ساختمان داده ها و الگوریتم ها

پیچیدگی زمانی و مکانی(توابع رشد)

محمدحسین اولیائی

مطالب این جلسه

توصیف زمان اجرای برنامه با توابع رشد

- مجانب بالا
- مجانب پایین
- نماد θ
- حل چند مثال

توابع رشد

سه نماد O,θ,Ω:

می گوییم:

- فرض کنید زمان اجرای الگوریتم حبابی $T(n) = 3n^2 + 4n + 7$ می باشد.
- بدترین حالت اجرای الگوریتم حبابی از مرتبه $T(n) = O(n^2)$ می باشد.
- یعنی از مرتبه دقیق n^2 می باشد، در حالت میانگین نیز $\theta(n^2)$ است.
- در بهترین حالت (زمانی که آرایه بصورت صعودی از قبل مرتب باشد) این الگوریتم از مرتبه $\Omega(n)$ است.

نماد O

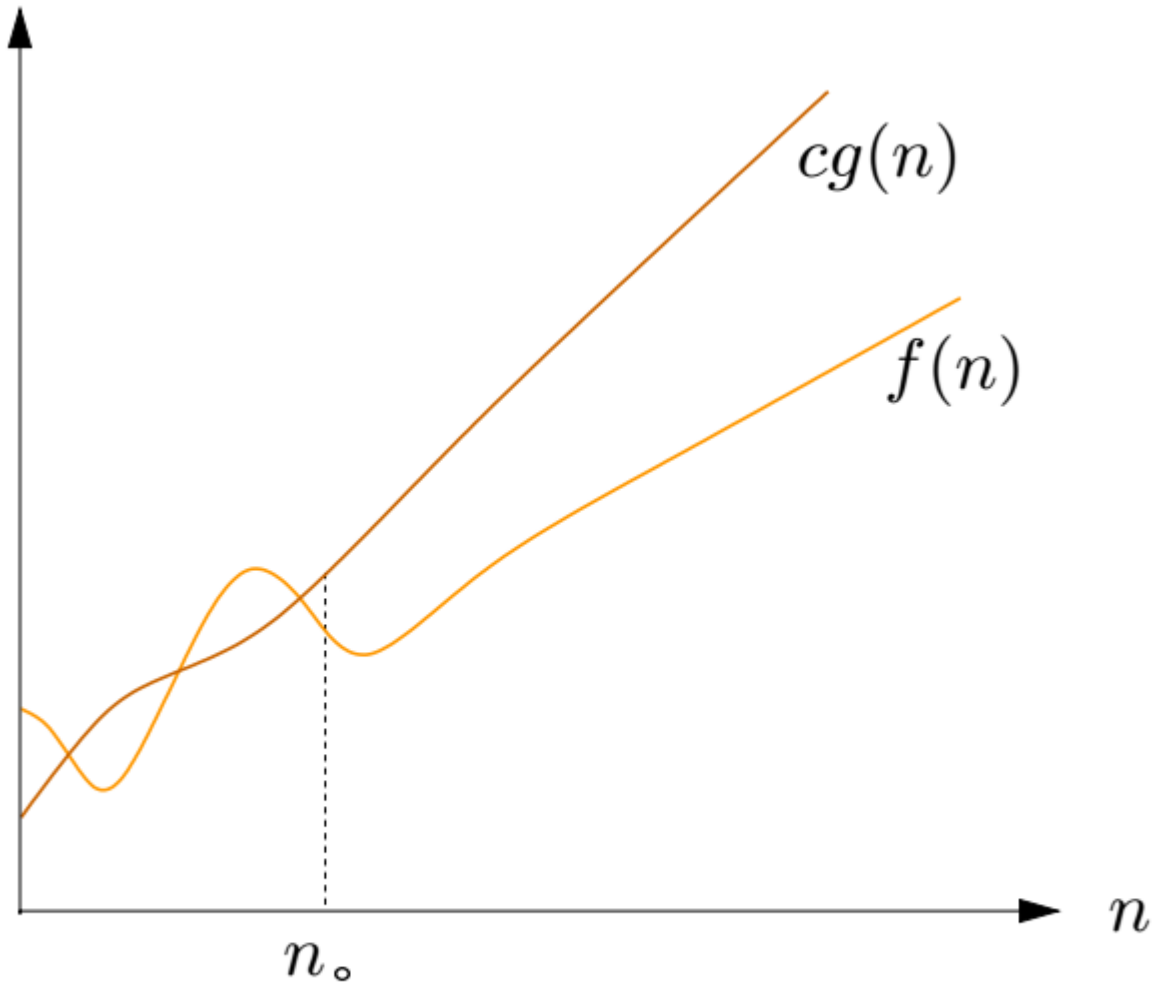
برای $g(n)$ داده‌شده، $O(g(n))$ را به شکل مجموعه‌ی توابع زیر تعریف می‌کنیم:

