Data Structures & Algorithms

21: Asymptotic Notations

(Runtime Analysis)

Dr Ram Prasad Krishnamoorthy

Associate Professor
School of Computing and Data Science

ram.krish@saiuniversity.edu.in

Asymptotic Notations

Asymptotic Analysis

Characterizing Running Times

- A way to describe behavior of functions *in the limit*.
  - Studying *asymptotic efficiency* (for large values of **n**, how algorithms behaves)
- Describes growth of the functions.
- Focus on what's important by abstracting away low-order terms and constant factors.
- How we indicate running times of algorithms.

Asymptotic Analysis

Way to compare size of functions (notations)

$O \approx \leq$ → Big oh of / Order of

$\Omega \approx \geq$ → Big omega of

$\Theta \approx =$ → Theta of

$o \approx <$ → Little oh

$\omega \approx >$ → Little omega

Asymptotic Analysis

Way to compare size of functions (notations)

O	$\approx$	$\leq$	Big oh of / Order of
Ω	$\approx$	$\geq$	Big omega of
Θ	$\approx$	$=$	Theta of
o	$\approx$	$<$	Little oh
ω	$\approx$	$>$	Little omega

Asymptotic Analysis

O-notation

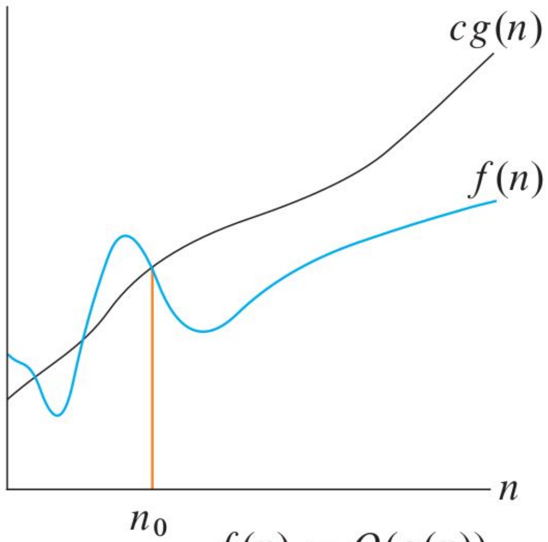
O-notation characterizes an *upper bound* on the asymptotic behavior of a function: it says that a function grows *no faster* than a certain rate. This rate is based on the highest order term.

For example, $\boxed{f(n) = 7n^3 + 100n^2 - 20n + 6 \text{ is } O(n^3)}$, since the highest order term is $7n^3$, and therefore the function grows no faster than n^3.

Asymptotic Analysis

O-notation

$$O(g(n)) = \{f(n) : \text{there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \le f(n) \le cg(n) \text{ for all } n \ge n_0\}.$$



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

Asymptotic Analysis

$g(n)$ is an ***asymptotic upper bound*** for $f(n)$.

If $f(n) \in O(g(n))$, we write $f(n) = O(g(n))$

$3n + 2 = O(n)$

$3n + 2 \leq 4n$ for all $n \geq 2$

$100n + 1 = O(n)$

$100n + 1 \leq 101n$ for all $n \geq 1$

$10n^2 + 4n + 2 = O(n^2)$

$10n^2 + 4n + 2 \leq 11n^2$ for all $n \geq 5$

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

$$\lim_{n\to\infty} \frac{n^2 + n}{n^2}$$

$$\lim_{n\to\infty} 1 + \frac{1}{n}$$

$$1$$

Ω-notation

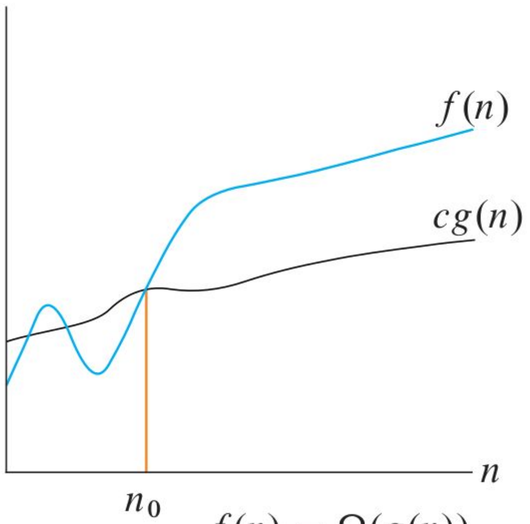
Ω-notation characterizes a *lower bound* on the asymptotic behavior of a function: it says that a function grows *at least as fast* as a certain rate. This rate is again based on the highest-order term.

For example, $\boxed{f(n) = 7n^3 + 100n^2 - 20n + 6 \text{ is } \Omega(n^3)}$, since the highest-order term, n^3, grows at least as fast as n^3.

Asymptotic Analysis

Ω-notation

$$\Omega(g(n)) = \{f(n) : \text{there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \le cg(n) \le f(n) \text{ for all } n \ge n_0\}.$$



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

Asymptotic Analysis

$g(n)$ is an ***asymptotic lower bound*** for $f(n)$.

