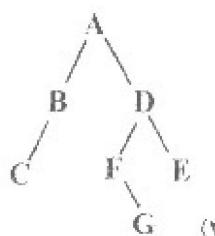
inorder (۲)

infix ()

۱۷۳ - کدام درخت مربوط به پیمایش رویرو است؟ (دولتی ۸۲)

preorder : ABCDEFO

inorder : CBAEFDG



۱۷۴ - کدامیک از مجموعه‌های زیر نمایش دهنده یک درخت است؟ (آزاد ۸۷)

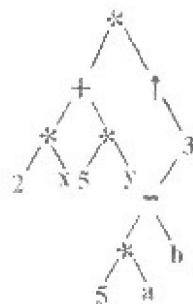
$$A = \{(0, 2), (0, 3), (2, 1), (2, 3)\} \quad (\text{۱})$$

$$A = \{(0, 1), (0, 3), (3, 2), (3, 3), (-4, 2)\} \quad (\text{۱})$$

$$A = \{(0, 1), (0, 3), (0, 4), (4, 1)\} \quad (\text{۱})$$

$$A = \{(0, 2), (0, 3), (3, 1), (3, 4)\} \quad (\text{۱})$$

۱۷۵ - شکل زیر نمایش گدام گفته است؟ (آزاد ۸۴)



$$(2x + 5y)(5a - b) \quad (\text{۱})$$

$$(2x + 5y)(b - 5a)^{-1} \quad (\text{۱})$$

$$(2x + 5y)(5a - b)^3 \quad (\text{۱})$$

$$(5a - b)(2x + 5y)^3 \quad (\text{۱})$$

۱۷۶ - لیست مربوط به گره‌های درخت دودویی را در نظر بگیرید. root (ریشه درخت) left (ریشه چپ) و Right (ریشه راست) است. پیمایش

درخت ۱ عبارت است از: (آزاد ۸۷) inorder

	INFO	left	right
root →	1	A	2
	2	B	4
	3	C	0
	4	D	0
	5	E	7
	6	F	9
	7	G	0
	8	H	0
	9	O	0

D B G E H A C O F ()

A B D E G H C F O (۱)

D B E G H A C F O (۲)

O F C A D G H E B (۳)

۱۷۷ - اگر پیمایش بین ترتیب (Inorder) یک درخت دودویی پر (FULL) به صورت زیر باشد، پیمایش پیش‌تریت آن گدام است؟

(علمی کاربردی ۸۲)

پیمایش بین ترتیب = dbagecf

۱) این پیمایش منحصر به فرد

adbegef (۱)

gfcdeba (۲)

abdecgf (۳)

نیست.

۱۷۸ - گذنیک از موارد از خواص ادرخت جستجوی دودویی نیست؟ (آزاد ۷۹)

۱) کلید‌های واقع در زیر درخت غیر تهی چب باید کمتر از مقدار واقع در ریشه زیر درخت راست باشد.

۲) هر عضو دارای یک کلید است و کلیدها منحصر به فرد نیستند.

۳) زیر درختان چب و راست نیز خود درختان جستجوی دودویی می‌باشند.

۴) کلیدهای زیر درخت غیر تهی راست، باید بزرگتر از مقدار کمید ریشه زیر درخت چب باشند.

۱۷۹ - کدامیک از مجموعه های زیر نمایش دهنده یک درخت است؟ (آزاد ۷۹)

$$E(G) = \{(0,1)(0,2)(0,3)(1,2)\} \quad (۱)$$

$$E(G) = \{(0,1)(0,2)(1,3)(1,4)(2,5)\} \quad (۲)$$

$$E(G) = \{(0,1)(0,2)(4,0)(4,3)(4,2)\} \quad (۴)$$

$$E(G) = \{(0,1)(0,3)(0,2)(2,0)\} \quad (۳)$$

۱۸۰ - چنانچه بخواهیم داده های تکراری را از لیست حذف کنیم، از کدام ساختار داده ای برای لیست مزبور استفاده می کنیم؟ (دولتشی)

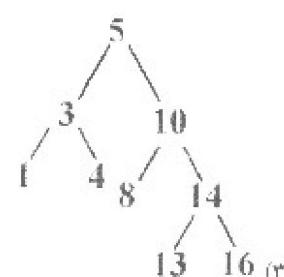
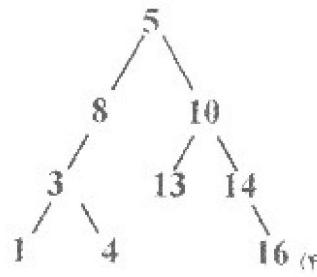
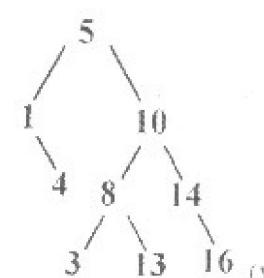
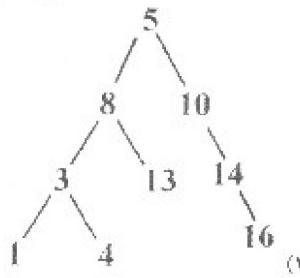
(۴) صفحه

(۳) پشتہ

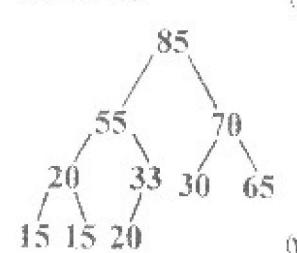
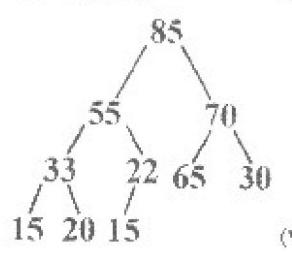
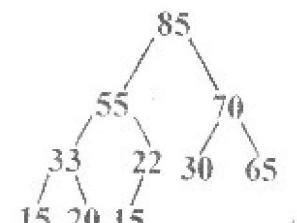
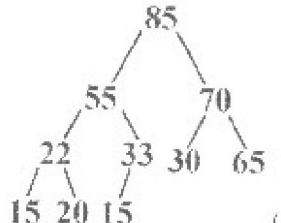
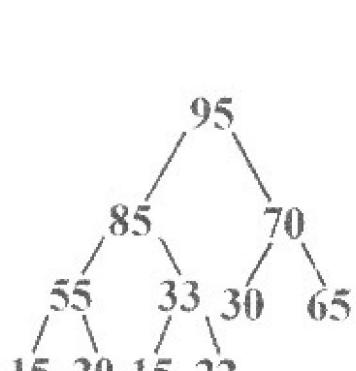
(۲) درخت Heap

۱۸۱ - درخت جستجوی دودویی، درختی است که در آن اولاً برای هر گره از درخت، تمام گره های زیر درخت سمت چپ آن کوچک تر با مساوی با گره ریشه و تمام گره های زیر درخت سمت راست آن بزرگتر از گره ریشه باشند و تاباً تمام گره های زیر درخت سمت چپ کوچک تر از تمام گره های زیر درخت راست باشند. فرض کنید اعداد زیر به ترتیب از سمت چپ وارد یک درخت جستجوی دودویی شوند. درخت حاصل کدام است؟ (مسابقات آموزشکده های فنی ۷۸)

$$5 - 10 - 8 - 3 - 14 - 1 - 13 - 4 - 16$$

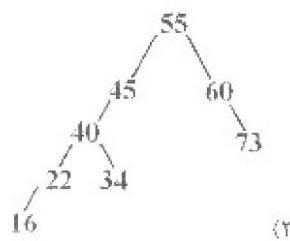


۱۸۲ - با توجه به درخت (Max Heap) زیر چنانچه ریشه آن (95) حذف شود شکل جدیدش کدام است؟ (مسابقات آموزشکده های فنی ۹۵)

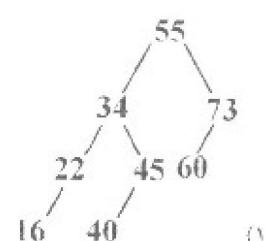


۱۸۳ - فرض کنید اعداد زیر به ترتیب در یک درخت جستجوی دودویی خالی درج شوند. درخت نهایی کدام است؟ (مسابقات آموزشکده‌های فنی)

۵۵ , ۷۳ , ۳۴ , ۴۵ , ۲۲ , ۱۶ , ۶۰ , ۴۰

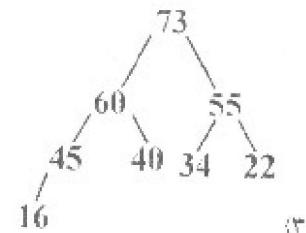


(۲)



(۱)

