طراحی الگوریتم‌ها – رشد توابع

Introduction to Algorithm

مهدی جوانمردی

۱۴۰۱

مرور مباحث قبل

روش تقسیم و حل

The divide-and-conquer approach:

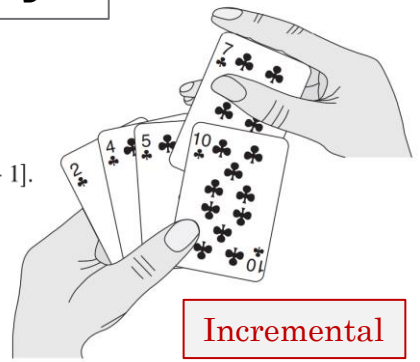
Divide the problem into a number of subproblems

Conquer the subproblems by solving them recursively

Combine subproblems and solve the original problem

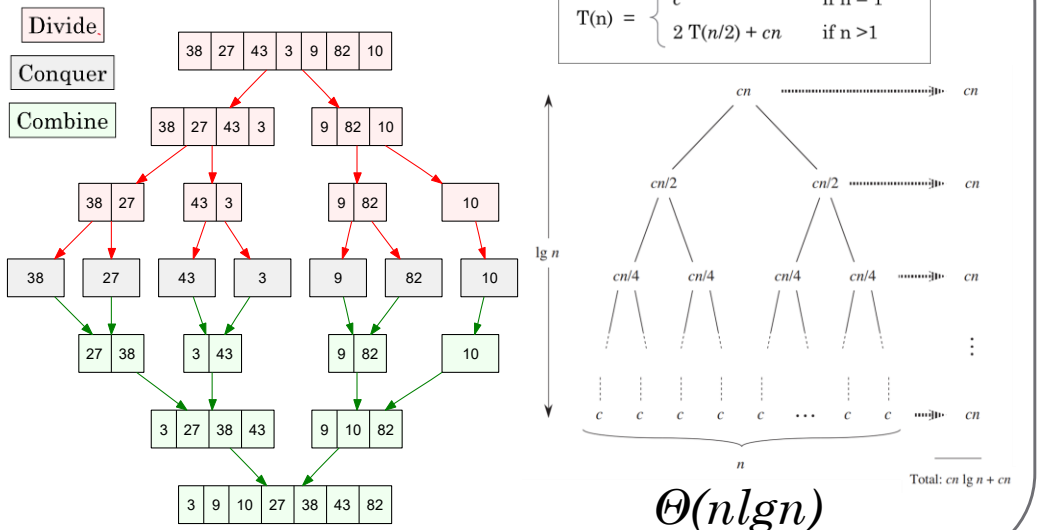
مرتب‌سازی درجی

```
INSERTION-SORT(A)
1  for j = 2 to A.length
2      key = A[j]
3      // Insert A[j] into the sorted sequence A[1 .. j − 1].
4      i = j − 1
5      while i > 0 and A[i] > key
6          A[i + 1] = A[i]
7          i = i − 1
