$\lg^*$

طراحی الگوریتم‌ها – جلسه چهارم

Introduction to Algorithm

استاد: جوانمردی

۱۳۹۹/۷/۷

مرور جلسه قبل

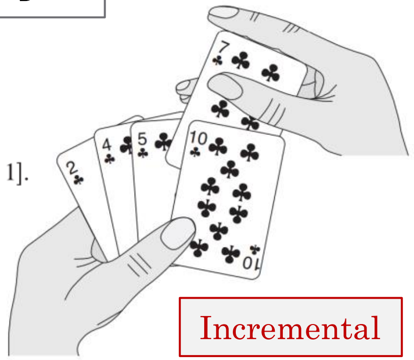
روش تقسیم و حل

The divide-and-conquer approach:

- **Divide** the problem into a number of subproblems
- **Conquer** the subproblems by solving them recursively
- **Combine** subproblems and solve the original problem

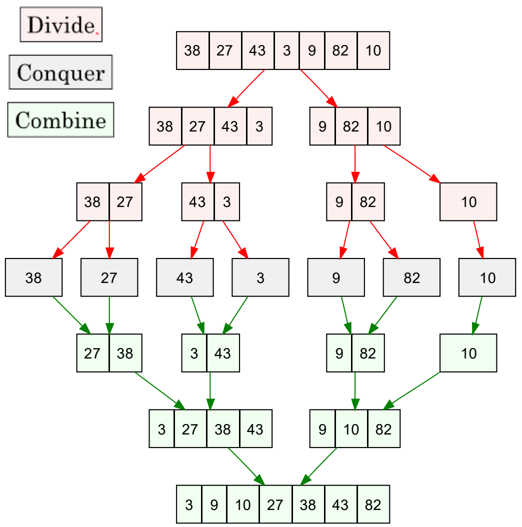
مرتب‌سازی درجی

```
INSERTION-SORT(A)
for j = 2 to A.length
    key = A[j]
    // Insert A[j] into the sorted sequence A[1..j-1].
    i = j - 1
    while i > 0 and A[i] > key
        A[i+1] = A[i]
        i = i - 1
    A[i+1] = key
```

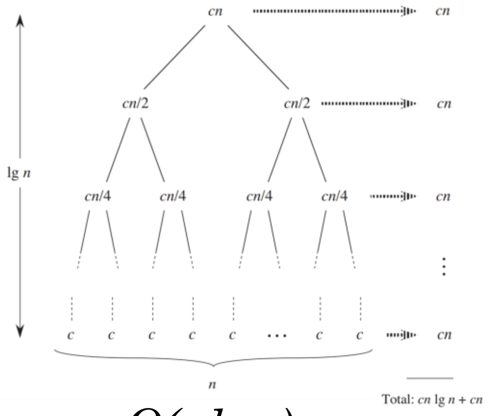


Incremental

مرتب‌سازی ادغامی



$$T(n) = \begin{cases} c & \text{if } n = 1 \\ 2\,T(n/2) + cn & \text{if } n > 1 \end{cases}$$



$\Theta(nlgn)$

بدترین و بهترین حالت

Best case	The array is already sorted

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

$\Theta(n)$

Worst case	The array is in reverse sorted order

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

$\Theta(n^2)$

فصل سوم: رشد توابع

- تعریف نمادهای رشد مجانبی
- نمادهای رشد مجانبی در معادلات
- ویژگی‌های مقایسه توابع با نمادهای رشد مجانبی
- توابع مرسوم و نمادهای استاندارد

رشد توابع و نمادهای تقریب مرتبه زمانی

- زمان اجرای دقیق الگوریتم اطلاعات اضافی ← نیاز به زمان اجرای تقریبی یا مرتبه زمانی
- نمادهای تقریب مرتبه زمانی

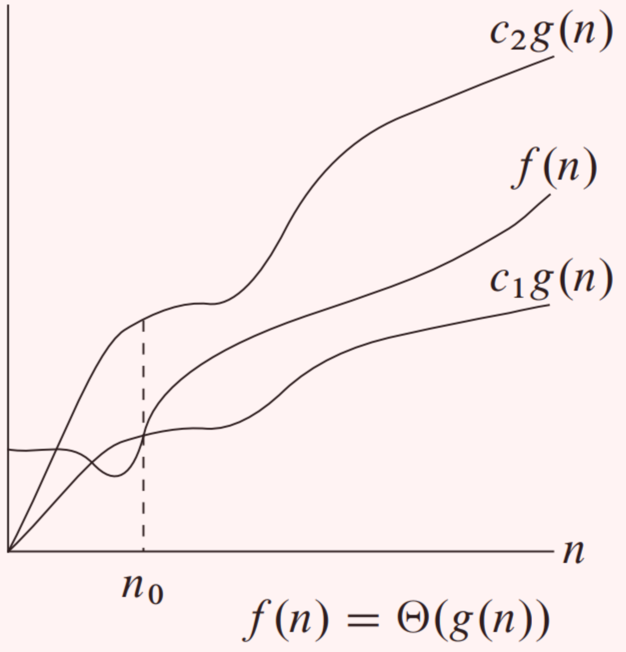
Name	Notation
Big	$\mathcal{O}$ or O
Big Omega	Ω
Big Theta	Θ
Small O	o
Small Omega	ω

- کاربرد این نمادها در طرح طراحی الگوریتم: مقایسه الگورتیم‌ها بر حسب زمان اجرا
- کاربردهای دیگر: تحلیل دیگر ویژگی‌های الگوریتم بر حسب تعداد ورودی ← حجم حافظه
- مهم: کدام زمان اجرا؟ بهترین – بدترین – متوسط؟ ← در نماد Θ اهمیت این موضوع بیشتر

