Analysis and Design of Algorithms

Asymptotic Notations- Rate of Growth

Contents

- Ω-Notation (Big-Omega Notation)
- Θ-Notation (Theta Notation)
- Asymptotic Order of Growth
- Bounding Function
- Properties of Asymptotic Notation

2. Ω (Big- Omega) – Notation

2. Ω(Big-Omega) – Notation

Big Omega Notation:

Big Omega notation is used to define the **lower bound** of any algorithm or we can say **the best case** of any algorithm.

This always indicates the minimum time required for any algorithm for all input values, therefore the best case of any algorithm.

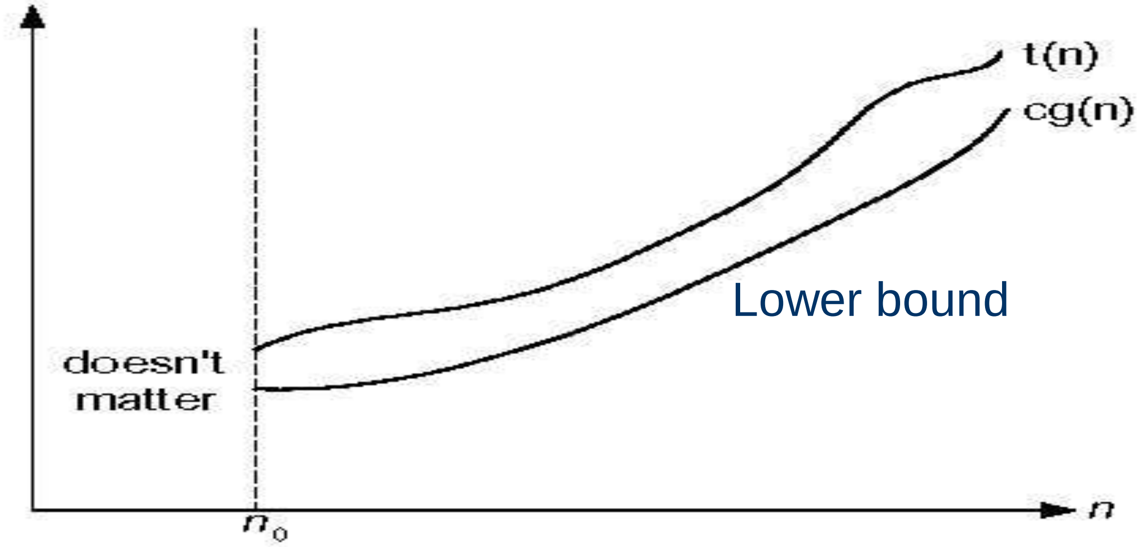
In simple words, when we represent a time complexity for any algorithm in the form of big-Ω, we mean that the algorithm will take at least this much time to complete its execution. It can definitely take more time than this too.

Ω(Big-Omega) – Notation

- f(n) = $\Omega(g(n))$ (read as “f of n is omega of g of n”)

f(n) ≥ c * g(n)

- iff there exist positive constants c and
- n_0 such that for all n, **$n \geq n_0$**



Example-1 of Big-Omega

Ex: $f(n) = 3n+2$.

Sol: $f(n) \geq c * g(n)$

$3n+2 \geq 2 * g(n)$ $\qquad c = 3-1 = 2$

for $g(n) = 1$, i.e

$3n+2 \geq 2*1$

for $n=1$: $3(1)+2 \geq 2*1$

$5 \geq 2$

True

$n=2$: $3(2)+2 \geq 2*1$

$8 \geq 2$ True

$n=3$: $3(3)+2 \geq 2*1$

$11 \geq 2$ True

$\Omega(1)$ is possible, because whatever values we take for n, it satisfies the condition.

Now, for $g(n)=n$, i.e

$3n+2 \geq 2*n$

$n=1$: $3(1)+2 \geq 2*1$

$5 \geq 2$ True

$n = 2: \quad 3(2) + 2 \geq 2 * 2$

$8 \geq 4$ True

$n = 3: \quad 3(3) + 2 \geq 2 * 3$

$11 \geq 6$ True

when we keep on substituting 'n' values, we will get true, i.e condition is satisfied

$\therefore \Omega(n)$ is possible.