8      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

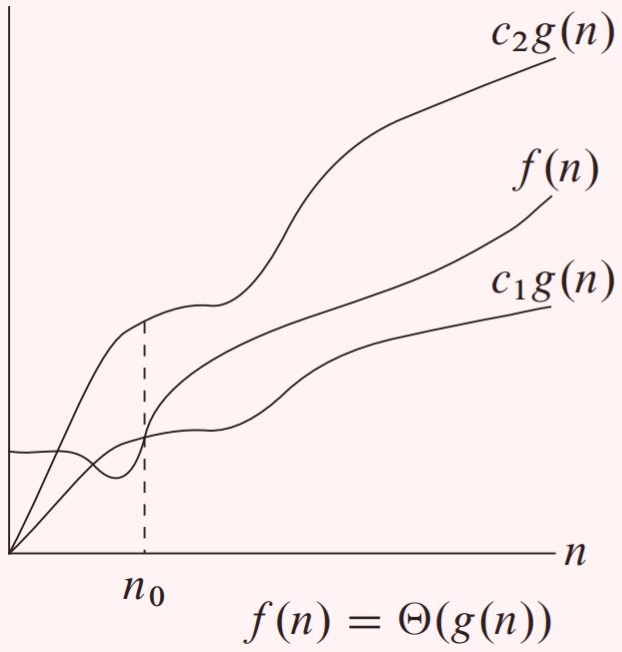
Name	Notation
Big	$\mathcal{O}$ or O
Big Omega	Ω
Big Theta	Θ
Small O	o
Small Omega	ω

- 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

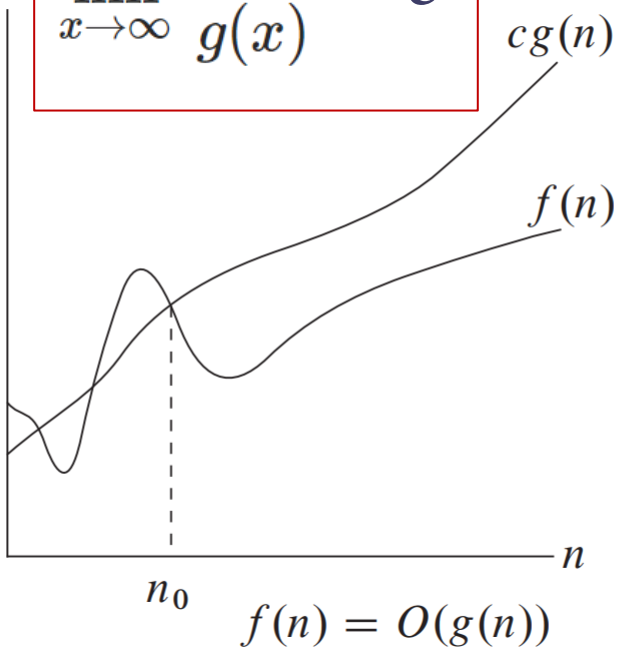
$$\Theta(g(n)) = \{ f(n) : \exists c_1, c_2 > 0, n_0 > 0 \text{ s.t. } \forall n \geq n_0: \quad c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n) \ \}$$

Tight bound

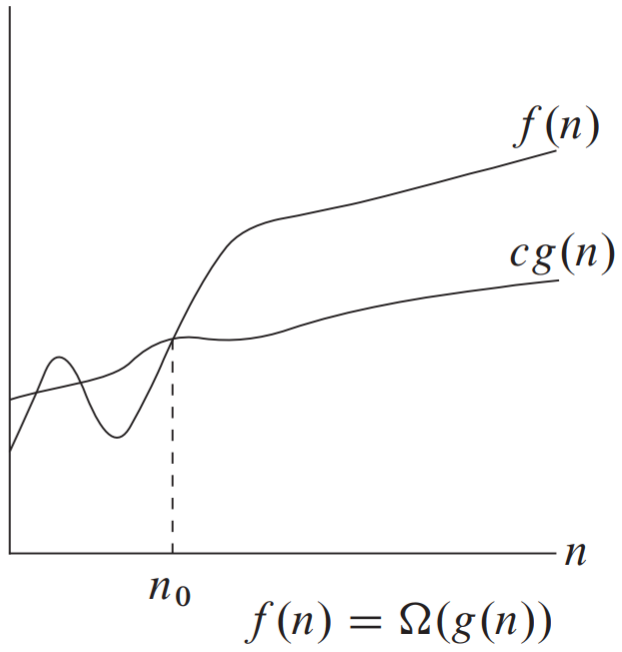


$f(n) = \Theta(g(n))$

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



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

