St. Pölten University of Applied Sciences

Landau Symbols (Complexity Classes)

Theoretical Computer Science

Dipl.-Ing. Hubert Schölnast, BSc

September 20, 2021

St. Pölten University of Applied Sciences, Campus-Platz 1, 3100 St. Pölten, T: +43 (2742) 313 228, F: +43 (2742) 313 228-339, E: csc@fhstp.ac.at, I: www.fhstp.ac.at

Table of Contents

1 Landau Symbols (Big O notation)

Laundau symbols are used to group functions together and give these groups individual names. Functions that grow at similar rates as their argument gets larger and larger are grouped together.

$O(f(x))$, $\Omega(f(x))$, $\mathcal{O}(f(x))$, $\Theta(f(x))$, $o(f(x))$, $\mathcal{o}(f(x))$ and $\omega(f(x))$ are different sets of functions, which will be explained in detail later. Each of these sets is a container, and inside these containers are functions. For each container there is a function $f(x)$, which defines as a kind of role model which other functions are contained in this set.

$f(x) \in \mathcal{O}(g(x))$ means that the function $f(x)$ is one of the functions in the set $\mathcal{O}(g(x))$ defined by the function $g(x)$.

When using Landau Symbols, you very often will also see this notation:

$f(x) = \mathcal{O}(g(x))$

Technically this is a wrong notation, because it would mean, that the function $f(x)$ is equal to a set of functions, which makes no sense. Nevertheless, it is a commonly used notation that everyone understands as a synonym for $f(x) \in \mathcal{O}(g(x))$.

You say: “f of x is of (the) order g of x”.

Various symbols are used ($O, \Omega, \mathcal{O}, \Theta$, etc.). Some of them are simply different letters that have the same meaning, but some other symbols have different meanings.

2 Big O (“big oh” or “big omicron”)

Equivalent notations (they all mean the same):

$$f(x) \in \mathcal{O}(g(x)) \qquad f(x) \in O(g(x))$$
$$f(x) = \mathcal{O}(g(x)) \qquad f(x) = O(g(x))$$

Formal definition

$$\exists k > 0 \; \exists x_0 > 0 \; \forall x > x_0 \colon \; |f(x)| \leq k \cdot g(x)$$

How to read this definition:

$\exists k > 0$	For at least one (constant) value for k that is greater than 0 and
$\exists x_0 > 0$	for at least one value for x_0 that is greater than 0
$\forall x > x_0$	it is always true for all values of x that are greater than x_0
:	that
$\lvert f(x)\rvert \leq k \cdot g(x)$	the magnitude (the absolute value) of $f(x)$ is less than the product of k and $g(x)$.

Or in other words:

When x is greater than a certain limit (which is x_0), then the quotient $\frac{|f(x)|}{g(x)}$ will always be less than a certain constant value (which is k).

2.1 Example 1:

$f(x) = 5x^2 + 23x + 144$ $\qquad$ $g(x) = x^2$

x	$f(x)$	$g(x)$	$\frac{\lvert f(x)\rvert}{g(x)}$
1	172	1	172.00
2	210	4	52.50
3	258	9	28.67
4	316	16	19.75
5	384	25	15.36
6	462	36	12.83
7	550	49	11.22
8	648	64	10.13
9	756	81	9.33
10	874	100	8.74
11	1002	121	8.28
12	1140	144	7.92
13	1288	169	7.62
14	1446	196	7.38
15	1614	225	7.17
16	1792	256	7.00
17	1980	289	6.85
18	2178	324	6.72
19	2386	361	6.61
20	2604	400	6.51

$f(x) \in \mathcal{O}(g(x))$ is true, because:

	k				x_0	
When you choose	$k = 7$	then	$\lvert f(x)\rvert \le k \cdot g(x)$	for all	$x > 16$	or
when you choose	$k = 200$	then	$\lvert f(x)\rvert \le k \cdot g(x)$	for all	$x > 1$	or
when you choose	$k = 5.1$	then	$\lvert f(x)\rvert \le k \cdot g(x)$	for all	$x > 237$	or …

It doesn't matter which pair of k and x_0 can be used to make this relation become true. If there is at least one such pair, this already is enough.

So: $5x^2 + 23x + 144 \;\in\; \mathcal{O}(x^2)$

2.2 Example 2:

$f(x) = 5x^2 + 23x + 144$ $g(x) = x^3$

Now $g(x)$ grows much faster than in example 1, and therefore it is much easier for the term $k \cdot g(x)$ to be greater than $|f(x)|$

This means, that also this is true: $5x^2 + 23x + 144 \in \mathcal{O}(x^3)$

2.3 Counter example:

$f(x) = x^3 + 5x^2 + 23x + 144$

$g(x) = x^2$

x	$f(x)$	$g(x)$	$\frac{\vert f(x)\vert}{g(x)}$
1	173	1	173.000
2	218	4	54.500
3	285	9	31.667
4	380	16	23.750
5	509	25	20.360
6	678	36	18.833
7	893	49	18.224
7.75	1088	60	18.115
8	1160	64	18.125
9	1485	81	18.333
10	1874	100	18.740
20	10604	400	26.510
50	138794	2500	55.518
100	1052444	10000	105.244
200	8204744	40000	205.119
500	126261644	250000	505.047
1000	1005023144	1000000	1005.023
2000	8020046144	4000000	2005.012
5000	1.25125E+11	25000000	5005.005
10000	1.0005E+12	100000000	10005.002

$\frac{|f(x)|}{g(x)}$ decreases at the beginning, but then reaches a minimum somewhere near 7.75 and then increases and grows forever.

