بنام حق

ساختمان داده (data-structure)

مرجع :

کتاب حمیدرضا مقسمی

بنام حق

فصل اول   پیچیدگی زمانی و مرتبه اجرا

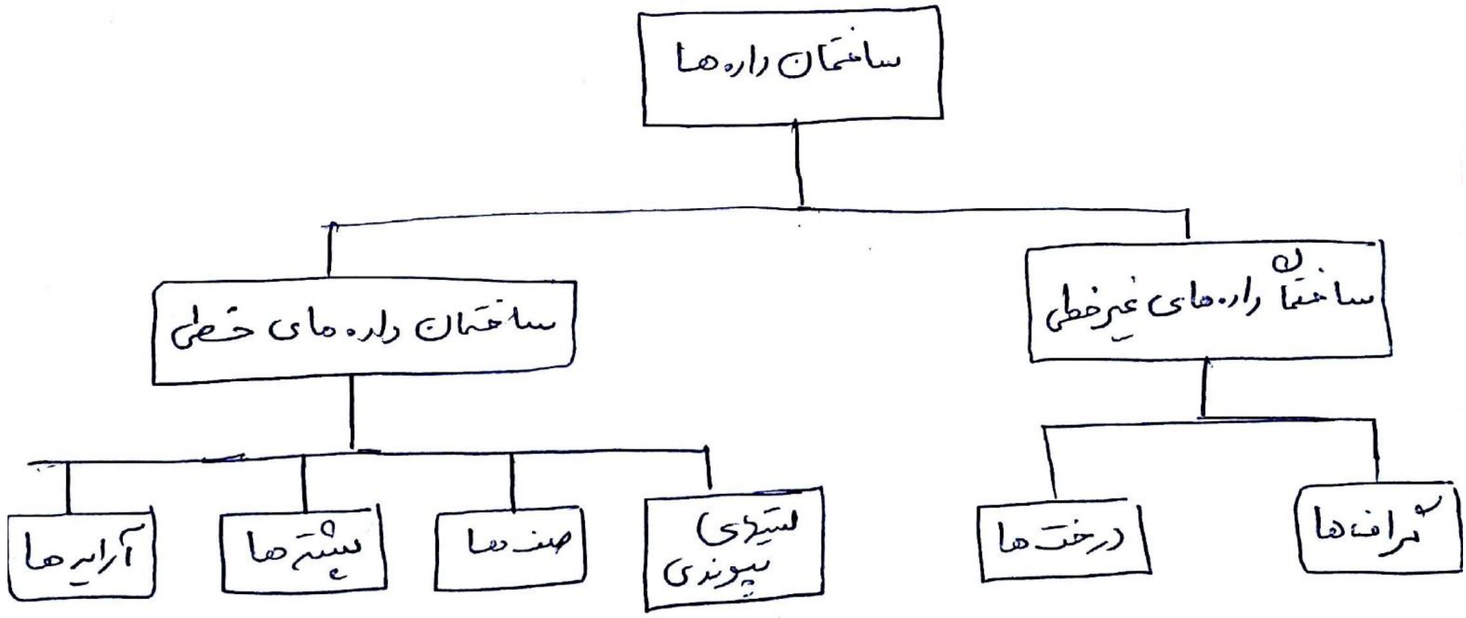
این درس درباره چیست ؟

۱- حل مساله و برنامه نویسی به همراه مثالهای کاربردی

2- الگوریتم : روشی برای حل مساله که دارای خصوصیات زیر است

۱- دقیق   2- با جزئیات کامل نوشته شود   3- پایان پذیر

3- ساختمان داده : روشی برای سازمان دهی داده ها و ذخیره اطلاعات در حافظه.



