Asymptotic Notation

1. Big-oh-Notation. (O) (Worst case)
2. Omega Notation. (Ω) (Best case)
3. Theta Notation. (θ) (Average case).

⇒ These notation are applicable for +ve functions.

$f(n) = n-10$; +ve

$= n-1000000$; +ve

⇒ Algo. is valid only for +ve functions.

⇒ -ve function :-

$f(n) = 10-n = -ve$ fun.

$= 10000-n = -ve$ fun.

Algorithms are not for -ve functions.

Asymptotic Notation :- To choose best algo we need to check efficiency of each algorithm. The efficiency can be measured by computing time complexity of algorithm. Asymptotic Notation is a shorthand way to represent time complexity.

⇒ Let $f(n)$ & $g(n)$ be 2 +ve functions and n_0 & c are two constant such that $c > 0$ & $n_0 \geq 1$ Then.

1. Big - Oh Notation :-

Big-oh notation is denoted by 'O' it is a method of representing the upper bound of algorithm's Running time.

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

or

$f(n)$ is order of $g(n)$

iff

$$\boxed{f(n) \leq c \cdot g(n)}, \forall n, n \geq n_0$$

Where c & n_0 are 2 +ve constant.
$c > 0$ & $n \geq 1$.

eg 1 :-

$$f(n) = n + 10$$
$$g(n) = n$$

$$n + 10 \leq c \cdot n \quad . \forall n \quad . n \geq n_0$$

$c := 2$ $\qquad$ $n_0 := 10$

for which point the condition true that value is consider as n_0.

$10 + 10 \leq 2 \cdot 10$

$20 \leq 20$

$$\boxed{n+10 = O(n)}$$

$n_0 = 10$, $c = 2$ Ans

eg - 2)

$f(n) = n$

$g(n) = n+10$

$\Downarrow$

$n \leq c \cdot (n+10)$, $\forall n$, $n \geq n_0$

$\Downarrow$ 1 $\Downarrow$ 1

For $n_0 = 1$ & $c = 1$ the condition is working.

$1 \leq 1 \cdot (11)$

For $n_0 = 1$ the condition is true.

eg - 3)

$f(n) = n^2$

$g(n) = n$

$\Downarrow$

$n^2 \leq c \cdot n$ $\forall n$, $n \geq n_0$

$\downarrow$

$c \neq n$

⇒ we not give c value as n because c is a constant. and if we make $c = n$ then c is not act as a constant.

$f(n) \leq c.\ g(n)$ (Not True)

⇒ With the help of constant (c) if we make the RHS greater than or equal to L.H.S. then we say that

$$\boxed{f(n) = O(g(n))}$$

2. Omega Notation (Ω) :-

Omega notation is denoted by Ω. This notation is used to represent lower bound of algo running time.

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

or

$f(n)$ is omega of $g(n)$.

iff

$$\boxed{f(n) \geq c.\ g(n)} \quad \forall n,\ n \geq n_0.$$

Such that, there exist 2 +ve constant c & n_0 where $c > 0$ & $n_0 \geq 1$.

eg 1)

$f(n) = n$

$g(n) = n+10$

$\Downarrow$

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

$\Downarrow$

$n \leq c_0 (n+10)$, $\forall n$, $n \geq n_0$

$\Downarrow$ $\Downarrow$

$c = 1/2$ 10

$\Rightarrow$ From 10 onwards LHS $\geq$ RHS.

so,

$f(n) = \Omega(g(n))$ { $c = 1/2$ always }.

eg 2)

$f(n) = n$

$g(n) = n$

$\Downarrow$

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

$\Downarrow$

$f(n) \geq c_0 g(n)$, $\forall n$, $n \geq n_0$

$\Downarrow$

$n \geq c_0 n$, $\forall n$, $n \geq n_0$

$\Downarrow$

1

It is already satisfied. No need to do anything.

so,

$\boxed{n_0 = 1}$

$f(n) = \Omega(g(n))$. ✓

eg 3)

$f(n) = n$

$g(n) = n^2$

$\Downarrow$

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

$\Downarrow$

$n \geq c_0 \cdot n^2$, $\forall n$, $n \geq n_0$

$\Downarrow$

c value not possible.