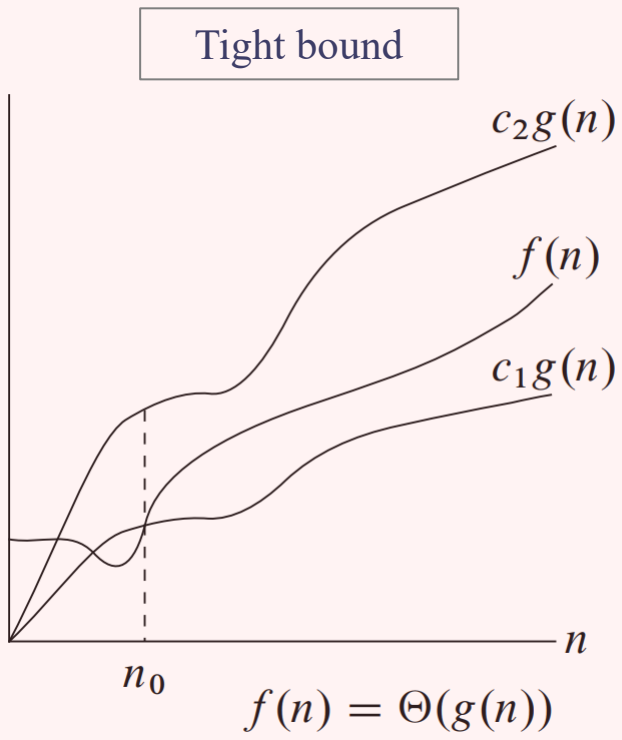


$f(n) = \Omega(g(n))$

n_0: minimum possible

The Θ-Notation

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



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

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

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

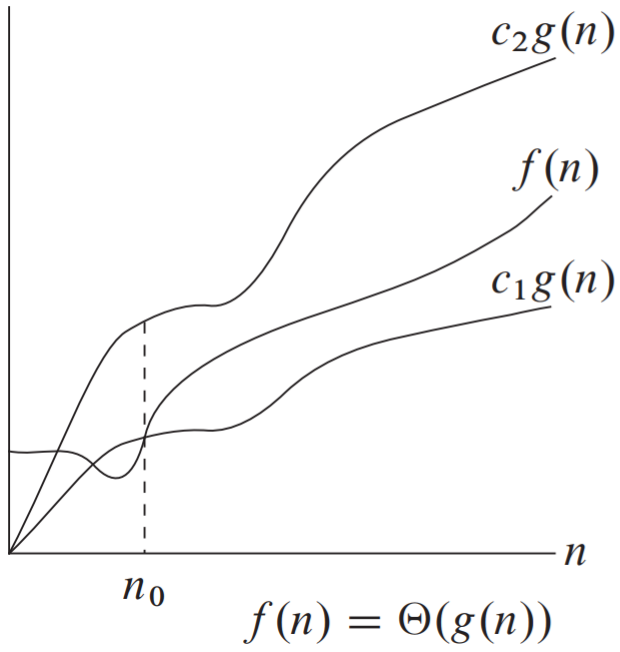
for all $n \geq n_0$. Dividing by n^2 yields

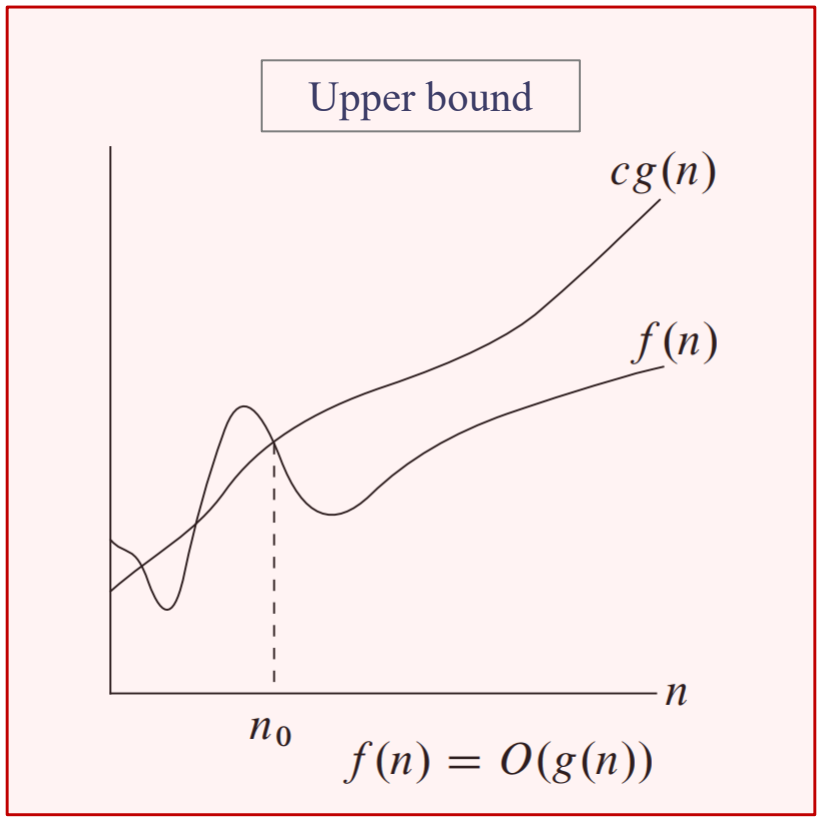
$$c_1 \leq \frac{1}{2} - \frac{3}{n} \leq c_2 .$$

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

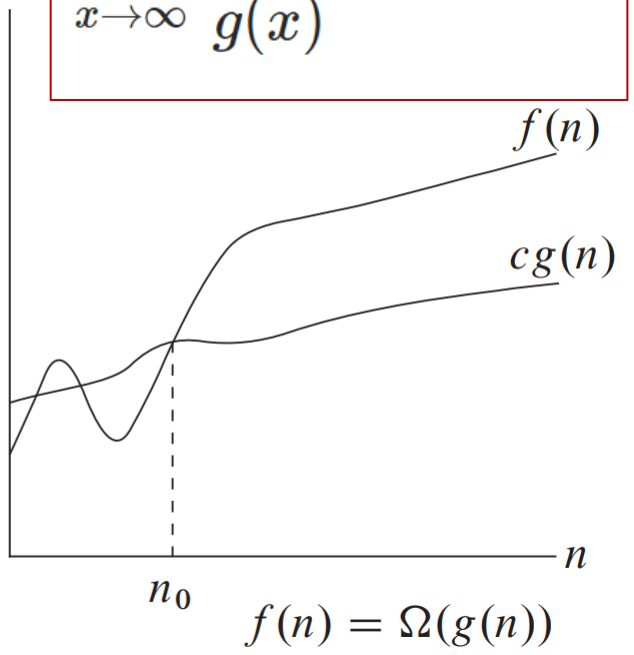
The O-Notation

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





