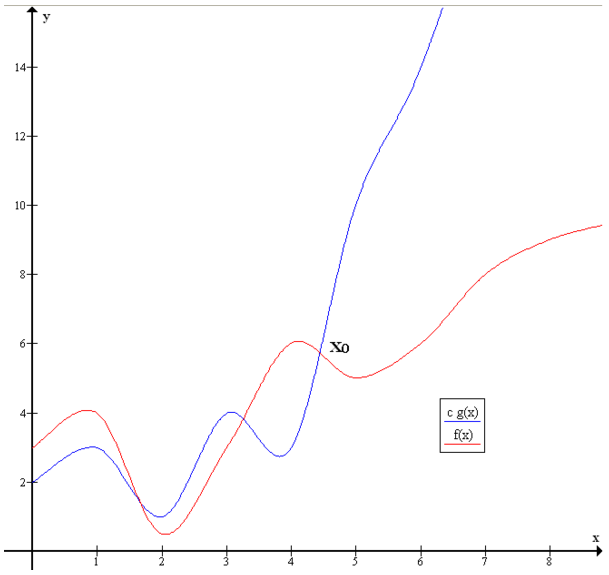
Big *O* notation

Big *O* notation is a mathematical notation that describes the limiting behavior of a function when the argument tends towards a particular value or infinity. Big O is a member of a family of notations invented by Paul Bachmann,[1] Edmund Landau,[2] and others, collectively called **Bachmann–Landau notation** or **asymptotic notation**. The letter O was chosen by Bachmann to stand for *Ordnung*, meaning the order of approximation.



Example of Big O notation: $f(x) \in O(g(x))$ as there exists $M > 0$ (e.g., $M = 1$) and x_0 (e.g., $x_0 = 5$) such that $f(x) \leq Mg(x)$ whenever $x \geq x_0$.

In computer science, big O notation is used to classify algorithms according to how their run time or space requirements grow as the input size grows.[3] In analytic number theory, big O notation is often used to express a bound on the difference between an arithmetical function and a better understood approximation; a famous example of such a difference is the remainder term in the prime number theorem. Big O notation is also used in many other fields to provide similar estimates.

Big O notation characterizes functions according to their growth rates: different functions with the same growth rate may be represented using the same O notation. The letter O is used because the growth rate of a function is also referred to as the **order of the function**. A description of a function in terms of big O notation usually only provides an upper bound on the growth rate of the function.

Associated with big O notation are several related notations, using the symbols o, Ω, ω, and Θ, to describe other kinds of bounds on asymptotic growth rates.

Contents

Formal definition

Let f, the function to be estimated, be a real or complex valued function and let g, the comparison function, be a real valued function. Let both functions be defined on some unbounded subset of the positive real numbers, and $g(x)$ be strictly positive for all large enough values of x.[4] One writes

$$f(x) = O\big(g(x)\big) \quad \text{as } x \to \infty$$

if the absolute value of $f(x)$ is at most a positive constant multiple of $g(x)$ for all sufficiently large values of x. That is, $f(x) = O\big(g(x)\big)$ if there exists a positive real number M and a real number x_0 such that

$$|f(x)| \leq Mg(x) \quad \text{for all } x \geq x_0.$$

In many contexts, the assumption that we are interested in the growth rate as the variable x goes to infinity is left unstated, and one writes more simply that

$$f(x) = O\big(g(x)\big).$$

The notation can also be used to describe the behavior of f near some real number a (often, $a = 0$): we say

$$f(x) = O\big(g(x)\big) \quad \text{as } x \to a$$

if there exist positive numbers δ and M such that for all defined x with $0 < |x - a| < \delta$,

$$|f(x)| \leq Mg(x).$$

As $g(x)$ is chosen to be strictly positive for such values of x, both of these definitions can be unified using the limit superior:

$$f(x) = O\big(g(x)\big) \quad \text{as } x \to a$$

if

$$\limsup_{x \to a} \frac{|f(x)|}{g(x)} < \infty.$$

And in both of these definitions the limit point a (whether ∞ or not) is a cluster point of the domains of f and g, i. e., in every neighbourhood of a there have to be infinitely many points in common. Moreover, as pointed out in the article about the limit inferior and limit superior, the $\limsup_{x \to a}$ (at least on the extended real number line) always exists.

In computer science, a slightly more restrictive definition is common: f and g are both required to be functions from some unbounded subset of the positive integers to the nonnegative real numbers; then $f(x) = O\big(g(x)\big)$ iff there exist positive integer numbers M and n_0 such that $f(n) \leq Mg(n)$ for all $n \geq n_0$.[5]

Example

In typical usage the O notation is asymptotical, that is, it refers to very large x. In this setting, the contribution of the terms that grow "most quickly" will eventually make the other ones irrelevant. As a result, the following simplification rules can be applied:

- If $f(x)$ is a sum of several terms, if there is one with largest growth rate, it can be kept, and all others omitted.
- If $f(x)$ is a product of several factors, any constants (terms in the product that do not depend on x) can be omitted.

For example, let $f(x) = 6x^4 - 2x^3 + 5$, and suppose we wish to simplify this function, using O notation, to describe its growth rate as x approaches infinity. This function is the sum of three terms: $6x^4$, $-2x^3$, and 5. Of these three terms, the one with the highest growth rate is the one with the largest exponent as a function of x, namely $6x^4$. Now one may apply the second rule: $6x^4$ is a product of 6 and x^4 in which the first factor does not depend on x. Omitting this factor results in the simplified form x^4. Thus, we say that $f(x)$ is a "big O" of x^4. Mathematically, we can write $f(x) = O(x^4)$. One may confirm this calculation using the formal definition: let $f(x) = 6x^4 - 2x^3 + 5$ and $g(x) = x^4$. Applying the formal definition from above, the statement that $f(x) = O(x^4)$ is equivalent to its expansion,

$$|f(x)| \leq Mx^4$$

for some suitable choice of x_0 and M and for all $x > x_0$. To prove this, let $x_0 = 1$ and $M = 13$. Then, for all $x > x_0$:

$$\begin{aligned} |6x^4 - 2x^3 + 5| &\leq 6x^4 + |2x^3| + 5 \\ &\leq 6x^4 + 2x^4 + 5x^4 \\ &= 13x^4 \end{aligned}$$

so

$$|6x^4 - 2x^3 + 5| \leq 13x^4.$$

Usage

Big O notation has two main areas of application:

- In mathematics, it is commonly used to describe how closely a finite series approximates a given function, especially in the case of a truncated Taylor series or asymptotic expansion
- In computer science, it is useful in the analysis of algorithms

In both applications, the function $g(x)$ appearing within the $O(\cdot)$ is typically chosen to be as simple as possible, omitting constant factors and lower order terms.