دسته بندی ساختمان داده ها

پیچیدگی زمانی و مرتبه اجرا

مثال زیر را در نظر بگیرید:

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

روش اول

$$\overset{30}{x} \leftarrow \overset{10}{x} + \overset{20}{y}$$
$$\overset{10}{y} \leftarrow \overset{30}{x} - \overset{20}{y}$$
$$\overset{20}{x} \leftarrow \overset{30}{x} - \overset{10}{y}$$

خروجی $\downarrow$

$(x=20,\ y=10)$

روش دوم

$$\overset{10}{t} \leftarrow \overset{10}{x}$$
$$\overset{20}{x} \leftarrow \overset{20}{y}$$
$$\overset{10}{y} \leftarrow \overset{10}{t}$$

خروجی $\downarrow$

$(x=20,\ y=10)$

روش سوم

$$\overset{200}{x} \leftarrow \overset{10}{x} * \overset{20}{y}$$
$$\overset{10}{y} \leftarrow \overset{200}{x} / \overset{20}{y}$$
$$\overset{20}{x} \leftarrow \overset{200}{x} / \overset{10}{y}$$

خروجی $\downarrow$

$(x=20,\ y=10)$

مثال بالا نشان می دهد که روش حل یک مسئله ممکن است منحصر به فرد نباشد.

سوال: معیار برتری یک الگوریتم نسبت به الگوریتمهای دیگر چیست؟

دو معیاری که غالبا برای ارزیابی الگوریتمها مورد استفاده قرار می گیرند:

1- زمان اجرا

2- حافظه مورد نیاز

معیار زمان اجرا وابسته به ماشینی است که الگوریتم بر روی آن اجرا می‌شود.

معیار حافظه مورد نیاز نیز به روش مدیریت حافظه توسط سیستم عامل بستگی دارد.

پس اگر بخواهیم دو الگوریتم را با همدیگر مقایسه کنیم باید شرایط کاملاً یکسانی برای آنها فراهم نماییم.

سوال: اگر الگوریتمها توسط افراد متفاوت، در مکانهای متفاوت و با امکانات متفاوت نوشته شوند، چگونه می‌توان شرایط یکسانی فراهم کرد؟

جواب: باید معیارهای فوق را به گونه‌ای تعریف کرد که مستقل از ماشین و سیستم عامل باشند. دو معیار زیر برای این منظور تعریف شده‌اند:

1- مرتبه زمانی اجرای الگوریتم ← (رشد زمان اجرای الگوریتم به ازای سایزهای ورودی متفاوت)

2- مرتبه مکانی اجرای الگوریتم

پس ما می‌توانیم زمان اجرای یک برنامه را بصورت تابعی از مشخصه‌های مهم داده ورودی آن بیان کنیم.

(در ادامه رشد زمان اجرای الگوریتم به ازای سایزهای ورودی متفاوت و مرتبه‌ی پیچیدگی الگوریتم‌ها را مورد بحث قرار می‌دهیم)

مثال: برنامه‌ای می‌نویسید که از ورودی یک آرایه را بگیرد و ...

این آرایه می‌تواند 3 عضوی باشد یا 50 عضوی یا 1000 عضوی

قطعاً واضح است که سرعت اجرای برنامه برای هر ورودی با سایزهای متفاوت متفاوت خواهد بود. بنابراین برای بررسی یک الگوریتم، تابع زمانی —

الگوریتم $T(n)$ را در نظر می‌گیریم. (تابع پیچیدگی)

مثال: برنامه‌ای بنویسید که از ورودی یک ماتریس را بگیرد و ...