Now when $g(n) = n^2$, i.e

$3n + 2 \geq 2 * n^2$

$n = 1: \quad 3(1) + 2 \geq 2 * (1)^2$

$5 \geq 2$ True

$n = 2: \quad 3(2) + 2 \geq 2 * (2)^2$

$8 \geq 8$ True

$n = 3: \quad 3(3) + 2 \geq 2 * (3)^2$

$11 \geq 18$ False

$n = 4: \quad 3(4) + 2 \geq 2 * (4)^2$

Here

$14 \geq 32$ False

As the conditions are not satisfied, $\Omega(n^2)$ is not possible.

$f(n) = 3n + 2$

$\Omega(1)$, $\Omega(n)$ are the possible omega notations.

Example-2

Ex2: $f(n) = 10n^2 + 3n + 3$

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

$f(n) \geq c * g(n)$

$10n^2 + 3n + 3 \geq 9 * g(n)$ $\qquad c = 10 - 1 = 9$

For $g(n) = 1$

$10n^2 + 3n + 3 \geq 9 * 1$

$n = 1$: $10(1)^2 + 3(1) + 3 \geq 9 * 1$

$16 \geq 9$ True

$n = 2$: $10(2^2) + 3(2) + 3 \geq 9 * 1$

$49 \geq 9$ True

$\Omega(1)$ is possible

For $g(n) = n$,

$10n^2 + 3n + 3 \geq 9 * n$

$n = 1$: $10(1)^2 + 3(1) + 3 \geq 9 * 1$

$16 \geq 9$ True

$n = 2$: $10(2^2) + 3(2) + 3 \geq 9 * 2$

$49 \geq 18$ True

$n = 3$: $10(3^2) + 3(3) + 3 \geq 9 * 3$

$102 \geq 27$ True

For any value of n, the condition gets satisfied

$\therefore \Omega(n)$ is possible

For $g(n) = n^2$, i.e

$$10n^2 + 3n + 3 \geq 9 * n^2$$

$n = 1$: $10(1)^2 + 3(1) + 3 \geq 9 * 1^2$

$16 \geq 9$ True

$n = 2$: $10(2^2) + 3(2) + 3 \geq 9 * 2^2$

$49 \geq 36$ True

$n = 3$: $10(3^2) + 3(3) + 3 \geq 9 * 3^2$

$102 \geq 81$ True

Here also, for any value of n, the condition is satisfied.

$\therefore \Omega(n^2)$ is possible

For $f(n) = 10n^2 + 3n + 3$

$\Omega(1)$, $\Omega(n)$ and $\Omega(n^2)$ are the possible omega notations.

3. Θ(Theta) – Notation

Θ(Theta) – Notation

Theta Notation:

When we say tight bounds, we mean that the time complexity represented by the Big-Θ notation is like the average value or range within which the actual time of execution of the algorithm will be.

For example, if for some algorithm the time complexity is represented by the expression $3n^2 + 5n$, and we use the Big-Θ notation to represent this, then the time complexity would be $\Theta(n^2)$, ignoring the constant coefficient and removing the insignificant part, which is $5n$.

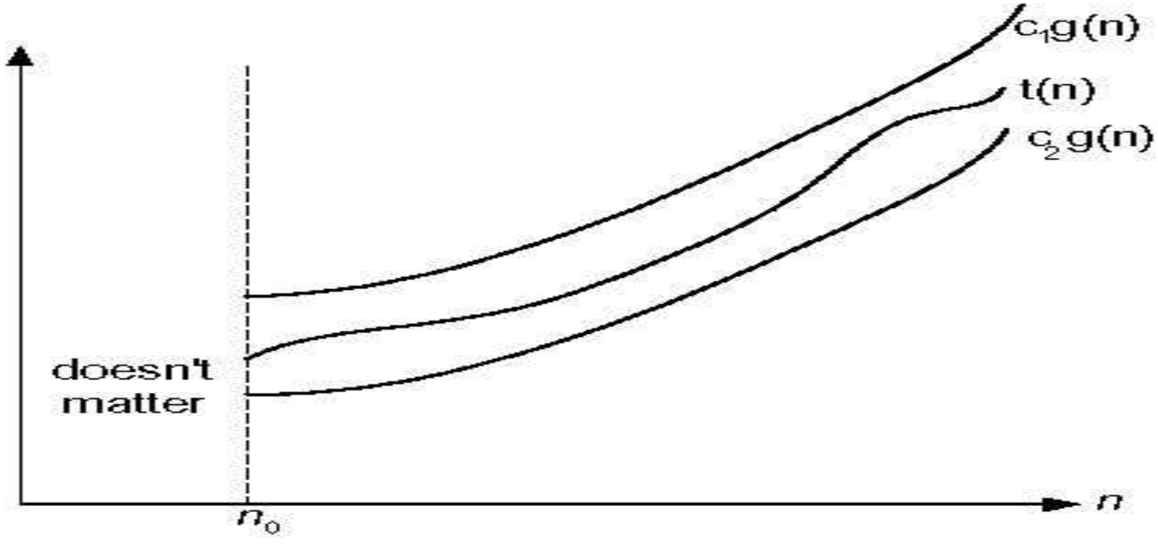
$f(n) = \Theta(g(n))$, if there exists a positive integer n_0 and a positive constants c1, c2, such that $c1 * g(n) \leq f(n) \leq c2 * g(n) \ \forall\ n \geq n_0$