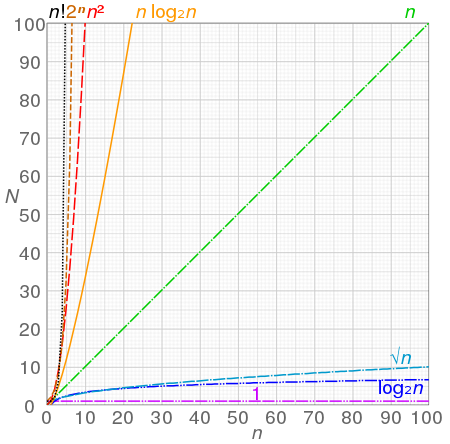
There are two formally close, but noticeably different, usages of this notation:

- infinite asymptotics
- infinitesimal asymptotics.

This distinction is only in application and not in principle, however—the formal definition for the "big O" is the same for both cases, only with different limits for the function argument.

Infinite asymptotics

Big O notation is useful when analyzing algorithms for efficiency. For example, the time (or the number of steps) it takes to complete a problem of size n might be found to be $T(n) = 4n^2 - 2n + 2$. As n grows large, the n^2 term will come to dominate, so that all other terms can be neglected—for instance when $n = 500$, the term $4n^2$ is 1000 times as large as the $2n$ term. Ignoring the latter would have negligible effect on the expression's value for most purposes. Further, the coefficients become irrelevant if we compare to any other order of expression, such as an expression containing a term n^3 or n^4. Even if $T(n) = 1{,}000{,}000n^2$, if $U(n) = n^3$, the latter will always exceed the former once n grows larger than $1{,}000{,}000$ ($T(1{,}000{,}000) = 1{,}000{,}000^3 = U(1{,}000{,}000)$). Additionally, the number of steps depends on the details of the machine model on which the algorithm runs, but different types of machines typically vary by only a constant factor in the number of steps needed to execute an algorithm. So the big O notation captures what remains: we write either



Graphs of functions commonly used in the analysis of algorithms, showing the number of operations N versus input size n for each function

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

or

$$T(n) \in O(n^2)$$

and say that the algorithm has *order of* n^2 time complexity. The sign "=" is not meant to express "is equal to" in its normal mathematical sense, but rather a more colloquial "is", so the second expression is sometimes considered more accurate (see the "Equals sign" discussion below) while the first is considered by some as an abuse of notation.[6]

Infinitesimal asymptotics

Big O can also be used to describe the error term in an approximation to a mathematical function. The most significant terms are written explicitly, and then the least-significant terms are summarized in a single big O term. Consider, for example, the exponential series and two expressions of it that are valid when x is small:

$$\begin{aligned} e^x &= 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots && \text{for all } x \\ &= 1 + x + \frac{x^2}{2} + O(x^3) && \text{as } x \to 0 \\ &= 1 + x + O(x^2) && \text{as } x \to 0 \end{aligned}$$

The second expression (the one with $O(x^3)$) means the absolute-value of the error $e^x - (1 + x + x^2/2)$ is at most some constant times $|x^3|$ when x is close enough to 0.

Properties

If the function f can be written as a finite sum of other functions, then the fastest growing one determines the order of $f(n)$. For example,

$$f(n) = 9\log n + 5(\log n)^4 + 3n^2 + 2n^3 = O(n^3) \qquad \text{as } n \to \infty.$$

In particular, if a function may be bounded by a polynomial in n, then as n tends to *infinity*, one may disregard *lower-order* terms of the polynomial. The sets $O(n^c)$ and $O(c^n)$ are very different. If c is greater than one, then the latter grows much faster. A function that grows faster than n^c for any c is called *superpolynomial*. One that grows more slowly than any exponential function of the form c^n is called *subexponential*. An algorithm can require time that is both superpolynomial and subexponential; examples of this include the fastest known algorithms for integer factorization and the function $n^{\log n}$.

We may ignore any powers of n inside of the logarithms. The set $O(\log n)$ is exactly the same as $O(\log(n^c))$. The logarithms differ only by a constant factor (since $\log(n^c) = c \log n$) and thus the big O notation ignores that. Similarly, logs with different constant bases are equivalent. On the other hand, exponentials with different bases are not of the same order. For example, 2^n and 3^n are not of the same order.

Changing units may or may not affect the order of the resulting algorithm. Changing units is equivalent to multiplying the appropriate variable by a constant wherever it appears. For example, if an algorithm runs in the order of n^2, replacing n by cn means the algorithm runs in the order of c^2n^2, and the big O notation ignores the constant c^2. This can be written as $c^2n^2 = \mathrm{O}(n^2)$. If, however, an algorithm runs in the order of 2^n, replacing n with cn gives $2^{cn} = (2^c)^n$. This is not equivalent to 2^n in general. Changing variables may also affect the order of the resulting algorithm. For example, if an algorithm's run time is $O(n)$ when measured in terms of the number n of *digits* of an input number x, then its run time is $O(\log x)$ when measured as a function of the input number x itself, because $n = O(\log x)$.

Product

$$f_1 = O(g_1) \text{ and } f_2 = O(g_2) \Rightarrow f_1 f_2 = O(g_1 g_2)$$
$$f \cdot O(g) = O(fg)$$

Sum

If $f_1 = O(g_1)$ and $f_2 = O(g_2)$ then $f_1 + f_2 = O(\max(g_1, g_2))$. It follows that if $f_1 = O(g)$ and $f_2 = O(g)$ then $f_1 + f_2 \in O(g)$. In other words, this second statement says that $O(g)$ is a convex cone.

Multiplication by a constant

Let k be a nonzero constant. Then $O(|k| \cdot g) = O(g)$. In other words, if $f = O(g)$, then $k \cdot f = O(g)$.

