Big O notation (and its companions)

Big O Notation

primary use

Describe function behaviour as input gets large

Applications

Computing: run time/storage of algorithms & data structures

Number theory: bounding approximations

Asymptotic notation

	Name	Symbolises	Introduced		Read
o	"Little oh"	Asymptotic loose upper bound	Landau 1909 	$<$	$o(f(n))$ - "order less than $f(n)$"
O	"Big Oh"	Asymptotic upper bound	Bachmann 1894 	$\leq$	$O(f(n))$ - "order at most $f(n)$"
Θ	"Big Theta"	Asymptotic tight bound	Knuth 1976 	$=$	$\Theta(f(n))$ - "order exactly $f(n)$"
Ω	"Big Omega"	Asymptotic lower bound	Knuth 1976 	$\geq$	$\Omega(f(n))$ - "order at least $f(n)$"
ω	"Little omega"	Asymptotic loose lower bound	Knuth 1976 	$>$	$\omega(f(n))$ - "order greater than $f(n)$"

References/Notes/Image credits:
(history of notation) D. E. Knuth, "The art of computer programming, vol. 1: fundamental algorithms", 3rd Ed. (1997)
P. Bachmann, "Die analytische Zahlentheorie" (1894)
(Image of Bachmann) https://commons.wikimedia.org/wiki/File:Paul_Bachmann.jpg
E. Landau, "Handbuch der Lehre yon der Verteilung der Primzahlen" (1909)
https://en.wikipedia.org/wiki/Edmund_Landau#/media/File:Edmund_Landau.jpg
D. E. Knuth, "Big omicron and big omega and big theta", ACM Sigact News 8.2 (1976)
(Donald Ervin Knuth) https://en.wikipedia.org/wiki/Donald_Knuth#/media/File:Donald_Ervin_Knuth_(cropped).jpg
(Mike Paterson) https://simons.berkeley.edu/people/michael-paterson
(Bob Tarjan) https://en.wikipedia.org/wiki/Robert_Tarjan#/media/File:Bob_Tarjan.jpg
(Godfrey Harold Hardy) https://en.wikipedia.org/wiki/G._H._Hardy#/media/File:Ghhardy@72.jpg
(John Edensor Littlewood) https://en.wikipedia.org/wiki/John_Edensor_Littlewood#/media/File:John_Edensor_Littlewood.jpg

Asymptotic upper bounds with $O(\cdot)$

Informal definition: we say that a function $f(n)$ is $O(g(n))$ if it grows no faster than $g(n)$

Examples:

highest order term dominates when n grows large

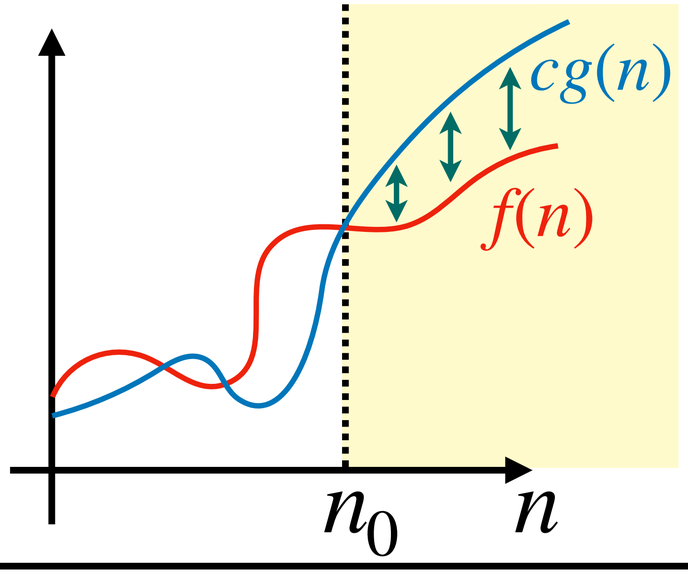
$f(n) = 2n^2 + n + 3$ is $O(n^2)$

$f(n) = 2n^2 + n + 3$ is also $O(n^3)$

Asymptotic upper bound

$f(n)$ never goes above $cg(n)$ whenever $n \geq n_0$

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



Claim: $f(n) = O(n^2)$ find n_0 and c

Pick $n_0 = 10$ then solve for c

$2 \cdot 10^2 + 10 + 3 \leq c \cdot 10^2$

$\implies c \geq 2.13$

$f(n) \leq 2.13n^2$ for all $n \geq 10$

$f(n) \leq cn^2$ for all $n \geq n_0$

$f(n) \in O(g(n))$ if there exist positive constants c and n_0 such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0$