$T(n) = 3n^2 + 4n + 7$

$$\mathcal{O}(g(n)) = \{f(n) : \exists c > \circ \text{ and } n_\circ \text{ such that } \forall n \geq n_\circ,\ \circ \leq f(n) \leq cg(n)\}$$

n^3

$g(n)$ **کران بالای مجانبی** (asymptotically upper bound) **برای** $f(n)$ **است.**



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

توابع رشد

مثلا در مورد مرتب سازی حبابی داریم:

$$
\begin{aligned}
T(n) = 3n^2 + 4n + 7 &= \mathcal{O}(n^{۲}) \\
&= \mathcal{O}(۱۰۰n^{۲}) \\
&= \mathcal{O}(n^{۳}) \\
&= \mathcal{O}(n^{۲} \lg n) \\
&\neq \mathcal{O}(n \lg n) \\
&\neq \mathcal{O}(n).
\end{aligned}
$$

می‌توان گفت مسئله از $\mathcal{O}(n^{۱۰۰})$ است، ولی این اطلاع چندان مفید نیست.

نماد Ω

برای $g(n)$ **داده‌شده،** $\Omega(g(n))$ **را به‌شکل مجموعه‌ی توابع زیر تعریف می‌کنیم:**

$$\Omega(g(n)) = \{f(n) : \exists c > \circ \text{ and } n_\circ \text{ such that } \forall n \geq n_\circ \circ \leq cg(n) \leq f(n)\}$$

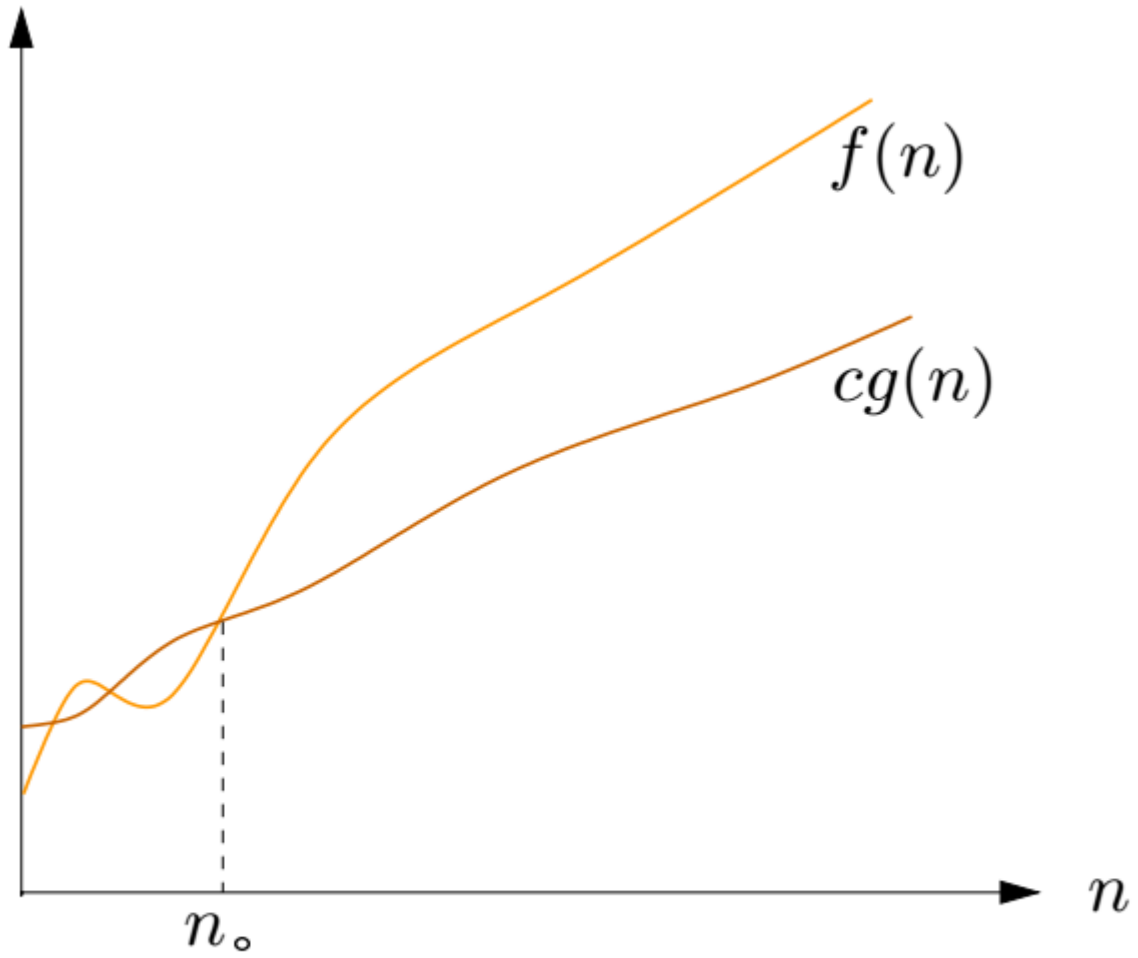
$g(n)$ **کران پایین مجانبی** (asymptotically lower bound) **برای** $f(n)$ **است.**