Multiple variables

Big O (and little o, Ω, etc.) can also be used with multiple variables. To define big O formally for multiple variables, suppose f and g are two functions defined on some subset of $\mathbb{R}^n$. We say

$$f(\mathbf{x}) \text{ is } O(g(\mathbf{x})) \quad \text{as } \mathbf{x} \to \infty$$

if and only if there exist constants M and $C > 0$ such that $|f(\mathbf{x})| \leq C|g(\mathbf{x})|$ for all $\mathbf{x}$ with $x_i \geq M$ for some i.[7] Equivalently, the condition that $x_i \geq M$ for some i can be written $\|\mathbf{x}\|_\infty \geq M$, where $\|\mathbf{x}\|_\infty$ denotes the Chebyshev norm. For example, the statement

$$f(n, m) = n^2 + m^3 + O(n + m) \quad \text{as } n, m \to \infty$$

asserts that there exist constants C and M such that

$$|f(n, m) - (n^2 + m^3)| \leq C|n + m|$$

whenever either $m \geq M$ or $n \geq M$ holds. This definition allows all of the coordinates of $\mathbf{x}$ to increase to infinity. In particular, the statement

$$f(n, m) = O(n^m) \quad \text{as } n, m \to \infty$$

(i.e., $\exists C\, \exists M\, \forall n\, \forall m\, \cdots$) is quite different from

$$\forall m\colon\ f(n, m) = O(n^m) \quad \text{as } n \to \infty$$

(i.e., $\forall m\, \exists C\, \exists M\, \forall n\, \cdots$).

Under this definition, the subset on which a function is defined is significant when generalizing statements from the univariate setting to the multivariate setting. For example, if $f(n, m) = 1$ and $g(n, m) = n$, then $f(n, m) = O(g(n, m))$ if we restrict f and g to $[1, \infty)^2$, but not if they are defined on $[0, \infty)^2$.

This is not the only generalization of big O to multivariate functions, and in practice, there is some inconsistency in the choice of definition.[8]

Matters of notation

Equals sign

The statement "$f(x)$ is $O(g(x))$" as defined above is usually written as $f(x) = O(g(x))$. Some consider this to be an abuse of notation, since the use of the equals sign could be misleading as it suggests a symmetry that this statement does not have. As de Bruijn says, $O(x) = O(x^2)$ is true but $O(x^2) = O(x)$ is not.[9] Knuth describes such statements as "one-way equalities", since if the sides could be reversed, "we could deduce ridiculous things like

$n = n^2$ from the identities $n = O(n^2)$ and $n^2 = O(n^2)$."[10] In another letter, Knuth also pointed out that "the equality sign is not symmetric with respect to such notations", as, in this notation, "mathematicians customarily use the = sign as they use the word "is" in English: Aristotle is a man, but a man isn't necessarily Aristotle".[11]

For these reasons, it would be more precise to use set notation and write $f(x) \in O(g(x))$ (read as: "$f(x)$ *is an element of* $O(g(x))$", or "$f(x)$ *is in the set* $O(g(x))$"), thinking of $O(g(x))$ as the class of all functions $h(x)$ such that $|h(x)| \leq C|g(x)|$ for some constant C.[10] However, the use of the equals sign is customary.[9][10]

Other arithmetic operators

Big O notation can also be used in conjunction with other arithmetic operators in more complicated equations. For example, $h(x) + O(f(x))$ denotes the collection of functions having the growth of $h(x)$ plus a part whose growth is limited to that of $f(x)$. Thus,

$$g(x) = h(x) + O(f(x))$$

expresses the same as

$$g(x) - h(x) = O(f(x)).$$

Example

Suppose an algorithm is being developed to operate on a set of n elements. Its developers are interested in finding a function $T(n)$ that will express how long the algorithm will take to run (in some arbitrary measurement of time) in terms of the number of elements in the input set. The algorithm works by first calling a subroutine to sort the elements in the set and then perform its own operations. The sort has a known time complexity of $O(n^2)$, and after the subroutine runs the algorithm must take an additional $55n^3 + 2n + 10$ steps before it terminates. Thus the overall time complexity of the algorithm can be expressed as $T(n) = 55n^3 + O(n^2)$. Here the terms $2n + 10$ are subsumed within the faster-growing $O(n^2)$. Again, this usage disregards some of the formal meaning of the "=" symbol, but it does allow one to use the big O notation as a kind of convenient placeholder.

Multiple uses

In more complicated usage, $O(\cdot)$ can appear in different places in an equation, even several times on each side. For example, the following are true for $n \to \infty$:

$$\begin{aligned}
(n+1)^2 &= n^2 + O(n), \\
(n + O(n^{1/2})) \cdot (n + O(\log n))^2 &= n^3 + O(n^{5/2}), \\
n^{O(1)} &= O(e^n).
\end{aligned}$$

The meaning of such statements is as follows: for *any* functions which satisfy each $O(\cdot)$ on the left side, there are *some* functions satisfying each $O(\cdot)$ on the right side, such that substituting all these functions into the equation makes the two sides equal. For example, the third equation above means: "For any function $f(n) = O(1)$, there is some function $g(n) = O(e^n)$ such that $n^{f(n)} = g(n)$." In terms of the "set notation" above, the meaning is that the class of functions represented by the left side is a subset of the class of functions represented by the right side. In this use the "=" is a formal symbol that unlike the usual use of "=" is not a symmetric relation. Thus for example $n^{O(1)} = O(e^n)$ does not imply the false statement $O(e^n) = n^{O(1)}$.

Typesetting

Big O is typeset as an italicized uppercase "O", as in the following example: $O(n^2)$.[12][13] In TeX, it is produced by simply typing O inside math mode. Unlike Greek-named Bachmann–Landau notations, it needs no special symbol. Yet, some authors use the calligraphic variant $\mathcal{O}$ instead.[14][15]

Orders of common functions

Here is a list of classes of functions that are commonly encountered when analyzing the running time of an algorithm. In each case, c is a positive constant and n increases without bound. The slower-growing functions are generally listed first.