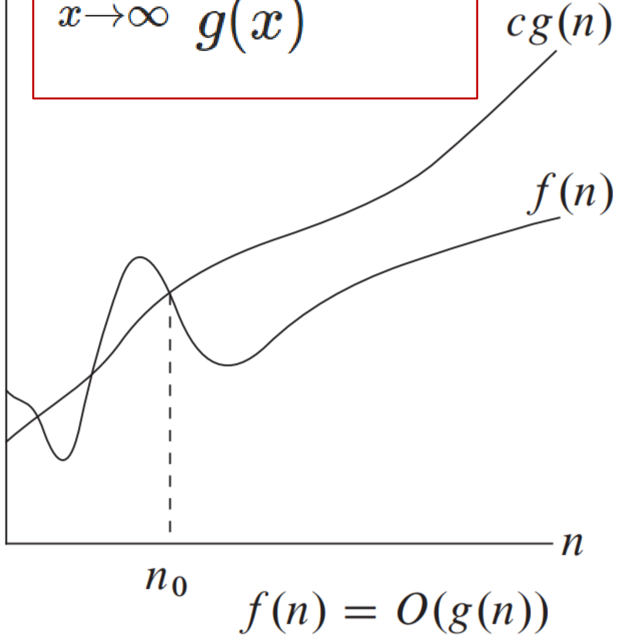
The Θ-Notation

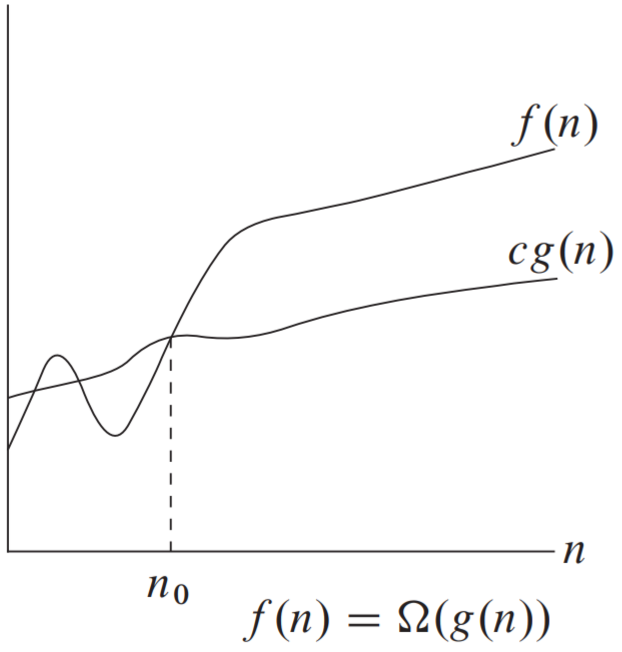
$\color{red}{\Theta(g(n))}$ = $\{ f(n) : \exists c_1, c_2 > 0, n_0 > 0 \text{ s.t. } \forall n \geq n_0: \quad c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n) \ \}$

Tight bound



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



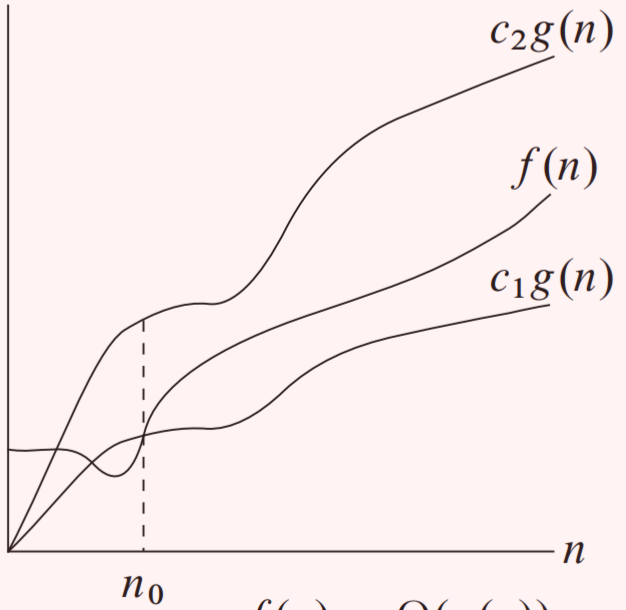


n_0: minimum possible

The Θ-Notation

$\Theta(g(n)) = \{ f(n) : \exists c_1, c_2 > 0, n_0 > 0 \text{ s.t. } \forall n \geq n_0: \quad c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n) \}$

Tight bound



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

$\frac{1}{2}n^2 - 3n = \Theta(n^2)$?

$6n^3 \neq \Theta(n^2)$?

$$c_1 n^2 \leq \frac{1}{2}n^2 - 3n \leq c_2 n^2$$

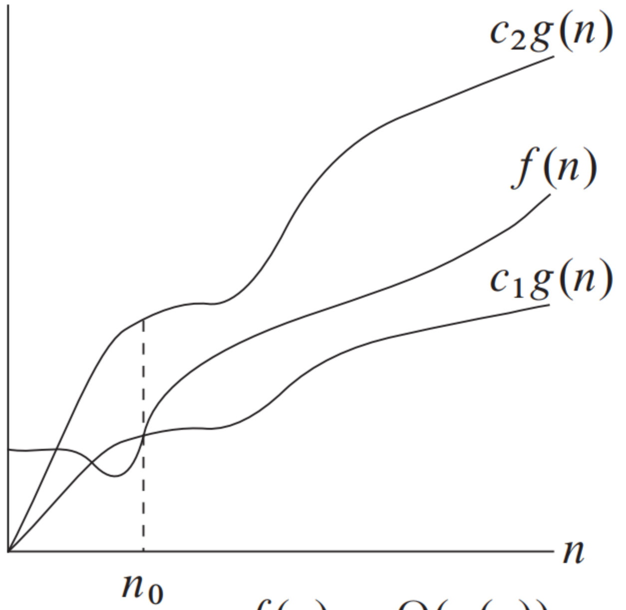
for all $n \geq n_0$. Dividing by n^2 yields