$T(n, m)$ تابع زمانی الگوریتم

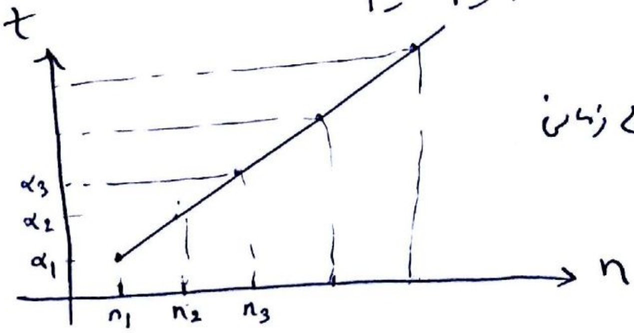
n: تعداد سطرها

m: تعداد ستونها

مثال: کد زیر را در نظر بگیرید:

```
int sum(int ar[], int n)
{
    int s=0;
    for(int i=0; i<n; i++)
        s+=ar[i];
    return s;
}
```

که به ازای سایزهای متفاوت نمودار را رسم کنیم داریم:



تابع زمانی $T(n) = an + b$

$a, b \in R$

زمان اجرای تابع sum براساس ورودی‌ها با سایز متفاوت

ممکن است یک الگوریتمی داشته باشیم که تابع زمانی (تابع پیچیدگی) آن بصورت $T(n) = an^2 + bn + c$ (درجه دو) یا بصورت درجه 3 و ... یا بصورت $T(n) = C$ باشد.

← زمان اجرا به n وابسته نیست.

مثال زیر را در نظر بگیرید:

```
1) x=0;
2) for (i=0; i<n; i++)
3) x++;
```

سطر	زمان	تعداد
1	c_1	1
2	c_2	$n+1$
3	c_3	n

$$\Rightarrow T(n) = c_1 + c_2(n+1) + c_3 n$$

حال c را بیشترین مقدار c_1، c_2، c_3 در نظر می‌گیریم. بنابراین:

$$T(n) = c(2n+2)$$

```
1) x=0;
2) for (i=0; i<n; i++)
3)     for (j=0; j<n; j++)
4)         x++;
```

سطر	زمان	تعداد
1	c_1	1
2	c_2	$n+1$
3	c_3	$n(n+1)$
4	c_4	$n \times n$

$$\Rightarrow T(n) = c(2n^2 + 2n + 2)$$

حال برای آنکه به اینها نوشتار ریاضی دهیم از نمادهای زیر استفاده می کنیم .

۱- نماد Θ (تتا)

2 - نماد O (اُ بزرگ)

3- نماد Ω (امگا بزرگ)

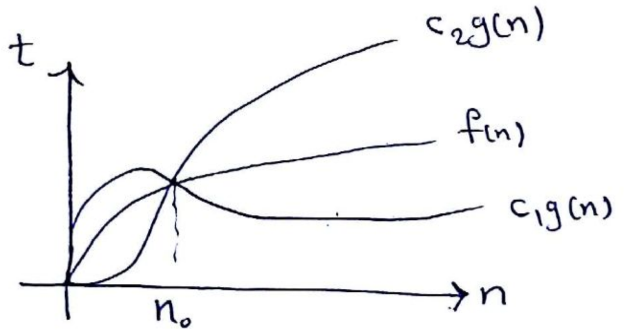
4) نماد o (اُ کوچک)

5) نماد ω (امگا کوچک)

انواع توابع رشد :

1) $\Theta(g(n)) = \{ f(n) : \exists c_1, c_2, n_0 \text{ s.t } \forall n \geq n_0$

$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \}$



آیا رابطه زیر برقرار است ؟

$$\underbrace{10n^2 + 8n + 5}_{f(n)} \in \Theta(\underbrace{n^2}_{g(n)})$$

$c_1 = 1,\ c_2 = 100,\ n_0 = 1 \Rightarrow \forall n \geq 1 \quad 0 \leq n^2 \leq 10n^2 + 8n + 5 \leq 100n^2$ ✓

* اگر $f(n) \in \Theta(g(n))$ آنگاه می گوییم : آهنگ رشد $f(n)$ و $g(n)$ برای مقادیر بزرگ n یکسان هست .