Notation	Name	Example
$O(1)$	constant	Determining if a binary number is even or odd; Calculating $(-1)^n$; Using a constant-size lookup table
$O(\log\log n)$	double logarithmic	Average number of comparisons spent finding an item using interpolation search in a sorted array of uniformly distributed values
$O(\log n)$	logarithmic	Finding an item in a sorted array with a binary search or a balanced search tree as well as all operations in a binomial heap
$O((\log n)^c)$ $c > 1$	polylogarithmic	Matrix chain ordering can be solved in polylogarithmic time on a parallel random-access machine.
$O(n^c)$ $0 < c < 1$	fractional power	Searching in a k-d tree
$O(n)$	linear	Finding an item in an unsorted list or in an unsorted array; adding two *n*-bit integers by ripple carry
$O(n \log^* n)$	*n* log-star *n*	Performing triangulation of a simple polygon using Seidel's algorithm, or the union–find algorithm. Note that $\log^*(n) = \begin{cases} 0, & \text{if } n \leq 1 \\ 1 + \log^*(\log n), & \text{if } n > 1 \end{cases}$
$O(n \log n) = O(\log n!)$	linearithmic, loglinear, quasilinear, or "*n* log *n*"	Performing a fast Fourier transform; fastest possible comparison sort; heapsort and merge sort
$O(n^2)$	quadratic	Multiplying two *n*-digit numbers by schoolbook multiplication; simple sorting algorithms, such as bubble sort, selection sort and insertion sort; (worst-case) bound on some usually faster sorting algorithms such as quicksort, Shellsort, and tree sort
$O(n^c)$	polynomial or algebraic	Tree-adjoining grammar parsing; maximum matching for bipartite graphs; finding the determinant with LU decomposition
$L_n[\alpha, c] = e^{(c+o(1))(\ln n)^\alpha (\ln\ln n)^{1-\alpha}}$ $0 < \alpha < 1$	L-notation or sub-exponential	Factoring a number using the quadratic sieve or number field sieve
$O(c^n)$ $c > 1$	exponential	Finding the (exact) solution to the travelling salesman problem using dynamic programming; determining if two logical statements are equivalent using brute-force search
$O(n!)$	factorial	Solving the travelling salesman problem via brute-force search; generating all unrestricted permutations of a poset; finding the determinant with Laplace expansion; enumerating all partitions of a set

The statement $f(n) = O(n!)$ is sometimes weakened to $f(n) = O\left(n^n\right)$ to derive simpler formulas for asymptotic complexity. For any $k > 0$ and $c > 0$, $O(n^c (\log n)^k)$ is a subset of $O(n^{c+\varepsilon})$ for any $\varepsilon > 0$, so may be considered as a polynomial with some bigger order.

Related asymptotic notations

Big O is widely used in computer science. Together with some other related notations it forms the family of Bachmann–Landau notations.

Little-o notation

Intuitively, the assertion "$f(x)$ is $o(g(x))$" (read "$f(x)$ is little-o of $g(x)$") means that $g(x)$ grows much faster than $f(x)$. Let as before f be a real or complex valued function and g a real valued function, both defined on some unbounded subset of the positive real numbers, such that $g(x)$ is strictly positive for all large enough values of x. One writes

$$f(x) = o(g(x)) \quad \text{as } x \to \infty$$

if for every positive constant ε there exists a constant x_0 such that

$$|f(x)| \leq \varepsilon g(x) \quad \text{for all } x \geq x_0.$$[16]

For example, one has

$$2x = o(x^2) \text{ and } 1/x = o(1), \quad \text{both as } x \to \infty.$$

The difference between the definition of the big-O notation and the definition of little-o is that while the former has to be true for *at least one* constant M, the latter must hold for *every* positive constant ε, however small.[17] In this way, little-o notation makes a *stronger statement* than the corresponding big-O notation: every function that is little-o of g is also big-O of g, but not every function that is big-O of g is also little-o of g. For example, $2x^2 = O(x^2)$ but $2x^2 \neq o(x^2)$.

As $g(x)$ is nonzero, or at least becomes nonzero beyond a certain point, the relation $f(x) = o(g(x))$ is equivalent to

$$\lim_{x \to \infty} \frac{f(x)}{g(x)} = 0$$ (and this is in fact how Landau[16] originally defined the little-o notation).

Little-o respects a number of arithmetic operations. For example,

if c is a nonzero constant and $f = o(g)$ then $c \cdot f = o(g)$, and
if $f = o(F)$ and $g = o(G)$ then $f \cdot g = o(F \cdot G)$.

It also satisfies a transitivity relation:

if $f = o(g)$ and $g = o(h)$ then $f = o(h)$.

Big Omega notation

Another asymptotic notation is Ω, read "big omega".[18] There are two widespread and incompatible definitions of the statement

$$f(x) = \Omega(g(x)) \text{ as } x \to a,$$

where a is some real number, ∞, or −∞, where f and g are real functions defined in a neighbourhood of a, and where g is positive in this neighbourhood.

The Hardy–Littlewood definition is used mainly in analytic number theory, and the Knuth definition mainly in computational complexity theory; the definitions are not equivalent.

The Hardy–Littlewood definition

In 1914 Godfrey Harold Hardy and John Edensor Littlewood introduced the new symbol Ω,[19] which is defined as follows:

$$f(x) = \Omega(g(x)) \text{ as } x \to \infty \text{ if } \limsup_{x\to\infty} \left|\frac{f(x)}{g(x)}\right| > 0.$$

Thus $f(x) = \Omega(g(x))$ is the negation of $f(x) = o(g(x))$.

In 1916 the same authors introduced the two new symbols Ω_R and Ω_L, defined as:[20]

$$f(x) = \Omega_R(g(x)) \text{ as } x \to \infty \text{ if } \limsup_{x\to\infty} \frac{f(x)}{g(x)} > 0;$$

$$f(x) = \Omega_L(g(x)) \text{ as } x \to \infty \text{ if } \liminf_{x\to\infty} \frac{f(x)}{g(x)} < 0.$$

These symbols were used by Edmund Landau, with the same meanings, in 1924.[21] After Landau, the notations were never used again exactly thus; Ω_R became Ω_+ and Ω_L became Ω_-.