Note: convention is to write $f(n) = O(g(n))$ rather than $f(n) \in O(g(n))$ $= O(\cdot)$ allows some algebra

More formal definition

References
Definitions: https://en.wikipedia.org/wiki/Big_O_notation
(Asymptotic notation) T. Cormen et al., "Introduction to algorithms", Chap 3.2, MIT press (2022)
Figure is based on https://en.wikipedia.org/wiki/Big_O_notation#/media/File:Big-O-notation.png
and similar figures in T. Cormen et al., "Introduction to algorithms", Chap 3.2, MIT press (2022)
D. E. Knuth, "Big omicron and big omega and big theta", ACM Sigact News 8.2 (1976)

Note: we are considering sets of function that are *asymptotically nonnegative* (nonnegative for sufficiently large values of n)

Asymptotic lower and tight bounds

Informal: $f(n)$ grows at least as fast as $g(n)$

Asymptotic lower bound

$f(n) = \Omega(g(n))$ if there exist positive constants c and n_0 such that

$0 \leq cg(n) \leq f(n)$ for all $n \geq n_0$

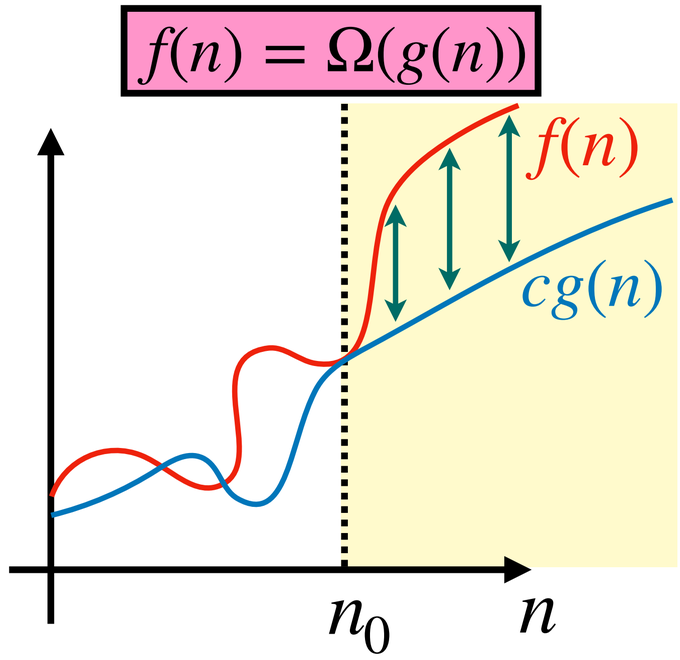
$f(n)$ is of order at least $g(n)$

Examples:

Examine highest order term

$f(n) = 2n^2 + n + 3$ is $\Omega(n^2)$

$f(n) = 2n^2 + n + 3$ is also $\Omega(n)$



Informal: $f(n)$ grows exactly as fast as $g(n)$

Asymptotic tight bound

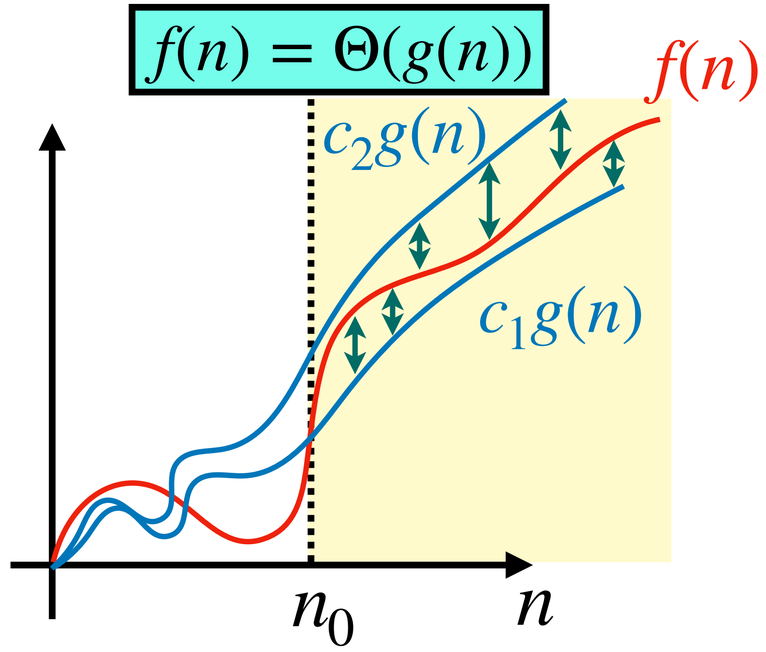
$f(n) = \Theta(g(n))$ if there exist positive constants c_1, c_2 and n_0 such that