$\Rightarrow$ with the help of processor speed we not

3. Theta Notation (θ) :-

Theta notation is denoted by θ. By this method running time is bitween upper bound & lower bound.

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

or

$f(n)$ is theta of $g(n)$.

if

(i) $\boxed{f(n) \leq c_1 \cdot g(n)}$, $\forall n$, $n \geq n_0$

&

(ii) $\boxed{f(n) \geq c_2 \cdot g(n)}$, $\forall n$, $n \geq n_0$.

Both should be satisfied.

eg 1)

$$f(n) = n$$
$$g(n) = n+10$$

$$\Downarrow$$

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

$$\Downarrow$$

1. $f(n) \leq c_1 \cdot g(n)$ $\forall n$, $n \geq n_0$

$$\Downarrow$$

$n \leq c_1 \cdot (n+10)$ $\forall n$, $n \geq n_0$

$$\Downarrow$$

1

$$\Downarrow$$

1

2. $f(n) \geq c_2 \cdot g(n)$, $\forall n \quad n \geq n_0$.

$\Downarrow$

$n \geq c_2(n+10)$, $\forall n \quad n \geq n_0$.

$c_2 \Downarrow 1/2$; $n_0 \Downarrow 10$

common $n_0 = 10$. ↓ working for 10 onwards. ✓

eg 2)

$f(n) = n$

$g(n) = n$

$\Downarrow$

$f(n) = \Theta(g(n))$.

$\Downarrow$

1. $n \geq G.n$, $\forall n, n \geq n_0$.

$G \Downarrow 1$; $n_0 \Downarrow 1$

2. $n \leq c_2.n$, $\forall n$, $n \geq n_0$.

$c_2 \Downarrow 1$; $n_0 \Downarrow 1$

Common no. $= 1$. ✓

$\Rightarrow c_1$ & c_2 may be equal, Need not be equal.

eg 3)

$f(n) = n$

$g(n) = n^2$

$\Downarrow$

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

$\Downarrow$

1. $n \leq c_1 . n^2 , \forall n , n \geq n_0$

$\Downarrow$ 1

$\Downarrow$ 1

2. $n \geq c_2 . n^2 , \forall n , n \geq n_0$

$\Downarrow$

$1/n$ (Not valid)

$$\boxed{f(n) \neq \Theta(g(n))}$$

$n \neq \Theta(n^2)$

① Big-oh only valid.