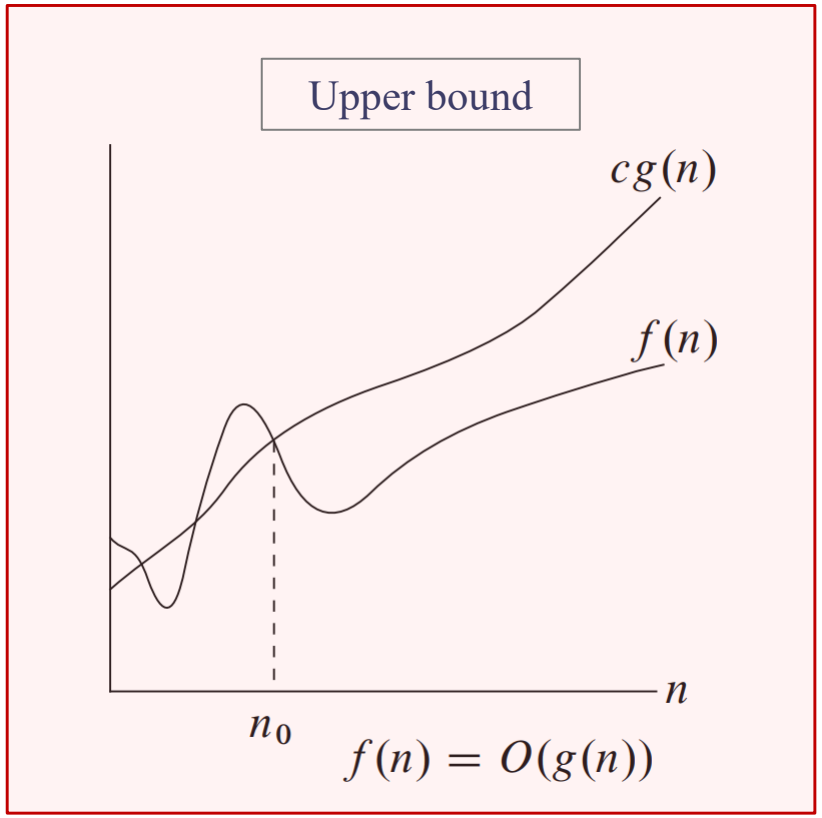
$$\lim_{x \to \infty} \frac{f(x)}{g(x)} = c \text{ or } 0$$



n_0: minimum possible

The O-Notation

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

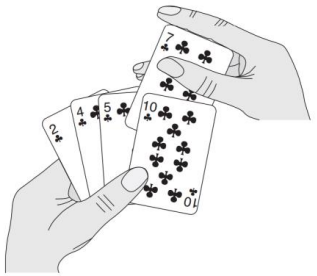


- در برخی منابع از نماد O معادل نماد Θ استفاده می شود
- نماد O : تخمین زمان اجرا از روی ساختار برنامه

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

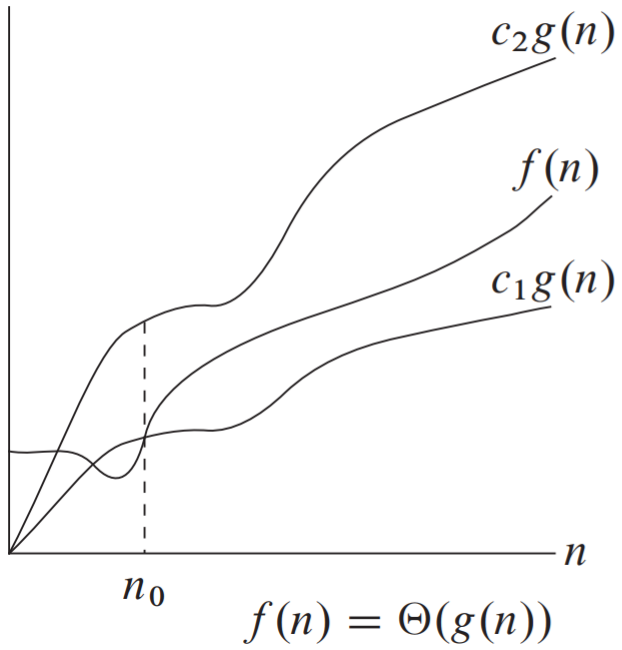
```
INSERTION-SORT(A)
1  for j = 2 to A.length
2      key = A[j]
3      // Insert A[j] into the sorted sequence A[1 . . j − 1].
4      i = j − 1
5      while i > 0 and A[i] > key
