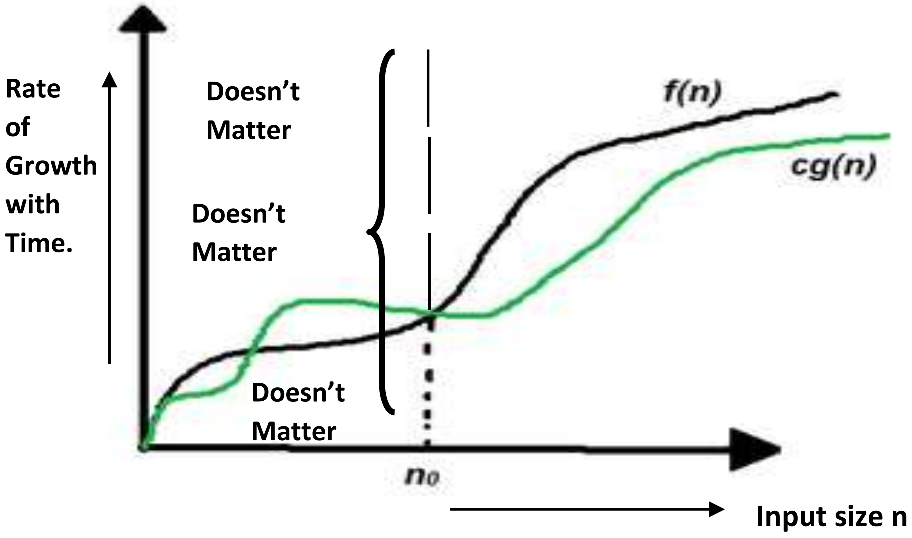
9.B. BIG OMEGA NOTATION

The lower bound of an algorithm is given by the big-omega (Ω) notation.



DEFINITION: A function $f(n)$ is said to be in $\Omega(g(n))$, denoted $f(n)\ \varepsilon\ \Omega(g(n))$, *if $f(n)$ is bounded below by some positive constant multiple of $g(n)$ for all large n, i.e., if there exist some positive constant c and some nonnegative integer n_0 such that*:

$$f(n) \geq c \times g(n)\ for\ all\ n \geq n_0$$

ON THE ABOVE DIAGRAM 'STARTING FROM n_0 AND BEYOND ONLY MATTERS', BUT THE PORTION LESSER THAN AND WITHOUT THE STARTING POINT OF n_0 DOESN'T MATTER.

ILLUSTRATION OF THE DEFINITION

- Let f and g be two functions that map a set of natural numbers to a set of positive real numbers , that is $f: \mathbb{N} \to \mathbb{R}_{\geq 0}$.

- Let $\Omega(g)$ be the set of all those functions that have a similar rate of growth.

- The relation $f(n) = \Omega\big(g(n)\big)$ holds good if there exist two positive constants c and n_0 such that $f(n) \geq c \times g(n)$.

- Thus, the function $f(n)$ is said to be in $\Omega\big(g(n)\big)$, which can be represented as $f(n) \epsilon\ \Omega\big(\mathrm{g}(\mathrm{n})\big)$.

- This notation implies that $f(n)$ grows at a faster rate than a constant time $g(n)$ for a sufficiently large n .

The "omega notation" is used when the lower bound of a polynomial is to be found.

THE NEED OF BIG OMEGA (Ω) NOTATION:

- The notation is helpful in finding out the minimum amount of resources, an algorithm requires, in order to run.

- Finding out the minimum amount of resources is important as this time complexity can help us to schedule the task accordingly.

- It is also helpful to compare the best suited algorithm amongst the set of algorithms, if more than one algorithm can accomplish a given task.

Hence:

$\boldsymbol{f(n) = \Omega(g(n)), if\ f(n) \geq c\ \times g(n), n \geq n_0, where\ c\ and\ n_0\ are}$ **constants.**

i.e.

$\boldsymbol{\Omega(g(n)) = \{f(n): there\ exists\ positive\ constants\ c\ and\ n_0}$

$\boldsymbol{such\ that\ 0 \leq c \times g(n) \leq f(n)\ for\ all\ n \geq n_0\}}$

And if we see the rate of growth of $\boldsymbol{f(n)\ and\ g(n)}$, if $\boldsymbol{f(n) = 5n^2 + 2n + 5\ and\ g(n) = 4n^2, then}$:

Comparison of $\boldsymbol{f(n)}$ and $\boldsymbol{g(n)}$

$\boldsymbol{N}$	$\boldsymbol{5n^2 + 2n + 5}$	$\boldsymbol{4n^2}$
1	12	4
2	29	16
3	56	36
4	93	64
5	140	100
6	197	144

- $\boldsymbol{g(n)\ is\ an\ asymptotic\ tight\ lower\ bound\ for\ f(n)}$.
- Hence the Big-Omega notation gives the tighter lower bound for the given algorithm.
- Our objective is to give the largest rate of growth $\boldsymbol{g(n)}$ which is less than or equal to the given algorithm's rate of growth $\boldsymbol{f(n)}$.

****