ASYMPTOTIC ANALYSIS

- The main idea of asymptotic Analysis is to have a "measure of the efficiency of algorithms" that don't depend on machine-specific constants and don't require algorithms to be implemented and time taken by programs to be compared.

- Comparing of ~~items~~ terms:

$$C < \log(\log n) < \log n < n^{1/3} < n^{1/2} < n < n^2 < n^3 < n^4 < 2^n < n^n.$$

- Time complexity of all computer algorithms is $\Omega(1)$.

- The ~~ty~~ time required by an algorithm comes under 3 categories:
  1. WORST CASE: It defines the input for which the algorithm takes a huge time.
  2. AVERAGE CASE: It takes average time for the execution of program.
  3. BEST CASE: It defines the input for which the algorithm takes lowest time.

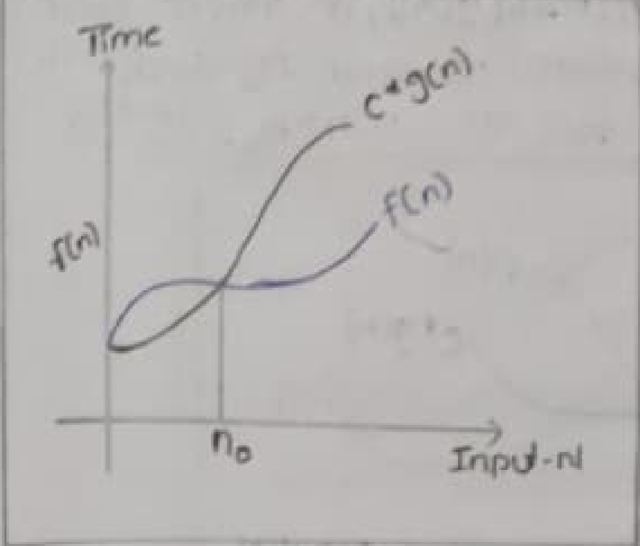
ASYMPTOTIC NOTATIONS

Asymptotic Notations are the "mathematical representations" of "time & Space complexities" in some approximate formats.

The notations are:

1. Big 'Oh' notation (O) $\leq$
2. Big 'omega' notation (Ω) $\geq$
3. Theta notation (Θ) $=$
4. Little 'oh' notation (o) $<$
5. Little 'omega' notation (ω) $>$

1. BIG 'Oh' NOTATION (O) : $\leq$



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

- The function "$f(n) = O(g(n))$", if there exist two positive constants c and n_0, such that "$f(n) \leq c * g(n)$" for all $n, n \geq n_0, c > 0, n_0 >= 1$.
- The O(n) is the way to express upper bound of an algorithm's running time.
- It measures the WORST CASE TIME complexity or Longest amount of time an algorithm can possibly take to complete.

EXAMPLE 1 :

Consider $f(n) = 3n+2$.
Assume that $3n+2 \leq 4n$.
Let

$n=1$	$3(1)+2 \leq 4(1)$	$\rightarrow 5 \leq 4$	False
$n=2$	$3(2)+2 \leq 4(2)$	$\rightarrow 8 \leq 8$	True ✓
$n=3$	$3(3)+2 \leq 4(3)$	$\rightarrow 11 \leq 12$	True

From this,

$c = 4$ $g(n) = n$ and $n_0 = 2$.

Hence, the function $3n+2 = O(n)$, if there exist two +ve's 4 & 2 such that $3n+2 \leq 4n$ for all $n, n \geq 2$.

Example 2 :

The function $n^2+n+3 = O(n^2)$, if there exist two +ve constants 2 and 3 such that, $n^2+n+3 \leq 2n^2$ for all $n, n \geq 3$.