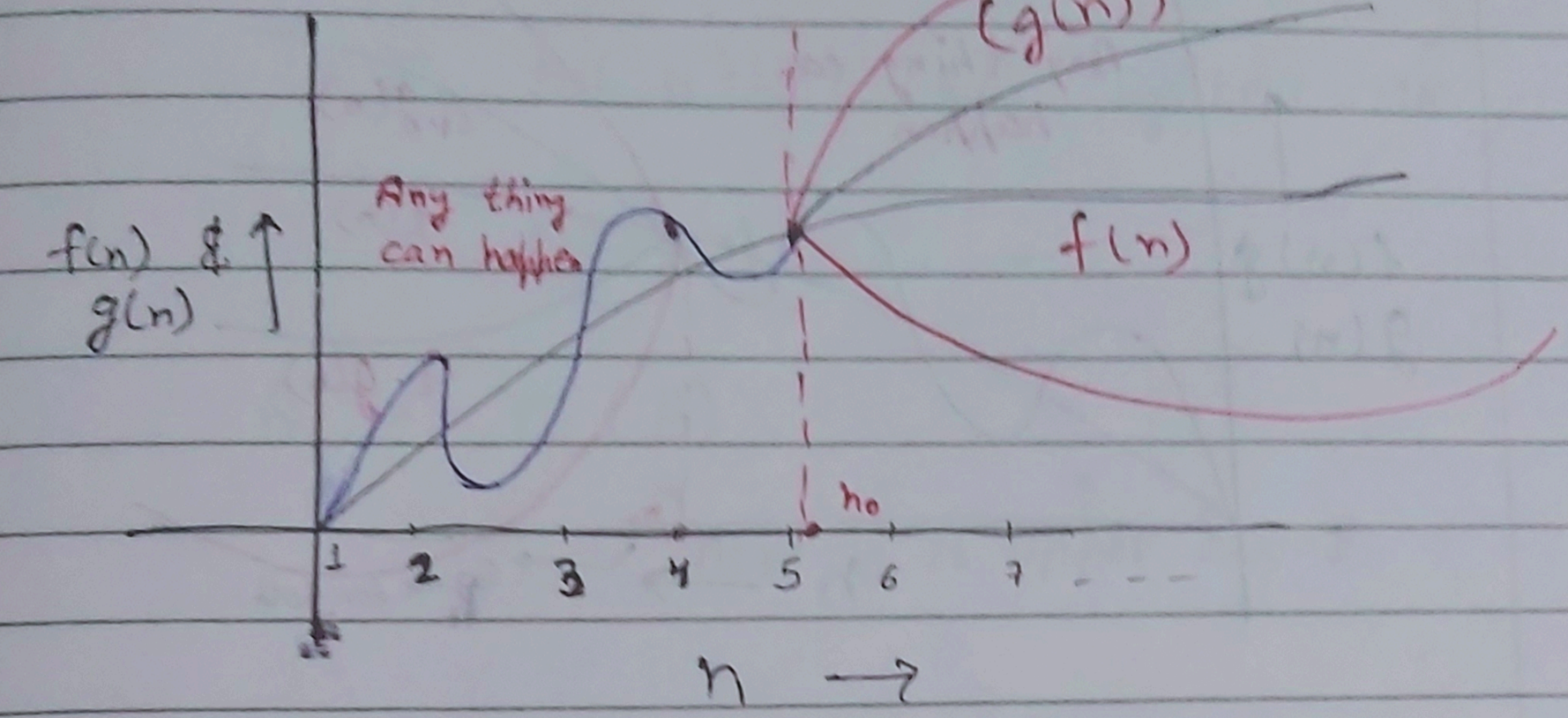
② omega is not valid.

⇒ If the function are differ by constant then theta (Θ) notation is valid with the help of constant (c).

⇒ If the function's are differ by function then it is not satisfied.