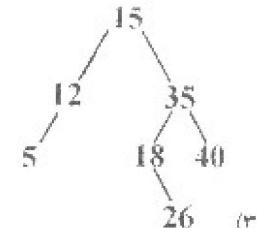
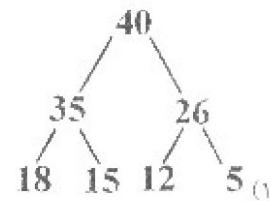
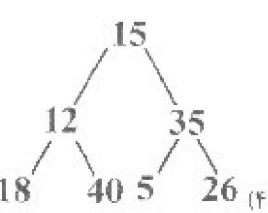
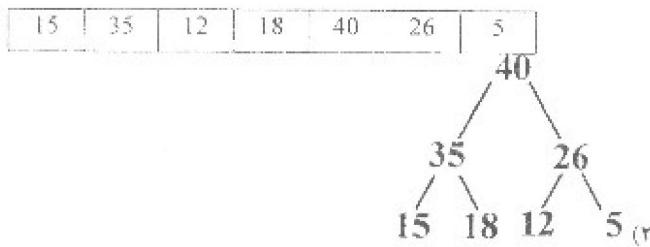
(۴) هیچ‌کدام



(۳)

۱۸۴ - فرض کنید آرایه زیر را با روش heap sort می‌خواهیم مرتب کنیم پس از اجرای مرحله اول الگوریتم، درخت حاصل شده کدام

است؟ (مسابقات آموزشکده‌های فنی)



۱۸۵ - حداقل تعداد مقایسه برای یافتن کلیدی در یک درخت جستجوی دودویی با n گره کدام است؟ (دولتی)

(۲) به اندازه تعداد گره‌های سطح آخر

(۱) به اندازه عمق درخت

$$\frac{n}{2} \quad (4)$$

$$n \log_2 \frac{n}{2} \quad (5)$$

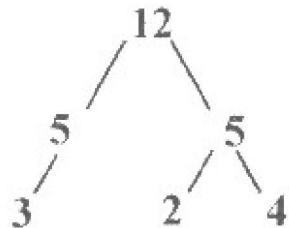
۱۸۶ - روش معرف نمایش درخت Max Heap چگونه است؟ (دولتی)

(۴) لیست مجاورتی

(۳) آرایه

(۲) پشته بیوندی

(۱) صف پیوندی



۱۸۷ - درخت روبه رو چه نوع درختی است؟ (دولتی ۸۱)

(۱) کامل

(۲) جستجوی دودویی

✓ (۳) max-tree

(۴) Max-Heap

۱۸۸ - کاربرد درخت Heap کدام است؟ (دولتی ۸۱)

(۱) جستجوی سریع

(۲) صفح و پشتہ

(۳) مرتب کردن داده‌ها - صفح اولویت‌دار

(۴) مرتب کردن داده‌ها

۱۸۹ - گزینه غلط کدام است؟ (دولتی ۸۱)

(۱) درخت پر، درخت کامل نیز هست.

(۲) درخت دودویی می‌تواند تهی باشد.

(۳) درخت Max-tree که کامل نیز باشد، Max-Heap است.

۱۹۰ - کدام درخت را نمی‌توان با آرایه نشان داد؟ (دولتی ۸۱)

Max - Heap

(۱) کامل

(۲) پر

(۳) اربی

(FULL)

(complete)

Min tree

Max Heap

۱۹۱ - برای نمایش کدام درخت، از روش پیوستی استفاده می‌شود؟ (آزاد ۸۱)

(۱) درخت پر

۱۹۲ - کدام گزینه صحیح است؟ (دولتی ۸۱)

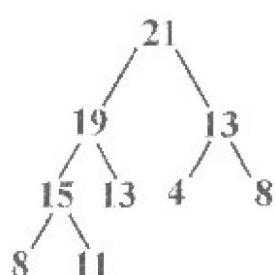
(۲) درخت min tree درخت کامل است.

(۱) درخت Heap min درخت کامل است.

(۳) درخت اربی، پک درخت min tree است.

(۲) درخت اربی، پک درخت min tree است.

۱۹۳ - اگر یک گره از درخت max Heap زیر حذف شود، داده آخرین سطح درخت چیست؟ (آزاد ۸۱)



۱۱ (۱)

۱۵ (۲)

۸ (۳)

21 (۴)

۱۹۴ - کاربرد درخت جستجوی دودویی چیست؟ (آزاد ۸۱)

(۱) حذف داده‌های تکراری از یک لیست

(۲) پیمایش VLR آن سبب مرتب شدن داده‌ها می‌شود.

(۳) اگر داده‌ها به ترتیب صعودی در آن وارد شوند، برای عمل جستجو بسیار مناسب است.

(۴) اگر داده‌ها به ترتیب نزولی در آن وارد شوند، برای عمل جستجو بسیار مناسب است.

۱۹۵ - کدام گزینه نادرست است؟ (آزاد ۸۰) (دوفتی)

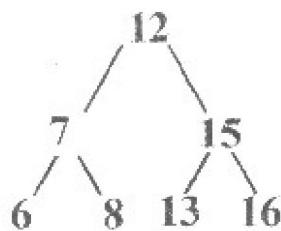
(۱) درخت min tree درخت min heap است که کامل نیز باشد.

(۲) روش نمایش درخت کامل، آرایه است.

(۳) تعداد گرهای سطح i درخت کامل ($i \geq 1$) $= 2^{i-1}$ گره است.

(۴) از روش پیوندی برای نمایش درخت اریب استفاده می‌شود.

۱۹۶ - درخت جستجوی دودویی روبرو شانده کدامیک از رشته عددی زیر می‌باشد؟ (مسابقات آموزشکده‌های فنی ۸۲)



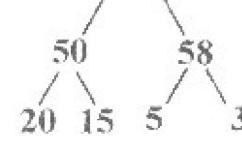
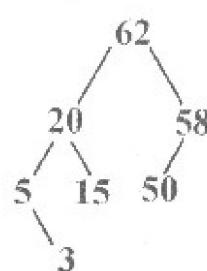
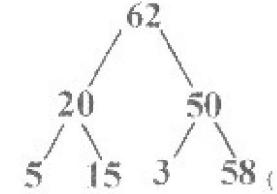
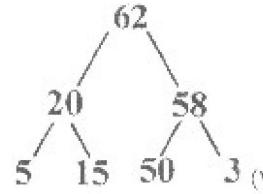
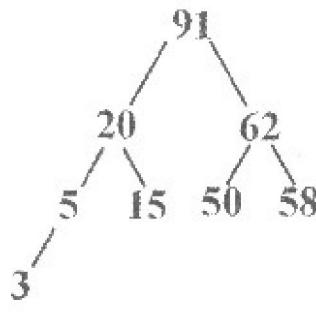
۱2, 6, 7, 8, 13, 15, 16 (۱)

۱2, 6, 7, 8, 15, 16, 13 (۲)

۱2, 7, 6, 8, 13, 15, 16 (۳)

۱2, 15, 7, 6, 8, 16, 13 (۴)

۱۹۷ - گر در درخت Max heap زیر ریشه حذف شود، درخت جدید کدام خواهد بود؟ (مسابقات آموزشکده‌های فنی ۸۲)



(۳)

۱۹۸ - در کدام نوع از درخت‌های دودویی پیمایش درخت به صورت *inorder* باعث پدیدآمدن یک لیست مرتب می‌شود؟ (آزاد ۸۲)

spanning tree (۴)

binary search tree (۳)

binary tree (۲)

heap (۱)

۱۹۹ - کدام گزینه نادرست است؟ (آزاد ۸۲) (دوفتی)

(۱) یک درخت دوایی کامل همیشه یک heap است.

(۲) یک heap همیشه یک درخت دوایی کامل است.

(۳) یک heap همیشه از نوع درخت جستجوی دودویی نیست.

(۴) یک درخت جستجوی دودویی (Binary Search) همیشه یک heap نیست.



۲۰۰ - درخت دودویی T این خاصیت را دارد که هر گره از فرزند چپش بزرگتر و از فرزند راستش کوچک‌تر است در این صورت T

(علمی کاربردی ۸۲)

۱) یک درخت دودویی جستجو است.

۲) هیچکدام.

۳) min heap.

۲۰۱ - فرض کنید زیر برنامه Test روی یک درخت دودویی جستجو اعمال گردد. پس از انجام تغیرات داده شده کدام گزینه صحیح است؟

(علمی کاربردی ۸۲)

Procedure Test (var T:tree);

var

p:tree

Begin

If T < > nil then

Begin

Test (T^.right); writeln (T^.data);

Test (T^.left);

P:=T^.right;

T^.right := T^.left;

T^.left = p;

End;

End;

۱) پیماش پس ترتیب postorder درخت، اعداد را به صورت نزولی چاپ می‌کند.