قضیه: اگر $T(n)=a_m n^m + a_{m-1} n^{m-1} + \dots + a_1 n + a_0$ زمان اجرای یک الگوریتم باشد و $a_m > 0$ آنگاه $T(n) \in \Theta(n^m)$

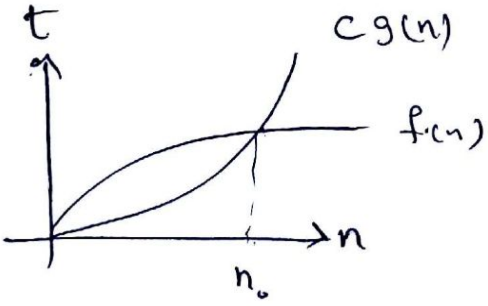
مثال

- $\frac{1}{100} n^4 + 100 n^3 - 1 \in \Theta(n^4)$

ضریب هیچ تاثیری ندارد.

- $n^2 \log n + n^2 \in \Theta(n^2 \log n)$
- $6n + 4 \in \Theta(n)$

2) $O(g(n)) = \{ f(n) : \exists c , n_0 \ s.t. \ \forall n \geq n_0 \quad 0 \leq f(n) \leq c g(n) \}$.



* اگر $f(n) \in O(g(n))$ آنگاه داریم:

آهنگ رشد $g(n)$ برای مقادیر بزرگ n بیشتر یا مساوی $f(n)$ است.

($g(n)$ یک کران بالا برای $f(n)$ است)

آیا رابطه زیر برقرار است؟

$$\underbrace{10n^2 + 5n - 1}_{f(n)} \in O(\underbrace{n^3}_{g(n)})$$

$c = 100$ $\quad \forall n \geq 1 \quad 0 \leq 10n^2 + 5n - 1 \leq 100 n^3$ ✓

$n_0 = 1$

آیا روابط زیر برقرارند؟

$10n^2+5n-1 \in \Theta(n^2\log n)$ ✓

$10n^2+5n-1 \in O(n^2)$ ✓

$10n^2+5n-1 \in O\left(\frac{n^3}{\log n}\right)$ ✓

$10n^2+5n-1 \in O\left(\frac{n^2}{\log n}\right)$ X

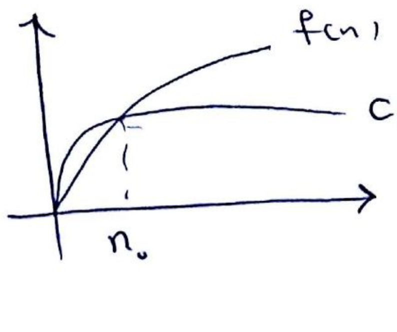
$10n^2+5n-1 \in O(n)$ X

$\frac{n^3}{\log n}$ و n^2 و $n^2\log n$ کران های بالا برای $10n^2+5n-1$ هستند.

با اینکه عبارات (1) و (3) صحیح هستند ولی بهتر است مورد استفاده قرار نگیرند چون هدف از بیان پیچیدگی با نماد O، نشان دادن کوچکترین کران بالا مرتبه پیچیدگی است.

3) $\Omega(g(n)) = \{ f(n) : \exists c, n_0 \text{ s.t. } \forall n \geq n_0$

$0 \leq cg(n) \leq f(n) \}$



* اگر $f(n) \in \Omega(g(n))$ آنگاه:

آهنگ رشد $g(n)$ برای مقادیر بزرگ n کوچکتر یا مساوی $f(n)$ است

($g(n)$ یک کران پایین برای $f(n)$ است)

مثال :

$$10n^2+5n-1 \in \Omega(n) \quad (1)$$

$$\in \Omega(n^2) \quad (2)$$

$$\in \Omega(n\log n) \quad (3)$$

$$\in \Omega\left(\frac{n^2}{\log n}\right) \quad (4)$$

$$\in \Omega(\log n) \quad (5)$$

$$10n^2+5n-1 \notin \Omega(n^3)$$

$$\notin \Omega(n^4)$$

$$\notin \Omega(n^2\log n)$$

$$\notin \Omega\left(\frac{n^3}{\log n}\right)$$

با این که عبارت‌های 1، 3، 4 و 5 صحیح هستند ولی بهتر است مورد استفاده قرار نگیرند. چون هدف از بیان پیچیدگی با امگا، نشان دادن بزرگترین کران پایین مرتبه پیچیدگی است.

