Data types:-

int, floating point, char, (1B) string,
(small int, long int)
boolean (1 bit).

Asymptotic Notations:-

To choose the best algorithm, we need to check efficiency of each algorithm.

The efficiency can be measured by computing time complexity of each algorithm.

Asymptotic notation is a shorthand way to represent the time complexity.

Using asymptotic notations we can give time complexity as "fatest possible", slowest possible" or "average time".

Various notations such as Ω, θ and O are used & called as asymptotic notions.

large values of n ↑

1. Big-Oh Notation : O - upper bound
2. Big-Omega notation : Ω - Lower bound
3. Theta Notatation : Θ - Avg./ Tight bound
4. Little-Oh notation : o
5. Little-Omega notation (ω)

↓ small values of n

$1 < \log n < \sqrt{n} < n < n\log n < n^2 < n^3 < \ldots 2^n < 3^n \ldots$

Big-Oh Notation (O) → worst case running time of algo.

A function f(n) is said to be in O(g(n)), denoted as $t(n) \in O(g(n))$, if t(n) is bounded above by some constant multiple of g(n) for all large n.

i.e. if there exist some positive constant c and some non-negative integer n_0 such that

$$\boxed{t(n) \le c(g(n)) \text{ for all } n \ge n_0.}$$

* O(g(n)): class of ftns/. t(n) that grow no faster than g(n)
* O → puts asymptotic UB on a ftn.

67.

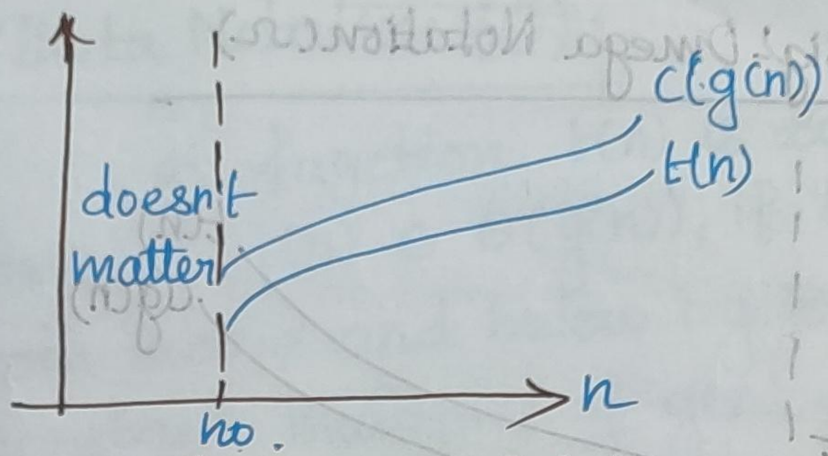


Fig:- Big-oh(O)

Let $t(n) = 2n+3$

$2n+3 \leq \_ ?$

$2n+3 \leq 5n \quad n \geq 1$

here $c = 5$

& $g(n) = n$.

$\boxed{\therefore t(n) = O(n)}$

$\leq n^2$

Also: $2n+3 \leq 5n^2 \quad n \geq 1$

Where $c = 5$ & $g(n) = n^2$

$\boxed{\therefore t(n) = O(n^2)}$ //.

Big-Omega Notation(Ω) → Best case running time of algo.