These three symbols $\Omega, \Omega_+, \Omega_-$, as well as $f(x) = \Omega_\pm(g(x))$ (meaning that $f(x) = \Omega_+(g(x))$ and $f(x) = \Omega_-(g(x))$ are both satisfied), are now currently used in analytic number theory.[22][23]

Simple examples

We have

$$\sin x = \Omega(1) \text{ as } x \to \infty,$$

and more precisely

$$\sin x = \Omega_\pm(1) \text{ as } x \to \infty.$$

We have

$$\sin x + 1 = \Omega(1) \text{ as } x \to \infty,$$

and more precisely

$$\sin x + 1 = \Omega_+(1) \text{ as } x \to \infty;$$

however

$$\sin x + 1 \neq \Omega_-(1) \text{ as } x \to \infty.$$

The Knuth definition

In 1976 Donald Knuth published a paper to justify his use of the Ω-symbol to describe a stronger property.[24] Knuth wrote: "For all the applications I have seen so far in computer science, a stronger requirement ... is much more appropriate". He defined

$$f(x) = \Omega(g(x)) \Leftrightarrow g(x) = O(f(x))$$

with the comment: "Although I have changed Hardy and Littlewood's definition of Ω, I feel justified in doing so because their definition is by no means in wide use, and because there are other ways to say what they want to say in the comparatively rare cases when their definition applies."[24]

Family of Bachmann–Landau notations

Notation	Name[24]	Description	Formal definition	Limit definition[25][26][27][24][19]
$f(n) = o(g(n))$	Small O; Small Oh	f is dominated by g asymptotically	$\forall k > 0\, \exists n_0\, \forall n > n_0 \colon \lvert f(n)\rvert < k \cdot g(n)$	$\lim_{n\to\infty} \frac{\lvert f(n)\rvert}{g(n)} = 0$
$f(n) = O(g(n))$	Big O; Big Oh; Big Omicron	$\lvert f\rvert$ is bounded above by g (up to constant factor) asymptotically	$\exists k > 0\, \exists n_0\, \forall n > n_0 \colon \lvert f(n)\rvert \leq k \cdot g(n)$	$\limsup_{n\to\infty} \frac{\lvert f(n)\rvert}{g(n)} < \infty$
$f(n) = \Theta(g(n))$	Big Theta	f is bounded both above and below by g asymptotically	$\exists k_1 > 0\, \exists k_2 > 0\, \exists n_0\, \forall n > n_0 \colon$ $k_1 \cdot g(n) \leq \lvert f(n)\rvert \leq k_2 \cdot g(n)$	$f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$ (Knuth version)
$f(n) \sim g(n)$	On the order of	f is equal to g asymptotically	$\forall \varepsilon > 0\, \exists n_0\, \forall n > n_0 \colon \left\lvert \frac{f(n)}{g(n)} - 1 \right\rvert < \varepsilon$	$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 1$
$f(n) = \Omega(g(n))$	Big Omega in complexity theory (Knuth)	f is bounded below by g asymptotically	$\exists k > 0\, \exists n_0\, \forall n > n_0 \colon \lvert f(n)\rvert \geq k \cdot g(n)$	$\liminf_{n\to\infty} \frac{\lvert f(n)\rvert}{g(n)} > 0$
$f(n) = \omega(g(n))$	Small Omega	f dominates g asymptotically	$\forall k > 0\, \exists n_0\, \forall n > n_0 \colon \lvert f(n)\rvert > k \cdot g(n)$	$\lim_{n\to\infty} \frac{\lvert f(n)\rvert}{g(n)} = \infty$
$f(n) = \Omega(g(n))$	Big Omega in number theory (Hardy–Littlewood)	$\lvert f\rvert$ is not dominated by g asymptotically	$\exists k > 0\, \forall n_0\, \exists n > n_0 \colon \lvert f(n)\rvert \geq k \cdot g(n)$	$\limsup_{n\to\infty} \frac{\lvert f(n)\rvert}{g(n)} > 0$

The limit definitions assume $g(n) > 0$ for sufficiently large n. The table is (partly) sorted from smallest to largest, in the sense that $o, O, \Theta, \sim,$ (Knuth's version of) Ω, ω on functions correspond to $<, \leq, \approx, =, \geq, >$ on the real line[27] (the Hardy-Littlewood version of Ω, however, doesn't correspond to any such description).

Computer science uses the big O, big Theta Θ, little o, little omega ω and Knuth's big Omega Ω notations.[28] Analytic number theory often uses the big O, small o, Hardy–Littlewood's big Omega Ω (with or without the +, − or ± subscripts) and $\sim$ notations.[22] The small omega ω notation is not used as often in analysis.[29]

Use in computer science

Informally, especially in computer science, the big O notation often can be used somewhat differently to describe an asymptotic tight bound where using big Theta Θ notation might be more factually appropriate in a given context. For example, when considering a function $T(n) = 73n^3 + 22n^2 + 58$, all of the following are generally acceptable, but tighter bounds (such as numbers 2 and 3 below) are usually strongly preferred over looser bounds (such as number 1 below).

1. $T(n) = O(n^{100})$
2. $T(n) = O(n^3)$
3. $T(n) = \Theta(n^3)$

The equivalent English statements are respectively:

1. $T(n)$ grows asymptotically no faster than n^{100}
2. $T(n)$ grows asymptotically no faster than n^3
3. $T(n)$ grows asymptotically as fast as n^3.

So while all three statements are true, progressively more information is contained in each. In some fields, however, the big O notation (number 2 in the lists above) would be used more commonly than the big Theta notation (items numbered 3 in the lists above). For example, if $T(n)$ represents the running time of a newly developed algorithm for input size n, the inventors and users of the algorithm might be more inclined to put an upper asymptotic bound on how long it will take to run without making an explicit statement about the lower asymptotic bound.

Other notation

In their book *Introduction to Algorithms*, Cormen, Leiserson, Rivest and Stein consider the set of functions f which satisfy

$$f(n) = O(g(n)) \quad (n \to \infty)\,.$$

In a correct notation this set can, for instance, be called $O(g)$, where

$$O(g) = \{f : \text{there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \leq f(n) \leq cg(n) \text{ for all } n \geq n_0\}.$$[30]

The authors state that the use of equality operator (=) to denote set membership rather than the set membership operator ($\in$) is an abuse of notation, but that doing so has advantages.[6] Inside an equation or inequality, the use of asymptotic notation stands for an anonymous function in the set $O(g)$, which eliminates lower-order terms, and helps to reduce inessential clutter in equations, for example:[31]

$$2n^2 + 3n + 1 = 2n^2 + O(n).$$

Extensions to the Bachmann–Landau notations

Another notation sometimes used in computer science is $\tilde{O}$ (read *soft-O*): $f(n) = \tilde{O}(g(n))$ is shorthand for $f(n) = O(g(n) \log^k n)$ for some k.[32] Some authors write O^* for the same purpose.[33] Essentially, it is big O notation, ignoring logarithmic factors because the growth-rate effects of some other super-logarithmic function indicate a growth-rate explosion for large-sized input parameters that is more important to predicting bad run-time performance than the finer-point effects contributed by the logarithmic-growth factor(s). This notation is often used to obviate the "nitpicking" within growth-rates that are stated as too tightly bounded for the matters at hand (since $\log^k n$ is always $o(n^\varepsilon)$ for any constant k and any $\varepsilon > 0$).