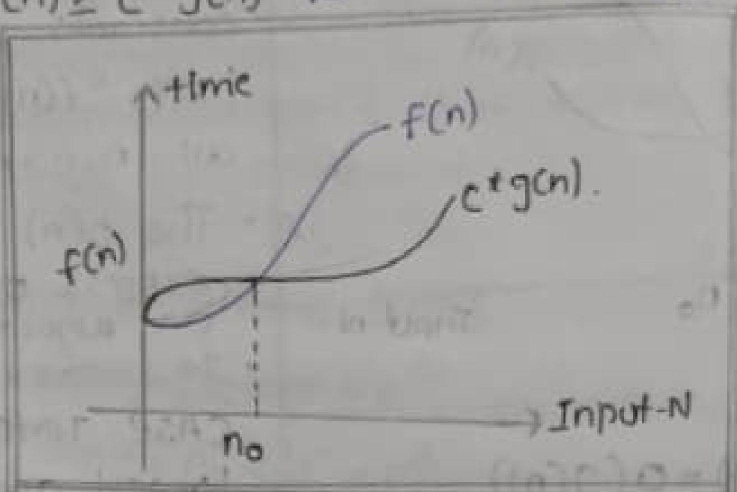
In these complexities,

- O(1) means Constant
- O(n) means linear
- O(log n) means Logarithmic.
- $O(n^2)$ means quadratic.
- $O(n^3)$ means cubic.
- $O(2^n)$ means exponential.

$$\boxed{O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2) < O(n^3) < \ldots\ldots < O(2^n)}$$

2. BIG 'Omega' Notation (Ω): $\geq$

The function $f(n) = \Omega(g(n))$, if there exist two positive constants c and n_0 such that $f(n) \geq c * g(n)$ for all $n, n \geq n_0$.



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

- The $\Omega(n)$ is the way to express the "LOWER BOUND" of an algorithm's running time.
- It measures the "BEST CASE" time complexity or "BEST AMOUNT" of time an algorithm can possibly take to complete.
- EXAMPLE:

consider $f(n) = 3n+2$

Assume that $3n+2 \geq 3n$.

Let

$n=1$ $3(1)+2 \geq 3(1)$ $\Rightarrow 5 \geq 3$ True

$n=2$ $3(2)+2 \geq 3(2)$ $\Rightarrow 8 \geq 6$ True

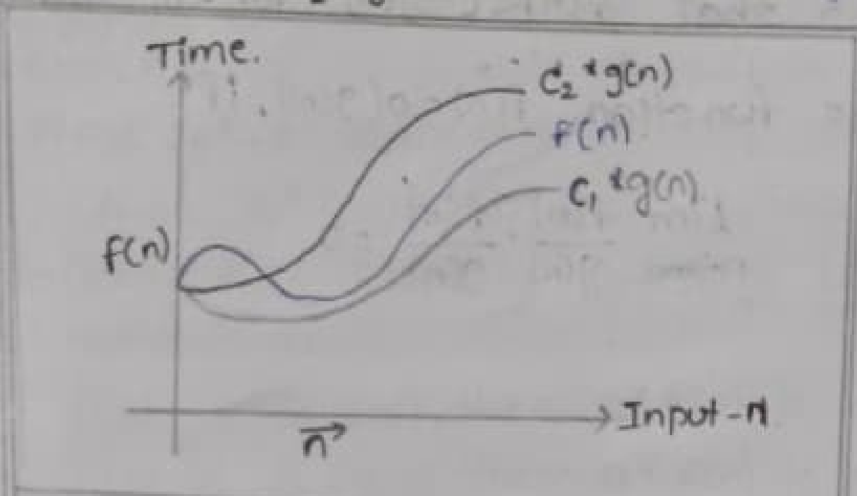
From this,

$c = 3$ $g(n) = n$ and $n_0 = 1$.

Hence, the function $3n+2 = \Omega(n)$, if there exist two positive constants 3 and 1 such that $3n+2 \geq 3n$ for all $n, n \geq 1$.

3. Theta - NOTATION (Θ):

The function $f(n) = \Theta(g(n))$, if there exist three 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 $n, n \ge n_0$.



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

- The $\Theta(n)$ is the way to express both "lower bound" and "upper bound" of an algorithm's running time.
- It measures the "AVERAGE CASE" time complexity.
- EXAMPLE:

  consider $f(n) = 3n+2$.

  Assume that $3n \le 3n+2 \le 4n$

let,

$n=1 \quad 3(1) \le 3(1)+2 \le 4(1) \rightarrow 3 \le 5 \le 4$ False

$n=2 \quad 3(2) \le 3(2)+2 \le 4(2) \rightarrow 6 \le 8 \le 8$ True

$n=3 \quad 3(3) \le 3(3)+2 \le 4(3) \rightarrow 9 \le 11 \le 12$ True.

From this,

$c_1 = 3 \quad c_2 = 4 \quad g(n) = n$ and $n_0 = 2$.