۲) پیماش پس ترتیب postorder درخت، اعداد را به صورت صعودی چاپ می‌کند.

۳) پیماش بین ترتیب inorder درخت، اعداد را به صورت نزولی چاپ می‌کند.

۴) پیماش پیش ترتیب preorder درخت اعداد را به صورت صعودی چاپ می‌کند.

۲۰۲ - خروجی ناتیج زیر با فرض آن که T یک درخت دودویی جستجو باشد چیست؟ (علمی کاربردی)

Function what (T:Tree) Integer;

Var

P: tree;

Begin

P := T;

while P^.left <> nil Do

P := P^.left;

what := P^.element;

end;

۱) ماکریم عنصر موجود در درخت

۲) عنصر وسط از تراکت بزرگی

۳) بستگی به نحوه قرار گرفتن اعداد در درخت دارد.

۴) مینیمم عنصر موجود در درخت

۲۰۳ - حداقل تعداد لبه‌های یک گراف جهت‌دار شامل n گره برابر است با: (آزاد ۷۹)

$$2n - n = n$$

$$n^2 - 1 = n^2 - 1$$

$$n^2 - n = n^2 - n$$

$$2n - 1 = 2n - 1$$

۲۰۴ - کدامیک از مجموعه‌های زیر نمایش دهنده یک درخت است؟ (آزاد ۷۹)

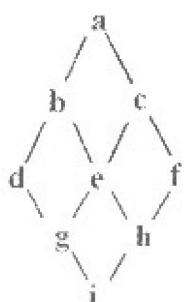
$$E(G) = \{(0,1), (0,2), (0,3), (1,2)\} \quad (۱)$$

$$E(G) = \{(0,1), (0,2), (1,3), (1,4), (2,5)\} \quad (۲)$$

$$E(G) = \{(0,1), (0,2), (4,0), (4,3), (4,2)\} \quad (۳)$$

$$E(G) = \{(0,1), (0,3), (0,2), (2,0)\} \quad (۴)$$

۲۰۵ - پیماش عمقی (depth-first search) گراف زیر چیست؟ اگر از گره a شروع کنیم (به ترتیب از چپ به راست) (مسابقات آموزشکده‌های فنی ۷۹)



a,b,c,d,e,f,g,h,i (۱)

a,b,d,g,i,h,c,e,f (۲)

a,b,c,f,h,e,j,g,d (۳)

a,b,d,g,i,h,f,c,e (۴)

۲۰۶ - کدام گزینه نادرست است؟ (مسابقات آموزشکده‌های فنی ۷۶)

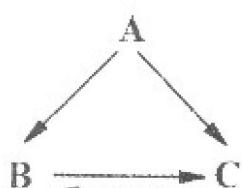
(۱) تفاوتی میان گراف و گراف چندگانه وجود ندارد.

(۲) حلقه یک مسیر ساده است که اولین و آخرین راس آن یکی باشد.

(۳) در یک گراف بدون جهت با n رأس بیشترین تعداد لبه‌ها $n(n-1)/2$ است.

(۴) در یک گراف جهت‌دار با n رأس بیشترین تعداد لبه‌ها $(n-1)n$ است.

۲۰۷ - در گراف زیر چند مسیر به طول ۲ وجود دارد؟ (مسابقات آموزشکده‌های فنی ۷۷ و ۸۰)



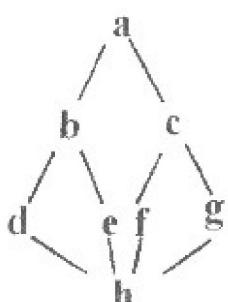
2 (۲)

1 (۱)

4 (۴)

3 (۳)

۲۰۸ - پیماش عمقی گراف زیر چه خواهد بود؟ اگر از گره a شروع کنیم، (پاسخ‌ها از چپ به راست نوشته شده‌اند). (دولتشی ۸۰)



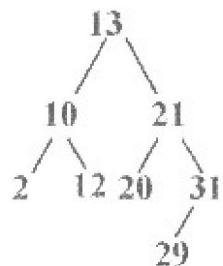
h,d,e,f,g,b,c,a (۱)

a,e,g,h,f,b,e,d (۲)

a,b,c,d,e,f,g,h (۳)

a,b,d,h,c,f,c,g (۴)

۲۰۹ - در شکل زیر کدامیک از انتخاب‌های روش Breadth - First - Traversal یک درخت است؟ (آزاد ۸۰)



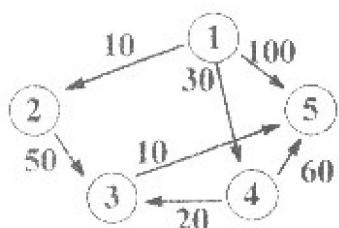
13, 21, 10, 31, 20, 12, 29, 2 (۰)

13, 10, 21, 31, 20, 12, 2, 29 (۱)

13, 10, 2, 12, 21, 20, 31, 29 (۲)

13, 10, 21, 2, 12, 20, 31, 29 (۳)

۲۱۰ - در گراف زیر کمترین هزینه از گره ۱ به ۵ درایی مسیری با طول؟ (مسابقات آموزشکده‌های فنی ۸۱)



1 (۰)

2 (۱)

3 (۲)

4 (۳)

۲۱۱ - از کدام روش برای تماش گراف استفاده نمی‌شود؟ (دولتی ۸۱)

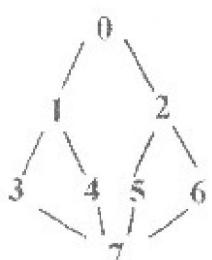
۱) ماتریس مجاورتی

۲) لیست یک طرفه دوبار

۳) لیست مجاورتی

۴) لیست چندگانه مجاورتی

۲۱۲ - پیمایش dfs(0) را چه کدام است؟ (دولتی ۸۱)



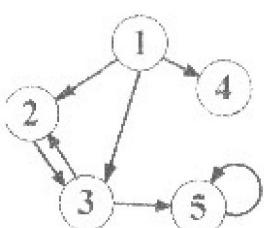
1, 3, 4, 6, 0, 2, 4, 6 (۰)

0, 1, 2, 3, 4, 5, 6, 7 (۱)

0, 1, 3, 7, 4, 5, 2, 6 (۲)

0, 2, 1, 4, 3, 5, 6, 7 (۳)

۲۱۳ - ماتریس مجاورتی (همجاوری) گراف زیر کدام است؟ (مسابقات آموزشکده‌های فنی ۸۲)



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

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

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

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

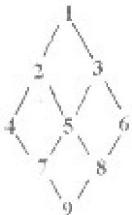
- ۲۱۴ - کدام گزینه نادرست است؟ (مسابقات آموزشکده‌های فنی ۸۲)
- (۱) در یک گراف بدون جهت با n رأس بیشترین تعداد لبه‌ها $\frac{n(n-1)}{2}$ است.
 - (۲) حلقه یک مسیر ساده است که اولین و آخرین رأس آن بکنی می‌باشد.
 - (۳) تفاوتی میان گراف و گراف چندگانه وجود ندارد.
 - (۴) در یک گراف جهت‌دار با n رأس بیشترین تعداد لبه‌ها $(n-1)n$ است.

- ۲۱۵ - اگر G یک گراف ساده از درجه n باشد، در صورتی که گراف G دارای ۵۶ یا n و مجموع درجات رئوس مکمل گراف G برابر ۱۶۰ باشد مقدار n کدام است؟ (دولتشی ۸۲)

- ۱۷ (۱) ۱۷ (۲) ۱۶ (۳) ۱۶ (۴)

- ۲۱۶ - اگر a, b دو گره در یک گراف بدون جهت G باشند و اگر دو مسیر P_1, P_2 از a به b وجود داشته باشد، آنگاه: (دولتشی ۸۲)
- (۱) a, b مجاورند.
 - (۲) G نمی‌تواند یک گراف باشد.
 - (۳) G دارای چرخه است.
 - (۴) احتیاج به جهت مسیر داریم.