Also the L notation, defined as

$$L_n[\alpha, c] = e^{(c+o(1))(\ln n)^\alpha (\ln \ln n)^{1-\alpha}}$$

is convenient for functions that are between polynomial and exponential in terms of $\ln n$.

Generalizations and related usages

The generalization to functions taking values in any normed vector space is straightforward (replacing absolute values by norms), where f and g need not take their values in the same space. A generalization to functions g taking values in any topological group is also possible. The "limiting process" $x \to x_0$ can also be generalized by introducing an arbitrary filter base, i.e. to directed nets f and g. The o notation can be used to define derivatives and differentiability in quite general spaces, and also (asymptotical) equivalence of functions,

$$f \sim g \iff (f - g) \in o(g)$$

which is an equivalence relation and a more restrictive notion than the relationship "f is $\Theta(g)$" from above. (It reduces to $\lim f / g = 1$ if f and g are positive real valued functions.) For example, $2x$ is $\Theta(x)$, but $2x - x$ is not $o(x)$.

History (Bachmann–Landau, Hardy, and Vinogradov notations)

The symbol O was first introduced by number theorist Paul Bachmann in 1894, in the second volume of his book *Analytische Zahlentheorie* ("analytic number theory").[1] The number theorist Edmund Landau adopted it, and was thus inspired to introduce in 1909 the notation o;[2] hence both are now called Landau symbols. These notations were used in applied mathematics during the 1950s for asymptotic analysis.[34] The symbol Ω (in the sense "is not an o of") was introduced in 1914 by Hardy and Littlewood.[19] Hardy and Littlewood also introduced in 1916 the symbols Ω_R ("right") and Ω_L ("left"),[20] precursors of the modern symbols Ω_+ ("is not smaller than a small o of") and Ω_- ("is not larger than a small o of"). Thus the Omega symbols (with their original meanings) are sometimes also referred to as "Landau symbols". This notation Ω became commonly used in number theory at least since the 1950s.[35] In the 1970s the big O was popularized in computer science by Donald Knuth, who introduced the related Theta notation, and proposed a different definition for the Omega notation.[24]

Landau never used the big Theta and small omega symbols.

Hardy's symbols were (in terms of the modern O notation)

$$f \preccurlyeq g \iff f \in O(g) \quad \text{and} \quad f \prec g \iff f \in o(g);$$

(Hardy however never defined or used the notation $\prec\prec$, nor $\ll$, as it has been sometimes reported). Hardy introduced the symbols $\preccurlyeq$ and $\prec$ (as well as some other symbols) in his 1910 tract "Orders of Infinity", and made use of them only in three papers (1910–1913). In his nearly 400 remaining papers and books he consistently used the Landau symbols O and o.

Hardy's notation is not used anymore. On the other hand, in the 1930s,[36] the Russian number theorist Ivan Matveyevich Vinogradov introduced his notation $\ll$, which has been increasingly used in number theory instead of the O notation. We have

$$f \ll g \iff f \in O(g),$$

and frequently both notations are used in the same paper.

The big-O originally stands for "order of" ("Ordnung", Bachmann 1894), and is thus a Latin letter. Neither Bachmann nor Landau ever call it "Omicron". The symbol was much later on (1976) viewed by Knuth as a capital omicron,[24] probably in reference to his definition of the symbol Omega. The digit zero should not be used.

See also

- Asymptotic expansion: Approximation of functions generalizing Taylor's formula
- Asymptotically optimal algorithm: A phrase frequently used to describe an algorithm that has an upper bound asymptotically within a constant of a lower bound for the problem
- Big O in probability notation: O_p, o_p
- Limit superior and limit inferior: An explanation of some of the limit notation used in this article
- Master theorem (analysis of algorithms): For analyzing divide-and-conquer recursive algorithms using Big O notation

- Nachbin's theorem: A precise method of bounding complex analytic functions so that the domain of convergence of integral transforms can be stated
- Orders of approximation
- Computational complexity of mathematical operations