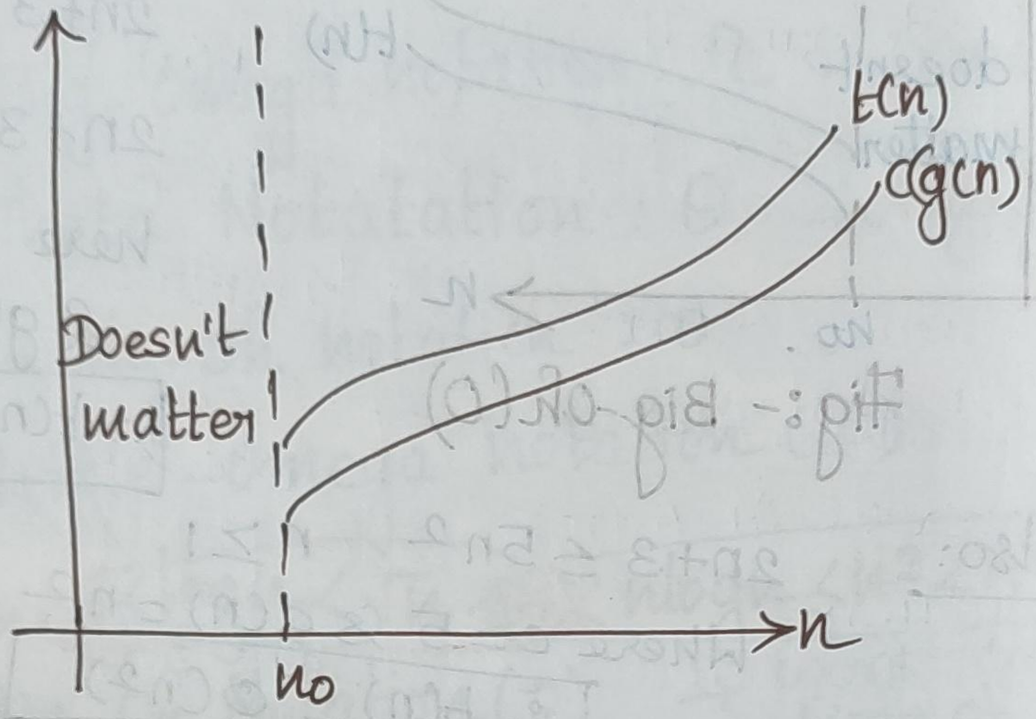
A function $t(n)$ is said to be in $\Omega(g(n))$, denoted as $t(n) \in \Omega(g(n))$, if $t(n)$ is bounded below by some positive constant multiple of $g(n)$ for all large n.

i.e., there exist some positive constant c & some non-negative integer no.

Such that $\boxed{t(n) \geq c(g(n)) \text{ for all } n \geq n_0.}$

★ It represents the lower bound of the resources required to solve a problem.

(68.) Fig: Big Omega Notation (Ω)



Let $t(n) = 2n+3$

$2n+3 \geq$ —— ??

$2n+3 \geq 1n$

here $c = 1$ and $g(n) = n$.

$\boxed{t(n) = \Omega(n)}$ //.

$2n+3 \geq 1 \log n \quad n \geq 1$

here $c=1$ and $g(n) = \log n$

$\boxed{t(n) = \Omega(\log n)}$ //.

69. # Theta Notation (Θ):-

A function $t(n)$ is said to be in $\Theta(g(n))$, denoted $t(n) \in \Theta(g(n))$, if $t(n)$ is bounded both above and below by some positive constant multiples of $g(n)$ for all large n. i.e., if there exist some positive constant $c1$ and $c2$ and some non-negative integer n_0 such that

$$c2\,g(n) \le t(n) \le c1\,g(n) \quad \forall\ n > n_0.$$

$$\boxed{\Theta(g(n)) = O(g(n)) \cap \Omega(g(n))}$$

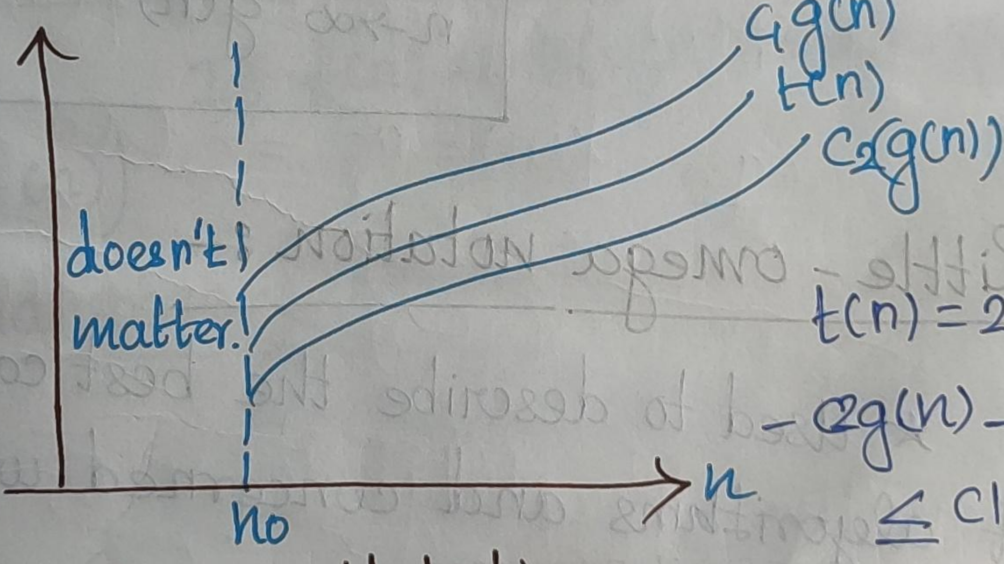


Fig: Θ- Notation.