در مثال بالا واضح است که n, n^2, $n\log n$, $\frac{n^2}{\log n}$, $\log n$ کران‌های پایین برای $10n^2+5n-1$ هستند ولی بزرگترین کران پایین n^2 می‌باشد.

قضیه: اگر $T(n)=a_mn^m+\dots+a_0$ و $a_m>0$، آنگاه

$$T(n)\in\Omega(n^m) \quad , \quad T(n)\in\Theta(n^m)$$

جمع‌بندی از سه آهنگ رشد :

$f(n) = \Theta(g(n))$: آهنگ رشد $f(n)$ = آهنگ رشد $g(n)$

$f(n) = O(g(n))$: آهنگ رشد $f(n)$ $\le$ آهنگ رشد $g(n)$

$f(n) = \Omega(g(n))$: آهنگ رشد $f(n)$ $\ge$ آهنگ رشد $g(n)$

سوال : $\Theta(g(n))$ و $O(g(n))$ چه نسبتی با هم دارند.

let $g(n) = n^2$

$$\Theta(g(n)) = \{n^2, 100n^2, \frac{1}{10}n^2, \ldots\}$$

بزرگ

$$O(g(n)) = \{n^2, 100n^2, \frac{1}{10}n^2, \ldots, n, \log n, \ldots\}$$

$$= \{\Theta(g(n)), n, \log n, \ldots\}$$

$$\Rightarrow \boxed{\Theta(g(n)) \subset O(g(n))}$$

$$\Omega(g(n)) = \{n^2, 100n^2, \frac{1}{10}n^2, \ldots, n^3, n^2\log n, \ldots\}$$

$$= \{\Theta(g(n)), n^3, n^2\log n, \ldots\}$$

$$\Rightarrow \boxed{\Theta(g(n)) \subset \Omega(g(n))}$$

او کوچک

4) $o(g(n)) = \{ f(n) : \exists\, c, n_0 \text{ s.t. } \forall n \geq n_0 \quad 0 \leq f(n) < c\,g(n) \}$

$f(n) \in o(g(n))$ $\Rightarrow$ آهنگ رشد $g(n)$ برای مقادیر بزرگ n اکیداً بزرگتر از $f(n)$ است

5) $\omega(g(n)) = \{ f(n) : \exists\, c, n_0 \text{ s.t. } \forall n \geq n_0 \quad 0 \leq c\,g(n) < f(n) \}$

$f(n) \in \omega(g(n))$ $\Rightarrow$ آهنگ رشد $g(n)$ برای مقادیر بزرگ n اکیداً کوچکتر از $f(n)$ است

نکته 1) $f(n) \overset{=}{\in} O(g(n))$ & $f(n) \overset{=}{\in} \Omega(g(n))$ $\Longleftrightarrow$

$f(n) \overset{=}{\in} \Theta(g(n))$

نکته 2) اگر $f(n) \overset{\in}{=} o(g(n))$ داریم:

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$$

نکته 3) اگر $f(n) = \omega(g(n))$ داریم:

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = \infty$$

نکته 4) $f(n) = o(g(n))$ اگر و فقط اگر $g(n) = \omega(f(n))$

نکته 5) $f(n) = O(g(n))$ اگر و فقط اگر $g(n) = \Omega(f(n))$

نکته 6) $f(n) = \Theta(g(n))$ اگر و فقط اگر $g(n) = \Theta(f(n))$

نکته 7) خاصیت تعدی :

$$f(n)=\Theta(g(n)) \quad و \quad g(n)=\Theta(h(n)) \implies f(n)=\Theta(h(n)) \checkmark$$

$$f(n)=O(g(n)) \quad , \quad g(n)=O(h(n)) \implies f(n)=O(h(n)) \checkmark$$

به همین ترتیب برای بقیه ی نمادها هم این خاصیت برقرار است .