Hence, the function $3n+2 = \Theta(g(n))$, if there exist three positive constants 3, 4 and 2 such that, $3n \le 3n+2 \le 4n$ for all $n, n \ge 2$.

4. LITTLE 'Oh' NOTATION (o): <

The function $f(n) = o(g(n))$, if there exist two positive constants c and n_0 such that "$f(n) < c^* g(n)$" for all $n, n \geq n_0$.

The function $f(n) = o(g(n))$, if

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} \cdot \frac{f(n)}{g(n)} = 0.$$

5. LITTLE 'Omega' NOTATION (ω) : >

The function "$f(n) = \omega(g(n))$", if there exist two positive constants c and n_0 such that "$f(n) > c^* g(n)$" for all $n, n \geq n_0$.

The function $f(n) = \omega(g(n))$, if

$$\lim_{n \to \infty} \frac{g(n)}{f(n)} \cdot \frac{g(n)}{f(n)} = 0.$$

PROPERTIES OF ASYMPTOTIC NOTATIONS:

1. GENARAL PROPERTIES:

- if $f(n)$ is $O(g(n))$, then $a^* f(n)$ is also $O(g(n))$; where 'n' is constant.
- EXAMPLE:

  $f(n) = 2n^2 + 5$ is $O(n^2)$.

  then $7^* f(n) = 14n^2 + 35$ is also $O(n^2)$.
- Similarly, this property satisfies both Θ and Ω notations.

∴ If $f(n)$ is $\Theta(g(n))$, then $a^* f(n)$ is also $\Theta(g(n))$

If $f(n)$ is $\Omega(g(n))$, then $a^* f(n)$ is also $\Omega(g(n))$

2. TRANSITIVE PROPERTIES:

- If $f(n)$ is $O(g(n))$ and $g(n)$ is $O(h(n))$, then $f(n) = O(h(n))$.
- EXAMPLE:

  If $f(n) = n$, $g(n) = n^2$ and $h(n) = n^3$

  n is $O(n^2)$ and n^2 is $O(n^3)$.

  then 'n' is $O(n^3)$.
- Similarly, this property satisfies both Θ & Ω.

3. REFLEXIVE PROPERTIES:

- If $f(n)$ is given, then $f(n)$ is $O(f(n))$. Since, MAXIMUM VALUE of $f(n)$ will be $f(n)$ ITSELF!. Hence, $x = f(n)$ and $y = O(f(n))$, tie themselves in reflexive relation always.
- EXAMPLE:

  $f(n) = n^2$; $O(n^2)$ i.e, $O(f(n))$.
- Similarly, this property satisfies both Θ and Ω notations.

4. SYMMETRIC PROPERTIES:

- If $f(n)$ is $\Theta(g(n))$, then $g(n)$ is $\Theta(f(n))$.
- EXAMPLE:
  $f(n)=n^2$ and $g(n)=n^2$, then
  $f(n)=\Theta(n^2)$ and $g(n)=\Theta(n^2)$.
- This property only satifies for Θ notation.

5. TRANSPOSE SYMMETRIC PROPERTY:

- If $f(n)$ is $O(g(n))$, then $g(n)$ is $\Omega(f(n))$.
- EXAMPLE:
  $f(n)=n$, $g(n)=n^2$.
  then n is $O(n^2)$ and n^2 is $\Omega(n)$.
- This property satifies only O & Ω notation

6. SOME MORE PROPERTIES:

1. If $f(n)=O(g(n))$ and $f(n)=\Omega(g(n))$
   then $f(n)=\Theta(g(n))$.
2. If $f(n)=O(g(n))$ and $d(n)=O(e(n))$
   then $f(n)+d(n)=O(\max(g(n),e(n)))$.
   EXAMPLE:
   $f(n)=n$ i.e $O(n)$
   $d(n)=n^2$ i.e $O(n^2)$
   then, $f(n)+d(n)=n+n^2$, i.e $O(n^2)$.
3. If $f(n)=O(g(n))$ and $d(n)=O(e(n))$
   then $f(n)*d(n)=O(g(n)*e(n))$
   EXAMPLE:
   $f(n)=n$, i.e $O(n)$
   $d(n)=n^2$ i.e $O(n^2)$
   then,
   $f(n)*d(n)=n*n^2=n^3$ i.e, $O(n^3)$.