Example

$\sqrt{n} = \Omega(\lg n)$, with $c = 1$ and $n_0 = 16$.

Examples of functions in $\Omega(n^2)$:

n^2	Also,
$n^2 + n$	n^3
$n^2 - n$	$n^{2.00001}$
$1000n^2 + 1000n$	$n^2 \lg\lg\lg n$
$1000n^2 - 1000n$	2^{2^n}

Θ-notation

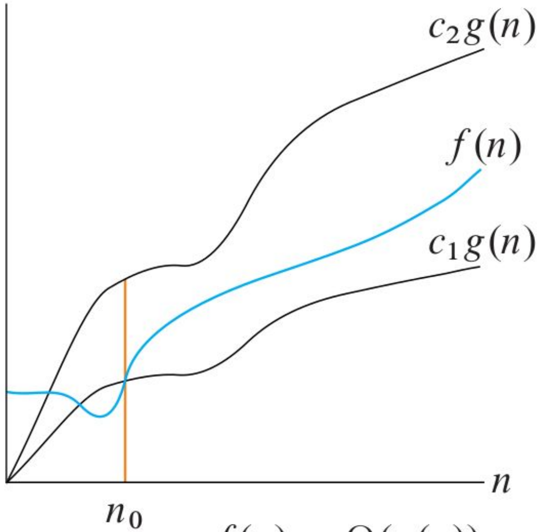
Θ-notation characterizes a *tight bound* on the asymptotic behavior of a function: it says that a function grows *precisely* at a certain rate, again based on the highest-order term.

If a function is both $O(f(n))$ and $\Omega(f(n))$, then a function is $\Theta(f(n))$.

Asymptotic Analysis

Θ-notation

$$\Theta(g(n)) = \{ f(n) : \text{there exist positive constants } c_1, c_2, \text{ and } n_0 \text{ such that } 0 \le c_1 g(n) \le f(n) \le c_2 g(n) \text{ for all } n \ge n_0 \}.$$



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

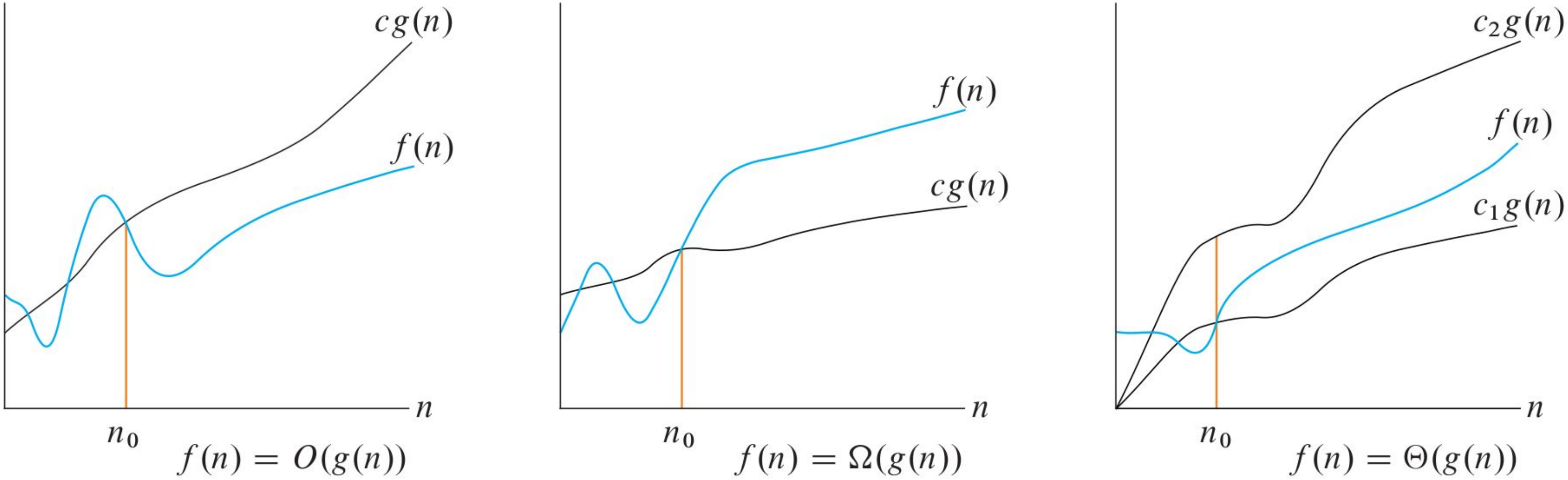
Asymptotic Analysis

$g(n)$ is an ***asymptotically tight bound*** for $f(n)$.

Example

$n^2/2 - 2n = \Theta(n^2)$, with $c_1 = 1/4$, $c_2 = 1/2$, and $n_0 = 8$.

