GROWTH OF FUNCTIONS AND ASYMPTOTIC NOTATION

Asymptotic notation is shorthand used to give a quick measure of the behavior of a function $f(n)$ as n grows large. Here, we focus on what's important by abstracting away low-order terms and constant factors.

O-notation

Big Oh is the most frequently used asymptotic notation. It is used to give an **upper bound** on the growth of a function, such as the running time of an algorithm.

Definition: Given functions $f, g: \mathbb{R} \rightarrow \mathbb{R}$, we say that

$$f = O(g)$$

iff there exists a constant $c \geq 0$ and an n_0 such that for all $n \geq n_0$, $|f(n)| \leq cg(n)$.

$O(g(n))$ is actually a set of functions, given by

$$O\big(g(n)\big) = \{\, f(n)\text{: there exist positive constants } c \text{ and } n_0 \text{ such that}$$

$$0 \leq f(n) \leq cg(n) \text{ for all } n \geq n_0 \}.$$

Thus, the notation $f = O(g)$ means that $f \in O(g)$ and should not be understood as $f(n)$ equals $O(\mathrm{g}(x))$.

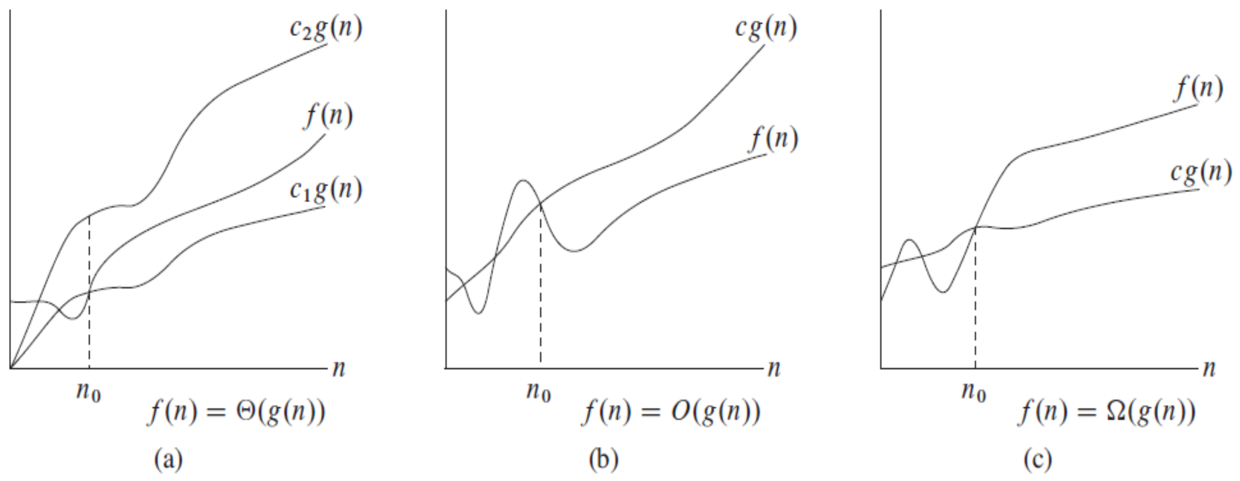

Figure: Graphic examples of the big oh, omega and theta notations.In each part, the value of n_0 shown is the minimum possible value; any greater value would also work.

Examples:

$1.\ 4x^2 + 5x = O(x^2)$ *because* $4x^2 + 5x \leq 9x^2\ \forall x \geq 0$ *, that is, for* $c = 9$ *and* $n_0 = 0$.

$2. 5x^3 + 5x^2 + 5 = O(x^3)\ because\ 5x^3 + 5x^2 + 5 \leq 15x^3\ \forall x \geq 0\ , that\ is, for\ c = 15\ and\ n_0 = 0.$

$3. 3\log n + 2 = O(\log n) because\ 3\log n + 2 \leq 3\log n + 2\log n \leq 5\log n\ \ \forall n \geq 2$ and c=5.

Note that c and n_0 are not unique (infinite number of such pairs exist if f(n)=O(g(n)) and it is not compulsory to take their minimum possible value. You can take any other value that makes the inequality true when n grows large.

