

Asymptotic Notation:

1. O-notation (upper bounds)

Define: $f(n) = O(g(n))$ (Read as “f of n equals big Oh of g of n”)

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

- $O(g(n))$ is a **set** of functions.

Abuse of equality:

We write $f(n) = O(g(n))$ as $f(n) \in O(g(n))$

- O-notation is an **upper-bound** notation.

Example: $2n^2 = O(n^3)$

There exist $c = 1$ and $n_0 = 2$ such that $0 \leq 2n^2 \leq cn^3$, for all $n \geq n_0$.

Check:

$2n^2$	cn^3
$2 * 2^2$	$1 * 2^3$
$2 * 3^2$	$1 * 3^3$
$2 * 4^2$	$1 * 4^3$

When determining the order of magnitude of $f(n)$, we shall always try to obtain the **smallest $g(n)$** such that $f(n) = O(g(n))$

So, the above example could be changed to

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

There exist $c = 2$ and $n_0 = 1$ such that $0 \leq 2n^2 \leq cn^2$, for all $n \geq n_0$.

Thm:

If $f(n) = a_m n^m + \dots + a_1 n + a_0$ is a polynomial of degree m then $f(n) = O(n^m)$.

That is, **you can ignore the leading term's constant coefficient and lower-order terms.**

2. Ω -notation (lower bounds)

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

Example: $\sqrt{n} = \Omega(\log_2 n)$

There exist $c = 1$ and we want to find n_0 such that $\log_2 n \leq \sqrt{n}$, for all $n \geq n_0$

Check: from below table, we get $n_0 = 16$

n	$\log_2 n$	\sqrt{n}
2	1	1.414
4	2	2.000
8	3	2.828
16	4	4.000
32	5	5.657
64	6	8.000
128	7	11.314

3. Θ - notation (tight bounds)

$\Theta(g(n)) = \{f(n) : \text{there exist positive constants } c_1, c_2 \text{ and } n_0 \text{ such that } 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n), \text{ for all } n \geq n_0\}$

Example: $\frac{1}{2}n^2 - 2n = \Theta(n^2)$

There exist $c_1 = \frac{1}{4}$, $c_2 = \frac{1}{2}$, and $n_0 = 8$ such that $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$, for all $n \geq n_0$.

$0 \leq c_1 n^2 \leq \frac{1}{2}n^2 - 2n$ <ul style="list-style-type: none"> • $c_1 > 0$ • $c_1 < \frac{1}{2}$ • pick up $c_1 = \frac{1}{4}$ $\frac{1}{4}n^2 \leq \frac{1}{2}n^2 - 2n$ $\frac{1}{4}n \leq \frac{1}{2}n - 2$ $2 \leq \frac{1}{4}n$ $8 \leq n$ <p style="text-align: right;">So, $n_0 = 8$</p>	$\frac{1}{2}n^2 - 2n \leq c_2 n$ <ul style="list-style-type: none"> • pick up $c_2 = \frac{1}{2}$ <p style="text-align: center;">since $\frac{1}{2}n^2 - 2n \leq \frac{1}{2}n^2$,</p> <p style="text-align: center;">for all $n > 0$</p>
---	---

4. Theorem 3.1 (page 46)

For any two functions $f(n)$ and $g(n)$, we have $f(n) = \Theta(g(n))$ if and only if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.

5. Fact: $O(1) < O(\lg n) < O(n) < O(n \lg n) < O(n^2) < O(n^3)$ and $O(2^n)$

$O(1)$: the number of execution of basic operations is fixed and the total time is bounded by a constant.

$\log_2 n$	n	$n \log_2 n$	n^2	n^3	2^n
0	1	0	1	1	2
1	2	2	4	8	4
2	4	8	16	64	16
3	8	24	64	512	256
4	16	64	256	4096	65536
5	32	160	1024	32768	4294967296

6. o-notation (little-oh)

Define: $f(n) = o(g(n))$ (Read as “f of n equals **little oh** of g of n”)

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

That is, $f(n) = o(g(n))$ if $f(n) = O(g(n))$ and $f(n) \neq \Theta(g(n))$