Asymptotic Analysis



Asymptotic Analysis

o-notation

$$o(g(n)) = \{f(n) : \text{for all constants } c > 0, \text{ there exists a constant } n_0 > 0 \text{ such that } 0 \leq f(n) < cg(n) \text{ for all } n \geq n_0\}.$$

Another view, probably easier to use: $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0.$

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

Asymptotic Analysis

ω-notation

$$\omega(g(n)) = \{f(n) : \text{for all constants } c > 0, \text{ there exists a constant } n_0 > 0 \text{ such that } 0 \le cg(n) < f(n) \text{ for all } n \ge n_0\} .$$

Another view, again, probably easier to use: $\lim_{n \to \infty} \frac{f(n)}{g(n)} = \infty.$

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

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

$n^2 \neq \omega(n^2)$

Asymptotic Analysis

Merge Sort

$T(n) = n + n\,\mathit{log}\,n \;\rightarrow O(n\,\mathit{log}\,n)$

Insertion Sort

Best: $an + b \;\rightarrow \square(n)$

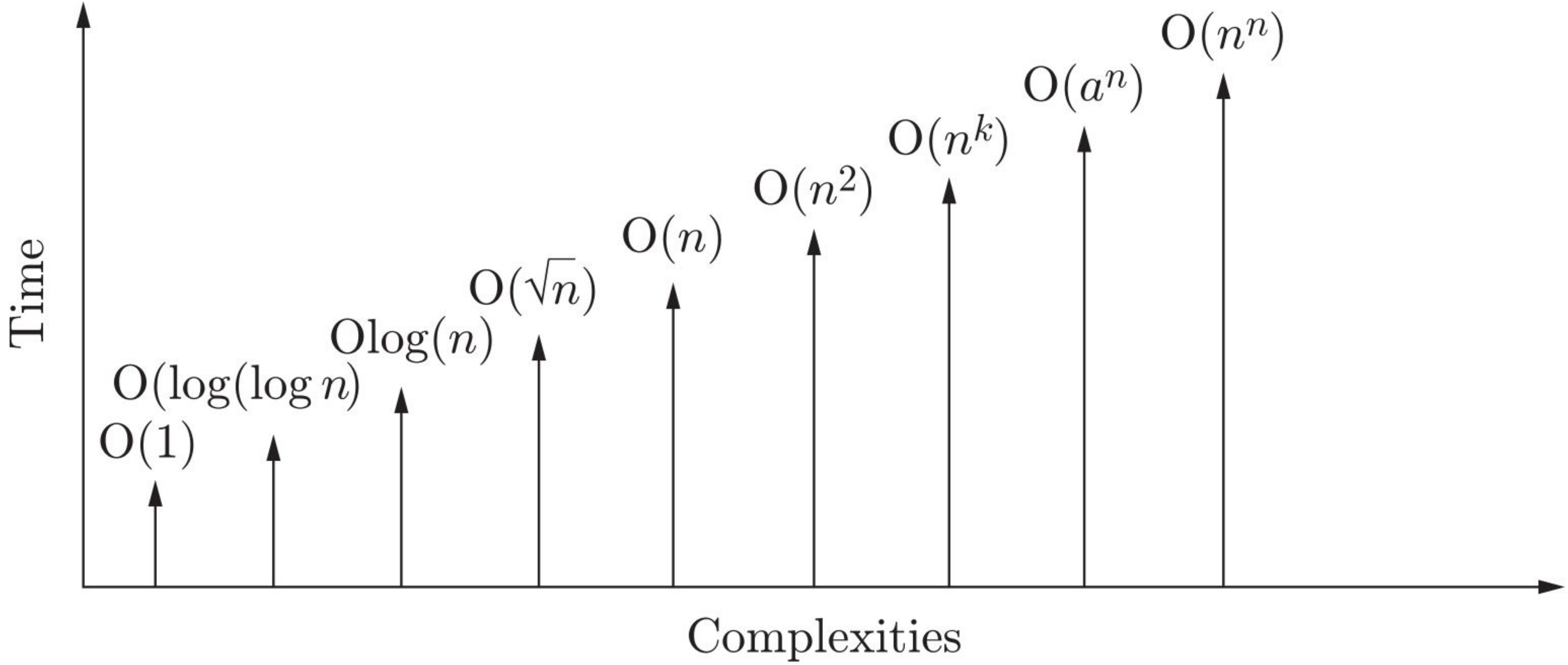
Worst: $an^2 + bn + c \;\rightarrow\; O(n^2)$

Average: $\Theta(n^2)$

Summary

$f = \Theta(g)$	f grows at the same rate as g
$f = O(g)$	f grows no faster than g
$f = \Omega(g)$	f grows at least as fast as g
$f = o(g)$	f grows slower than g
$f = \omega(g)$	f grows faster than g

Asymptotic Analysis



Types of complexity functions

1. Constant time complexity: $O(1)$
2. Logarithmic time complexity: $O(\log(\log n))$, $O(\sqrt{\log n})$ and $O(\log n)$
3. Linear time complexity: $O(\sqrt{n})$ and $O(n)$
4. Polynomial time complexity: $O(n^k)$, where k is a constant and is >1
5. Exponential time complexity: $O(a^n)$, where $a > 1$