6          A[i + 1] = A[i]
7          i = i − 1
8      A[i + 1] = key
```

$O(n^2)$



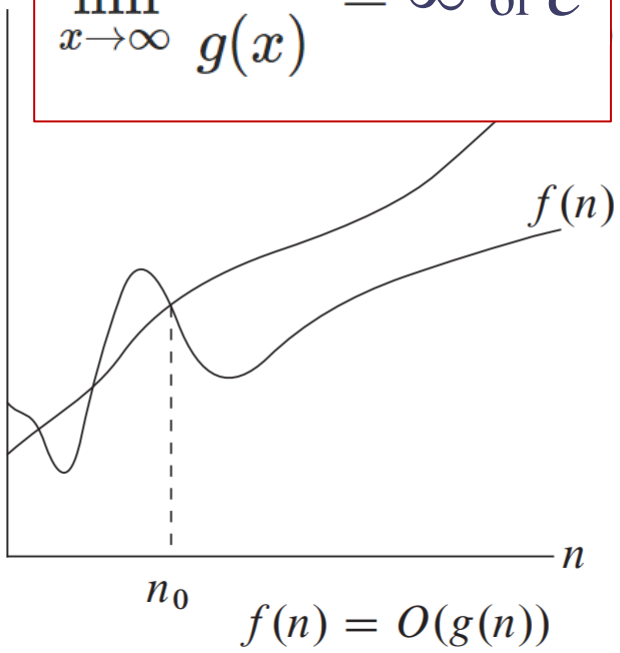
The Ω-Notation

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

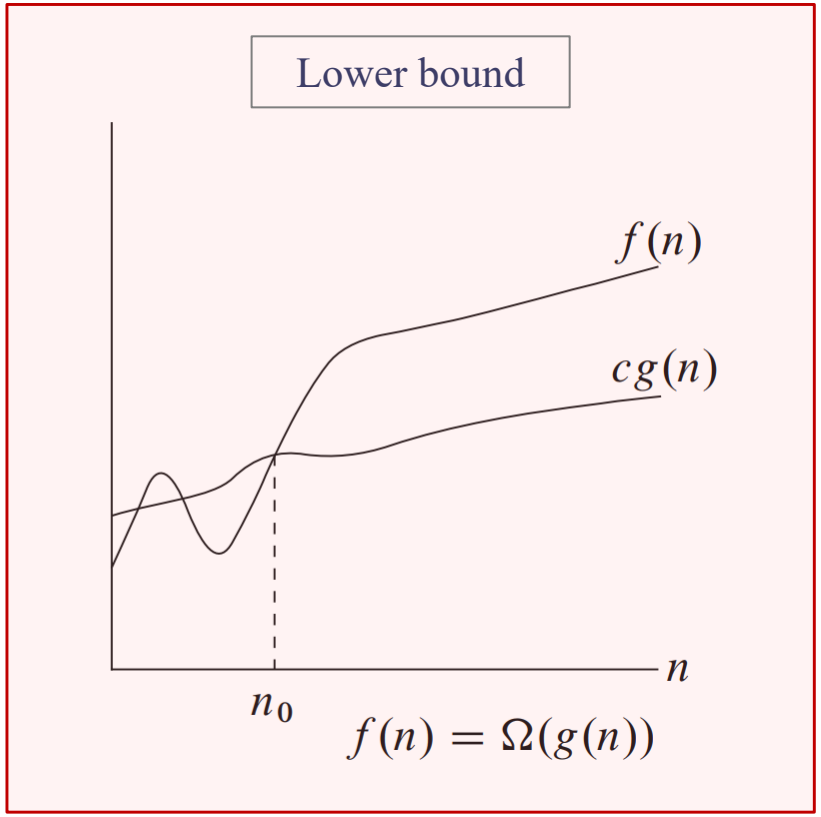


$f(n) = \Theta(g(n))$

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



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

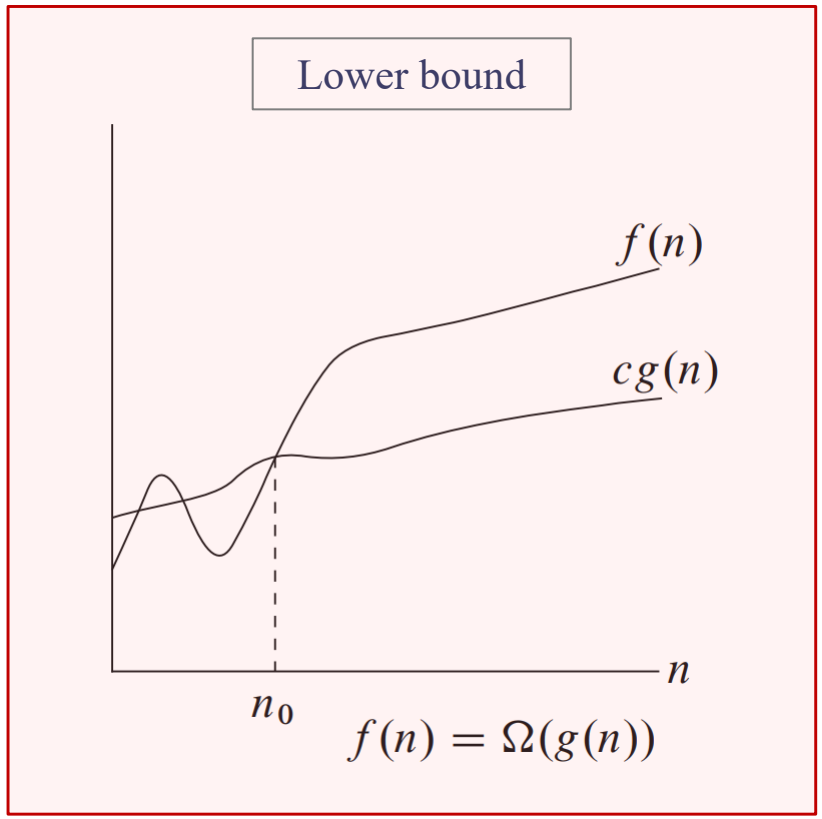


$f(n) = \Omega(g(n))$

n_0: minimum possible

The Ω-Notation

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

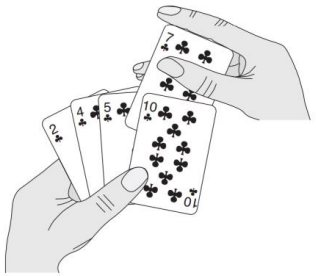


- نماد Ω : تخمین زمان اجرا از روی ساختار برنامه

