Theorem

If $f(n) = a_m n^m + \cdots + a_1 n + a_0$ then $f(n) = O(n^m)$

Proof:- $f(n) \leq \sum_{i=0}^{m} |a_i| n^i$

$$\leq n^m \sum_{i=0}^{m} |a_i| n^{i-m}$$

$$\leq n^m \sum_{i=0}^{m} |a_i| \quad \text{for } n \geq 1$$

So, $f(n) = O(n^m)$ (assuming that m is fixed).

Big-Ω notation

Definition:- $f(n) = \Omega(g(n))$ iff there are two positive constants c and n_0 such that

$$|f(n)| \geq c|g(n)| \text{ for all } n \geq n_0.$$

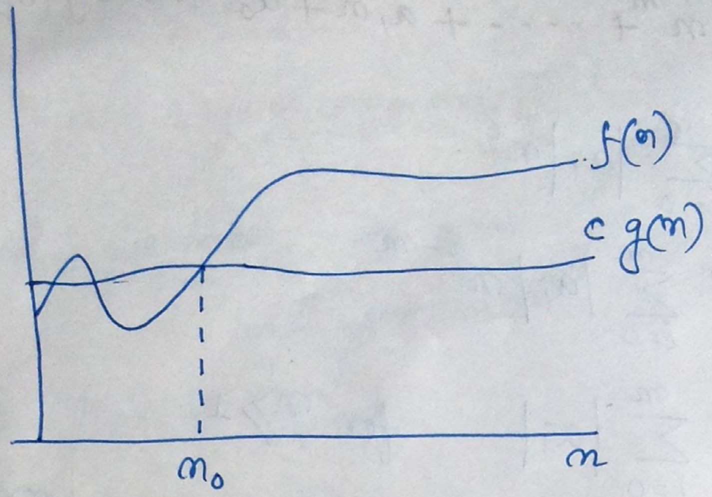
If $f(n)$ is non negative, we can simplify the last condition to

$$0 \leq c\, g(n) \leq f(n), \text{ for all } n \geq n_0.$$

We say that <u>$f(n)$ is big-omega of $g(n)$</u>.

As n increases, $f(n)$ grows no slower ~~that~~ than $g(n)$.
In other words, $g(n)$ is an <u>asymptotic lower bound</u> on $f(n)$.

Scanned by CamScanner



Example : $\sqrt{n} = \Omega(\lg n)$, with $c = 1$ and $n_0 = 16$.

Examples of functions in $\Omega(n^2)$:

n^2
$n^2 + n$
$n^2 - n$
$1000n^2 + 1000n$
$1000n^2 - 1000n$

also,
n^3
$n^{2.0001}$
$n^2 \lg\lg\lg n$
2^{2^n}

Explain with Example :-

$$n^3 + 4n^2 = \Omega(n^2)$$

Proof :-
here we have, $f(n) = n^3 + 4n^2$
and $g(n) = n^2$

Notice that, if $n \geq 0$.

$$n^3 \leq n^3 + 4n^2$$

and if $n \geq 1$

$$n^2 \leq n^3$$

Thus when,

$n \geq 1,$

$n^2 \leq n^3 \leq n^3 + 4n^2$

Therefore,

$1.n^2 \leq n^3 + 4n^2$ for all $n \geq 1$

Thus, we have shown that $n^3 + 4n^2 = \Omega(n^2)$ with $n_0 = 1$, and $c = 1$

Some Examples:-

(2) $3n+2 = \Omega(n)$ as $3n+2 \geq 3n$, for all $n \geq 1$

(3) $3n+3 = \Omega(n)$ as $3n+3 \geq 3n$ for all $n \geq 1$.

(4) $10n^2+4n+2 = \Omega(n^2)$ as $10n^2+4n+2 \geq n^2$ for $n \geq 1$.

(5) $6 \cdot 2^n + n^2 = \Omega(2^n)$ as $6 \cdot 2^n + n^2 \geq 2^n$ for $n \geq 1$.

But here we also observe that,

$3n+3 = \Omega(1)$, $10n^2+4n+2 = \Omega(n)$,

$10n^2+4n+2 = \Omega(1)$.

Big-Θ notation.