$$c_1 \leq \frac{1}{2} - \frac{3}{n} \leq c_2 \,.$$

$c_1 = 1/14 \quad c_2 = 1/2 \quad n_0 = 7$

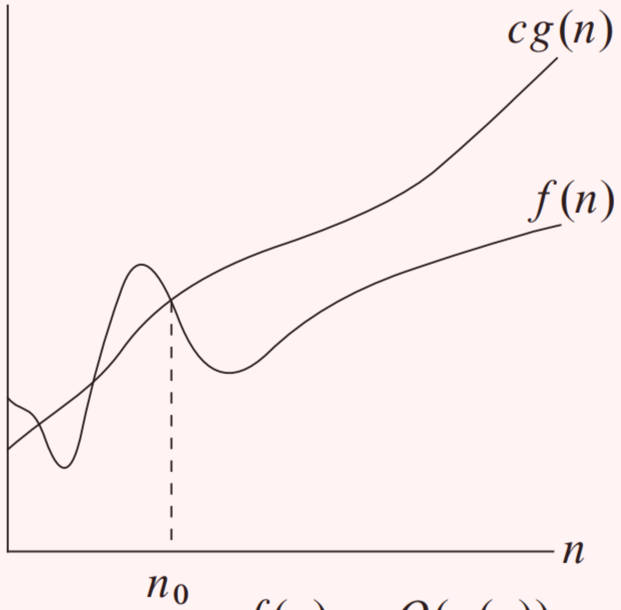
The O-Notation

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



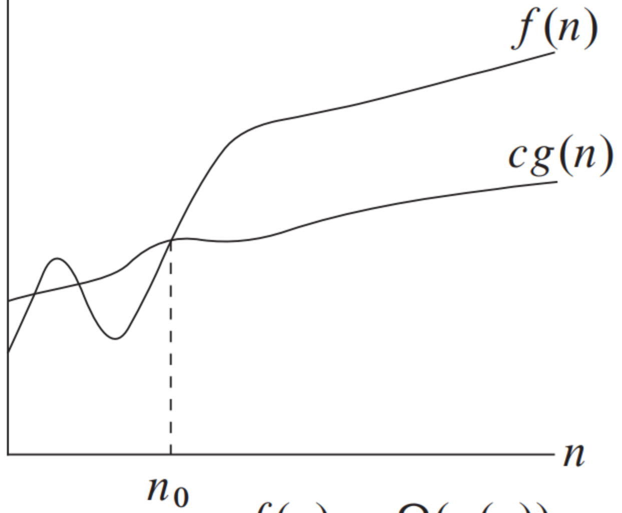
$f(n) = \Theta(g(n))$

Upper bound



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

$$\lim_{x \to \infty} \frac{f(x)}{g(x)} = c \text{ or } 0$$

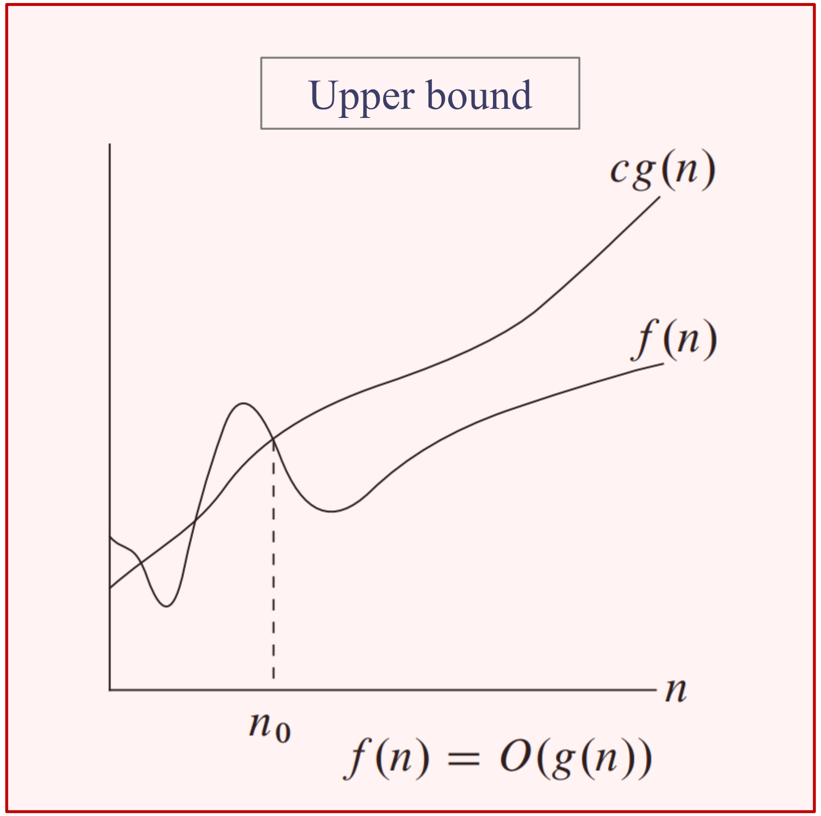


$f(n) = \Omega(g(n))$

n_0: minimum possible

The O-Notation

$\color{red}{\boldsymbol{O(g(n))}} = \{ f(n) : \exists c > 0,\ n_0 > 0 \text{ s.t. } \forall n \geq n_0 : f(n) \leq c \cdot g(n) \}$

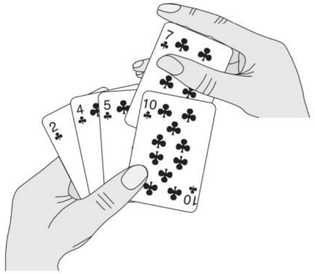


- در برخی منابع از نماد O معادل نماد Θ استفاده می شود
- نماد O : تخمین زمان اجرا از روی ساختار برنامه

  بر اساس بدترین زمان اجرا
- نماد Θ این ویژگی را ندارد

```
INSERTION-SORT(A)
for j = 2 to A.length
    key = A[j]
    // Insert A[j] into the sorted sequence A[1..j-1].
    i = j - 1
    while i > 0 and A[i] > key
        A[i + 1] = A[i]
        i = i - 1
    A[i + 1] = key
```