$4. if, f(n) = \frac{n(n-1)}{2}, then\ \ show\ that\ f(n) = O(n^2)?$

$$Solution : We\ have, \frac{n(n-1)}{2} = \frac{n^2-n}{2}$$

$$= \frac{n^2}{2} - \frac{n}{2}$$

$$\leq \frac{n^2}{2}$$

$$\leq \frac{1}{2}n^2\ \forall n \geq 0.$$

$Thus, f(n) = O(n^2)\ for\ c = \frac{1}{2} and\ n_0 = 0(other\ values\ for\ c\ and\ n_0\ also\ exist)$

It can be easily seen that any logarithmic or linear function is in $O(n^2)$. For example,$an + b = O(n^2), 5\log n = O(n^2)$.Similarly, any logarithmic or linear or quadratic function is in $O(n^2)$. This is because big oh gives only the upper bound on a function.

The following theorem illustrates that if degree of a polynomial function is n, then it is $O(x^n)$.

Theorem Let $f(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$, where $a_0, a_1, \ldots, a_{n-1}, a_n$ are real numbers. Then $f(x)$ is $O(x^n)$.

Proof: Using the triangle inequality, if $x > 1$ we have

$$\begin{aligned}
|f(x)| &= |a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0| \\
&\leq |a_n|x^n + |a_{n-1}|x^{n-1} + \cdots + |a_1|x + |a_0| \\
&= x^n \left(|a_n| + |a_{n-1}|/x + \cdots + |a_1|/x^{n-1} + |a_0|/x^n\right) \\
&\leq x^n \left(|a_n| + |a_{n-1}| + \cdots + |a_1| + |a_0|\right).
\end{aligned}$$

This shows that

$$|f(x)| \leq Cx^n,$$

Using big oh notation, the running time of an algorithm is often described by just inspecting the algorithms overall structure. For example, the doubly nested loop structure of Insertion sort immediately yields the $O(n^2)$ upper bound on the worst case running time.

Since O- notation describes an upper bound on a function, we use it to bound the worst case running time of an algorithm. For example, when it is given that the running time of an algorithm is $O(n^2)$, we mean that the worst case running time is $O(n^2)$.

Ω-notation (big omega)

Suppose you want to make a statement of the form "the running time of the algorithm is a least. . . ". Can you say it is "at least $O(n^2)$"? No! This statement is meaningless since big-oh can only be used for upper bounds. For **lower bounds**, we use a different symbol, called "big-Omega."

Definition: Given functions $f, g: \mathbb{R} \to \mathbb{R}$, we say that

$$f = \Omega(g)$$

iff there exists a constant $c \geq 0$ and an n_0 such that for all $n \geq n_0$, we have $f(n) \geq c|g(n)|$.

$\Omega(g(n))$ is actually a set of functions given by,

$$\Omega\big(g(n)\big) = \{\, f(n) \text{: there exist positive constants } c \text{ and } n_0 \text{ such that}$$

$$0 \leq cg(n) \leq f(n) \text{ for all } n \geq n_0 \,\}$$

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

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

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

***θ*-notation**

Sometimes we want to specify that a running time $T(n)$ is precisely quadratic up to constant factors (both upper bound and lower bound). We could do this by saying that
$T(n) = O(n^2) and\ T(n)\ \Omega(n^2)$ but rather than say both, mathematicians have devised yet another symbol, θ, to do the job.

Definition: $f = \theta(g)$ $\quad iff \quad$ $f = O(g)\ and\ f = \Omega(g)$

Equivalently, for a given function $g(n)$,we denote by $\theta(g(n))$ the set of functions

$$\theta\big(g(n)\big) = \{\, f(n)\text{: there exist positive constants } c_1,\ c_2 \text{ and } n_0 \text{ such that}$$

$$0 \le c_1 g(n) \le f(n) \le c_2 g(n) \text{ for all } n \ge n_0 \,\}$$

For example, if the running time of an algorithm is $T(n) = 10n^3 - 20n^2 + 1$, then we can more simply write $T(n) = \theta(n^3)$. Note that $10n^3 - 20n^2 + 1$ is both $O(n^3)$ *and* $\Omega(\mathrm{n}^3)$.

o-notation (Little oh)

We use o-notation to denote an upper bound that is not asymptotically tight. The set $o(g(n))$ is defined as:

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

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

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

$n! = o(n^n)$

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

Example: Show that $2^{n+1} \ne o(2^n)$?