1. Big - oh Notation → To calculate upper Bound.

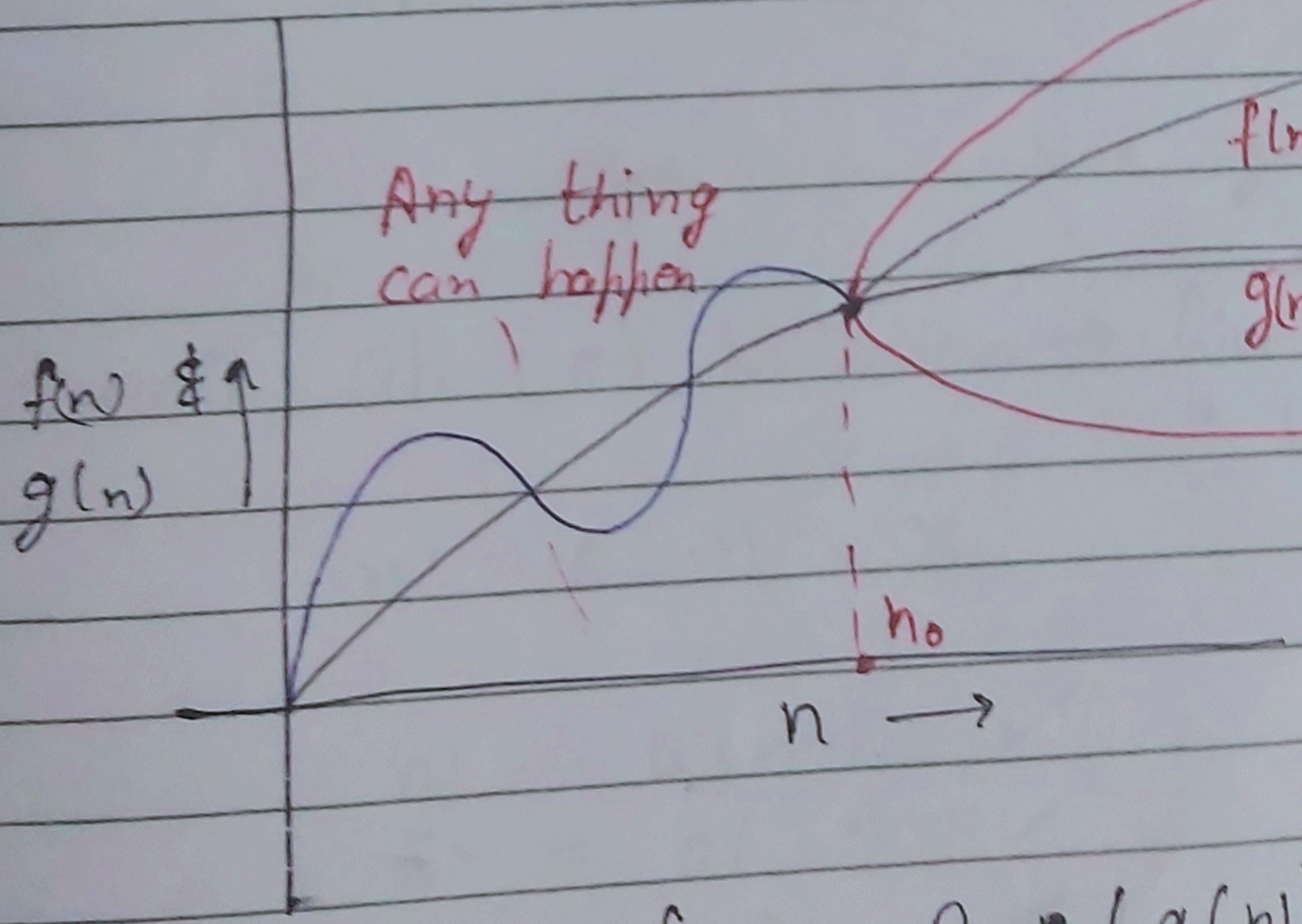
$n_0 \leq n \leq \infty$



$f(n) = O(g(n)) \Rightarrow f(n) \leq C \cdot g(n) \quad \forall n, n \geq n_0$

2. Omega (Ω) Notation → To calculate lower Bound.

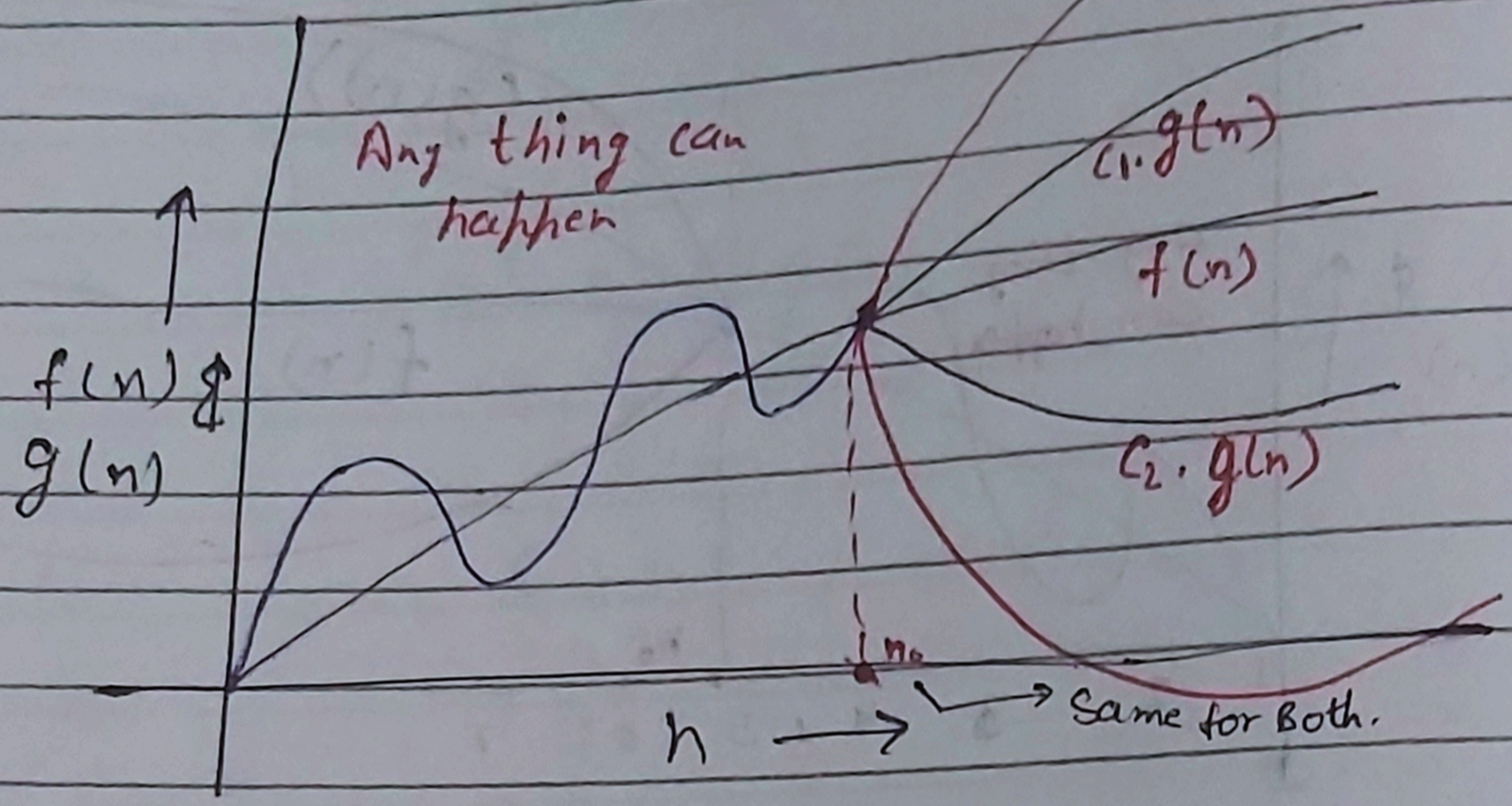
$n_0 \leq n \leq \infty$



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

$f(n) \geq C \cdot |g(n)|$

3. Theta (Θ) Notation →



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

$$\Downarrow$$

$$f(n) \leq c_1 \cdot g(n)$$

$$\&$$

$$f(n) \geq c_2 \cdot g(n)$$

1. a) Big - oh - notation →

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

$= O(n^3)$ Not Tightest UB

$= O(n^{10})$ ✓

$A = O(B)$

⇓

Here B is upper Bound

⇒ In all upper Bound which one is equal to the actual value is called as Tightest Upper Bound.

eg :- $n^2 = O(n^2)$ Tightest UB

Here B is UB. = (Both)