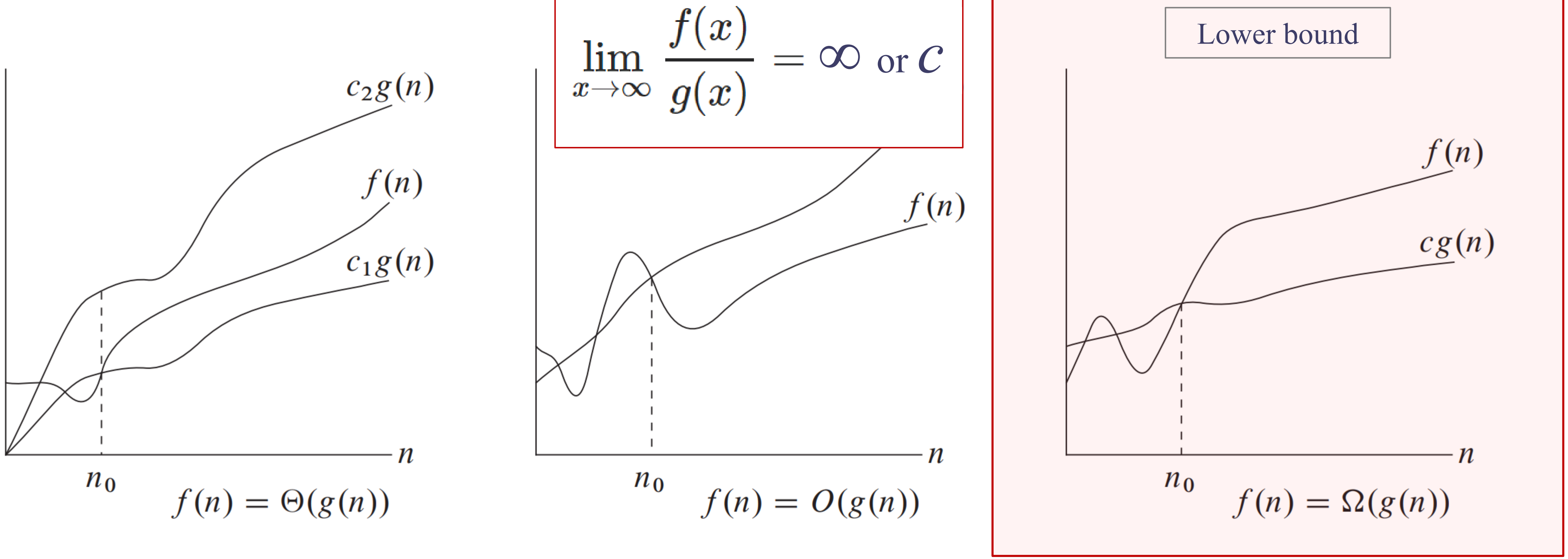
$\boxed{O(n^2)}$



The Ω-Notation

$\boldsymbol{\Omega(g(n))} = \{ f(n) : \exists c > 0, n_0 > 0 \text{ s.t. } \forall n \geq n_0 : f(n) \geq c \cdot g(n) \}$

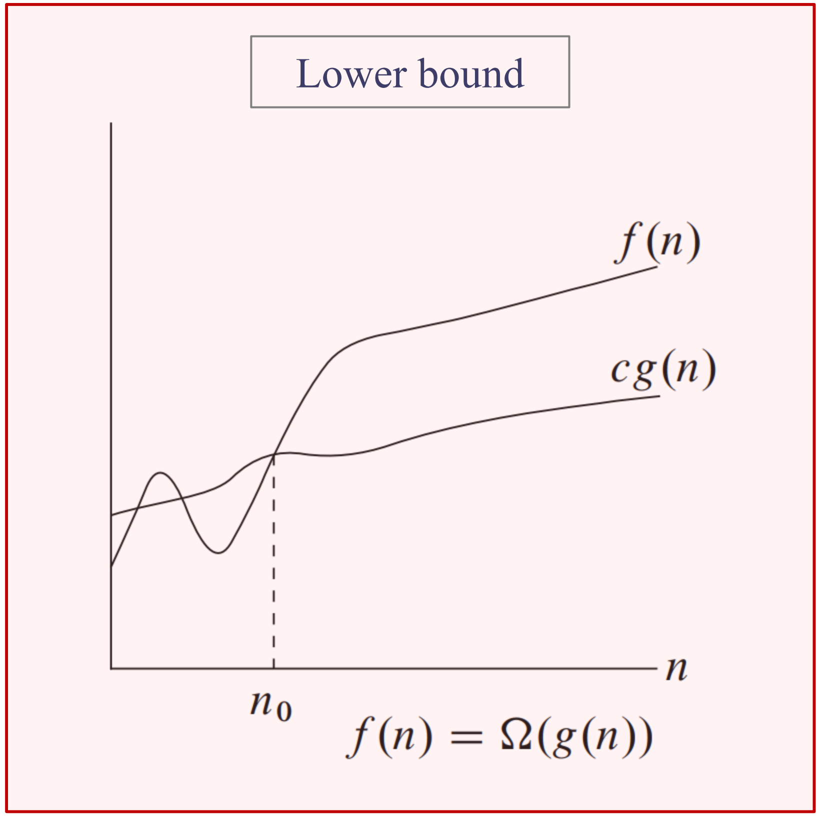
$$\lim_{x \to \infty} \frac{f(x)}{g(x)} = \infty \text{ or } C$$



n_0: minimum possible

The Ω-Notation

$$\Omega(g(n)) = \{ f(n) : \exists c > 0, n_0 > 0 \text{ s.t. } \forall n \geq n_0 : f(n) \geq c \cdot g(n) \}$$