نکته 8) خاصیت بازتابی (انعکاسی) :

$$f(n)=\Theta(f(n))$$

$$f(n)=O(f(n))$$

$$f(n)=\Omega(f(n))$$

نکته 9) خاصیت تقارن :

$$f(n)=\Theta(g(n)) \implies g(n)=\Theta(f(n))$$

ترتیب پیچیدگی های زمانی :

$$\Theta(\log n) < \Theta(n) < \Theta(n\log n) < \Theta(n^2) < \Theta(n^j) < \quad k>j,\ k,j>2$$

$$\Theta(n^k) < \Theta(a^n) < \Theta(b^n) < \Theta(n!) < \Theta(n^n)$$

$$a<b$$

متوجه شویم که برای بدست آوردن مرتبه پیچیدگی $T(n)$ دو قانون داریم :

1- در بین جملات ، جمله ای با بیشترین رشد را پیدا می کنیم .

2- ضریب جمله را حذف می کنیم .

$$\text{let } T(n)=\underbrace{3n}_{\text{بیشترین رشد}}+5 \implies \Theta(n)$$

$$\text{let } T(n)=\underbrace{5n^2}_{\text{بیشترین رشد}}+10n+3 \implies \Theta(n^2)$$

کدام یک از گزاره‌های زیر درست است .

1) $n! = \overset{\text{بزرگ}}{O}(n^n)$

2) $10^n + n^{20} = \Theta(n^n)$

3) $\dfrac{n^2}{\log n} = \Theta(n^2)$

4) $(\log n)! = \Omega(n!)$

5) $(n+1)(n^2 - 2n + 1) = \omega(n^3)$

6) $(0.9)^n = O(n^2)$

7) $2^n = O(n^2)$

8) $n^2 \times 2^n = O(3^n)$

9) $\log_7 n = \Theta(\log_5 n)$

10) $\sqrt{n} = \omega(\log n)$

11) $\log n = O(\sqrt{n})$

12) $\log(n!) = \Theta(n \log n)$

توجه: برای تابع ثابت $T(n) = c$ داریم $\Theta(1)$ (دارای مرتبه زمانی $O(1)$ است)

$T(n) = \underset{\text{ضریب}}{\underset{\downarrow}{c}} \times 1$

Scanned by CamScanner

جواب:

مورد 1 درست است. چون رشد n^n بزرگتر از رشد $n!$ است

مورد 2 غلط است.

مورد 3 درست نیست. چون رشد n^2 بزرگتر از رشد $\frac{n^2}{\log n}$ است.

مورد 4 درست نیست. چون رشد $n!$ بزرگتر از رشد $(\log n)!$ است

مورد 5 درست نیست. چون رشد n^3 برابر است با رشد $\underbrace{(n+1)(n^2-2n+1)}$

$\underbrace{n^3}-n^2-n+1$

رشد بیشتری دارد

مورد 6 - درست است. چون 0.9 کوچکتر از 1 است و $(0.9)^n$ نزولی است پس برای مقادیر بزرگ n، رشد n^2 بزرگتر از رشد $(0.9)^n$ است.

مورد 7 ولی داریم: اگر $a>1$ آنگاه $\longleftarrow$ رشد a^n $>$ رشد n^b

مثال $n^3 = O(2^n) \longrightarrow$ یعنی رشد 2^n $>$ رشد n^3

مورد 7 - نادرست است بنا به توضیح بالا.

مورد 8 درست است.