نماد Ω

مثلا در مورد مرتب سازی حبابی داریم:

$$
\begin{aligned}
n^2 = T(n) &= \Omega(n \lg n) \\
&= \Omega(n) \\
&= \Omega(n^2) \\
&\neq \Omega(n^2 \lg n)
\end{aligned}
$$

در عمل نماد Ω برای نشان دادن کران پایین یک تابع دیگر به‌کار می‌رود.

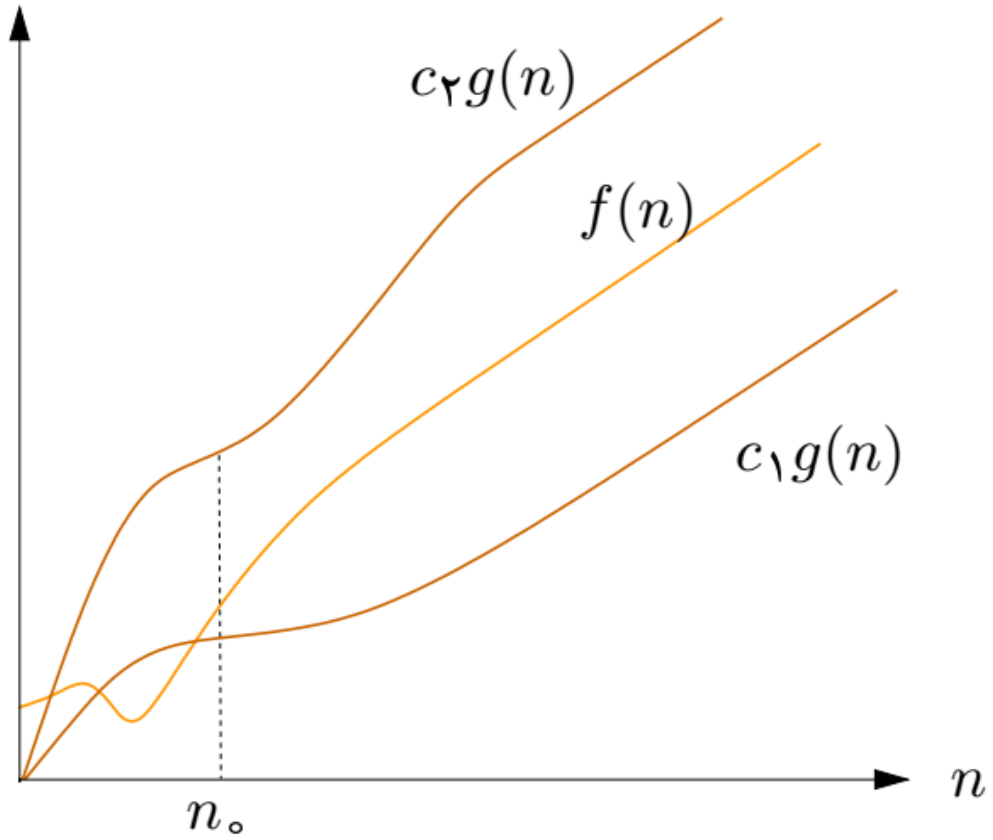


$$f(n) = \Omega(g(n))$$

نماد θ

$$\Theta(g(n)) = \{f(n) : \exists c_1, c_2 > 0 \text{ and } n_0 > 0 \text{ such that } \forall n \geq n_0,\ 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)\}$$

عبارت $c_1 g(n) \leq f(n) \leq c_2 g(n)$ به این معنی است که برای مقادیر بزرگ n درجه‌ی رشد f و g یکسان است.



