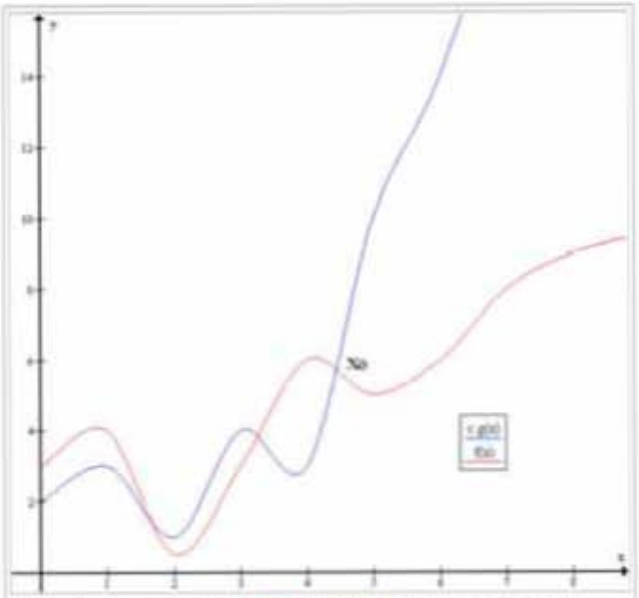
Big O notation

From Wikipedia, the free encyclopedia

In mathematics, **big O notation** describes the limiting behavior of a function when the argument tends towards a particular value or infinity, usually in terms of simpler functions. It is a member of a larger family of notations that is called **Landau notation**, **Bachmann–Landau notation** (after Edmund Landau and Paul Bachmann),[1][2] or **asymptotic notation**. In computer science, big O notation is used to classify algorithms by how they respond (*e.g.*, in their processing time or working space requirements) to changes in input size.[3] In analytic number theory, it is used to estimate the "error committed" while replacing the asymptotic size, or asymptotic mean size, of an arithmetical function, by the value, or mean value, it takes at a large finite argument. A famous example is the problem of estimating the remainder term in the prime number theorem.



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

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 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.

Big O notation is also used in many other fields to provide similar estimates.

Contents

NETWORK-1 EXHIBIT 2009
Google Inc. v. Network-1 Technologies, Inc.

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.

Formal definition

Let f and g be two functions defined on some subset of the real numbers. One writes

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

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

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

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

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

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

if and only if there exist positive numbers δ and M such that

$$|f(x)| \leq M|g(x)| \text{ for } |x - a| < \delta.$$

If $g(x)$ is non-zero for values of x sufficiently close to a, both of these definitions can be unified using the limit superior:

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

if and only if

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

Additionally, the notation $O(g(x))$ is also used to denote the set of all functions $f(x)$ that satisfy the relation $f(x)=O(g(x))$. In this case we write

$$f(x) \in O(g(x))$$

Example

In typical usage, the formal definition of O notation is not used directly; rather, the O notation for a function f is derived by the following simplification rules:

- If $f(x)$ is a sum of several terms, the one with the largest growth rate is 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) are 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

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.

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-oh" 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,

$$|f(x)| \leq M|g(x)|$$

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 $O(...)$ 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 and 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

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

or

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

and say that the algorithm has *order of* n^2 time complexity. Note that "=" is not meant to express "is equal to" in its normal mathematical sense, but rather a more colloquial "is", so the second expression is technically accurate (see the "Equals sign" discussion below) while the first is considered by some as an abuse of notation.[4]

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:

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.

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

The second expression (the one with $O(x^3)$) means the absolute-value of the error $e^x - (1 + x + x^2/2)$ is smaller than 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

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

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. Another thing to notice is 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 $c^2n^2 = O(n^2)$. 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

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

Sum

$$f_1 = O(g_1) \text{ and } f_2 = O(g_2) \Rightarrow f_1 + f_2 = O(|g_1| + |g_2|)$$

This implies $f_1 = O(g) \text{ and } f_2 = O(g) \Rightarrow f_1 + f_2 \in O(g)$, which means that $O(g)$ is a convex cone.

If f and g are positive functions, $f + O(g) = O(f + g)$

Multiplication by a constant

Let k be a constant. Then:
$O(kg) = O(g)$ if k is nonzero.
$f \in O(g) \Rightarrow kf = O(g)$.

Multiple variables

Big O (and little o, and Ω...) can also be used with multiple variables. To define Big O formally for multiple variables, suppose $f(\vec{x})$

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.

and $g(\vec{x})$ are two functions defined on some subset of $\mathbb{R}^n$. We say

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

if and only if[5]

$$\exists M\, \exists C > 0 \text{ such that for all } \vec{x} \text{ with } x_i \geq M \text{ for some } i, |f(\vec{x})| \leq C|g(\vec{x})|.$$

Equivalently, the condition that $x_i \geq M$ for some i can be replaced with the condition that $\|\vec{x}\| \geq M$, where $\|\vec{x}\|$ denotes the Chebyshev distance. For example, the statement

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

asserts that there exist constants C and M such that

$$\forall \|(n,m)\| \geq M : |g(n,m)| \leq C(n+m),$$

where $g(n,m)$ is defined by

$$f(n,m) = n^2 + m^3 + g(n,m).$$

Note that this definition allows all of the coordinates of $\vec{x}$ to increase to infinity. In particular, the statement

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

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

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

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

This is not the only generalization of big O to multivariate functions, and in practice, there is some inconsistency in the choice of definition.[6]

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.[7] 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 $n^2 = O(n^2)$."[8] For these reasons, it would be more precise to use set notation and write $f(x) \in O(g(x))$, 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.[8] However, the use of the equals sign is customary. Knuth pointed out that "mathematicians customarily use the = sign as they use the word 'is' in English: Aristotle is a man, but a man isn't necessarily Aristotle."[9]

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

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.