So, no matter how big you choose k, there never will be any x_0 for which it is true, that for every $x > x_0$ the relation $|f(x)| \leq k \cdot g(x)$ will be true.

And therefor:

$x^3 + 5x^2 + 23x + 144 \notin \mathcal{O}(x^2)$

2.4 Conclusion:

$f(x) \in \mathcal{O}(g(x))$ means, that the rate of growth of function $f(x)$ is less or equal than the rate of growth of $g(x)$. Or in other words: $g(x)$ is growing as fast or even faster than $f(x)$.

3 Big Θ ("big theta")

Equivalent notations (they all mean the same):

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

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

Formal definition

$$\exists k_1 > 0 \; \exists k_2 > 0 \; \exists x_0 > 0 \; \forall x > x_0 \colon \; k_1 \cdot g(x) \leq |f(x)| \leq k_2 \cdot g(x)$$

While in big O there was only an upper limit (which here became $k_2 \cdot g(x)$), now, in big theta we also have a lower limit $k_1 \cdot g(x)$, and both limits are constant multiples of the same function $g(x)$.

This means:

$$5x^2 + 23x + 144 \;\in\; \mathcal{O}(x^3)$$

but $\quad 5x^2 + 23x + 144 \;\notin\; \Theta(x^3)$

You will see big O and big theta quite often, because they are the most important symbols, but there are also some others which are less often used:

4 Big Ω ("big omega")

Equivalent notations (they all mean the same):

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

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

Formal definition

$$\exists k > 0 \; \exists x_0 > 0 \; \forall x > x_0 \colon \; |f(x)| \geq k \cdot g(x)$$

So, big omega defines a lower boundary. It is very rarely used in complexity theory.

5 Small symbols

There are also the symbols *small o* and *small* ω ("small omega"). They are defined similar to their big cousins but are more restrict.

So, the arrows in

$$f(x) \in o\big(g(x)\big) \;\Rightarrow\; f(x) \in \mathcal{O}\big(g(x)\big)$$

$$f(x) \in \omega\big(g(x)\big) \;\Rightarrow\; f(x) \in \Omega\big(g(x)\big)$$

only point from left to right, not in the other direction.

The small symbols are not used in complexity theory.

6 Some important defining functions in big O notation

$f(n) \in \mathcal{O}(1)$ "constant"

$f(x)$ is limited to a constant value. No matter how big n grows, $f(n)$ will never become greater than this constant value.
When the time complexity of an algorithm is constant, it means, that the algorithm always terminates within a constant time, no matter how big it's input was.
Example: Test, if a given decimal number of any length is a multiple of 5.

$f(n) \in \mathcal{O}(\log n)$ "logarithmic"

$f(n)$ grows by roughly a constant amount if n will be doubled.
Example: Perform a binary search in a sorted list of n elements.

$f(n) \in \mathcal{O}(\sqrt{n}) = \mathcal{O}\left(n^{\frac{1}{2}}\right)$ "square root"

$f(n)$ doubles if n will be multiplied by 4.
Example: Number of divisions when performing a naïve primality test for the number n.

$f(n) \in \mathcal{O}(n^c),\ 0 < c < 1$ "fractional power"

Generalized version of square root

$f(n) \in \mathcal{O}(n)$ "linear"

$f(n)$ doubles if you double n.
Example: Search an element in an unsorted list of n elements.

$f(n) \in \mathcal{O}(n \log n)$ "superlinear", "loglinear", "n log n"

$f(n)$ grows faster than $\mathcal{O}(n)$, but slower than $\mathcal{O}(n^c)$ for any $c > 1$
Example: Perform a merge sort on a list of n elements.

$f(n) \in \mathcal{O}(n^2)$ "quadratic"

$f(n)$ multiplies by 4 if you double n.
Example: Perform a bubble sort on a list of n elements.

$f(n) \in \mathcal{O}(n^c),\ c > 1$ "polynomial", "algebraic"

Generalized version of quadratic. Note, that c don't have to be an integer. Also $f(n) \in \mathcal{O}(n^{1.0001})$ is polynomial (and slower than $f(n) \in \mathcal{O}(n \log n)$).
A problem that can be solved with an algorithm who's time complexity is $\mathcal{O}(n^c)$ is often called a "simple" problem.

$f(n) \in \mathcal{O}(2^n) = \mathcal{O}(c^n)$ "exponential"

$f(n)$ doubles (multiplies by a constant factor) if you increase n by 1.
A problem whose fastest solving algorithm has a time complexity of $\Omega(e^n)$ is often called a "hard" problem.