- ۲۱۷ - پیغایش عمقی (depth - First - Search) گراف زیر کدام است؟ (دولتشی ۸۲)



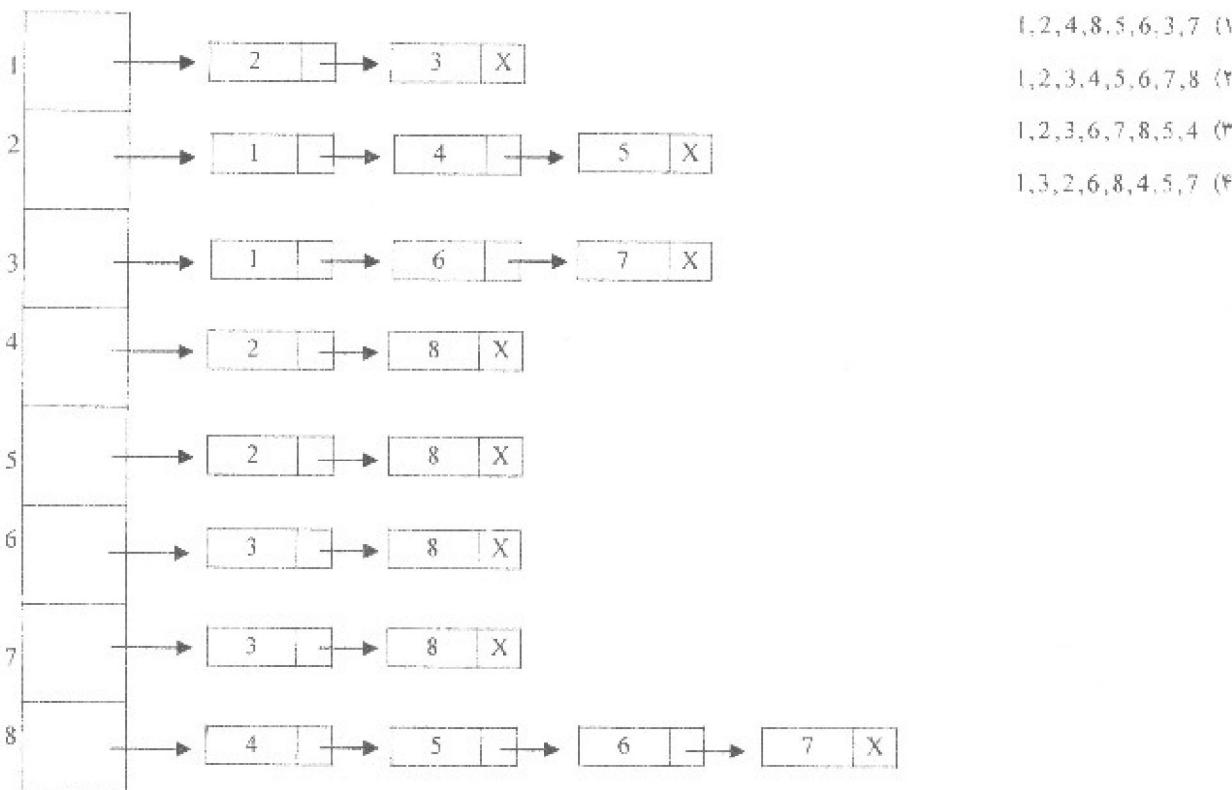
۱, ۲, ۳, ۴, ۵, ۶, ۷, ۸, ۹ (۱)

۱, ۲, ۴, ۷, ۹, ۸, ۵, ۳, ۶ (۲)

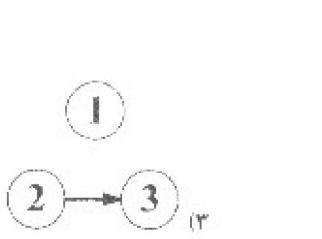
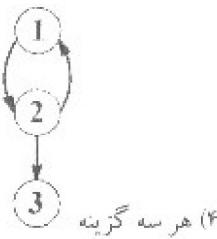
۱, ۲, ۳, ۶, ۸, ۵, ۹, ۷, ۴ (۳)

۱, ۲, ۴, ۷, ۹, ۸, ۶, ۳, ۵ (۴)

- ۲۱۸ - لیست مجازی گراف غیرجهت‌دار زیر را در نظر بگیرید. جستجوی عمقی (dfs) عبارت است از: (آزاد ۸۲)



۲۱۹ - گراف زیر را در نظر بگیرید. کدام گزینه، زیر گراف گراف فوق است؟ (آزاد ۸۲)



۲۲۰ - خروجی کدام الگوریتم درخت پوشابرای گراف نیست؟ (علمی کاربردی ۸۲)

BFS (۴)

kruskal (۳)

Dijkstra (۴)

Prim (۱)

۲۲۱ - اگر A ماتریس مجاورت یک گراف جهت دار از مرتبه n بوده و یک آرایه α عنصری که تمام عناصر آن در ابتدا True هستند، باشد.

تابع ذیل با دریافت دو راس i, j چه کاری انجام می دهد؟ (علمی کاربردی ۸۲)

```
Function F (i, j : Byte) Boolean;
```

```
Var
```

```
T : Boolean;
```

```
Begin
```

```
V [j] := False;
```

```
if A[i, j] = 1 then F := True
```

```
else Begin
```

```
T := False;
```

```
for k := 1 To n Do
```

```
if (A[i, k] = 1) and V [k] then
```

```
T := T or F (i, j);
```

```
F := T;
```

```
End;
```

```
End;
```

۱) تشخیص می دهد آیا از راس آتابه راس i مسیر وجود دارد یا خیر؟

۲) تشخیص می دهد آیا دوری با شروع از راس i با گذر از راس j در گراف وجود دارد یا خیر؟

۳) تشخیص می دهد آیا یالی از راس آتابه راس i در گراف می باشد یا خیر؟

۴) تشخیص می دهد آیا مسیری به طول n از آتابه وجود دارد یا خیر؟

مرتب سازی

۲۲۲ - اگر یک لیست مرتب شده با n خانه را با استفاده از الگوریتم Binary search برای یک مقدار خاص جستجو کنیم، تعداد دفعات مقایسه

چه خواهد بود؟

$O(\log_2 n)$ (۴)

$O(\log n)$ (۳)

$O(n^2)$ (۲)

$O(n/2)$ (۱)

۲۲۳ - برای مرتب سازی یک آرایه n تابی به روش Bubble - sort حداقل چند تعویض لازم است؟ (مسابقات آموزشکده های فنی ۷۹)

$\frac{N^2 - 3N - 4}{2}$ (۴)

$\frac{N^2}{4} + 3(N-1)$ (۳)

$\log_2(N-1)$ (۲)

$\frac{N(N-1)}{2}$ (۱)

ساختار داده ها



۲۲۴ - کدام کار روی همه ساختارهای داده‌ی الجام می‌شود؟ (مسابقات آموزشکده‌های فنی)

(۱) پردازش Process

(۲) ادغام (Merge)

(۳) درج (Insert)

(۴) مرتب‌سازی (sort)

۲۲۵ - نماد $O(\log n)$ نشان‌دهنده پیچیدگی کدام الگوریتم است؟ (مسابقات آموزشکده‌ها)

(۱) جستجوی دودویی (Binary Search)

(۲) جستجوی خطی (Linear Search)

(۳) مرتب‌سازی حبابی (Bubble sort)

(۴) مرتب‌سازی سریع (Quick sort)

۲۲۶ - بدترین حالت در روش مرتب‌سازی سریع (Quick sort) (جیست؟ (دولتی))

(۱) عنصر لیست به ترتیب معکوس باشد.

(۲) عنصر لیست از قبل مرتب باشد.

(۳) یک نیمه لیست مرتب باشد.

(۴) یک نیمه لیست معکوس باشد.

۲۲۷ - چنانچه بخواهیم داده‌های تکرار را از لیست حذف کنیم، از کدام ساختار داده‌ای برای لیست مذبور استفاده می‌کیم؟ (دولتی)

(۱) صف (Queue)

(۲) پشته (Stack)

(۳) درخت (Tree)

۲۲۸ - الگوریتم Quick sort یک رشته n تایی را با چه سرعانی مرتب می‌کند؟

$\log 2^n$

$O(n^2)$

$O(n)$

$O(n \log n)$

۲۲۹ - الگوریتم Quick sort یک رشته n تایی را در بدترین حالت با چه سرعانی مرتب می‌کند؟ (مسابقات آموزشکده‌های فنی)

$O(n \log n)$

$O(\log n)$

$O(n^2)$

$O(n)$

۲۳۰ - اگر آرایه مرتب A دارای r عنصر و آرایه مرتب B دارای p عنصر باشد، حداقل تعداد مقایسه برای ترتیب (Merge) دو آرایه کدام است؟

امست؟

$\log(r+p)$

$\text{Max}(r, p)$

$\sqrt{r+p}$

$\frac{r+p}{2}$

۲۳۱ - اگر N رکورد داشته باشیم تعداد کل مقایسه در روش liner insertion sort برابر است با:

$\frac{N^2}{4}$

$\frac{N(N+1)}{4}$

$\frac{N(N-1)}{2}$

$\frac{N^2}{4}$

۲۳۲ - در کدام روش مرتب‌سازی، لیست داده‌ها همواره به دو نیمه تقسیم می‌شود؟

Selection sort

Quick sort

Merge sort

Bubble sort

۲۳۳ - آرایه زیر یک Heap است. برای درج عدد ۹۵ در آرایه به گونه‌ای که آرایه نهایی نیز وضعیت Heap داشته باشد، چند عمل exchange (تعویض دو کمیت) لازم است؟

100
90
82
85
74
75
73
68
70

(۱) دو

(۲) چهار

(۳) شش

(۴) هشت

۲۳۴ - کاربرد درخت Heap کدام است؟

(۱) جستجوی سریع

(۲) مرتب کردن داده‌ها - صف اوپریت‌دار

(۳) مرتب کردن داده‌ها

۲۳۵ - برای مرتب‌سازی یک آرایه n تابی به روش مرتب‌سازی حبابی، حداقل چند تعویض لازم است؟

$$\frac{n(n-1)}{2}$$

$$\log_2(n-1)$$

$$n \log n$$

$$n^2$$

۲۳۶ - الگوریتم زیر چه نوع مرتب‌سازی است و تعداد مقایسه‌های آن بر حسب تعداد داده‌ها (۱) کدام است؟

1. Repeat step 2 and 3 for $k = 1$ to $N-1$

2. Set PTR := 1

3. Repeat while PTR $\leq N - k$

if DATA [PTR] > DATA [PRT + 1] then

swap DATA [PRT] and DATA [PTR + 1]

PTR := PTR + 1

End of step 3 loop

End of step 1 loop

4. Exit

$$\frac{n(n-1)}{2}$$

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

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

$$\frac{n(n-1)}{2}$$

۲۳۷ - می خواهیم یک آرایه ۵ عضوی را با استفاده از روش Selection sort مرتب کنیم. اگر آرایه در اینجا به ترتیب عکس مرتب شده باشد،

چه تعداد مقایسه برای مرتب کردن آن مورد نیاز است؟

20 (۴)

15 (۳)

10 (۲)

1 (۱)



ساختمن دادهها

۲۲۸ - جنایجه یک بردار کاملاً مرتب را به الگوریتم liner insertion sort بدهیم تعداد جابجایی و تعداد مقایسه چگونه است؟

(۱) تعداد جابجایی صفر و حداقل مقایسه را خواهیم داشت.

(۲) تعداد جابجایی و مقایسه در بالاترین مقطع می باشد.

(۳) تعداد جابجایی صفر و حداقل مقایسه را خواهیم داشت.

۲۲۹ - اگر بخواهیم یک لیست متصل (linked list) که آدرس اول آن first می باشد را با کمترین تعداد عملیات مرتب نماییم (sort) به جای

علامت؟ در الگوریتم زیر چه مقداری به ترتیب قرار دهیم؟

```
for (P = first ; ? ; P = P → link)
```

```
for (q = ? ; q ; q = q → link)
```

```
if (P → Data > q → Data)
```

```
swap (& P → Data , & q → Data)
```

P → link
P → link

P
P

P → link
P

P
P → link

۲۴۰ - کدامیک از روش‌های sort زیر در بدترین حالت از $O(n^2)$ است؟

(۱) هر سه

Quick sort (۴)

Insertion sort (۲)

Bubble sort (۱)

۲۴۱ - آرایه A به طول n را در نظر بگیرید. الگوریتم زیر یانگر کدام نوع عمل مرتب‌سازی است؟ (دلتی ۸۲)

For j ← 2 to n do

key ← A[j]

Insert A[j] Into the sorted sequence A[1:j'-1]

I ← j-1

while I > 0 and A[I] > key

Do A[I+1] ← A[I]

I = I - 1

A[I+1] ← key

Quick sort (۴)

selection sort (۴)

Insertion sort (۲)

Bubble sort (۱)

۲۴۲ - کدام الگوریتم مرتب‌سازی یک آرایه تقریباً مرتب شده را سریعتر مرتب می نماید؟ (علمی کاربردی ۸۲)

Insertion sort (۴)

Bubble sort (۴)

Quick sort (۲)

Selection sort (۱)

۲۴۳ - برای ادغام (merge) دو آرایه مرتب شده A و B که به ترتیب m و n عنصری هستند حداقل چند مقایسه لازم است؟ (علمی کاربردی

(۸۲)

$\frac{(m+n)(m+n-1)}{2}$ (۴)

$\max(n,m)$ (۴)

n + m - 1 (۲)

n + m (۱)

۲۴۴ - اگر آرایه‌ای مرتب از اعداد صحیح ۱ تا 1024 باشد، الگوریتم جستجوی دودوئی با چند بار تکرار عدد 4 را پیدا می کند؟

10 (۴)

9 (۳)

7 (۲)

8 (۱)

۸۳ دولتی

۲۴۵ - الگوریتم مقابله کدام عمل را انجام می‌دهد؟

```

function L(x: pointer): integer;
var p: pointer;
begin
L := 0;
if x < nil then begin
p := x;
repeat
L := L + 1;
p := p^.link;
until p = x;
end;
end;

```

(۲) پیمايش لیست پیوندی چرخشی

(۱) پیمايش لیست پیوندی خطی

(۴) تعیین طول لیست چرخشی

(۳) تعیین طول لیست پیوندی خطی

۲۴۶ - با توجه به نکه برنامه مقابل مرتبه اجرایی (۸) اعبارت است از:

```

i = 1;
while (i <= n)
{
j = 1;
while (j <= n)
{
j = j * 2;
}
i = i + 1;
}

```

$$8 \left(\log_2 8 + 1 \right) \quad (۱) \quad 8 \left(\frac{8+1}{2} \right) \quad (۲) \quad \log_2 8 + 1 \quad (۳) \quad \log_2 8 \quad (۴)$$

۲۴۷ - کدام مورد در ساختار یک صفحه حلقوی ۱۰۰۰ عنصری بیان کننده خالی و پر بودن صفحه است؟

(۱) front = (rear - 1) mod 1000 front = 0 , rear = 0 (۱)

(۲) front = 0 , rear = 1000 (خالی) front = 0 , rear = 0 (۲)

(۳) rear = (rear + 1) mod 1000 front = 0 , rear = 1 (۳)

(۴) rear = 999 , front = 0 (خالی) front = 1000 , rear = 1001 (۴)

۲۴۸. ساختار یک صفت پیوندی (نهاش ترتیبی صفت به وسیله لیست پیوندی) اگر front و rear به ترتیب اشاره به بخش‌های آغازین صفت داشته باشد، کدام الگوریتم نمایش اضافه کردن به بک صفت پیوندی است؟

```
procedure add (i , y : integer);
var x : pointer;
begin
new (x);
x ^ . data := x; x ^ . link := y;           (۱)
if front [i] = nil then rear [i] := x
else rear [i] := y;
end;
```

```
procedure add (i , y : integer);
var x : pointer;
begin
new (x);
x ^ . data := y; x ^ . link := nil;          (۲)
if front [i] = nil then front [i] := x
else rear [i] ^ . link := x;
rear [i] := x;
end;
```

```
procedure add (i , y : integer);
var x : pointer;
begin
new (x);
x ^ . data := i; y ^ . link := nil;           (۳)
if front [i] = nil then rear [i] := x
else rear [i] ^ . link := x;
front [i] := x;
end;
```

```
procedure add (i , y : integer );
var x : pointer;
begin
new (x);
y ^ . data := x; x ^ . link := nil;
rear [i] := x;
end;
```

۲۴۹. حداقل تعداد عناصر یک درخت دودویی کامل کدام است؟

4 (۴)

3 (۳)

2 (۲)

1 (۱)

۲۵۰. لیست s که $n = 6$ حرف تشکیل شده به صورت A, B, C, D, E, F می‌باشد، مقایسه‌های لازم برای مرتب کردن s با الگوریتم

عبارت است از:

36 (۲)

(۱) لیست مرتب است و مقایسه‌ای نداریم.