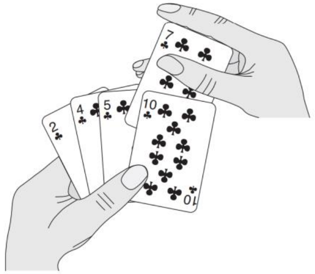


- نماد Ω : تخمین زمان اجرا از روی ساختار برنامه بر اساس بهترین زمان اجرا

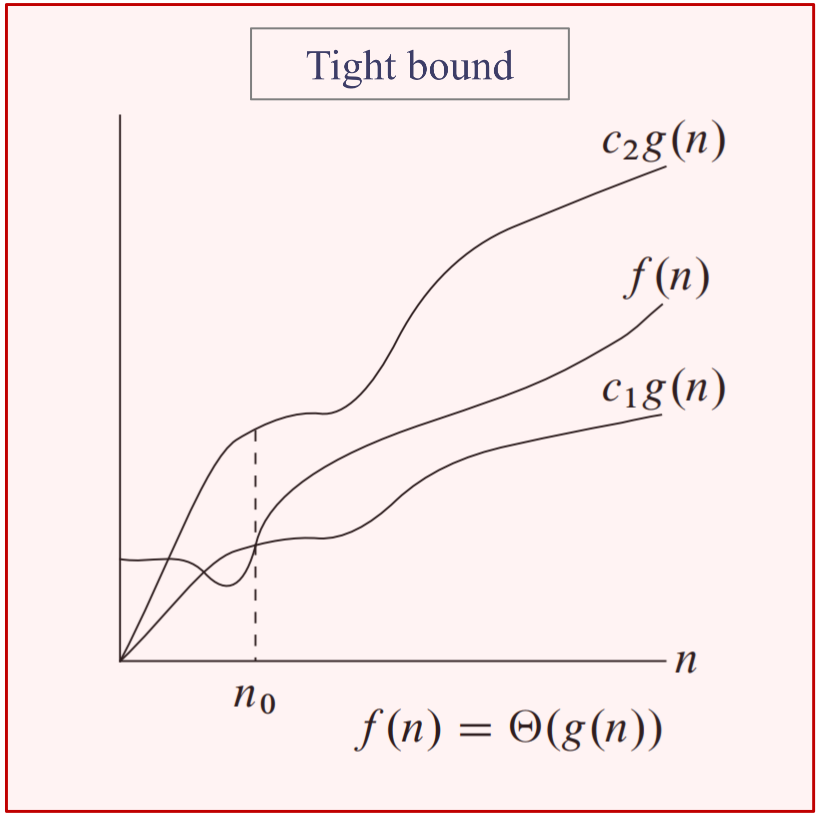
INSERTION-SORT(A)

```
for j = 2 to A.length
    key = A[j]
    // Insert A[j] into the sorted sequence A[1 .. j - 1].
    i = j - 1
    while i > 0 and A[i] > key
        A[i + 1] = A[i]
        i = i - 1
    A[i + 1] = key
```

$\boxed{\Omega(n)}$



The Θ, O and Ω-Notation



Theorem

$f(n) = \Theta(g(n))$ if and only if $f = O(g(n))$ and $f = \Omega(g(n))$

نمادهای رشد مجانبی در معادلات

- حضور نماد رشد مجانبی در سمت راست معادلات

$$2n^2 + 3n + 1 = 2n^2 + f(n)$$

وجود دارد $f(n) \in \Theta(n)$ که معادله صادق باشد

$$f(n) = 3n + 1$$

$$\boxed{2n^2 + 3n + 1 = 2n^2 + \Theta(n)}$$

- حضور نماد رشد مجانبی در سمت چپ معادلات

برای همه توابع $f(n) \in \Theta(n)$

وجود دارد تابع $g(n) \in \Theta(n^2)$

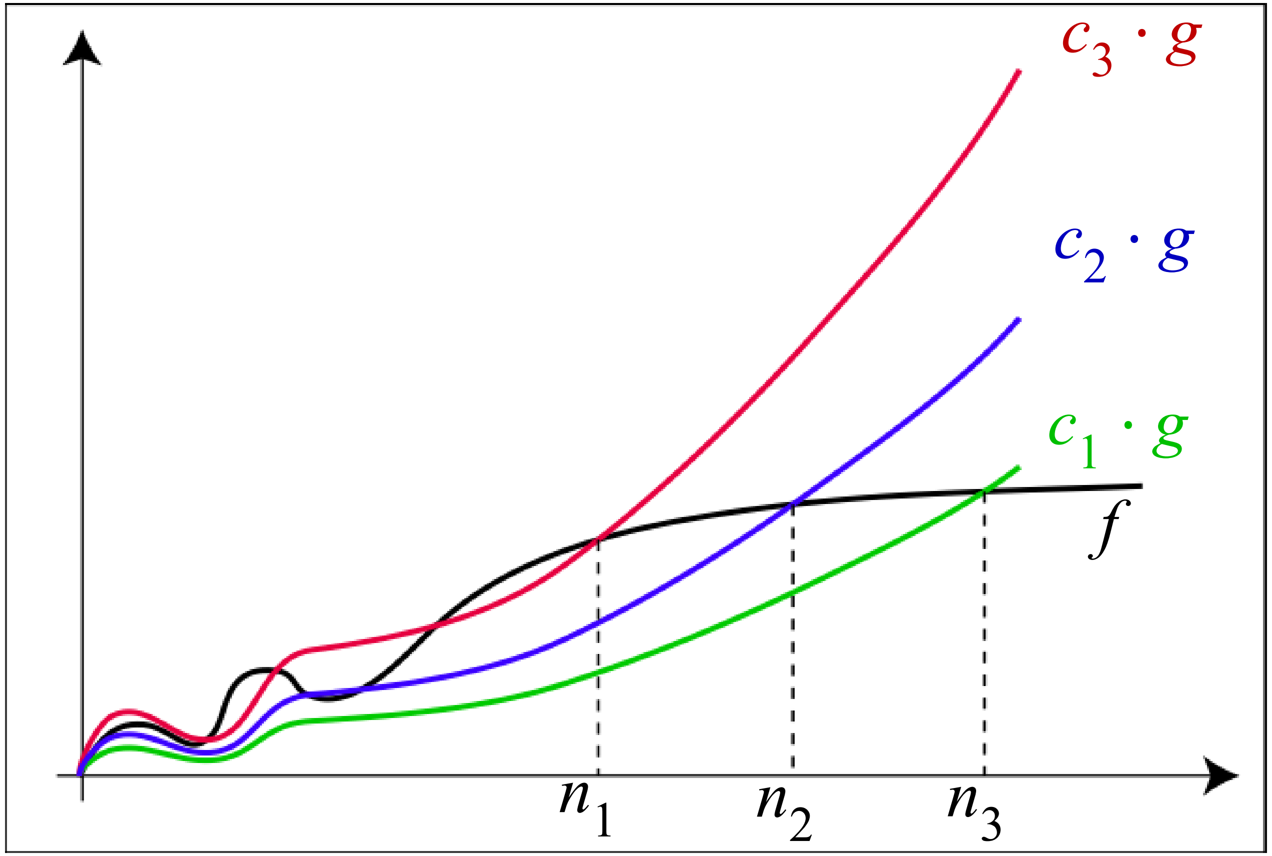
بطوریکه $2n^2 + f(n) = g(n)$

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

The o-Notation

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

$$\boxed{o(g(n)) = \{ f(n) : \forall c > 0\ \exists n_0 > 0 \text{ s.t. } \forall n \geq n_0 : f(n) \leq c \cdot g(n) \}}$$