Here, in the example above, complexity of $\Theta(n^2)$ means, that the average time for any input n will remain in between, $k1 * n^2$ and $k2 * n^2$, where k1, k2 are two constants, thereby tightly binding the expression representing

Θ(Theta) – Notation

- f(n) = Θ (g(*n*)) (read as “f of n is theta of g of n”)
- iff there exist positive constants c_1 , c_2 and n_0 such that $c_1 * g(n) \le f(n) \le c_2 * g(n)$

  for all n, $n \ge n_0$



Θ(Theta) – Notation Examples

Let us take an example 1 :

$f(n) = 4n + 3$

here $c_1 = 3$

$c_2 = 5$

$\Rightarrow 3 * g(n) \leq 4n + 3 \leq 5 * g(n)$

let us take

$g(n) = 1$

$3 \leq 4n + 3 \leq 5$

if $n = 1$ then

$3 \leq 7 \leq 5 \longrightarrow$ false

if $n = 2$ then

$3 \leq 11 \leq 5 \longrightarrow$ false

⋮

$\therefore \Theta(1)$ is not possible

let us consider

$g(n) = n$

$3*n \leq 4n+3 \leq 5*n$

if $n=1$ then

$3 \leq 7 \leq 5 \longrightarrow$ false

if $n=2$ then

$6 \leq 11 \leq 10 \longrightarrow$ false

if $n=3$ then

$9 \leq 15 \leq 15 \longrightarrow$ True.

$\therefore$ $\Theta(n)$ is possible

let us consider

$g(n) = n^2$

$3*n^2 \leq 4n+3 \leq 5n^2$

if $n=1$ then

$3 \leq 7 \leq 5 \longrightarrow$ false

if $n=2$ then

$12 \leq 11 \leq 20 \longrightarrow$ false

if $n=3$ then

$27 \leq 15 \leq 45 \longrightarrow$ false.

$\vdots$

$\therefore$ $\Theta(n^2)$ is not possible.

In theta notation only $\Theta(n)$ is possible in this function.

$\therefore$ $\Theta(n)$ is possible notation.

Let us take another example 2:

$$f(n) = 10n^2 + 3n + 3$$

$$c_1 = 9$$

$$c_2 = 11$$

$$c_1 * g(n) \leq f(n) \leq c_2 * g(n)$$

$$9 * g(n) \leq 10n^2 + 3n + 3 \leq 11 * g(n)$$

$g(n) = 1$

$$9 \leq 10n^2 + 3n + 3 \leq 11$$

if $n = 1$ then

$9 \leq 16 < 11 \rightarrow$ false

if $n = 2$ then

$9 \leq 49 < 11 \rightarrow$ false

$\vdots$

$\therefore$ $\Theta(1)$ is not possible

for $g(n) = n$.