10 (۴)

15 (۳)

۲۵۱. لیست همایش مجاورت مربوط به گراف G را در نظر بگیرید، بیانیش عمقی این گراف عبارتست از:

adj list
A: B, C, D
B: A, D, E
C: A, D, F
D: A, B, C
E: B, F
F: C, E

A D C B E F (۱)

A C F D E B (۲)

A B E F C A (۳)

A B D C F E (۴)

۲۵۲ - حاصل عبارت $- \cdot \cdot \cdot * \uparrow 2 \cdot 3 \cdot - \cdot 1 \cdot 1 \cdot / \cdot 10 \cdot / \cdot \cdot 5 \cdot + \cdot 2 \cdot 3$ کدام است؟

۱۰ (۴)

۱۰*

-۲ (۲)

-۹ (۱)

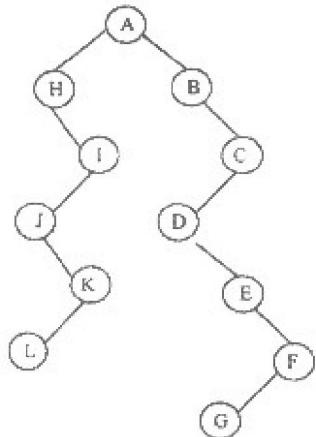
۲۵۳ - پیماش VR درخت دودویی مغایل کدام است؟

A H I J K L B C D E F G (۱)

H J L K I A B D E G F C (۰)

H J L K I A D E C F G (۰)

L K I J H A G F E D C B (۰)



۲۵۴ - روان بازگشتنی مغایل را در نظر بگیرید مطلوب است $M[30, 5]$

 $M(A, B)$
 $P \leftarrow 0$

 while $A < > 0$

 do if $A \bmod 2 = 1$

 then $P \leftarrow P + B$
 $A \leftarrow \lfloor A / 2 \rfloor$
 $B \leftarrow 2B;$

 return p

۱۵۰ (۴)

۷۰ (۰*)

۴۰ (۲)

۲۰ (۱)

۲۵۵ - درخت زیر با روائل فید شده مشخص شده است. آرایه A نمایش درخت فوق به صورت ترتیبی است و

طول T زینه و heapsize طون درخت heap می باشد کدام گزینه صحیح است؟

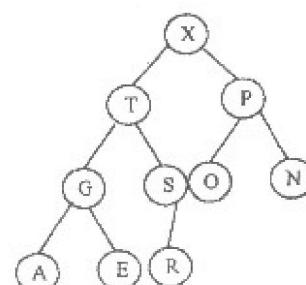
```

parent(A, i)
return (i / 2)
left(A, i)
return 2i
right(A, i)
return (2i + 1)
  
```

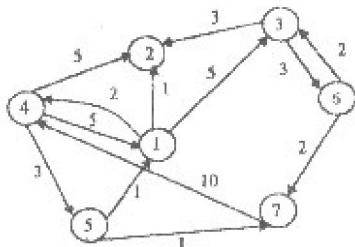
 $\text{length}(A) = 10, \text{heapsize} = 11$ (۰)

 $\text{length}(A) = 11, \text{heapsize} = 11$ (۴)

 $\text{length}(A) = 10, \text{heapsize} = 10$ (۱)

 $\text{length}(A) = 11, \text{heapsize} = 10$ (۰)


۲۵۶ - در گراف زیر کمترین هزینه از گره ۱ به گره ۷ عبارت است از:



8 (۱)

10 (۲)

20 (۳)

6 (۴)

۲۵۷ - شکن بک زیر رشته substring به صورت زیر است:

`substring(string, initial, length)`

که در آن string نام رشته اصلی و initial مکان اولین کاراکتر زیر رشته در رشته اصلی و length طول زیر رشته می‌باشد. متن داده شده T را در نظر بگیرید می‌خواهیم رشته (String) را طوری به آن اضافه کنیم که از مکان K شروع می‌شود insert(t, k, string) دو رشته s₁ و s₂ به شکل مقابل به یکدیگر متصل می‌شوند: s₁ || s₂

$$s_1 = 'ALJ' \\ s_2 = 'reza' \Rightarrow s_1 | s_2 = 'ALIreza'$$

اگر عمل insert به شکل زیر تعریف شود:

$$\text{insert}(T, K, \text{string}) = \text{substring}(T, 1, K-1) | \text{string} | \text{substring}(T, K, \text{length}(T)-K+1)$$

حاصل insert('A B C D E F G', 6, 'x y z') چیست؟

A B C D E F G x y z (۱)

A B C D x y z E F G (۱)

A B C D E x y z F G (۱)

A B C D x y z F G E (۱)

۲۵۸ - فرض کنید Maze یک آرایه سه بعدی است که به صورت (2..8, -4..1, 6..10) تعریف می‌شود، زبان برنامه‌نویسی Maze را به روش سطحی در حافظه ذخیره می‌کند و Base(Maze) = 200 و w = 4 - Base(Maze) = 200 است.

چیست Loc(Maze[5, -1, 8])

628 (۴)

428 (۳)

207 (۲)

200 (۱)

۲۵۹ - در عمل حذف در یک لیست بیوندی با در نظر گرفتن لیست در دسترس حافظه (حاوی خانه‌های خالی لیست) چند آدرس جایگزینی

الهام می‌شود؟

1 (۱)

2 (۲)

3 (۳)

4 (۴)

پاسخ تشریحی

پشته و صف

- ۱ - گزینه ۴ صحیح می‌باشد.
- ۲ - گزینه ۲ صحیح می‌باشد.
- ۳ - گزینه ۳ صحیح می‌باشد.
- ۴ - گزینه ۳ صحیح می‌باشد.
- ۵ - گزینه ۱ صحیح می‌باشد.
- ۶ - گزینه ۳ صحیح می‌باشد.
- ۷ - گزینه ۳ صحیح می‌باشد.
- ۸ - گزینه ۱ صحیح می‌باشد.
- ۹ - گزینه ۱ صحیح می‌باشد.
- ۱۰ - گزینه ۱ صحیح می‌باشد.
- ۱۱ - گزینه ۴ صحیح می‌باشد.
- ۱۲ - گزینه ۲ صحیح می‌باشد.
- ۱۳ - گزینه ۱ صحیح می‌باشد.
- ۱۴ - گزینه ۳ صحیح می‌باشد.
- ۱۵ - گزینه ۲ صحیح می‌باشد.
- ۱۶ - گزینه ۱ صحیح می‌باشد.
- ۱۷ - گزینه ۴ صحیح می‌باشد.
- ۱۸ - گزینه ۳ صحیح می‌باشد.
- ۱۹ - گزینه ۳ صحیح می‌باشد.
- ۲۰ - گزینه ۴ صحیح می‌باشد.
- ۲۱ - گزینه ۲ صحیح می‌باشد.
- ۲۲ - گزینه ۱ صحیح می‌باشد.
- ۲۳ - گزینه ۴ صحیح می‌باشد.
- ۲۴ - گزینه ۴ صحیح می‌باشد.
- ۲۵ - گزینه ۲ صحیح می‌باشد.

- ۲۶ - گزینه ۱ صحیح می باشد.
- ۲۷ - گزینه ۴ صحیح می باشد.
- ۲۸ - گزینه ۲ صحیح می باشد.
- ۲۹ - گزینه ۳ صحیح می باشد.
- ۳۰ - گزینه ۴ صحیح می باشد.
- ۳۱ - گزینه ۴ صحیح می باشد.
- ۳۲ - گزینه ۱ صحیح می باشد.
- ۳۳ - گزینه ۲ صحیح می باشد.
- ۳۴ - گزینه ۳ صحیح می باشد.
- ۳۵ - گزینه ۳ صحیح می باشد.
- ۳۶ - گزینه ۴ صحیح می باشد.
- ۳۷ - گزینه ۳ صحیح می باشد.
- ۳۸ - گزینه ۱ صحیح می باشد.
- ۳۹ - گزینه ۲ صحیح می باشد.
- ۴۰ - گزینه ۱ صحیح می باشد.
- ۴۱ - گزینه ۴ صحیح می باشد.
- ۴۲ - گزینه ۲ صحیح می باشد.
- ۴۳ - گزینه ۴ صحیح می باشد.
- ۴۴ - گزینه ۳ صحیح می باشد.
- ۴۵ - گزینه ۲ صحیح می باشد.
- ۴۶ - گزینه ۴ صحیح می باشد.
- ۴۷ - گزینه ۳ صحیح می باشد.
- ۴۸ - گزینه ۴ صحیح می باشد.
- ۴۹ - گزینه ۴ صحیح می باشد.
- ۵۰ - گزینه ۲ صحیح می باشد.
- ۵۱ - گزینه ۴ صحیح می باشد.
- ۵۲ - گزینه ۴ صحیح می باشد.
- ۵۳ - گزینه ۲ صحیح می باشد.