Solution: we have $\lim_{n\to\infty} \frac{2^{n+1}}{2^n} = \lim_{n\to\infty} \frac{2.2^n}{2^n} = 2 \ne 0$. Thus, $2^{n+1} \ne o(2^n)$.

The definitions of O-notation and o-notation are similar. The main difference is that in $f(n) = O(g(n))$, the bound $0 \le f(n) \le cg(n)$ holds for *some* constant c > 0, but in $f(n) = o(g(n))$, the bound $0 \le f(n) \le cg(n)$ holds for *all* constants c > 0.

ω-notation (little omega)

By analogy, ω-notation is to Ω -notation as o-notation is to O-notation.We use ω-notation to denote a lower bound that is not asymptotically tight. One way to define it is by

$$f(n) \in \omega\big(g(n)\big) \quad iff \;\; g(n) \in o(f(n)$$

Formally, however, we define $\omega\big(g(n)\big)$ ("little-omega of g of n") as the set

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

Equivalently, the relation $f(n) = \omega\big(g(n)\big)$ implies that

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

if the limit exists.

For example,

1. $\frac{n^2}{2} = \omega(n)$ because $\lim_{n\to\infty} \frac{\frac{n^2}{2}}{n} = \frac{n}{2} = \infty$
2. $\frac{n^2}{2} \neq \omega(n^2)$ because $\lim_{n\to\infty} \frac{\frac{n^2}{2}}{n^2} = \frac{1}{2} \neq \infty$
3. $n^{2.0001} = \omega(n^2)$
4. $n^2 \log n = \omega(n^2)$
5. $n^2 \neq \omega(n^2)$

Comparing functions

Many of the relational properties of real numbers apply to asymptotic comparisons as well. For the following, assume that $f(n)$ and $g(n)$ are asymptotically positive.

Transitivity:

$$\begin{array}{llll}
f(n) = \Theta(g(n)) & \text{and } g(n) = \Theta(h(n)) & \text{imply} & f(n) = \Theta(h(n)) , \\
f(n) = O(g(n)) & \text{and } g(n) = O(h(n)) & \text{imply} & f(n) = O(h(n)) , \\
f(n) = \Omega(g(n)) & \text{and } g(n) = \Omega(h(n)) & \text{imply} & f(n) = \Omega(h(n)) , \\
f(n) = o(g(n)) & \text{and } g(n) = o(h(n)) & \text{imply} & f(n) = o(h(n)) , \\
f(n) = \omega(g(n)) & \text{and } g(n) = \omega(h(n)) & \text{imply} & f(n) = \omega(h(n)) .
\end{array}$$

Reflexivity:

$$\begin{aligned}
f(n) &= \Theta(f(n)) , \\
f(n) &= O(f(n)) , \\
f(n) &= \Omega(f(n)) .
\end{aligned}$$

Symmetry:

$$f(n) = \Theta(g(n)) \text{ if and only if } g(n) = \Theta(f(n)) .$$

Transpose symmetry:

$$\begin{aligned}
f(n) = O(g(n)) &\text{ if and only if } g(n) = \Omega(f(n)) , \\
f(n) = o(g(n)) &\text{ if and only if } g(n) = \omega(f(n)) .
\end{aligned}$$

Because these properties hold for asymptotic notations, we can draw an analogy between the asymptotic comparison of two functions f and g and the comparison of two real numbers a and b:

$$\begin{array}{lll}
f(n) = O(g(n)) & \text{is like} & a \le b , \\
f(n) = \Omega(g(n)) & \text{is like} & a \ge b , \\
f(n) = \Theta(g(n)) & \text{is like} & a = b , \\
f(n) = o(g(n)) & \text{is like} & a < b , \\
f(n) = \omega(g(n)) & \text{is like} & a > b .
\end{array}$$

We say that $f(n)$ is ***asymptotically smaller*** than $g(n)$ if $f(n) = o(g(n))$, and $f(n)$ is ***asymptotically larger*** than $g(n)$ if $f(n) = \omega(g(n))$.