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

$n^{1.9999} = o(n^2)$

$n^2 / \lg n = o(n^2)$

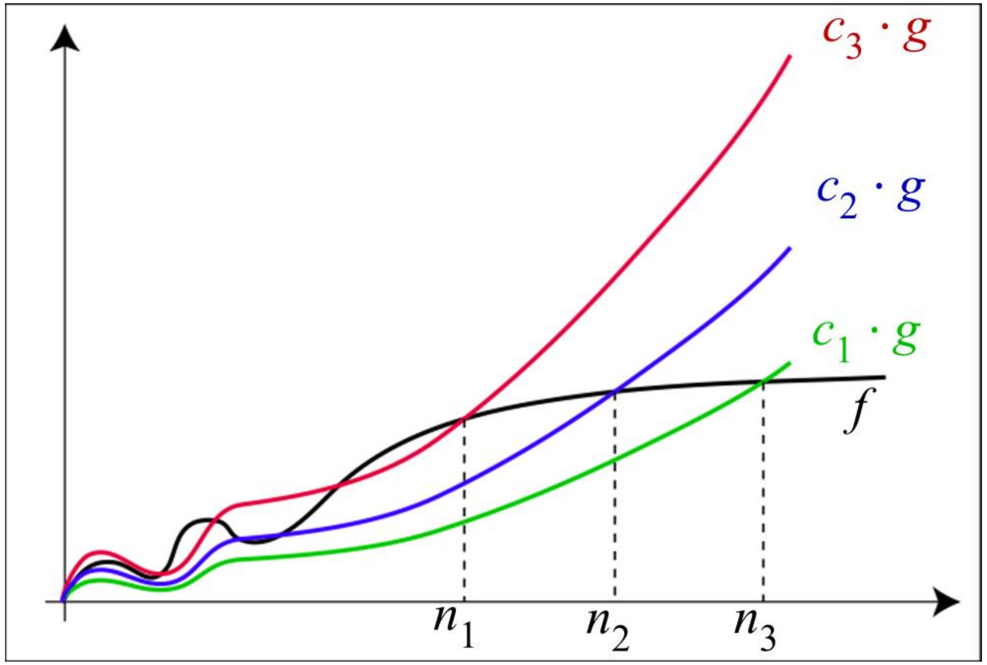
$n^2 \neq o(n^2)$ (just like $2 \not< 2$)

$n^2/1000 \neq o(n^2)$

The o-Notation

$$\boldsymbol{o(g(n))} = \{ f(n) : \forall c > 0\ \exists n_0 > 0 \text{ s.t. } \forall n \geq n_0 : f(n) \leq c \cdot g(n) \}$$

