- Big O Notation (complexity theory) $\boxed{\text{Ordo } (O)}$.

  - it's called Landau Notation
  - it describes the limiting behaviour of a function. when the argument tends towards a perticular value or infinity.
  - used to classify algorithems by how they respond (time and space requirements) to changes in the N input size
  - usually we are interested in large input sizes (Asymtotic Analysis)
  - we will drop all the terms that grow slowly & only keep the ones that grow fast as N (so the input size) becomes larger
  - Mathematical definition:

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

-> where

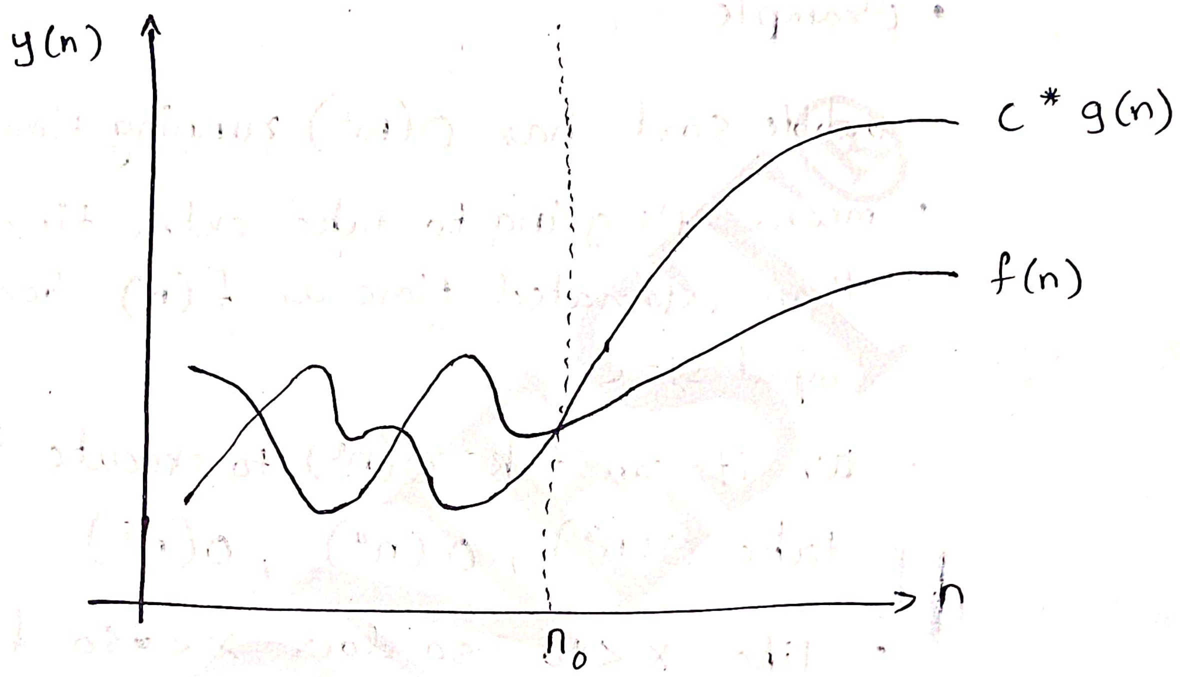
- f = measures the running time of ~~function~~ algorithem
- n = input size
- g = also a function.

- this means that there is some $c > 0$ value & some $n_0 > 0$ threshhold value such that for

- $n > n_0$ the absolute (| |) value ~~f(n)|≤c*|g(~~

$$\boxed{|f(n)| \leq c * |g(n)|}$$

- graph:



- when n (input size) is small we don't really care which one is greater f(n) or g(n)
- in this above image as you can see sometimes f(n) is greater and sometimes c*g(n)
- whats more crucial is that if n is larger than n_0, as we can see in right side of illustration c*g(n) tends to go larger than f(n)
- if we find c = constant, n_0 threshold such that the f(n) function is smaller or equals than c*g(n) ~~that~~ that's where we say

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

- essentially $O(g(n))$ defines the upper bound for the $f(n)$ function.

- <u>Example</u>

- bubble sort has $O(n^2)$ running time complexity
- means it's going to take extra time to execute than estimated time as $f(n)$ based on the input size
- as it can tak $O(n^2)$ to execute it also may take $O(n^3)$, $O(n^4)$, $O(n^{15})$
- like $x < 10$ so does $x < 50$ & $x < 100$

- we only consider larger elements based tests because that's where you exactly understand how it's effecient otherwise with less elements every algorithm is going to be fast.

there are 7 types of notation

$O(1)$ = Constant
$O(\log n)$ = logarithmic
$O(n)$ = linear
$O(n \log n)$ = linearithmic
$O(n^2)$ = quadratic
$O(2^n)$ = exponential
$O(n!)$ = factorial.