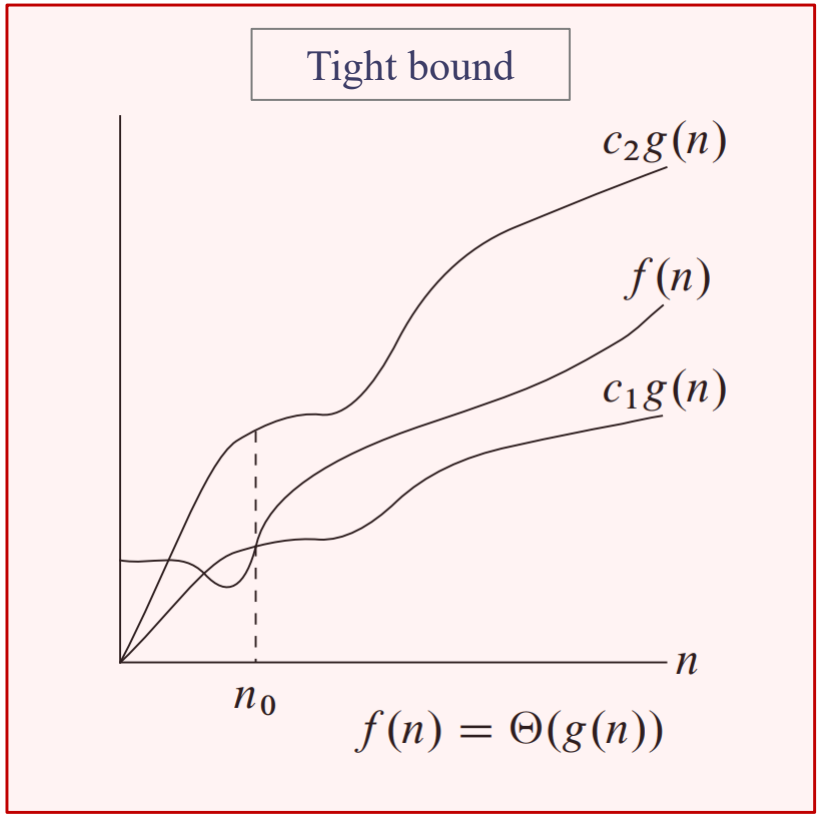
  بر اساس بهترین زمان اجرا

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

$\Omega(n)$



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

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

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

- حضور نماد رشد مجانبی در سمت چپ معادلات

برای همه توابع $f(n) \in \Theta(n)$

وجود دارد تابع $g(n) \in \Theta(n^2)$

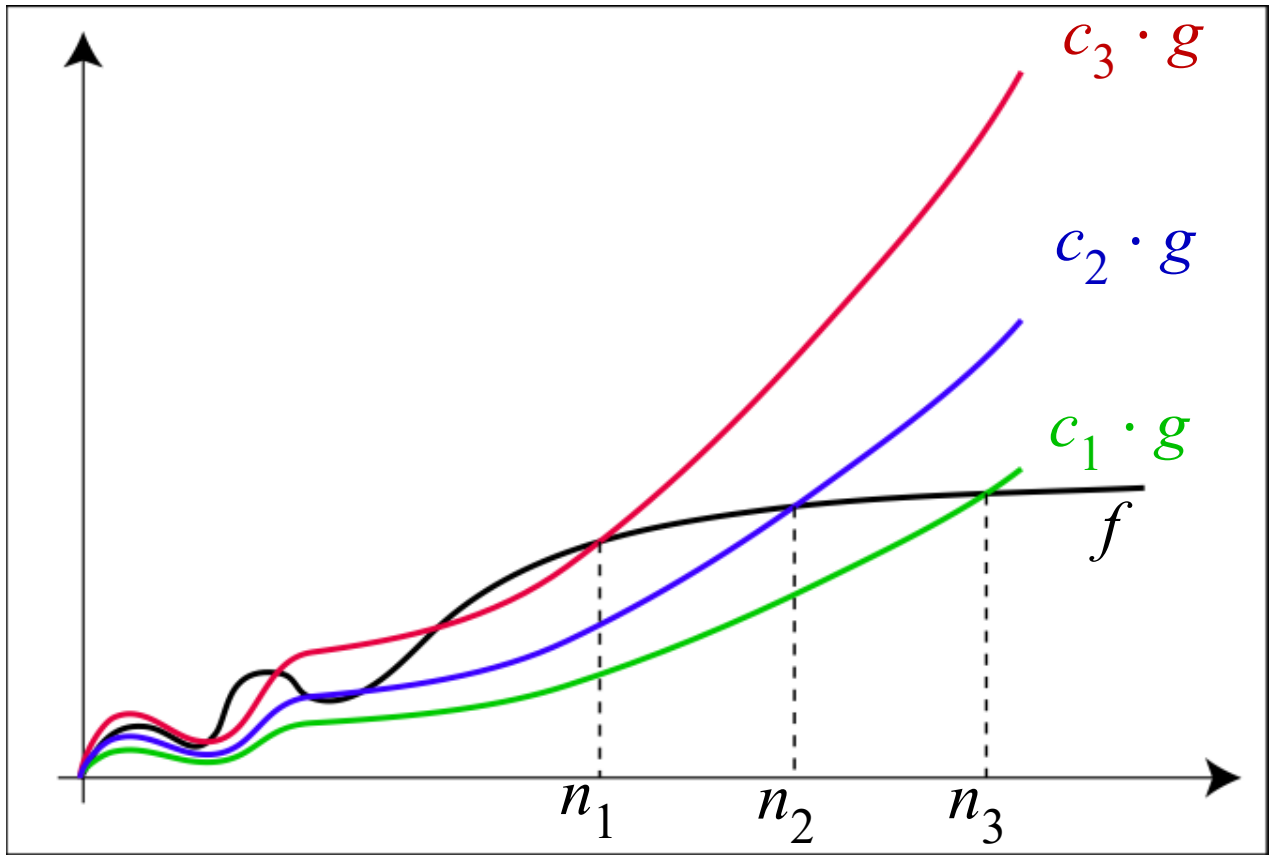
بطوریکه $2n^2 + f(n) = g(n)$

$$2n^2 + \Theta(n) = \Theta(n^2)$$

The o-Notation

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

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