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

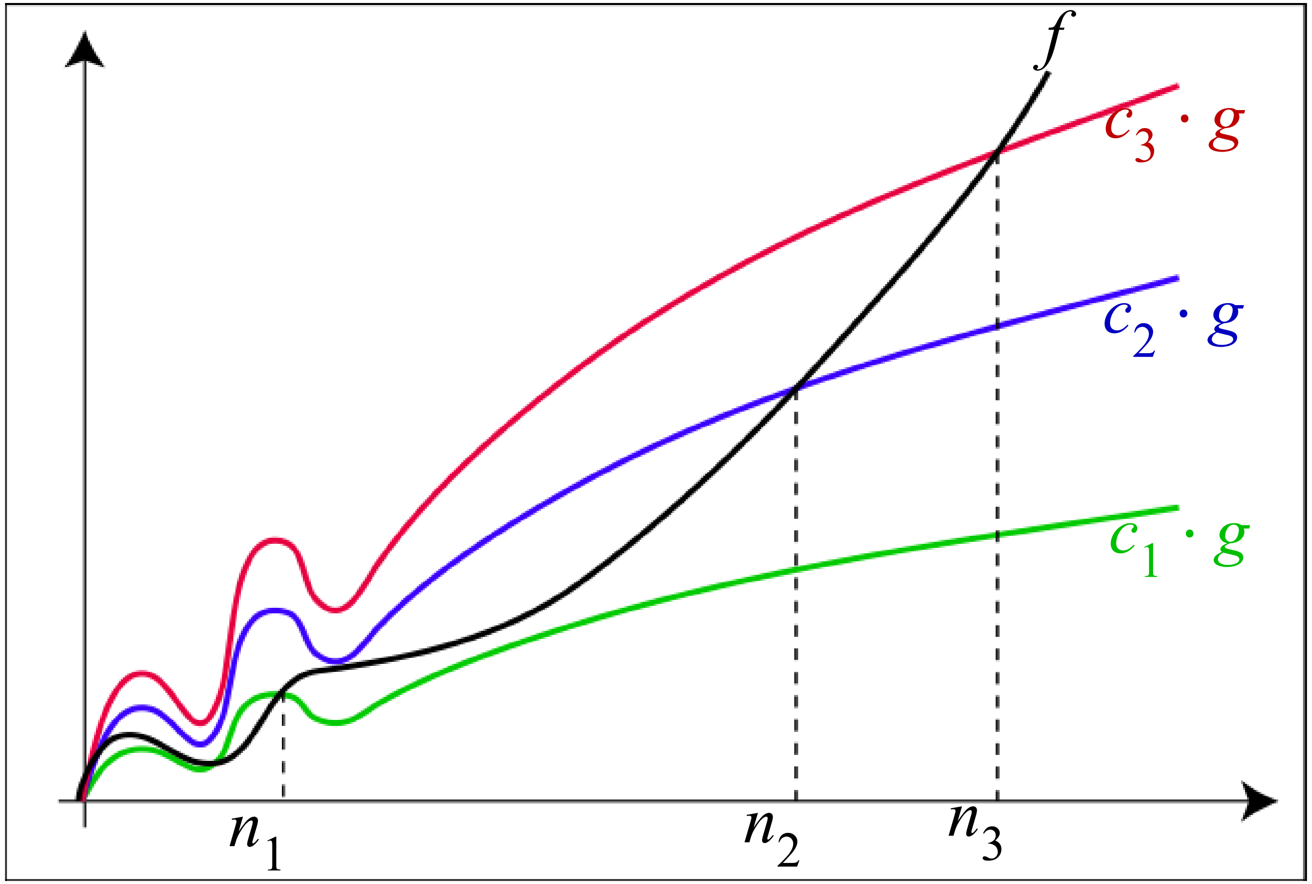


$T_A(n) = o(T_B(n))$ ⟵ الگوریتم A بهتر است الگوریتم B است

The ω-Notation

$\Omega(g(n)) = \{ f(n) : \exists c > 0, n_0 > 0 \text{ s.t. } \forall n \geq n_0: f(n) \geq c \cdot g(n) \}$

$\boxed{\omega(g(n)) = \{ f(n) : \forall c > 0\ \exists n_0 > 0 \text{ s.t. } \forall n \geq n_0: f(n) \geq c \cdot g(n) \}}$



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

$$n^{2.0001} = \omega(n^2)$$
$$n^2 \lg n = \omega(n^2)$$
$$n^2 \neq \omega(n^2)$$