۵۴. گزینه ۳ صحیح می باشد.
۵۵. گزینه ۲ صحیح می باشد.
۵۶. گزینه ۳ صحیح می باشد.
۵۷. گزینه ۲ صحیح می باشد.
۵۸. گزینه ۳ صحیح می باشد.
۵۹. گزینه ۴ صحیح می باشد.
۶۰. گزینه ۱ صحیح می باشد.
۶۱. گزینه ۱ صحیح می باشد.
۶۲. گزینه ۳ صحیح می باشد.
۶۳. گزینه ۲ صحیح می باشد.
۶۴. گزینه ۲ صحیح می باشد.
۶۵. گزینه ۲ صحیح می باشد.
۶۶. گزینه ۴ صحیح می باشد.
۶۷. گزینه ۲ صحیح می باشد.
۶۸. گزینه ۳ صحیح می باشد.
۶۹. گزینه ۳ صحیح می باشد.
۷۰. گزینه ۱ صحیح می باشد.
۷۱. گزینه ۳ صحیح می باشد.
۷۲. گزینه ۲ صحیح می باشد.
۷۳. گزینه ۴ صحیح می باشد.
۷۴. گزینه ۱ صحیح می باشد.
۷۵. گزینه ۴ صحیح می باشد.
۷۶. گزینه ۴ صحیح می باشد.
۷۷. گزینه ۲ صحیح می باشد.
۷۸. گزینه ۱ صحیح می باشد.
۷۹. گزینه ۳ صحیح می باشد.
۸۰. گزینه ۳ صحیح می باشد.
۸۱. گزینه ۳ صحیح می باشد.

- ۸۲ - گزینه ۱ صحیح می‌باشد.
- ۸۳ - گزینه ۱ صحیح می‌باشد.
- ۸۴ - گزینه ۴ صحیح می‌باشد.
- ۸۵ - گزینه ۲ صحیح می‌باشد.
- ۸۶ - گزینه ۴ صحیح می‌باشد.
- ۸۷ - گزینه ۱ صحیح می‌باشد.
- ۸۸ - گزینه ۲ صحیح می‌باشد.
- ۸۹ - گزینه ۱ صحیح می‌باشد.
- ۹۰ - گزینه ۳ صحیح می‌باشد.

لیست پیوندی

- ۹۱ - گزینه ۳ صحیح می‌باشد.
- ۹۲ - گزینه ۱ صحیح می‌باشد.
- ۹۳ - گزینه ۲ صحیح می‌باشد.
- ۹۴ - گزینه ۳ صحیح می‌باشد.
- ۹۵ - گزینه ۲ صحیح می‌باشد.
- ۹۶ - گزینه ۳ صحیح می‌باشد.
- ۹۷ - گزینه ۲ صحیح می‌باشد.
- ۹۸ - گزینه ۱ صحیح می‌باشد.
- ۹۹ - گزینه ۳ صحیح می‌باشد.
- ۱۰۰ - گزینه ۳ صحیح می‌باشد.
- ۱۰۱ - گزینه ۱ صحیح می‌باشد.
- ۱۰۲ - گزینه ۴ صحیح می‌باشد.
- ۱۰۳ - گزینه ۴ صحیح می‌باشد.
- ۱۰۴ - گزینه ۱ صحیح می‌باشد.
- ۱۰۵ - گزینه ۳ صحیح می‌باشد.
- ۱۰۶ - گزینه ۲ صحیح می‌باشد.
- ۱۰۷ - گزینه ۳ صحیح می‌باشد.

- ۱۰۸ - گزینه ۲ صحیح می باشد.
- ۱۰۹ - گزینه ۴ صحیح می باشد.
- ۱۱۰ - گزینه ۲ صحیح می باشد.
- ۱۱۱ - گزینه ۴ صحیح می باشد.
- ۱۱۲ - گزینه ۳ صحیح می باشد.
- ۱۱۳ - گزینه ۱ صحیح می باشد.
- ۱۱۴ - گزینه ۲ صحیح می باشد.
- ۱۱۵ - گزینه ۱ صحیح می باشد.
- ۱۱۶ - گزینه ۳ صحیح می باشد.
- ۱۱۷ - گزینه ۲ صحیح می باشد.
- ۱۱۸ - گزینه ۳ صحیح می باشد.
- ۱۱۹ - گزینه ۳ صحیح می باشد.
- ۱۲۰ - گزینه ۴ صحیح می باشد.
- ۱۲۱ - گزینه ۲ صحیح می باشد.
- ۱۲۲ - گزینه ۳ صحیح می باشد.
- ۱۲۳ - گزینه ۲ صحیح می باشد.
- ۱۲۴ - گزینه ۴ صحیح می باشد.
- ۱۲۵ - گزینه ۳ صحیح می باشد.
- ۱۲۶ - گزینه ۲ صحیح می باشد.
- ۱۲۷ - گزینه ۳ صحیح می باشد.
- ۱۲۸ - گزینه ۲ صحیح می باشد.
- ۱۲۹ - گزینه ۴ صحیح می باشد.
- ۱۳۰ - گزینه ۲ صحیح می باشد.
- ۱۳۱ - گزینه ۱ صحیح می باشد.
- ۱۳۲ - گزینه ۳ صحیح می باشد.
- ۱۳۳ - گزینه ۴ صحیح می باشد.
- ۱۳۴ - گزینه ۱ صحیح می باشد.
- ۱۳۵ - گزینه ۳ صحیح می باشد.

۱۳۶ - گزینه ۴ صحیح می‌باشد.

۱۳۷ - گزینه ۳ صحیح می‌باشد.

۱۳۸ - گزینه ۳ صحیح می‌باشد.

درخت

۱۳۹ - گزینه ۳ صحیح می‌باشد.

۱۴۰ - گزینه ۴ صحیح می‌باشد.

۱۴۱ - گزینه ۳ صحیح می‌باشد.

۱۴۲ - گزینه ۱ صحیح می‌باشد.

۱۴۳ - گزینه ۳ صحیح می‌باشد.

۱۴۴ - گزینه ۲ صحیح می‌باشد.

۱۴۵ - گزینه ۲ صحیح می‌باشد.

۱۴۶ - گزینه ۴ صحیح می‌باشد.

۱۴۷ - گزینه ۱ صحیح می‌باشد.

۱۴۸ - گزینه ۲ صحیح می‌باشد.

۱۴۹ - گزینه ۱ صحیح می‌باشد.

۱۵۰ - گزینه ۴ صحیح می‌باشد.

۱۵۱ - گزینه ۲ صحیح می‌باشد.

۱۵۲ - گزینه ۳ صحیح می‌باشد.

۱۵۳ - گزینه ۱ صحیح می‌باشد.

۱۵۴ - گزینه ۱ صحیح می‌باشد.

۱۵۵ - گزینه ۴ صحیح می‌باشد.

۱۵۶ - گزینه ۲ صحیح می‌باشد.

۱۵۷ - گزینه ۳ صحیح می‌باشد.

۱۵۸ - گزینه ۴ صحیح می‌باشد.

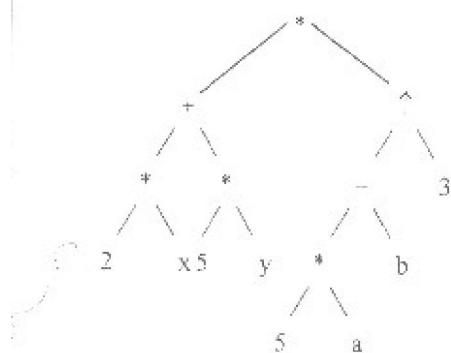
۱۵۹ - گزینه ۲ صحیح می‌باشد.

۱۶۰ - گزینه ۲ صحیح می‌باشد.

۱۶۱ - گزینه ۱ صحیح می‌باشد.

- ۱۶۲ - گزینه ۴ صحیح می باشد.
- ۱۶۳ - گزینه ۲ صحیح می باشد.
- ۱۶۴ - گزینه ۴ صحیح می باشد.
- ۱۶۵ - گزینه ۳ صحیح می باشد.
- ۱۶۶ - گزینه ۳ صحیح می باشد.
- ۱۶۷ - گزینه ۲ صحیح می باشد.
- ۱۶۸ - گزینه ۳ صحیح می باشد.
- ۱۶۹ - گزینه ۱ صحیح می باشد.
- ۱۷۰ - گزینه ۱ صحیح می باشد.
- ۱۷۱ - گزینه ۲ صحیح می باشد.
- ۱۷۲ - گزینه ۴ صحیح می باشد.
- ۱۷۳ - گزینه ۱ صحیح می باشد.
- ۱۷۴ - گزینه ۳ صحیح می باشد.
- ۱۷۵ - گزینه ۳ صحیح می باشد.