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

$$f(n) \in \Theta(g(n))$$

می‌گوییم:

تابع $g(n)$ **کران بالای بسته‌ی مجانبی** (asymptotically tight bound) **برای** $f(n)$ **است.**

به‌شکل ساده‌تر

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

در مورد الگوریتم مرتب‌ساز حبابی $T(n) = \Theta(n^۲)$.

زیرا می‌توان مقادیر مثبتی برای $c_۱$ و $c_۲$ را طوری یافت که

$$c_۱ n^۲ \leq an^۲ + bn + c \leq c_۲ n^۲$$

مسئله: ثابت کنید که $100n^2 + 5n - 4 = \Theta(n^2)$.

حل:

اگر $c_1 = 1$ **و** $c_2 = 200$، **برای همه‌ی مقادیر** $n > 1$ **داریم**

$$c_1 n^2 \leq 100n^2 + 5n - 4 \leq c_2 n^2$$

مسئله: نشان دهید که $100n^2 + 5n - 4 \neq \Theta(n^3)$.

حل:

باید نشان دهیم که هیچ مقادیر مثبت برای c_1 و c_2 و یک مقدار برای n_0 پیدا نمی‌شود که رابطه‌ی

$$c_1 n^3 \leq 100n^2 + 5n - 4 \leq c_2 n^3$$

برای همه‌ی مقادیر $n > n_0$ برقرار باشد.

به‌وضوح به‌ازای هر مقدار $c_1 > 0$ و برای nهای بزرگ داریم $c_1 n^3 \not\leq 100n^2 + 5n - 4$.

می‌توان ثابت کرد که هر چندجمله‌ای از مرتبه‌ی دقیق جمله‌ی با بزرگ‌ترین توانش است، یعنی

$$a_k n^k + a_{k-1} n^{k-1} + \cdots = \Theta(n^k)$$

در مورد مرتب سازی حبابی داریم:

$$\begin{aligned} T(n) &= \Theta(n^۲) \\ &= \Theta(۱۰۰n^۲) \\ &\neq \Theta(n^۳) \\ &\neq \Theta(n^۲ \lg n) \\ &\neq \Theta(n \lg n) \end{aligned}$$

برای اثبات $T(n) \neq \Theta(n \lg n)$ باید نشان داد که نمی‌توان ثابت‌های $c_۱$ و $c_۲$ را پیدا کرد که برای $n > n_۰$ داشته‌باشیم:

$$c_۱ n \lg n \leq n^۲ \leq c_۲ n \lg n$$

تابع Θ در واقع عطف $\mathcal{O}$ و Ω است یعنی

$$f(n) = \Theta(g(n)) \Leftrightarrow f(n) = \mathcal{O}(g(n)) \wedge f(n) = \Omega(g(n))$$