Theorem: $f(x) = O(g(x))$ *if and only if* $g(x) = \Omega(f(x))$.

Proof.

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

$$\text{iff} \quad \exists c > 0, x_0.\ \forall x \geq x_0.\ |f(x)| \leq cg(x)$$

$$\text{iff} \quad \exists c > 0, x_0.\ \forall x \geq x_0.\ g(x) \geq \frac{1}{c}|f(x)|$$

$$\text{iff} \quad \exists c' > 0, x_0.\ \forall x \geq x_0.\ g(x) \geq c'|f(x)| \qquad (\text{set } c' = 1/c)$$

$$\text{iff} \quad g(x) = \Omega(f(x))$$

---

1. Show that $n! = O(n^n)$?

$we\ have, n! = n.(n-1) \ldots..3.2.1$

$or \quad n! \leq n.n.n \ldots..n$

$or \quad n! \leq n^n$

$thus, n! = O(n^n) \quad for\ c = 1\ and\ n \geq 2$

---

2. Show that $6n^3 \neq O(n^2)$

To show that $6n^3 \neq O(n^2)$, we suppose $6n^3 = O(n^2)$. Thus there exist c and k such that

$6n^3 \leq c.n^2 \quad \forall n \geq k.$

$or \quad 6n \leq c \quad (dividing\ by\ n^2)$

$or \quad n \leq \frac{c}{6}$

Since $n \leq \frac{c}{6}$ cannot hold for arbitrarily large n as c is constant.

Thus, $6n^3 \neq O(n^2)$.

---

3. Show that $n! = O(n^n)$?

We have, $n! \leq n^n$.

Taking log on both sides, we get

$\log(n!) \leq \log n^n$

$or\ \log(n!) \leq n \log n$

Thus, $\log(n!) = O(n \log n)$.

---

4. Show that $2^{n+1} = O(2^n)$ *and* $2^{2n} \neq O(2^n)$?

Solution:

$2^{n+1} = O(2^n)$, but $2^{2n} \neq O(2^n)$.

To show that $2^{n+1} = O(2^n)$, we must find constants $c, n_0 > 0$ such that

$0 \le 2^{n+1} \le c \cdot 2^n$ for all $n \ge n_0$.

Since $2^{n+1} = 2 \cdot 2^n$ for all n, we can satisfy the definition with $c = 2$ and $n_0 = 1$.

To show that $2^{2n} \neq O(2^n)$, assume there exist constants $c, n_0 > 0$ such that

$0 \le 2^{2n} \le c \cdot 2^n$ for all $n \ge n_0$.

Then $2^{2n} = 2^n \cdot 2^n \le c \cdot 2^n \Rightarrow 2^n \le c$. But no constant is greater than all 2^n, and so the assumption leads to a contradiction.

Example : Show that $(n+a)^b = \theta(n^b)$, *where a and b are real constants?*

Solution:

To show that $(n+a)^b = \Theta(n^b)$, we want to find constants $c_1, c_2, n_0 > 0$ such that $0 \le c_1 n^b \le (n+a)^b \le c_2 n^b$ for all $n \ge n_0$.

Note that

$$\begin{aligned} n + a &\le n + |a| \\ &\le 2n \qquad \text{when } |a| \le n \text{ ,} \end{aligned}$$

and

$$\begin{aligned} n + a &\ge n - |a| \\ &\ge \frac{1}{2}n \qquad \text{when } |a| \le \tfrac{1}{2}n \text{ .} \end{aligned}$$

Thus, when $n \ge 2|a|$,

$$0 \le \frac{1}{2}n \le n + a \le 2n \text{ .}$$

Since $b > 0$, the inequality still holds when all parts are raised to the power b:

$$0 \le \left(\frac{1}{2}n\right)^b \le (n+a)^b \le (2n)^b \text{ ,}$$

$$0 \le \left(\frac{1}{2}\right)^b n^b \le (n+a)^b \le 2^b n^b \text{ .}$$

Thus, $c_1 = (1/2)^b$, $c_2 = 2^b$, and $n_0 = 2|a|$ satisfy the definition.