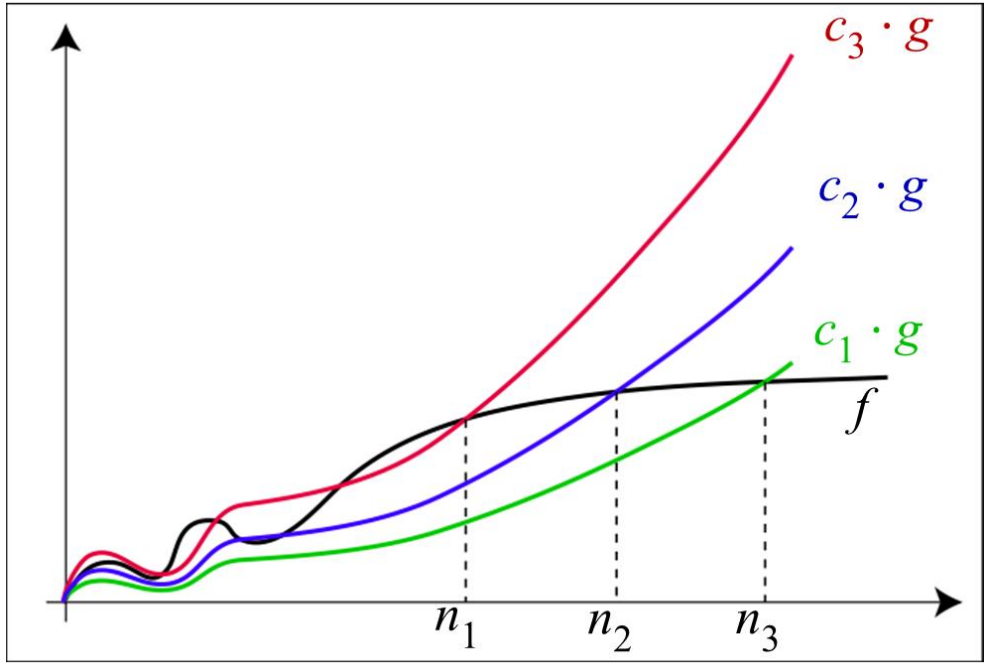
$$\lim_{x \to \infty} \frac{f(x)}{g(x)} = 0$$

$n^{1.9999} = o(n^2)$
$n^2 / \lg n = o(n^2)$
$n^2 \neq o(n^2)$ (just like $2 \not< 2$)
$n^2/1000 \neq o(n^2)$

The o-Notation

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

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

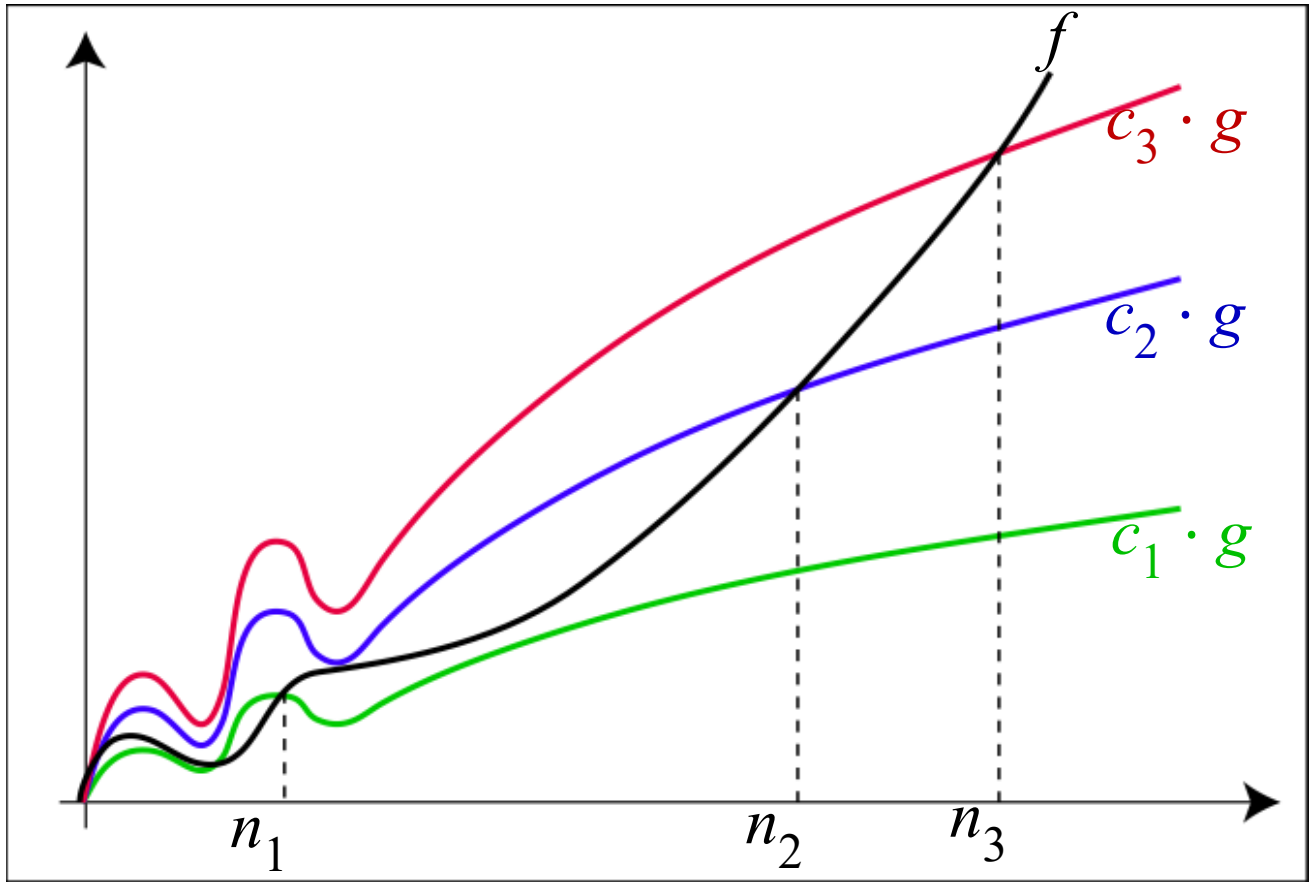


$T_A(n) = o(T_B(n))$ ← الگوریتم A بهتر است الگوریتم B است

The ω-Notation

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

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



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