معادل سازی نمادهای رشد مجانبی

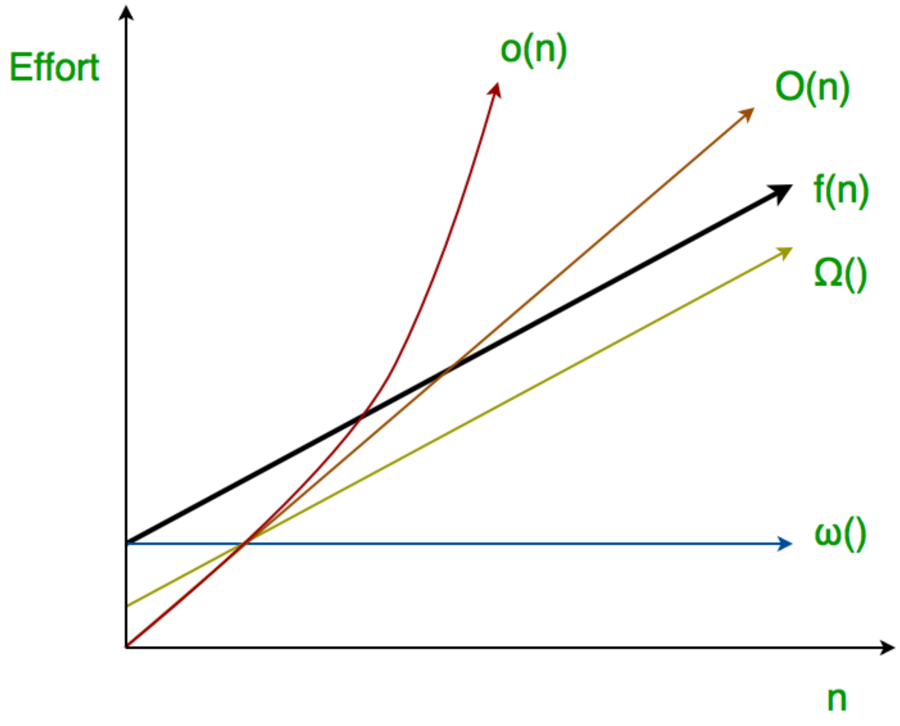
$$f(n) = O(g(n)) \approx a \leq b,$$

$$f(n) = \Omega(g(n)) \approx a \geq b,$$

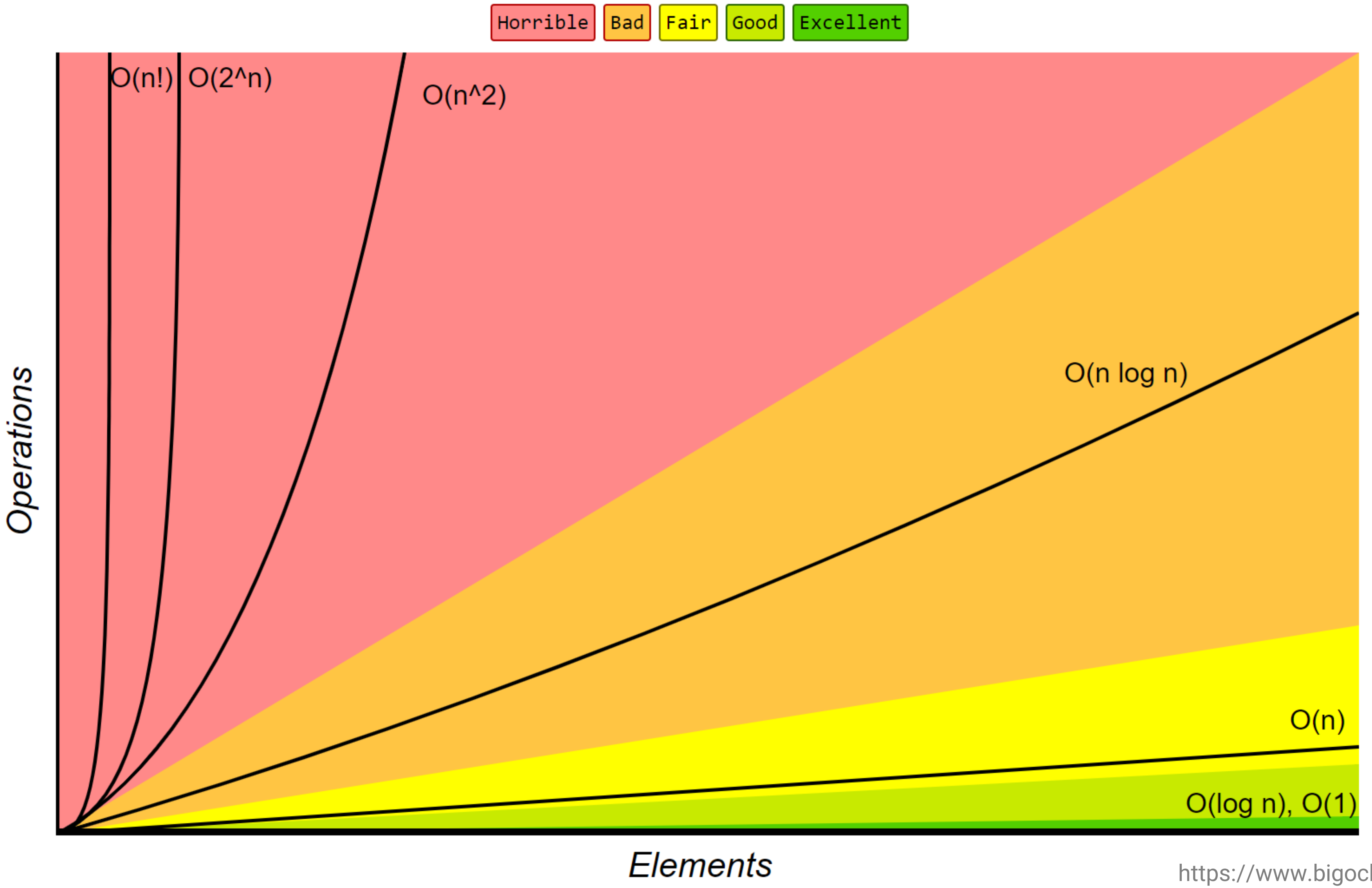
$$f(n) = \Theta(g(n)) \approx a = b,$$

$$f(n) = o(g(n)) \approx a < b,$$

$$f(n) = \omega(g(n)) \approx a > b.$$



مرسوم‌ترین بدترین زمان‌های اجرا



منبع: https://www.bigocheatsheet.com/

مقایسه خواص توابع

- $f(n) = O(g(n))$ and $g(n) = O(h(n)) \Rightarrow f(n) = O(h(n))$
- $f(n) = \Omega(g(n))$ and $g(n) = \Omega(h(n)) \Rightarrow f(n) = \Omega(h(n))$
- $f(n) = \Theta(g(n))$ and $g(n) = \Theta(h(n)) \Rightarrow f(n) = \Theta(h(n))$

تراگذری

Transitivity

- $f(n) = O(f(n))$
- $f(n) = \Omega(f(n))$
- $f(n) = \Theta(f(n))$

بازتابی

Reflexivity

مقایسه خواص توابع

تقارنی

Symmetry

- $f(n) = \Theta(g(n)) \Longleftrightarrow g(n) = \Theta(f(n))$
- $f(n) = O(g(n)) \overset{?}{\Longleftrightarrow} g(n) = O(f(n))$

لزوما برقرار نیست

ضد تقارنی

Transpose Symmetry

- $f(n) = O(g(n)) \Longleftrightarrow g(n) = \Omega(f(n))$
- $f(n) = o(g(n)) \Longleftrightarrow g(n) = \omega(f(n))$

توابع مرسوم و نمادهای استاندارد

یکنوایی تابع

Monotonicity

A function $f(n)$ is ***monotonically increasing*** if $m \leq n$ implies $f(m) \leq f(n)$

A function $f(n)$ is ***monotonically decreasing*** if $m \leq n$ implies $f(m) \geq f(n)$

A function $f(n)$ is ***strictly increasing*** if $m < n$ implies $f(m) < f(n)$

A function $f(n)$ is ***strictly decreasing*** if $m < n$ implies $f(m) > f(n)$

توابع مرسوم و نمادهای استاندارد

توابع کف و سقف

Floors and ceilings

هردو تابع یکنوای افزایشی

$$x - 1 < \lfloor x \rfloor \le x \le \lceil x \rceil < x + 1$$ (۱

$$\{x\} = x - \lfloor x \rfloor$$ (۲

$$\lceil n/2 \rceil + \lfloor n/2 \rfloor = n$$ (۳

$$\left\lceil \frac{\lceil x/a \rceil}{b} \right\rceil = \left\lceil \frac{x}{ab} \right\rceil$$ (۴

for any real number $x \ge 0$ and integers $a, b > 0$.

$$\left\lfloor \frac{\lfloor x/a \rfloor}{b} \right\rfloor = \left\lfloor \frac{x}{ab} \right\rfloor$$ (۵

توابع مرسوم و نمادهای استاندارد

$$\lim_{n\to\infty} \frac{n^b}{a^n} = 0 \qquad a > 1$$

$\longrightarrow$ $n^b = o(a^n)$

$n \to \log n$

$a \to 2^a$

?

$\lg^b n = o(n^a)$

$\lim_{n\to\infty} \frac{\log^b n}{(2^a)^{\log n}} = \lim_{n\to\infty} \frac{\log^b n}{(n^a)^{\cancel{\log 2}}} = 0 \Rightarrow \log^b n \in o(n^2)$

Factorials | فاکتوریل

$$\boxed{n! = n \times (n-1) \times (n-2) \times (n-3) \times \ldots \times 2 \times 1}$$

$$n! = o(n^n)$$

$$n! = \omega(2^n)$$

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

توابع مرسوم و نمادهای استاندارد

تکرار تابع

Functional Iteration

$$f^{(i)}(n) = \begin{cases} n & \text{if } i = 0\,, \\ f(f^{(i-1)}(n)) & \text{if } i > 0\,. \end{cases}$$

If $\quad f(n) = 2n$

then $\quad f^{(2)}(n) = f(2n) = 2(2n) = 2^2 n$

then $\quad f^{(3)}(n) = f(f^{(2)}(n)) = 2(2^2 n) = 2^3 n$

then $\quad f^{(i)}(n) = 2^i n$

توابع مرسوم و نمادهای استاندارد

لگاریتم استار

Iterated logarithmic function

$$\lg^* n = \min\{i \geq 0 : \lg^{(i)} n \leq 1\}$$

$$\begin{aligned} \lg^* 2 &= 1, \\ \lg^* 4 &= 2, \\ \lg^* 16 &= 3, \\ \lg^* 65536 &= 4, \\ \lg^*(2^{65536}) &= 5. \end{aligned}$$