$c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all $n \geq n_0$

$f(n)$ is of order exactly $g(n)$

$f(n) = 2n^2 + n + 3$ is $\Theta(n^2)$

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



References

Definitions follow P. Black, "Ω", in "Dictionary of Algorithms and Data Structures" and T. Cormen et al., "Introduction to algorithms", Chap 3.2, MIT press, (2022)

Figures based on https://xlinux.nist.gov/dads/HTML/omegaCapital.html, https://xlinux.nist.gov/dads/HTML/theta.html and similar figures in T. Cormen et al., "Introduction to algorithms", Chap 3.2, MIT press, (2022)

Equivalence of $f = \Theta(g(n))$ to $f = O(g(n))$ and $f = \Theta(g(n))$: D. E. Knuth, "The art of computer programming, vol. 1: fundamental algorithms", 3rd Ed. (1997)

Asymptotic loose bounds

Informal: $f(n)$ grows slower than $g(n)$

Asymptotic loose upper bound

$f(n) = o(g(n))$ if for any $c > 0$ there exists $n_0 > 0$ such that

$0 \le f(n) < cg(n)$ for all $n \ge n_0$

$f(n)$ is of order less than $g(n)$

Example: $f(n) = 2n^2 + n + 3$ is $o(n^3)$ but it is not $o(n^2)$

Informal: $f(n)$ grows faster than $g(n)$

Asymptotic loose lower bound

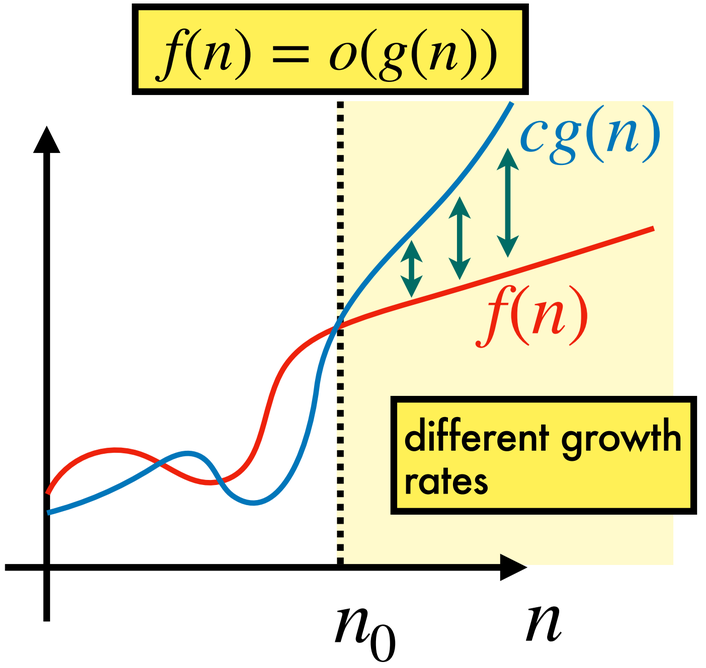
$f(n) = \omega(g(n))$ if for any $c > 0$ there exists $n_0 > 0$ such that