References and notes

1. Bachmann, Paul (1894). *Analytische Zahlentheorie* (https://archive.org/stream/dieanalytischeza00bachuoft#page/402/mode/2up) [*Analytic Number Theory*] (in German). Vol. 2. Leipzig: Teubner.
2. Landau, Edmund (1909). *Handbuch der Lehre von der Verteilung der Primzahlen* (https://archive.org/details/handbuchderlehre01landuoft) [*Handbook on the theory of the distribution of the primes*] (in German). Leipzig: B. G. Teubner. p. 883.
3. Mohr, Austin. "Quantum Computing in Complexity Theory and Theory of Computation" (http://www.austinmohr.com/Work_files/complexity.pdf) (PDF). p. 2. Retrieved 7 June 2014.
4. Landau, Edmund (1909). *Handbuch der Lehre von der Verteilung der Primzahlen* (https://archive.org/stream/handbuchderlehre01landuoft#page/31/mode/2up) [*Handbook on the theory of the distribution of the primes*] (in German). Leipzig: B.G. Teubner. p. 31.
5. Michael Sipser (1997). *Introduction to the Theory of Computation*. Boston/MA: PWS Publishing Co. Here: Def.7.2, p.227
6. Cormen,Thomas H.; Leiserson, Charles E.; Rivest, Ronald L. (2009). *Introduction to Algorithms* (https://archive.org/details/introductiontoal00corm_805) (3rd ed.). Cambridge/MA: MIT Press. p. 45 (https://archive.org/details/introductiontoal00corm_805/page/n65). ISBN 978-0-262-53305-8. "Because $\theta(g(n))$ is a set, we could write "$f(n) \in \theta(g(n))$" to indicate that $f(n)$ is a member of $\theta(g(n))$. Instead, we will usually write $f(n) = \theta(g(n))$ to express the same notion. You might be confused because we abuse equality in this way, but we shall see later in this section that doing so has its advantages."
7. Cormen, Thomas; Leiserson, Charles; Rivest, Ronald; Stein, Clifford (2009). *Introduction to Algorithms* (https://archive.org/details/introductiontoal00corm_805) (Third ed.). MIT. p. 53 (https://archive.org/details/introductiontoal00corm_805/page/n73).
8. Howell, Rodney. "On Asymptotic Notation with Multiple Variables" (http://people.cis.ksu.edu/~rhowell/asymptotic.pdf) (PDF). Retrieved 2015-04-23.
9. N. G. de Bruijn (1958). *Asymptotic Methods in Analysis* (https://books.google.com/books?id=_tnwmvHmVwMC&q=%22The+trouble+is%22&pg=PA5). Amsterdam: North-Holland. pp. 5–7. ISBN 978-0-486-64221-5.
10. Graham, Ronald; Knuth, Donald; Patashnik, Oren (1994). *Concrete Mathematics* (https://books.google.com/books?id=pntQAAAAMAAJ) (2 ed.). Reading, Massachusetts: Addison–Wesley. p. 446. ISBN 978-0-201-55802-9.
11. Donald Knuth (June–July 1998). "Teach Calculus with Big O" (https://www.ams.org/notices/199806/commentary.pdf) (PDF). *Notices of the American Mathematical Society*. **45** (6): 687. (Unabridged version (http://www-cs-staff.stanford.edu/~knuth/ocalc.tex))
12. Donald E. Knuth, The art of computer programming. Vol. 1. Fundamental algorithms, third edition, Addison Wesley Longman, 1997. Section 1.2.11.1.
13. Ronald L. Graham, Donald E. Knuth, and Oren Patashnik, *Concrete Mathematics: A Foundation for Computer Science (2nd ed.)*, Addison-Wesley, 1994. Section 9.2, p. 443.
14. Sivaram Ambikasaran and Eric Darve, An $\mathcal{O}(N \log N)$ Fast Direct Solver for Partial Hierarchically Semi-Separable Matrices, *J. Scientific Computing* **57** (2013), no. 3, 477–501.
15. Saket Saurabh and Meirav Zehavi, $(k, n-k)$-Max-Cut: An $\mathcal{O}^*(2^p)$-Time Algorithm and a Polynomial Kernel, *Algorithmica* **80** (2018), no. 12, 3844–3860.
16. Landau, Edmund (1909). *Handbuch der Lehre von der Verteilung der Primzahlen* (https://archive.org/stream/handbuchderlehre01landuoft#page/61/mode/2up) [*Handbook on the theory of the distribution of the primes*] (in German). Leipzig: B. G. Teubner. p. 61.
17. Thomas H. Cormen et al., 2001, Introduction to Algorithms, Second Edition, Ch. 3.1 (http://highered.mcgraw-hill.com/sites/0070131511/)

18. Cormen TH, Leiserson CE, Rivest RL, Stein C (2009). *Introduction to algorithms* (https://www.worldcat.org/oclc/676697295) (3rd ed.). Cambridge, Mass.: MIT Press. p. 48. ISBN 978-0-262-27083-0. OCLC 676697295 (https://www.worldcat.org/oclc/676697295).
19. Hardy, G. H.; Littlewood, J. E. (1914). "Some problems of diophantine approximation: Part II. The trigonometrical series associated with the elliptic θ-functions" (http://projecteuclid.org/download/pdf_1/euclid.acta/1485887376). *Acta Mathematica*. **37**: 225. doi:10.1007/BF02401834 (https://doi.org/10.1007%2FBF02401834).
20. G. H. Hardy and J. E. Littlewood, « Contribution to the theory of the Riemann zeta-function and the theory of the distribution of primes », *Acta Mathematica*, vol. 41, 1916.
21. E. Landau, "Über die Anzahl der Gitterpunkte in gewissen Bereichen. IV." Nachr. Gesell. Wiss. Gött. Math-phys. Kl. 1924, 137–150.
22. Aleksandar Ivić. The Riemann zeta-function, chapter 9. John Wiley & Sons 1985.
23. Gérald Tenenbaum, Introduction to analytic and probabilistic number theory, Chapter I.5. American Mathematical Society, Providence RI, 2015.
24. Knuth, Donald (April–June 1976). "Big Omicron and big Omega and big Theta" (http://www.phil.uu.nl/datastructuren/10-11/knuth_big_omicron.pdf) (PDF). *SIGACT News*. **8** (2): 18–24. doi:10.1145/1008328.1008329 (https://doi.org/10.1145%2F1008328.1008329). S2CID 5230246 (https://api.semanticscholar.org/CorpusID:5230246).
25. Balcázar, José L.; Gabarró, Joaquim. "Nonuniform complexity classes specified by lower and upper bounds" (http://archive.numdam.org/article/ITA_1989__23_2_177_0.pdf) (PDF). *RAIRO – Theoretical Informatics and Applications – Informatique Théorique et Applications*. **23** (2): 180. ISSN 0988-3754 (https://www.worldcat.org/issn/0988-3754). Retrieved 14 March 2017.
26. Cucker, Felipe; Bürgisser, Peter (2013). "A.1 Big Oh, Little Oh, and Other Comparisons". *Condition: The Geometry of Numerical Algorithms*. Berlin, Heidelberg: Springer. pp. 467–468. doi:10.1007/978-3-642-38896-5 (https://doi.org/10.1007%2F978-3-642-38896-5). ISBN 978-3-642-38896-5.
27. Vitányi, Paul; Meertens, Lambert (April 1985). "Big Omega versus the wild functions" (http://www.kestrel.edu/home/people/meertens/publications/papers/Big_Omega_contra_the_wild_functions.pdf) (PDF). *ACM SIGACT News*. **16** (4): 56–59. CiteSeerX 10.1.1.694.3072 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.694.3072). doi:10.1145/382242.382835 (https://doi.org/10.1145%2F382242.382835). S2CID 11700420 (https://api.semanticscholar.org/CorpusID:11700420).
28. Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford (2001) [1990]. *Introduction to Algorithms* (2nd ed.). MIT Press and McGraw-Hill. pp. 41–50. ISBN 0-262-03293-7.
29. for example it is omitted in: Hildebrand, A.J. "Asymptotic Notations" (http://www.math.uiuc.edu/~ajh/595ama/ama-ch2.pdf) (PDF). Department of Mathematics. *Asymptotic Methods in Analysis*. Math 595, Fall 2009. Urbana, IL: University of Illinois. Retrieved 14 March 2017.
30. Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L. (2009). *Introduction to Algorithms* (3rd ed.). Cambridge/MA: MIT Press. p. 47. ISBN 978-0-262-53305-8. "When we have only an asymptotic upper bound, we use O-notation. For a given function $g(n)$, we denote by $O(g(n))$ (pronounced "big-oh of g of n" or sometimes just "oh of g of n") the set of functions $O(g(n)) = \{ f(n)$: there exist positive constants c and n_0 such that $0 \le f(n) \le cg(n)$ for all $n \ge n_0\}$"
31. Cormen,Thomas H.; Leiserson, Charles E.; Rivest, Ronald L. (2009). *Introduction to Algorithms* (https://archive.org/details/introductiontoal00corm_805) (3rd ed.). Cambridge/MA: MIT Press. p. 49 (https://archive.org/details/introductiontoal00corm_805/page/n69). ISBN 978-0-262-53305-8. "When the asymptotic notation stands alone (that is, not within a larger formula) on the right-hand side of an equation (or inequality), as in $n = O(n^2)$, we have already defined the equal sign to mean set membership: $n \in O(n^2)$. In general, however, when asymptotic notation appears in a formula, we interpret it as standing for some anonymous function that we do not care to name. For example, the formula $2n^2 + 3n + 1 = 2n^2 + \theta(n)$ means that $2n^2 + 3n + 1 = 2n^2 + f(n)$, where $f(n)$ is some function in the set $\theta(n)$. In this case, we let $f(n) = 3n + 1$, which is indeed in $\theta(n)$. Using asymptotic notation in this manner can help eliminate inessential detail and clutter in an equation."
32. *Introduction to algorithms* (https://archive.org/details/introductiontoal00corm_805). Cormen, Thomas H. (Third ed.). Cambridge, Mass.: MIT Press. 2009. p. 63 (https://archive.org/details/introductiontoal00corm_805/page/n83). ISBN 978-0-262-27083-0. OCLC 676697295 (https://www.worldcat.org/oclc/676697295).