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

معادل سازی نمادهای رشد مجانبی

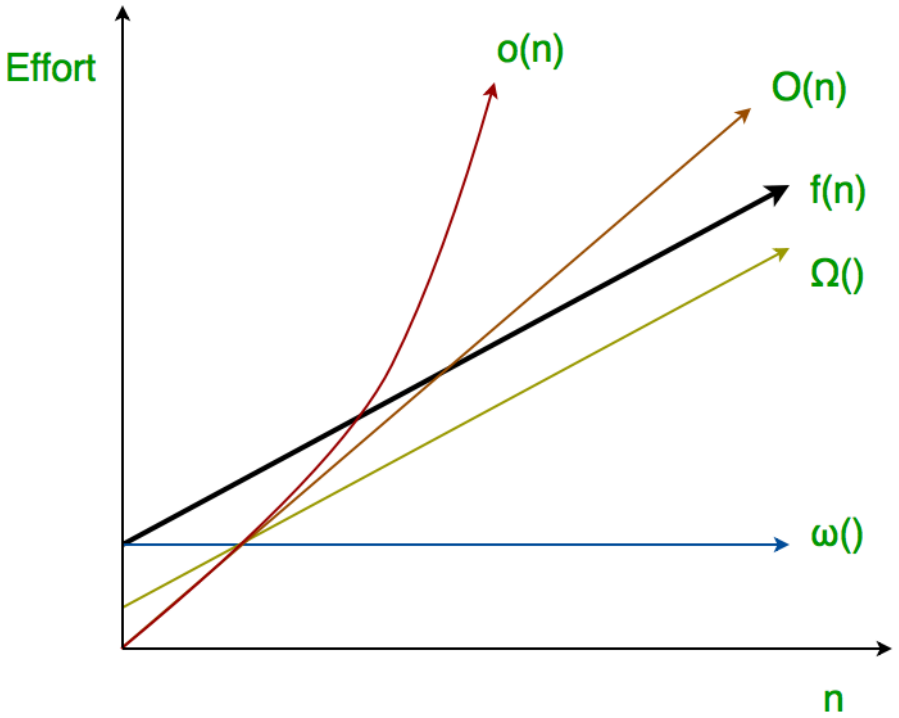
$$f(n) = O(g(n)) \approx a \leq b,$$

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

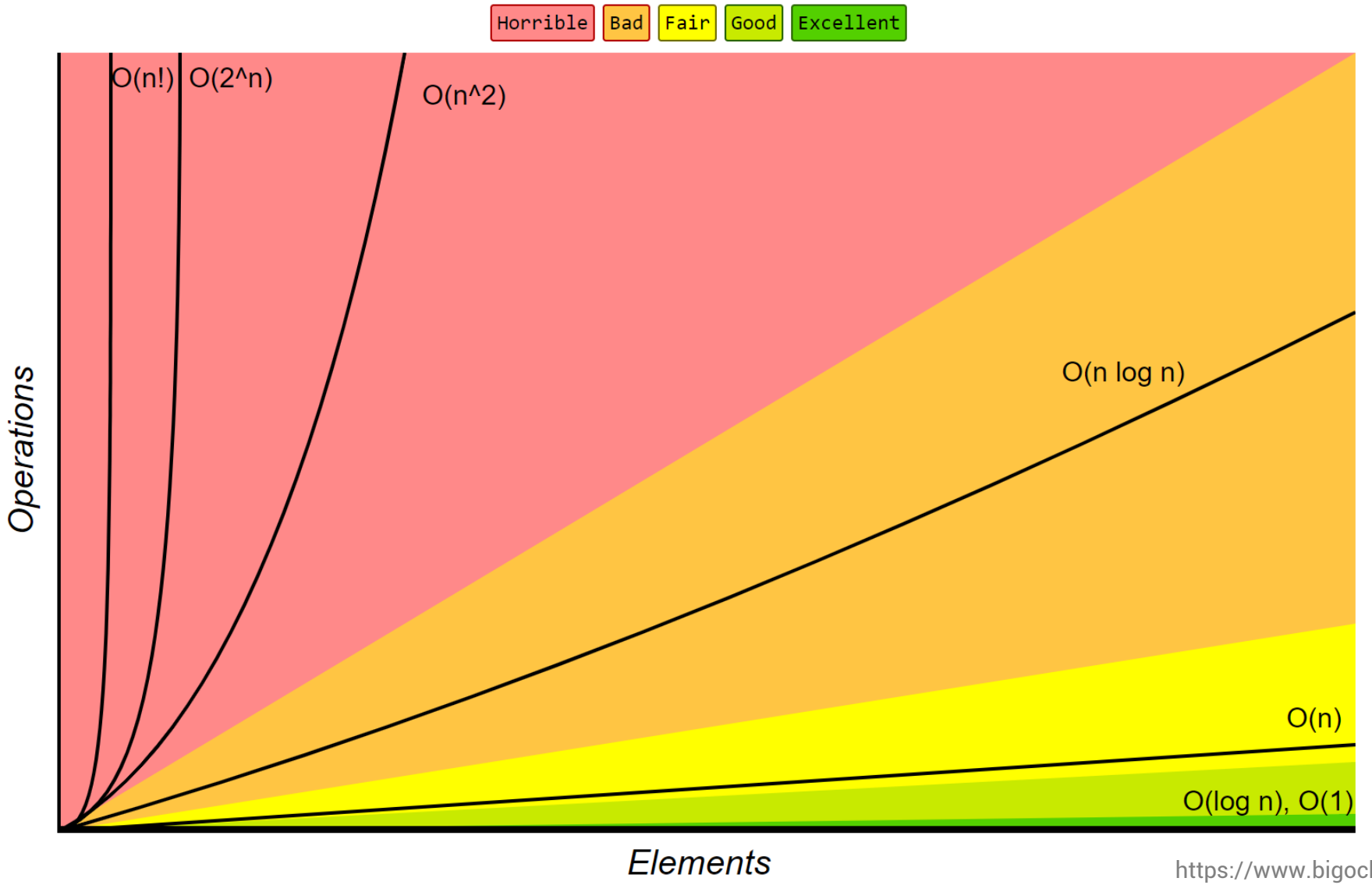
$$f(n) = \Theta(g(n)) \approx a = b,$$

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

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



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

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

لزوما برقرار نیست

تقارنی

Symmetry

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

ضد تقارنی

Transpose Symmetry