Tightest Not Tightest

b) Small Oh (o) Notation → Give only Not-Tightest U.B.

$n^2 = o(n^2)$ ✗

$n^2 = o(n^3)$ ✓

$n^2 = o(n^{10})$ ✓

$n^2 \leq n^2$

$n^2 < n^2$

$A = O(B)$

⇓

Here B is NTUB.

2. Omega Notation :

$n^3 = \Omega$ | (n^3) | Tightest Lower Bound

$n^3 = \Omega$ | (n^2) | NTLB

$n^3 = \Omega$ | (n) | NTLB

$A = \Omega(B)$

Here B is LB → T, NT

3. Small - omega - Notation (ω) →

$n^3 = \omega(n^3)$ ✗

$n^3 = \omega$ | (n^2) | NTLB

$n^3 = \omega$ | (n) | NTLB

$A = \omega(B)$

Here B is NTLB

PAGE NO.:

3. Theta Notation →

$n^3 = O(n^3)$ ⇒ TUB

$n^3 = \Omega(n^3)$ ⇒ TLB

$\Rightarrow n^3 = \Theta(n^3)$

⇓

Both TUB & TLB.

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

&

$n^2 = \Omega(n^2)$ TLB

⇓

$n^2 = \Theta(n^2)$ = TUB & TLB

$A = \Theta(B)$

⇓

Here B is Both TUB & TLB.

:- Conclusion

	O	o	Ω	ω	Θ
i)	≤	<	≥	>	=
ii)	TUB NTUB	NTUB	TLB NTLB	NTLB	TLB + TUB