شکل صحیح سؤال به صورت زیر بوده است:



- ۱۷۶ - گزینه ۱ صحیح می باشد.
- ۱۷۷ - گزینه ۱ صحیح می باشد.
- ۱۷۸ - گزینه ۲ صحیح می باشد.
- ۱۷۹ - گزینه ۱ صحیح می باشد.
- ۱۸۰ - گزینه ۱ صحیح می باشد.
- ۱۸۱ - گزینه ۳ صحیح می باشد.
- ۱۸۲ - گزینه ۲ صحیح می باشد.

۱۸۷	- گزینه ۳	صحیح می باشد.
۱۸۸	- گزینه ۴	صحیح می باشد.
۱۸۹	- گزینه ۴	صحیح می باشد.
۱۹۰	- گزینه ۱	صحیح می باشد.
۱۹۱	- گزینه ۲	صحیح می باشد.
۱۹۲	- گزینه ۱	صحیح می باشد.
۱۹۳	- گزینه ۳	صحیح می باشد.
۱۹۴	- گزینه ۱	صحیح می باشد.
۱۹۵	- گزینه ۱	صحیح می باشد.
۱۹۶	- گزینه ۴	صحیح می باشد.
۱۹۷	- گزینه ۲	صحیح می باشد.
۱۹۸	- گزینه ۳	صحیح می باشد.
۱۹۹	- گزینه ۱	صحیح می باشد.
۲۰۰	- گزینه ۴	صحیح می باشد.
۲۰۱	- گزینه ۳	صحیح می باشد.
۲۰۲	- گزینه ۳	صحیح می باشد.
۲۰۳	- گزینه ۲	صحیح می باشد.
۲۰۴	- گزینه ۱	صحیح می باشد.
۲۰۵	- گزینه ۲	صحیح می باشد.
۲۰۶	- گزینه ۱	صحیح می باشد.
۲۰۷	- گزینه ۴	صحیح می باشد.
۲۰۸	- گزینه ۴	صحیح می باشد.
۲۰۹	- گزینه ۴	صحیح می باشد.
۲۱۰	- گزینه ۳	صحیح می باشد.

- ۲۱۱ - گزینه ۲ صحیح می‌باشد.
- ۲۱۲ - گزینه ۳ صحیح می‌باشد.
- ۲۱۳ - گزینه ۱ صحیح می‌باشد.
- ۲۱۴ - گزینه ۳ صحیح می‌باشد.
- ۲۱۵ - گزینه ۲ صحیح می‌باشد.
- ۲۱۶ - گزینه ۳ صحیح می‌باشد.
- ۲۱۷ - گزینه ۲ صحیح می‌باشد.
- ۲۱۸ - گزینه ۱ صحیح می‌باشد.
- ۲۱۹ - گزینه ۴ صحیح می‌باشد.
- ۲۲۰ - گزینه ۲ صحیح می‌باشد.
- ۲۲۱ - گزینه ۱ صحیح می‌باشد.
- ۲۲۲ - گزینه ۴ صحیح می‌باشد.
- ۲۲۳ - گزینه ۱ صحیح می‌باشد.
- ۲۲۴ - گزینه ۴ صحیح می‌باشد.
- ۲۲۵ - گزینه ۱ صحیح می‌باشد.
- ۲۲۶ - گزینه ۲ صحیح می‌باشد.
- ۲۲۷ - گزینه ۱ صحیح می‌باشد.
- ۲۲۸ - گزینه ۱ صحیح می‌باشد.
- ۲۲۹ - گزینه ۲ صحیح می‌باشد.
- ۲۳۰ - گزینه ۲ صحیح می‌باشد.
- ۲۳۱ - گزینه ۲ صحیح می‌باشد.
- ۲۳۲ - گزینه ۳ صحیح می‌باشد.
- ۲۳۳ - گزینه ۱ صحیح می‌باشد.
- ۲۳۴ - گزینه ۴ صحیح می‌باشد.
- ۲۳۵ - گزینه ۴ صحیح می‌باشد.
- ۲۳۶ - گزینه ۲ صحیح می‌باشد.
- ۲۳۷ - گزینه ۲ صحیح می‌باشد.
- ۲۳۸ - گزینه ۳ صحیح می‌باشد.

۲۳۹ - گزینه ۴ صحیح می‌باشد.

۲۴۰ - گزینه ۴ صحیح می‌باشد.

۲۴۱ - گزینه ۲ صحیح می‌باشد.

۲۴۲ - گزینه ۳ صحیح می‌باشد.

۲۴۳ - گزینه ۲ صحیح می‌باشد.

۲۴۴ - گزینه ۱ صحیح می‌باشد.

پاسخ تشریحی دولتی ۸۳

۲۴۵ - گزینه ۴ صحیح می‌باشد.

۲۴۶ - گزینه ۴ صحیح می‌باشد.

۲۴۷ - گزینه ۱ صحیح می‌باشد.

۲۴۸ - گزینه ۴ صحیح می‌باشد.

۲۴۹ - گزینه ۱ صحیح می‌باشد.

۲۵۰ - گزینه ۳ صحیح می‌باشد.

۲۵۱ - گزینه ۱ صحیح می‌باشد.

۲۵۲ - گزینه ۲ و ۴ صحیح می‌باشد.

۲۵۳ - گزینه ۲ صحیح می‌باشد.

۲۵۴ - گزینه ۴ صحیح می‌باشد.

A	B	P	$A < > 0$
30	5	0	TRUE
15	10	10	TRUE
7	20	30	TRUE
3	40	70	TRUE
1	80	150	TRUE
0	160		FALSE

(۲۴۵) صد و سهاده ایشتمنه است.

۲۵۸ - گزینه ۴ صحیح می‌باشد.

۲۵۹ - گزینه ۲ صحیح می‌باشد.

پاسخ تشریحی ۵۰ اول

آرایه

۱. گزینه ۴ صحیح می باشد.
۲. گزینه ۲ صحیح می باشد.
۳. گزینه ۱ صحیح می باشد.
۴. گزینه ۲ صحیح می باشد.
۵. گزینه ۲ صحیح می باشد.
۶. گزینه ۳ صحیح می باشد.
۷. گزینه ۲ صحیح می باشد.
۸. گزینه ۲ صحیح می باشد.
۹. گزینه ۳ صحیح می باشد.
۱۰. گزینه ۳ صحیح می باشد.
۱۱. گزینه ۳ صحیح می باشد.
۱۲. گزینه ۲ صحیح می باشد.
۱۳. گزینه ۳ صحیح می باشد.
۱۴. گزینه ۲ صحیح می باشد.
۱۵. گزینه ۱ صحیح می باشد.
۱۶. گزینه ۲ صحیح می باشد.
۱۷. گزینه ۲ صحیح می باشد.
۱۸. گزینه ۱ صحیح می باشد.
۱۹. گزینه ۳ صحیح می باشد.
۲۰. گزینه ۱ صحیح می باشد.
۲۱. گزینه ۲ صحیح می باشد.
۲۲. گزینه ۱ صحیح می باشد.
۲۳. گزینه ۲ صحیح می باشد.
۲۴. گزینه ۱ صحیح می باشد.
۲۵. گزینه ۴ صحیح می باشد.



- ۲۶. گزینه ۳ صحیح می باشد.
- ۲۷. گزینه ۴ صحیح می باشد.
- ۲۸. گزینه ۳ صحیح می باشد.
- ۲۹. گزینه ۴ صحیح می باشد.
- ۳۰. گزینه ۳ صحیح می باشد.
- ۳۱. گزینه ۲ صحیح می باشد.
- ۳۲. گزینه ۳ صحیح می باشد.
- ۳۳. گزینه ۲ صحیح می باشد.
- ۳۴. گزینه ۴ صحیح می باشد.
- ۳۵. گزینه ۱ صحیح می باشد.
- ۳۶. گزینه ۲ صحیح می باشد.

توابع بازگشتی

- ۳۷. گزینه ۳ صحیح می باشد.
- ۳۸. گزینه ۱ صحیح می باشد.
- ۳۹. گزینه ۲ صحیح می باشد.
- ۴۰. گزینه ۴ صحیح می باشد.
- ۴۱. گزینه ۳ صحیح می باشد.
- ۴۲. گزینه ۱ صحیح می باشد.
- ۴۳. گزینه ۳ صحیح می باشد.
- ۴۴. گزینه ۳ صحیح می باشد.
- ۴۵. گزینه ۲ صحیح می باشد.
- ۴۶. گزینه ۲ صحیح می باشد.
- ۴۷. گزینه ۳ صحیح می باشد.
- ۴۸. گزینه ۲ صحیح می باشد.
- ۴۹. گزینه ۱ صحیح می باشد.
- ۵۰. گزینه ۲ صحیح می باشد.
- ۵۱. گزینه ۱ صحیح می باشد.