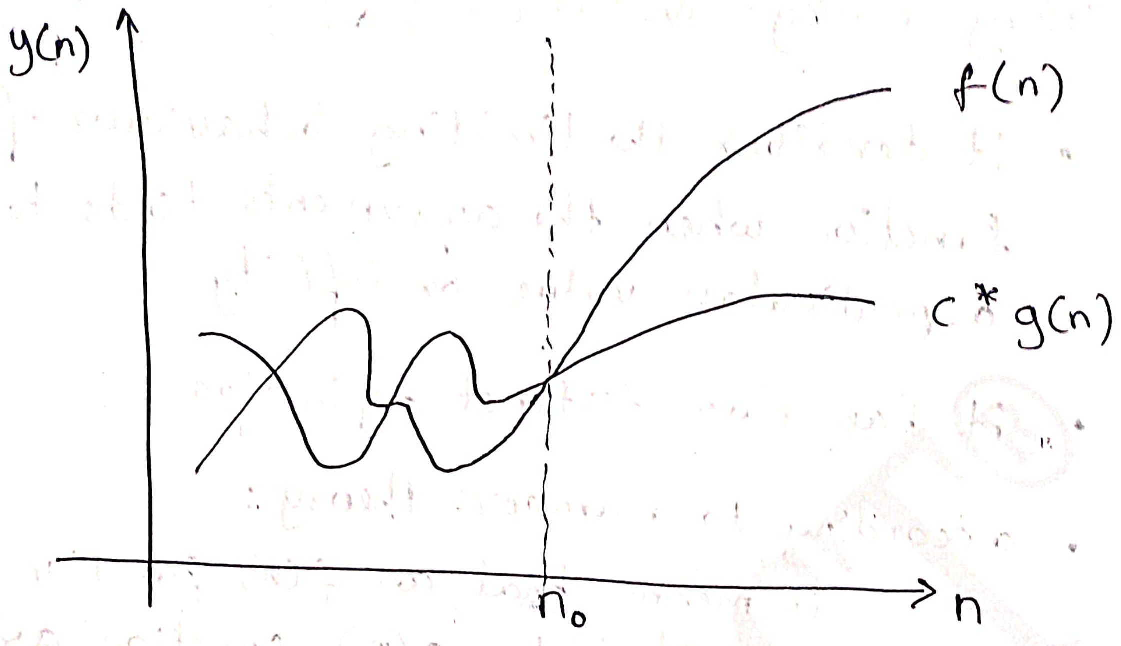
Big Omega Notation (Ω)

- it describes the limiting behaviour of an f(n) function when the arguments tends towards a perticular value or infinity
- it has two distinct definition
- according to numbers theory:
  it means that an f(n) function is not dominated by g(n) function asymptotically
- according to complexity theory:
  it means that an f(n) function is bounded by an g(n) function asymptotically
- the ~~f(n)~~ Mathematical definition

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

- this means, that there is some $c > 0$ value & some $n_0 > 0$ threshhold value such that for $n > n_0$ absolute value would be

$$\boxed{|f(n)| \geq c * |g(n)|}$$



- when n is smaller than threshhold (n_0) ~~the~~ sometimes f(n) is greater & somethimes the c* g(n)

- but whe n is larger than threshhold (n_0) the f(n) function tends to be larger than the c* g(n)

- essentially $\Omega(g(n))$ defines the lower bound for the f(n) function.

- Example: let's assume that algorithem has the $\Omega(N^2)$ ~~com~~ time complexity

- so, it's also can also $\Omega(N)$, $\Omega(\log N)$, $\Omega(1)$

- like, if $x > 10$ then $\rightarrow$ $x > 1$ & $x > -5$

Big theta Notation (Θ)

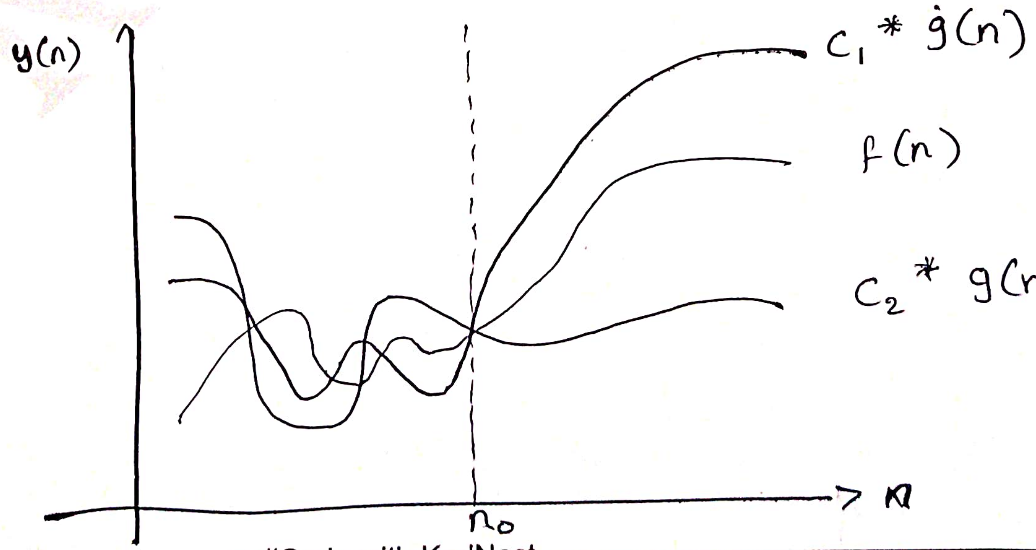
- it describes the limiting behaviour of f(n) function when the argument tends towards a perticular value or infinity
- the f(n) function is bounded both below & above by a g(n) function asymptotically such that $f(n) = \Omega(g(n))$ & $f(n) = O(g(n))$
- mathematical definition

c = constant.

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

- means that, there is some $c_1, c_2 > 0$ value and some $n_0 > 0$ threshhold value such that for $n > n_0$ the

$$\boxed{|c_1 * |g(n)| \geq |f(n)| \geq c_2 * |g(n)|}$$



- the upper bound of the f(n) function will be $c_1 * g(n)$
- and the lower bound of the f(n) function will be $c_2 * g(n)$
- essentially $\Theta(g(n))$ defines the lower & upper bounds for the f(n) function.