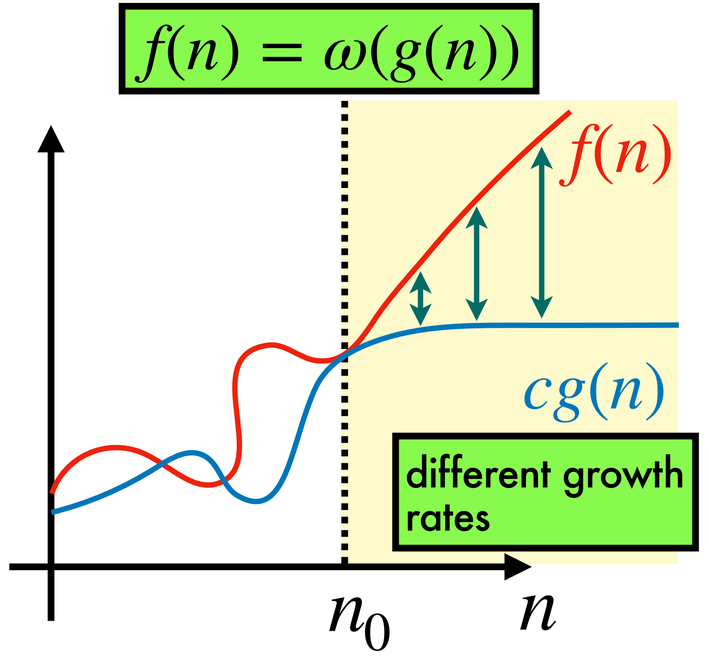
$0 \le cg(n) < f(n)$ for all $n \ge n_0$

$f(n)$ is of order greater than $g(n)$

$f(n) = 2n^2 + n + 3$ is $\omega(n)$ but it is not $\omega(n^2)$

Definition	? $c > 0$	? $n_0 > 0$	$f(n)$? $cg(n)$
$o(\cdot)$	$\forall$	$\exists$	$<$
$O(\cdot)$	$\exists$	$\exists$	$\le$
$\Omega(\cdot)$	$\exists$	$\exists$	$\ge$
$\omega(\cdot)$	$\forall$	$\exists$	$>$





References
L. McCann, http://www2.cs.arizona.edu/classes/cs345/summer14/files/bigO.pdf
T. Cormen et al., "Introduction to algorithms", Chap 3.2, MIT press, (2022)
(Table) http://www2.cs.arizona.edu/classes/cs345/summer14/files/bigO.pdf

O-notation algebra

With O-notation (and other variants) we can perform some algebra (with caution)

Why caution is needed

We can write $2n^2 + n = O(n^2)$ but not $O(n^2) = 2n^2 + n$

Otherwise, since $n^2 = O(n^2)$, we'd have $n^2 = 2n^2 + n$ = is "one-way" equality

Knuth: RHS of equation does not give more information than LHS RHS is a "crudification"

This stems from using $f(n) = O(g(n))$ to denote set membership $f(n) \in O(g(n))$