برای مقایسه رشد توابع می‌توان از حدگیری استفاده کرد.

$$\lim_{n\to\infty} \frac{n^2 \times 2^n}{3^n} = \frac{\infty}{\infty} \xrightarrow{H} \lim \frac{2n \times 2^n + n^2 \times 2^n \ln 2}{3^n \ln 3} = \frac{\infty}{\infty}$$

اگر به همین ترتیب ادامه دهیم رفع ابهام نمی‌شود پس صورت و مخرج را بر 2^n تقسیم می‌کنیم (یا ضرب در 2^{-n})

$$\lim_{n\to\infty} \frac{n^2 \times 2^{-n} \cdot 2^n}{3^n \times 2^{-n}} = \lim_{n\to\infty} \frac{n^2}{\frac{3^n}{2^n}} = \lim_{n\to\infty} \frac{n^2}{(\frac{3}{2})^n} \longrightarrow$$

ادامه در صفحه بعد

$$\lim_{n\to\infty} \frac{n^2}{(\frac{3}{2})^n} = \frac{\infty}{\infty} \xrightarrow{H} \lim_{n\to\infty} \frac{2n}{(\frac{3}{2})^n \ln\frac{3}{2}} = \frac{\infty}{\infty} \xrightarrow{H}$$

$$\lim_{n\to\infty} \frac{2}{\ln\frac{3}{2}\cdot(\frac{3}{2})^n \ln\frac{3}{2}} = 0$$

پس رشد 3^n بیشتر است.

نتیجه گیری: رشد $a^n >$ رشد $n^c \times b^n$ $\quad a>b$

مورد 9 درست است.

$$\lim_{n\to\infty} \frac{\log_5^n}{\log_7^n} = \frac{\infty}{\infty} \xrightarrow{H} \lim_{n\to\infty} \frac{\frac{1}{n\ln5}}{\frac{1}{n\ln7}} = \frac{\ln7}{\ln5} = \text{عدد}$$

پس $\log_5^n$ و $\log_7^n$ رشد یکسانی دارند. این نشان می‌دهد که در رشد توابع مبنای log مهم نیست.

مورد 10 درست است.

$$\lim_{n\to\infty} \frac{\sqrt{n}}{\log n} = \frac{\infty}{\infty} \xrightarrow{H} \lim_{n\to\infty} \frac{\frac{1}{2\sqrt{n}}}{\frac{1}{n}} = \lim_{n\to\infty} \frac{n}{2\sqrt{n}} = \lim_{n\to\infty} \frac{\sqrt{n}\sqrt{n}}{2\sqrt{n}} = \infty$$

رشد $\sqrt{n}$ $>$ رشد $\log(n)$

مورد 11 درست است. بنابه توضیحات مورد ۱۰.

مورد 12 درست است . برای اثبات مورد 12 باید ثابت کنیم که

$$\underset{①}{\log(n!) = O(n\log n)} \quad و \quad \underset{②}{\log(n!) = \Omega(n\log n)}$$

اثبات ①

$$\log(n!) = \log(n \times (n-1) \times (n-2) \times \cdots \times 1) =$$

$$\log n + \log(n-1) + \cdots + \log 1 <$$

$$\log n + \log n + \cdots + \log n = n\log n$$

$\Rightarrow$ رشد $\log(n!)$ < رشد $n\log n$

$$\Rightarrow \underbrace{\log(n!) = O(n\log n)}_{*} \checkmark$$

اثبات ②

$$\log(n!) = \log n + \log(n-1) + \cdots + \log\frac{n}{2} + \underbrace{\cdots + \log 1}_{} >$$

$$\log\frac{n}{2} + \log\frac{n}{2} + \cdots + \log\frac{n}{2} =$$

$$\frac{n}{2}\log\frac{n}{2} = \frac{n}{2}(\log n - \cancel{\log 2}^{\,1}) =$$

$$\underbrace{\frac{n}{2}\log n}_{\text{رشدش بیشتر}} - \frac{n}{2}$$

$$\Rightarrow \underbrace{\log(n!) = \Omega(n\log n)}_{**} \checkmark$$

لذا * و ** نتیجه می‌گیریم که

$$\log(n!) = \Theta(n\log n)$$