$t(n) = 2n+3$

$c2\,g(n) \le 2n+3 \le c1 \cdot g(n)$

$1n \le 2n+3 \le 5n$

$n > 1$.

here $c1 = 5$ & $c2 = 1$

& $g(n) = n$.

$\therefore \boxed{t(n) = \Theta(n)}$

70.

Little-oh notation:- (o)

* Used to describe the worst case analysis of algorithms and concerned with small values of n.

A ftn. f(n) is said to be in o(g(n)), denoted t(n) ∈ o(g(n)), if there exist some positive constant c and some non-negative integer such that

$$\boxed{\begin{gathered} t(n) \leq c.g(n) \\ \lim_{n \to \infty} \frac{t(n)}{g(n)} = 0. \end{gathered}}$$

Little-omega notation:-

* Used to describe the best case analysis of algorithms and concerned with small values of n

The function t(n) = ω(g(n)) iff

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = \infty \text{ or } \lim_{n \to \infty} \frac{g(n)}{f(n)} = 0$$

(7)

Properties of O, Ω and θ

(1) General property:— (Ω & θ)

If $f(n)$ is $O(g(n))$ and then $a*f(n)$ is $O(g(n))$. Similar for Ω and θ.

(2) Transitive property:— ($\forall$ Ω, θ, o & ω)

If $f(n) \in O(g(n))$ and $g(n) \in O(h(n))$, then $f(n) \in O(h(n))$; that is O is transitive. Also Ω, θ, o and ω are transitive.

(3) Reflexive property:—

If $f(n)$ is $\theta(g(n))$ then $g(n)$ is $\theta(f(n))$

(4) Transpose property:—

If $f(n) = O(g(n))$ then $g(n)$ is Ω $f(n)$.

(5) Symmetric property:—

If $f(n)$ is $\theta(g(n))$ then $g(n)$ is $\theta(f(n))$

72

(1) General properties :- (True ∀ Ω & Θ)

If $f(n)$ is $O\,g(n)$ then $a * f(n)$ is $O\,g(n)$

eg:- $f(n) = 2n^2+5$ is $O(n^2)$

then $\Rightarrow 7 \cdot f(n) = 7(2n^2+5)$

$= 14n^2+35$ is

$\Rightarrow O(n^2)$

$\therefore f(n) = \Omega g(n) \rightarrow a * f(n) \rightarrow \Omega g(n)$

(2) Reflexive property :-

If $f(n)$ is given then $f(n)$ is $O f(n)$

eg:- $f(n) = n^2 \Rightarrow O(n^2)$ //.

$\uparrow$

$f(n)$

(3) Transitive property :- [∀ O, Ω & Θ]

If $f(n)$ is $O\,g(n)$ and $g(n)$ is $O(h(n))$

then $f(n) = O(h(n))$

(13.) eg. $f(n) = n \xrightarrow{UB} g(n) = n^2 \xrightarrow{UB} h(n) = n^3$

n is $O(n^2)$ & n^2 is $O(n^3)$

then n is $O(n^3)$

(4) Symmetric property :-

If $f(n)$ is $\theta(g(n))$ then $g(n)$ is $\theta(f(n))$

eg:- $f(n) = n^2 \rightarrow g(n) = n^2$) *

$f(n) = \theta(n^2)$

$g(n) = \theta(n^2)$

When both the functions are same, then they are symmetric.

(5.) Transpose symmetric :- $[O \,\&\, \Omega]^{ly.}$

If $f(n) = \underset{UB}{O(g(n))}$ then $g(n)$ is $\underset{LB.}{\Omega(f(n))}$

eg:- $f(n) = n$ $\quad g(n) = n^2$

then n is $O(n^2)$ and n^2 is $\Omega(n)$.

If one function forms an upper bound for other ftn. then the other ftn. will form a lower bound for the other function.

* If $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$

When same ftn. is acting both as UB & LB $\Big\}$ $g(n) \leq f(n) \leq g(n)$

$$\boxed{f(n) = \Theta(g(n))}$$ //.

* If $f(n) = O(g(n))$
and $d(n) = O(e(n))$
then $f(n) + d(n) = O(\max(g(n), e(n)))$

eg:- $f(n) = n \Rightarrow O(n)$
$d(n) = n^2 \Rightarrow O(n^2)$
$f(n) + d(n) = n + n^2 = O(n^2)$ //

* If $f(n) = O(g(n))$

$d(n) = O(e(n))$

then $f(n) * d(n) = O(g(n) * e(n))$

$n \quad n^2 = \quad n^3$ //.

— x —