$O(f(n)) = O(g(n))$ really means $O(f(n)) \subseteq O(g(n))$

Example properties

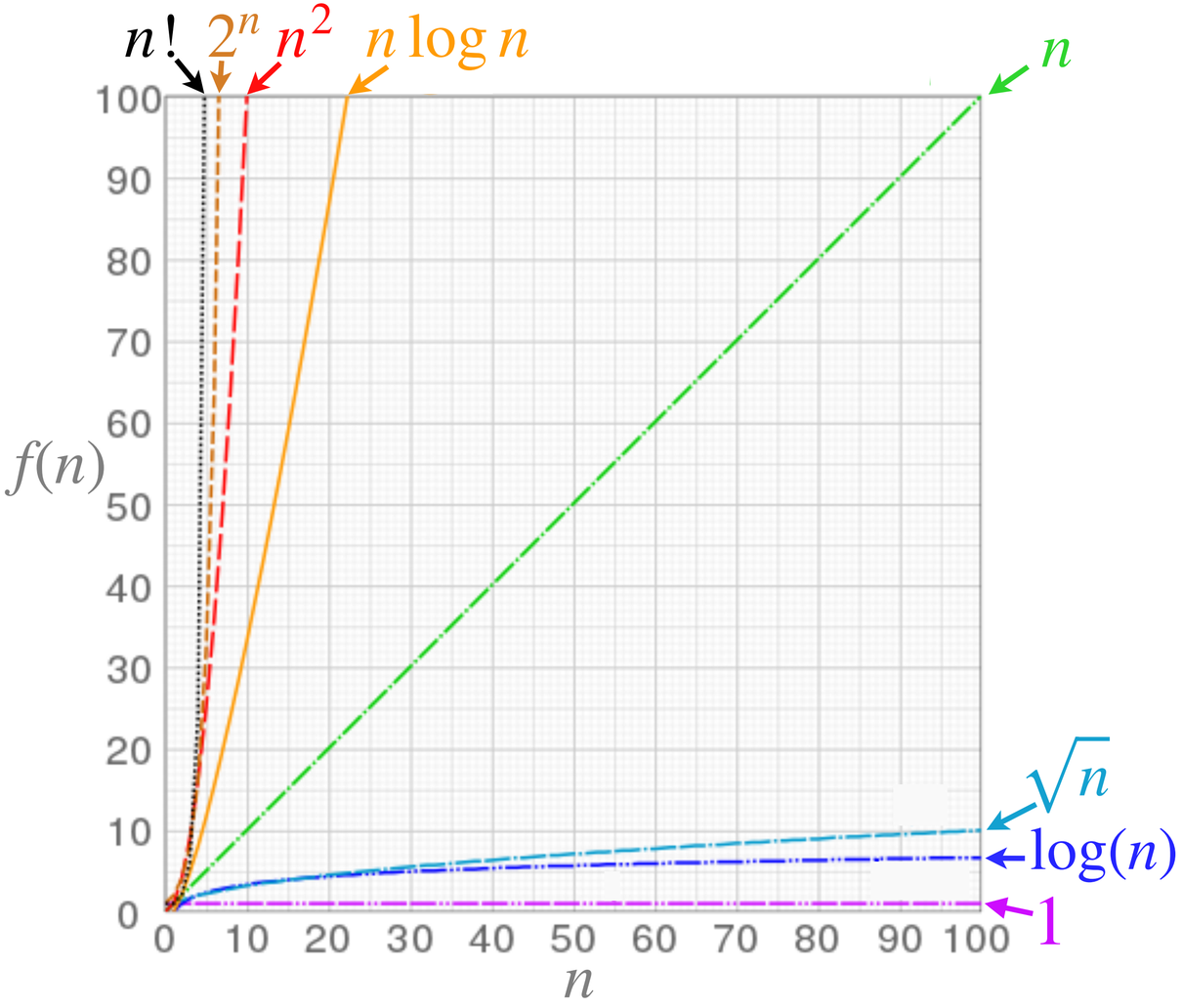
$f(n) = O(f(n))$ $\quad c \cdot f(n) = O(f(n))$ $\quad O(f(n)) + O(f(n)) = O(f(n))$

$f(n) = O(g(n)) \iff g(n) = \Omega(f(n))$ $\quad f(n) = o(g(n)) \iff g(n) = \omega(f(n))$

useful for derivations

References
(one-way equality) D. E. Knuth, "The art of computer programming, vol. 1: fundamental algorithms", Chap 1.2, 3rd Ed. (1997)
T. Cormen et al., "Introduction to algorithms", Chap 3.2, MIT press, (2022)

Functions commonly used for algorithm analysis



$\Theta(n!)$ - factorial

$\Theta(2^n)$ - exponential

$\Theta(n^2)$ - quadratic

$\Theta(n \log(n))$ - linearithmic

$\Theta(n)$ - linear

$\Theta(\sqrt{n})$ - square root

$\Theta(\log(n))$ - logarithmic

$\Theta(1)$ - constant

References/image credit
https://en.wikipedia.org/wiki/Time_complexity
https://upload.wikimedia.org/wikipedia/commons/7/7e/Comparison_computational_complexity.svg

See video description below for links to:

Further resources

Slides

References

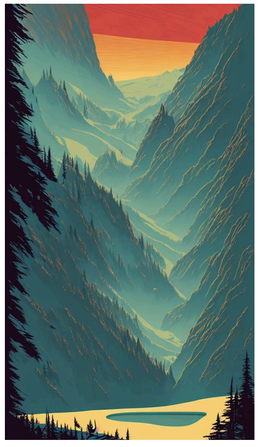


Image credit: Stable diffusion (lexica.art)
https://lexica.art/prompt/16135179-6a39-496a-9a7f-c4a06cdd8ff5