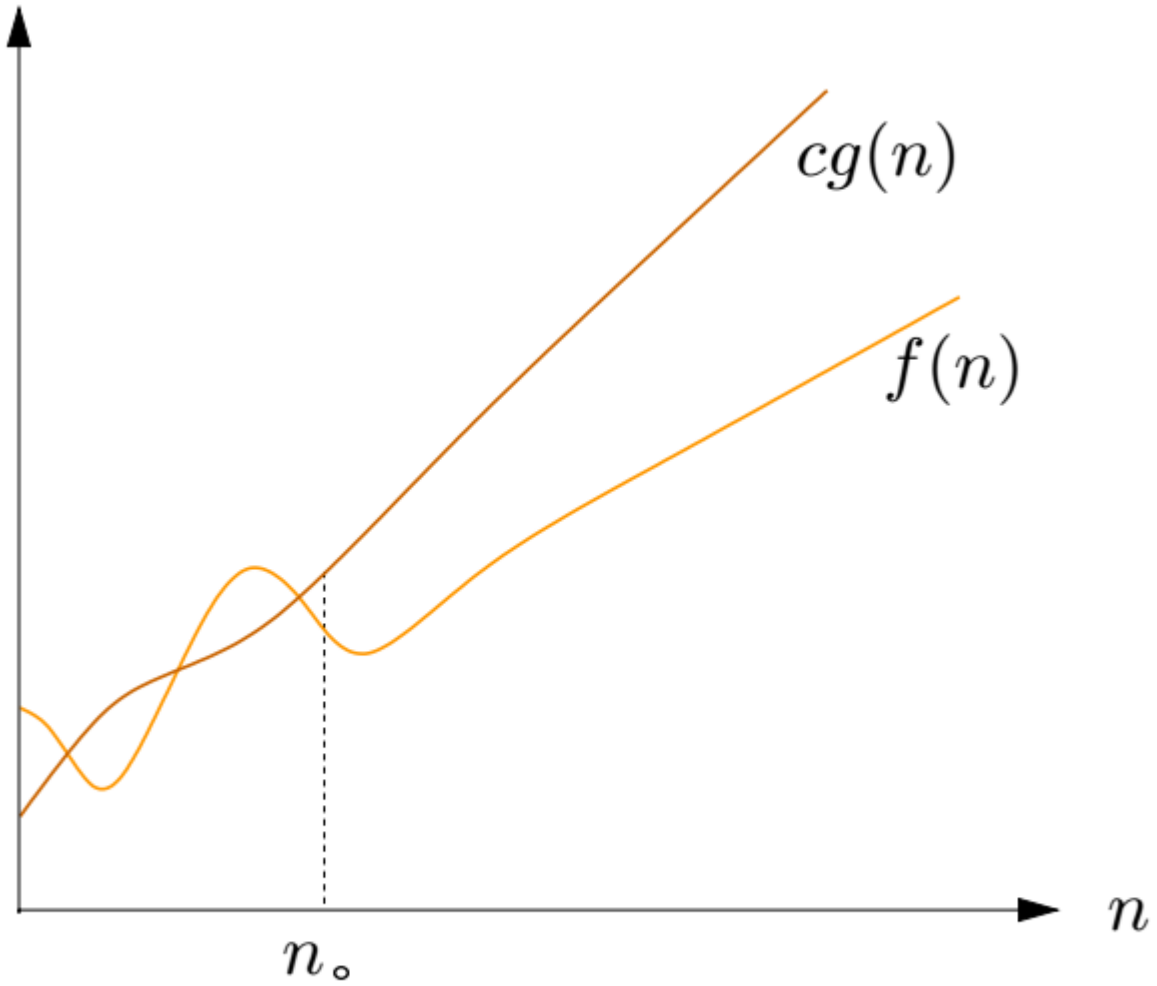
$n^2, n^3, n^2\text{logn}, \ldots$

یا $n, c, nlogn, n^2, \ldots$

قضیه: شرط لازم و کافی برای $f(n) = \Theta(g(n)$ آن است که $f(n) = \mathcal{O}(g(n)$ و $f(n) = \Omega(g(n)$.

نماد o

$$o(g(n)) = \{f(n) | \quad \text{for any positive constant} \quad c > \circ,$$
$$\exists n_\circ > \circ, s.t. \forall n > n_\circ : \circ \leq f(n) < cg(n)\}$$



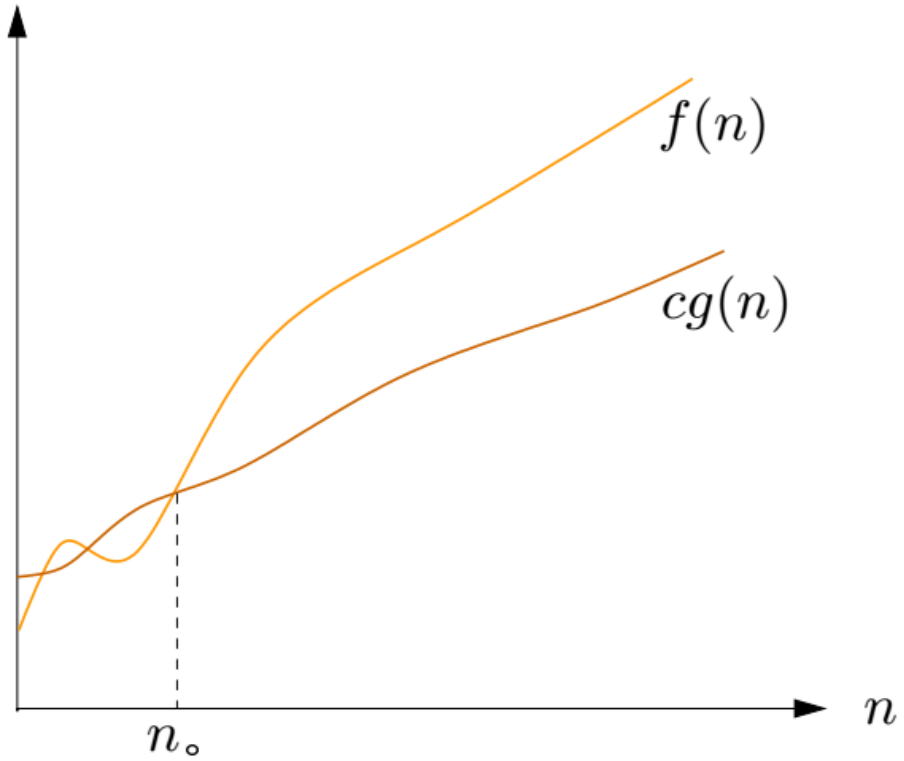
$$f(n) = o(g(n))$$

این نماد به $\mathcal{O}$ شبیه است، بااین تفاوت که درجه‌ی رشد $f(n)$ نمی‌تواند با $g(n)$ برابر باشد و باید الزاماً کوچک‌تر باشد.