$$9 * n \leq 10n^2 + 3n + 3 \leq 11 * n$$

if $n = 1$ then

$9 \leq 16 \leq 11 \rightarrow$ false

if $n = 2$ then $18 \leq 49 \leq 22 \rightarrow$ false

⋮

$\therefore \Theta(n)$ is not possible

for $g(n) = n^2$

$$9 * n^2 \le 10n^2 + 3n + 3 \le 11 * n^2$$

$n = 1 \quad 9 \le 16 \le 11 \rightarrow$ false

$n = 2 \quad 36 \le 49 \le 44 \rightarrow$ false

$n = 3 \quad 81 \le 102 \le 99 \rightarrow$ false

$n = 4 \quad 144 \le 175 \le 176 \rightarrow$ True

⋮

$\Theta(n^2)$ is possible

for $g(n) = n^3$

$$9 * n^3 \le 10n^2 + 3n + 3 \le 11 * n^3$$

if $n = 2$ then

$72 \le 49 \le 88 \rightarrow$ false

⋮

$\Theta(n^3)$ is not possible

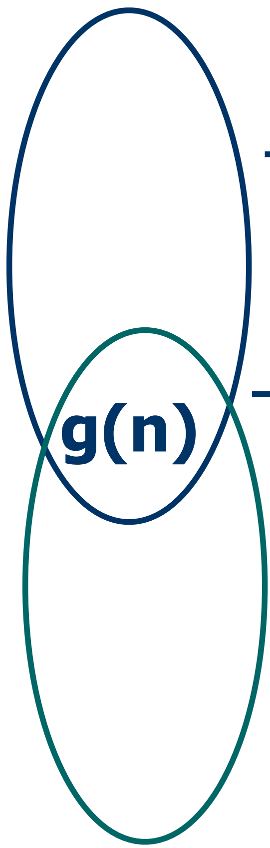
In this function, $\Theta(n^2)$ is the possible notation.

Asymptotic Order of Growth

A way of comparing functions that ignores constant factors and small input sizes :

- $O(g(n))$: class of functions $f(n)$ that grow no faster than $g(n)$
- $\Theta(g(n))$: class of functions $f(n)$ that grow at same rate as $g(n)$
- $\Omega(g(n))$: class of functions $f(n)$ that grow at least as fast as $g(n)$

Asymptotic Order of Growth



$\Omega(g(n))$, functions that grow at least as fast as g(n)

$\Theta(g(n))$, functions that grow at the same rate as g(n)

$O(g(n))$, functions that grow no faster than g(n)

Bounding Function

- $f(n) = O(g(n))$ means c X $g(n)$ is upper bound on $g(n)$.
- $f(n) = \Omega(g(n))$ means c X $g(n)$ is a lower bound on $g(n)$.
- $f(n) = \Theta(g(n))$ means c_1 X $g(n)$ is upper bound on $f(n)$ and c_2 X $g(n)$ is a lower bound on $f(n)$

Where, c, c_1 and c_2 are all constant independent of *n.*

All of these definitions imply a constants, n_0 beyond which they are satisfied.

Comparison between O, Ω, θ →

What is the practical significance of all these notations –

O → worst case time or upper bound (Time will not exceed)

Ω → Best case or lower bound.

θ → Average case

– We generally search for worst case in which an Algorithm take more time for any input. So generally we not interested in Best case.

worst	Best
✓	✗

– Average case → when any Algorithm is same perform on Best & worst case then we have to move for Average case.

Example →

Assume that we have an array of numbers →

2	4	6	8	7	5	4	5	13

size of Array = n.

→ Search for an element say x. in Linear search or sequential search

→ ① Best case ⇒ $x = 2$ (It is found in 1 comparison)

↓ $\Omega(1)$

② worst case → $O(n)$ (It is found in n comparison) say $x = 5$

③ Average case → $\theta(n/2)$ → $\theta(n)$ (It is found in n/2 comparison) say $x = 7$

Properties of Asymptotic Notations

Properties of Asymptotic notations:

1. Transitive

- If $f(n) = \Theta(g(n))$ and $g(n) = \Theta(h(n))$, then $f(n) = \Theta(h(n))$
- If $f(n) = O(g(n))$ and $g(n) = O(h(n))$, then $f(n) = O(h(n))$
- If $f(n) = o(g(n))$ and $g(n) = o(h(n))$, then $f(n) = o(h(n))$
- If $f(n) = \Omega(g(n))$ and $g(n) = \Omega(h(n))$, then $f(n) = \Omega(h(n))$
- If $f(n) = \omega(g(n))$ and $g(n) = \omega(h(n))$, then $f(n) = \omega(h(n))$

2. Reflexivity

- $f(n) = \Theta(f(n))$
- $f(n) = O(f(n))$
- $f(n) = \Omega(f(n))$

Properties of Asymptotic Notations

3. Symmetry

- $f(n) = \Theta(g(n))$ if and only if $g(n) = \Theta(f(n))$

4. Transpose Symmetry

- $f(n) = O(g(n))$ if and only if $g(n) = \Omega(f(n))$
- $f(n) = o(g(n))$ if and only if $g(n) = \omega(f(n))$

5. Some other properties of asymptotic notations are as follows:

- If $f(n)$ is $O(h(n))$ and $g(n)$ is $O(h(n))$, then $f(n) + g(n)$ is $O(h(n))$.
- The function $\log_a n$ is $O(\log_b n)$ for any positive numbers a and $b \neq 1$.
- $\log_a n$ is $O(\log n)$ for any positive $a \neq 1$, where $\log n = \log_2 n$.