33. Andreas Björklund and Thore Husfeldt and Mikko Koivisto (2009). "Set partitioning via inclusion-exclusion" (https://www.cs.helsinki.fi/u/mkhkoivi/publications/sicomp-2009.pdf) (PDF). *SIAM Journal on Computing*. **39** (2): 546–563. See sect.2.3, p.551.
34. Erdelyi, A. (1956). *Asymptotic Expansions*. ISBN 978-0-486-60318-6.
35. E. C. Titchmarsh, The Theory of the Riemann Zeta-Function (Oxford; Clarendon Press, 1951)
36. See for instance "A new estimate for *G*(*n*) in Waring's problem" (Russian). Doklady Akademii Nauk SSSR 5, No 5-6 (1934), 249–253. Translated in English in: Selected works / Ivan Matveevič Vinogradov; prepared by the Steklov Mathematical Institute of the Academy of Sciences of the USSR on the occasion of his 90th birthday. Springer-Verlag, 1985.

Further reading

- Hardy, G. H. (1910). *Orders of Infinity: The 'Infinitärcalcül' of Paul du Bois-Reymond* (https://archive.org/details/ordersofinfinity00harduoft). Cambridge University Press.
- Knuth, Donald (1997). "1.2.11: Asymptotic Representations". *Fundamental Algorithms*. The Art of Computer Programming. Vol. 1 (3rd ed.). Addison-Wesley. ISBN 978-0-201-89683-1.
- Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford (2001). "3.1: Asymptotic notation". *Introduction to Algorithms* (2nd ed.). MIT Press and McGraw-Hill. ISBN 978-0-262-03293-3.
- Sipser, Michael (1997). *Introduction to the Theory of Computation* (https://archive.org/details/introductiontoth00sips_928). PWS Publishing. pp. 226 (https://archive.org/details/introductiontoth00sips_928/page/n239)–228. ISBN 978-0-534-94728-6.
- Avigad, Jeremy; Donnelly, Kevin (2004). *Formalizing O notation in Isabelle/HOL* (http://www.andrew.cmu.edu/~avigad/Papers/bigo.pdf) (PDF). International Joint Conference on Automated Reasoning. doi:10.1007/978-3-540-25984-8_27 (https://doi.org/10.1007%2F978-3-540-25984-8_27).
- Black, Paul E. (11 March 2005). Black, Paul E. (ed.). "big-O notation" (https://xlinux.nist.gov/dads/HTML/bigOnotation.html). *Dictionary of Algorithms and Data Structures*. U.S. National Institute of Standards and Technology. Retrieved December 16, 2006.
- Black, Paul E. (17 December 2004). Black, Paul E. (ed.). "little-o notation" (https://xlinux.nist.gov/dads/HTML/littleOnotation.html). *Dictionary of Algorithms and Data Structures*. U.S. National Institute of Standards and Technology. Retrieved December 16, 2006.
- Black, Paul E. (17 December 2004). Black, Paul E. (ed.). "Ω" (https://xlinux.nist.gov/dads/HTML/omegaCapital.html). *Dictionary of Algorithms and Data Structures*. U.S. National Institute of Standards and Technology. Retrieved December 16, 2006.
- Black, Paul E. (17 December 2004). Black, Paul E. (ed.). "ω" (https://xlinux.nist.gov/dads/HTML/omega.html). *Dictionary of Algorithms and Data Structures*. U.S. National Institute of Standards and Technology. Retrieved December 16, 2006.
- Black, Paul E. (17 December 2004). Black, Paul E. (ed.). "Θ" (https://xlinux.nist.gov/dads/HTML/theta.html). *Dictionary of Algorithms and Data Structures*. U.S. National Institute of Standards and Technology. Retrieved December 16, 2006.

External links

- Growth of sequences — OEIS (Online Encyclopedia of Integer Sequences) Wiki (http://oeis.org/wiki/Growth_of_sequences)
- Introduction to Asymptotic Notations (https://classes.soe.ucsc.edu/classes/cmps102/Spring04/TantaloAsymp.pdf)
- Landau Symbols (http://mathworld.wolfram.com/LandauSymbols.html)
- Big-O Notation – What is it good for (http://www.perlmonks.org/?node_id=573138)
- Big O Notation explained in plain english (https://stackoverflow.com/questions/487258/what-is-a-plain-english-explanation-of-big-o-notation/50288253#50288253)
- An example of Big O in accuracy of central divided difference scheme for first derivative (https://autarkaw.org/2013/01/30/making-sense-of-the-big-oh/) Archived (https://web.archive.org/web/20181007223123/https://a

utarkaw.org/2013/01/30/making-sense-of-the-big-oh/) 2018-10-07 at the Wayback Machine
- A Gentle Introduction to Algorithm Complexity Analysis (https://discrete.gr/complexity/)