نکته: اگر $f(n) = o(g(n))$، داریم: $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$

نماد ω

$$\omega(f(n)) = \{g(n)| \quad \text{for any positive constant} \quad c > \circ, \exists n_\circ > \circ,$$
$$\text{such that} \quad \forall n > n_\circ, \circ \leq cg(n) < f(n)\}$$



$$f(n) = \omega(g(n))$$

رابطه‌ی ω با Ω مثل o با $\mathcal{O}$ است.

نکته: $f(n) = o(g(n))$ **اگر و فقط اگر** $g(n) = \omega(f(n))$.

نکته: اگر $f(n) = \omega(g(n))$**، داریم:** $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$.

مثلاً $n^۲/۲ = \omega(n)$ **ولی** $n^۲/۲ \neq \omega(n^۲)$.

خواص توابع رشد

خاصیت تراگذری (transitive)

۱) از $f(n) = \Theta(g(n))$ و $g(n) = \Theta(h(n))$ **نتیجه می‌شود که** $f(n) = \Theta(h(n))$

۲) از $f(n) = \mathcal{O}(g(n))$ و $g(n) = \mathcal{O}(h(n))$ **نتیجه می‌شود که** $f(n) = \mathcal{O}(h(n))$

۳) از $f(n) = \Omega(g(n))$ و $g(n) = \Omega(h(n))$ **نتیجه می‌شود که** $f(n) = \Omega(h(n))$

۴) از $f(n) = o(g(n))$ و $g(n) = o(h(n))$ **نتیجه می‌شود که** $f(n) = o(h(n))$

۵) از $f(n) = \omega(g(n))$ و $g(n) = \omega(h(n))$ **نتیجه می‌شود که** $f(n) = \omega(h(n))$

خاصیت بازتابی

۱) $f(n) = \Theta(f(n))$

۲) $f(n) = \mathcal{O}(f(n))$

۳) $f(n) = \Omega(f(n))$

$f(n) = o(f(n))?$

$f(n) = \omega(f(n))?$

خاصیت تقارنی

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

$$g(n) = O(f(n)) \iff f(n) = O(g(n))$$

تمرین: همین‌جا حل کنید!

۱) آیا $f(n) + g(n) = \Theta(\min(f(n), g(n))$ است؟

۲) آیا $f(n) = \Theta(f(n)/۲)$ است؟

۳) آیا $f(n) + o(f(n)) = \Theta(f(n))$ ؟ $T(n) = o(f(n))$

روش‌های تحلیل الگوریتم‌ها

الگوریتم‌های ترتیبی

- یک یا تعداد ثابتی عبارت ساده (Statement) $\Leftarrow \Theta(1)$

- $T(n)$ زمان اجرای یک تکه‌برنامه که به‌صورت زیر است:

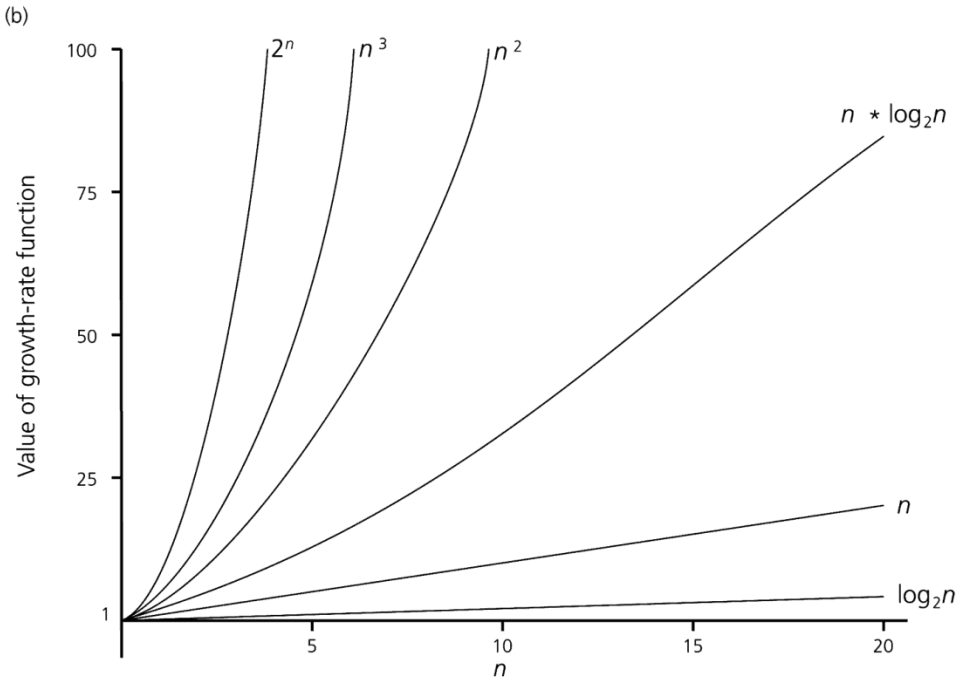
  ◁ k تکه‌برنامه با زمان‌های اجرای زیر

$$
\begin{aligned}
T_۱(n) &= \mathcal{O}(f_۱(n)) \\
T_۲(n) &= \mathcal{O}(f_۲(n)) \\
&\cdots \\
T_k(n) &= \mathcal{O}(f_k(n))
\end{aligned}
$$

در آن‌صورت،

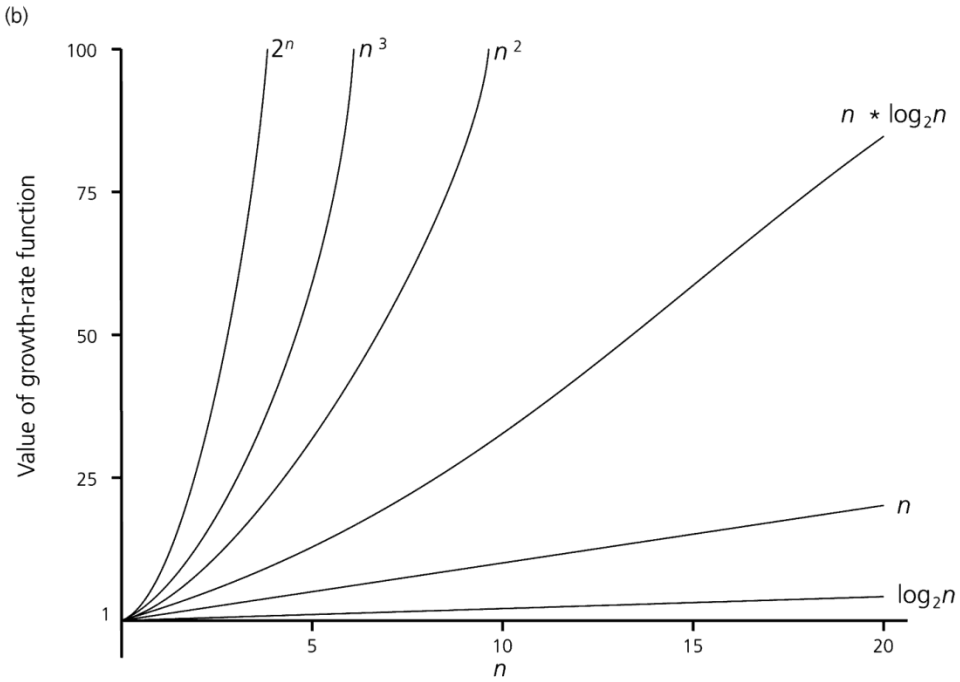
$$T(n) = \sum_{i=۱}^{k} T_i(n) = \mathcal{O}(\max_{۱ \le i \le k}(f_i(n))$$

مقایسه مقادیر توابع برای n های مختلف



	Time for $f(n)$ instructions on a 10^9 instr/sec computer						
n	$f(n)=n$	$f(n)=\log_2 n$	$f(n)=n^2$	$f(n)=n^3$	$f(n)=n^4$	$f(n)=n^{10}$	$f(n)=2^n$
10	.01μs	.03μs	.1μs	1μs	10μs	10sec	1μs
20	.02μs	.09μs	.4μs	8μs	160μs	2.84hr	1ms
30	.03μs	.15μs	.9μs	27μs	810μs	6.83d	1sec
40	.04μs	.21μs	1.6μs	64μs	2.56ms	121.36d	18.3min
50	.05μs	.28μs	2.5μs	125μs	6.25ms	3.1yr	13d
100	.10μs	.66μs	10μs	1ms	100ms	3171yr	$4*10^{13}$yr
1,000	1.00μs	9.96μs	1ms	1sec	16.67min	$3.17*10^{13}$yr	$32*10^{283}$yr
10,000	10.00μs	130.03μs	100ms	16.67min	115.7d	$3.17*10^{23}$yr	
100,000	100.00μs	1.66ms	10sec	11.57d	3171yr	$3.17*10^{33}$yr	
,000,000	1.00ms	19.92ms	16.67min	31.71yr	$3.17*10^{7}$yr	$3.17*10^{43}$yr	

