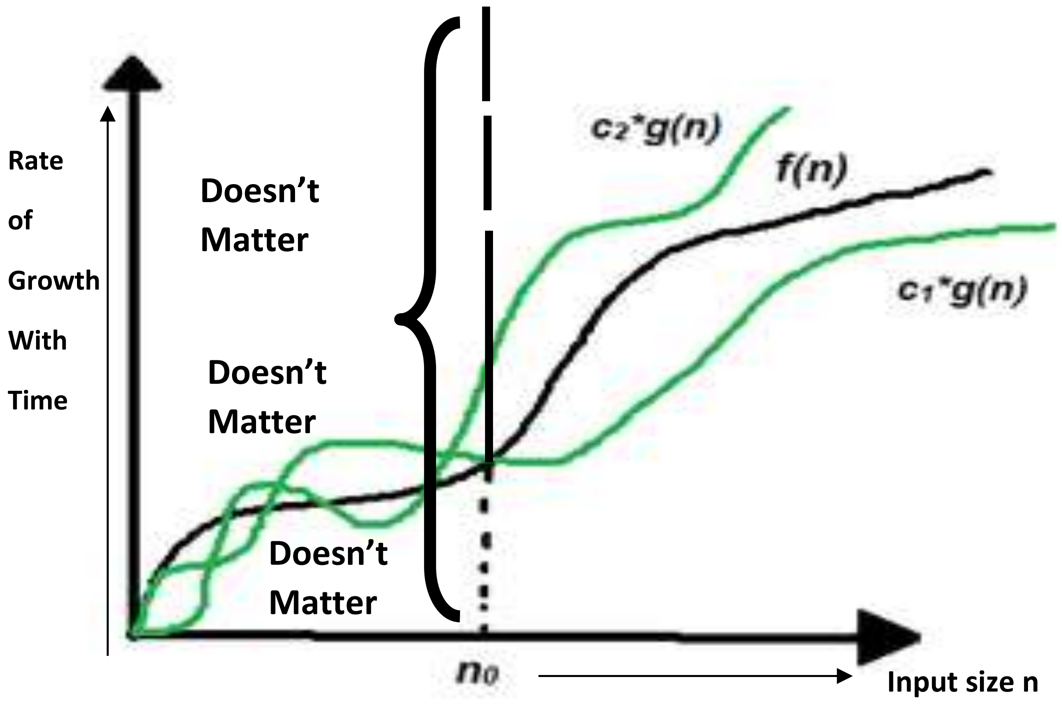
9.C. BIG THETA (Θ) NOTATION



Definition: $A\ function\ f(n) is\ said\ to\ be\ in\ \Theta(g(n)), denoted$ $f(n) \in \Theta(\mathrm{g(n)}), \mathrm{if\ f(n)}\ is\ bounded\ both\ above\ and$ $below\ by\ some\ positive\ constant\ multiples\ of\ g(n) for\ all$ $large\ n, i.e. if\ there\ exist\ some\ positive\ constants\ c_1\ and\ c_2$ $and\ some\ nonnegative\ integer\ n_0\ such\ that:$

$$c_2 g(n) \leq f(n) \leq c_1 g(n)\ \ for\ all\ n \geq n_0$$

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 $t\ and\ g$ be two functions that map a set of natural numbers to a set of positive real numbers.

- If the relationship $c_1 \times g(n) \leq f(n) \leq c_2 \times g(n)$ holds good for all $n \geq n_0\ and\ for\ constants\ c_1 and\ c_2, then,$

$$f(n) can\ be\ denoted\ as\ f(n) = \Theta\big(g(n)\big)$$

- This is equivalent to saying that $f(n)$ grows at the same rate as a constant time $g(n)$ for all sufficiently large values of n.

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

THE NEED OF BIG THETA (Θ) NOTATION:

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

- Finding out the bounds of resources (or time) is important as this can help us to schedule the task accordingly.

- The notation is also helpful in finding the best-suited algorithm amongst the set of algorithms, if more than one algorithm can accomplish a given task.

- Both the upper and the lower bound can be illustrated by this notation. The average-case complexity is also given by the Θ notation.

- This notation decides whether the upper and lower bounds of a given function (Algorithm) are the same.

- The average running time of an algorithm is always between the lower bound and the upper bound.

- If the upper bound $(\mathbf{O})$ and lower bound $(\boldsymbol{\Omega})$ give the same result, then the $(\boldsymbol{\Theta})$ notation will also have the same rate of growth, this also means when the rates of growth in the best case and worst case are the same. Then, the average case will also be the same.

- For a given function(algorithm) , if the rates of growth (bounds) for $\mathbf{O}$ *and* $\boldsymbol{\Omega}$ are not the same, then the rate of growth for the $\boldsymbol{\Theta}$ case may not be the same. For such a case, we need to consider all possible time complexities and take the average of those.

Hence by definition we get:

$f(n) = \Theta(g(n)), if\ C_1 g(n) \leq f(n) \leq C_2 g(n), n \geq n_0, C\ and\ n_0$

$are\ constants.$

i.e.

$\Theta(g(n\,)) = \{f(n): there\ exist\ positive\ constants\ c_1, c_2$ and n_0 $such\ that\ 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)\ for\ all$ $n \geq n_0\}.$

Say,

$f(n) = 3n^2 + 2n + 1$

$c_1 g(n); here\ g(n) = 2n^2$

$c_2 g(n); here\ g(n) = 4n^2$

n	$3n^2 + 2n + 1$	$4n^2$	$2n^2$
1	6	4	2
2	17	16	8
3	34	36	18
4	57	64	32
5	86	100	50
6	121	144	72
7	162	196	98
8	209	256	128
9	262	324	162
10	321	400	200
11	386	484	242

Hence $4n^2 \geq\ 3n^2 + 2n + 1\ and\ 3n^2 + 2n + 1 \leq 2n^2$

$i.e, 0 \leq c_1 g(n) = 2n^2 \leq f(n) = 3n^2 + 2n + 1 \leq c_2 g(n) = 4n^2$

$Where\ c_1 = 2\ and\ c_2 = 4.$

- g(n) is an asymptotic tight bound for $f(n)$.
- $\Theta(g(n))\ is\ the\ set\ of\ functions\ with\ the\ same$ order of growth as $g(n)$.

****