مقایسه مقادیر توابع برای n های مختلف



	Time for $f(n)$ instructions on a 10^9 instr/sec computer						
n	$f(n)=n$	$f(n)=\log_2 n$	$f(n)=n^2$	$f(n)=n^3$	$f(n)=n^4$	$f(n)=n^{10}$	$f(n)=2^n$
10	.01μs	.03μs	.1μs	1μs	10μs	10sec	1μs
20	.02μs	.09μs	.4μs	8μs	160μs	2.84hr	1ms
30	.03μs	.15μs	.9μs	27μs	810μs	6.83d	1sec
40	.04μs	.21μs	1.6μs	64μs	2.56ms	121.36d	18.3min
50	.05μs	.28μs	2.5μs	125μs	6.25ms	3.1yr	13d
100	.10μs	.66μs	10μs	1ms	100ms	3171yr	$4*10^{13}$yr
1,000	1.00μs	9.96μs	1ms	1sec	16.67min	$3.17*10^{13}$yr	$32*10^{283}$yr
10,000	10.00μs	130.03μs	100ms	16.67min	115.7d	$3.17*10^{23}$yr	
100,000	100.00μs	1.66ms	10sec	11.57d	3171yr	$3.17*10^{33}$yr	
.000,000	1.00ms	19.92ms	16.67min	31.71yr	$3.17*10^{7}$yr	$3.17*10^{43}$yr	

$$\log n < (\log n)^r < n^b < a^n < n! < n^n$$

درستی و نادرستی هریک از گزاره های را بررسی کنید

$n! = O(n^n)$

$n^2 \log n = \theta(n^2)$

$5n^2 - 6n = \theta(n^3)$

$3^n = O(2^n)$

$\sum_{i=0}^{n} i^2 = \theta(n^3)$

$33n^3 + 4n^2 = \Omega(n^3)$

$\sum_{i=0}^{n} i^3 = O(n^3)$

$\frac{6n^3}{logn+1} = O(n^3)$

$n^{2^n} + 10n^2 = O(n^{2^n})$

$n^{1.001} + nlogn = \theta(n^{1.001})$

زمان اجرای الگوریتم‌های زیر را محاسبه کنید:

```
MYSTRY(n)
1  for i ← 1 to n − 1
2      do for j ← i + 1 to n
3             do for k ← 1 to j
4                    do some O(1) statements
```

زمان اجرای الگوریتم‌های زیر را محاسبه کنید:

```
MYSTRY(n)
1  for i ← 1 to n − 1
2      do for j ← i + 1 to n
3          do for k ← 1 to j
4              do some O(1) statements
```

i	j	k	تعداد اجرا
1	2	1	2
		2	
	3	1	3
		2	
		3	
	4	1	4
		...	

جواب: $\mathcal{O}(n^3)$

بدون درنظر گرفتن حلقه اول: $2 + 3 + 4 + \cdots n = \frac{n(n+1)}{2} - 1 = \theta(n^2)$

```
VeryOdd(n)
 1  for i ← 1 to n
 2      do if odd(i)
 3           then for j ← 1 to n
 4                  do x ← x + 1
 5         for k ← 1 to i
 6             do y ← y + 1
```

```
VeryOdd(n)
 1  for i ← 1 to n
 2      do if odd(i)
 3           then for j ← 1 to n
 4                  do x ← x + 1
 5         for k ← 1 to i
 6             do y ← y + 1
```

جواب: $\mathcal{O}(n^2)$

1	21	34	12	123	31	434	5	15	210
5	10	11	16	123	210	220	300	1000	X

- زمان اجرای الگوریتمی که عنصر جدیدی را به یک آرایه مرتب n عنصری اضافه می کند
- زمان اجرای الگوریتمی که عنصر جدیدی را به یک آرایه نامرتب اضافه می کند
- زمان اجرای تابعی که میانگین عناصر یک آرایه $M \times N$ را محاسبه و این مقدار را به همه عناصر سطر اول اضافه می کند
- تعیین عناصر مشترک بین دو لیست که هر دو نامرتب هستند