Definition:- $f(n) = \Theta(g(n))$ iff there are three positive constants c_1, c_2 and n_0 such that,

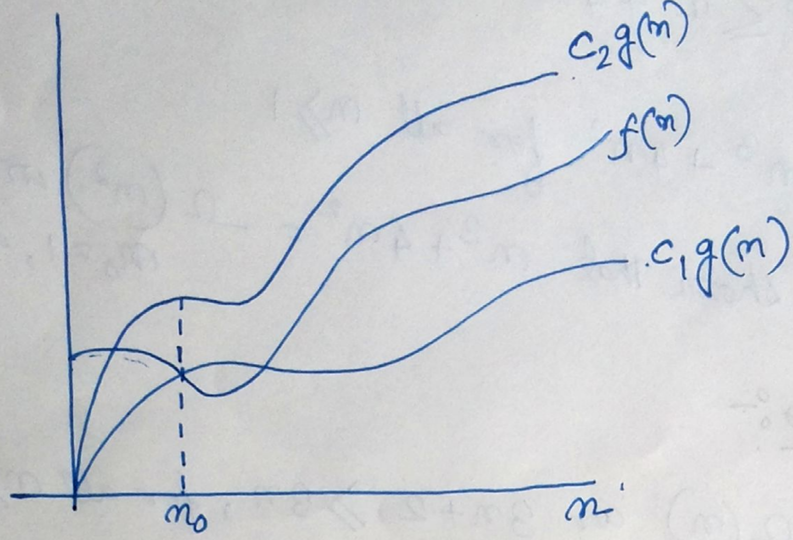
$c_1 |g(n)| \leq |f(n)| \leq c_2 |g(n)|$ for all $n \geq n_0$.

If $f(n)$ is nonnegative, we can simplify the last condition to

$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all $n \geq n_0$.

we say that $f(n)$ is theta of $g(n)$.

$g(n)$ is an asymptotically tight bound for $f(n)$.



As n increases, $f(n)$ grows at the same rate as $g(n)$.

Example :-

$$n^2+5n+7 = \Theta(n^2)$$

Proof :-.

- when $n \geq 1$,

$$n^2+5n+7 \leq n^2+5n^2+7n^2 \leq 13n^2$$

- when $n \geq 0$

$$n^2 \leq n^2+5n+7$$

- Thus when $n \geq 1$

$$1.n^2 \leq n^2+5n+7 \leq 13n^2.$$

Thus, $n^2+5n+7 = \Theta(n^2)$ with $n_0 = 1$, $c_1 = 1$ and $c_2 = 13$.

Some examples:-

2. $3n+2=\theta(n)$ as $3n+2 \geq 3n$ for all $n \geq 2$ and $3n+2 \leq 4n$ for all $n \geq 2$, so $c_1=3$, $c_2=4$ and $n_0=2$.

3. $3n+3=\theta(n)$,

4. $10n^2+4n+2=\theta(n^2)$

But,

$3n+2 \neq \theta(1)$,

$3n+3 \neq \theta(n^2)$

$10n^2+4n+2 \neq \theta(n)$

$10n^2+4n+2 \neq \theta(1)$.

Growth of functions:-

2^n

n^3

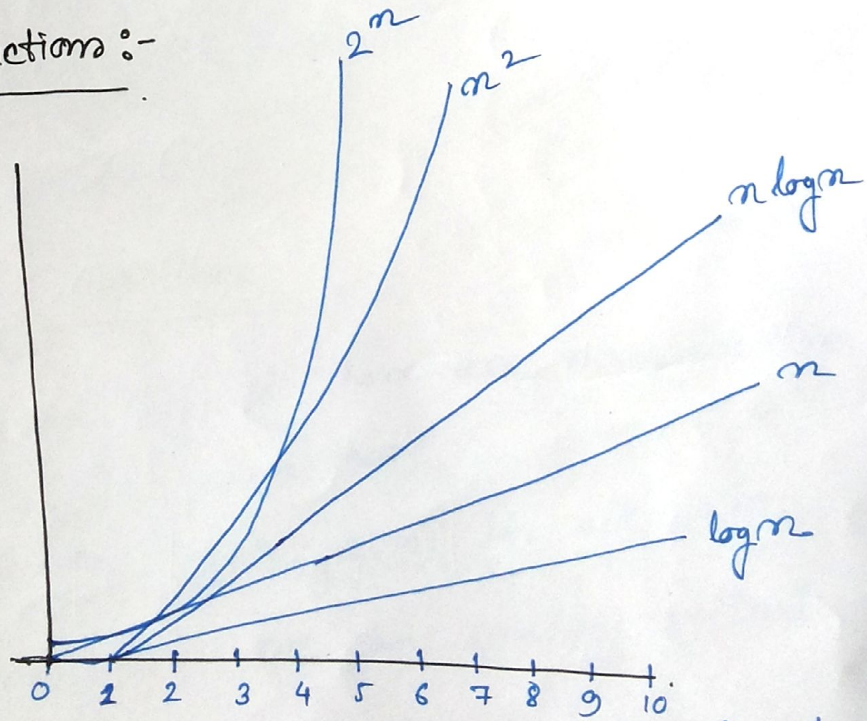
n^2

$n \log n$

n

$\log n$

1



If an algorithm takes time $O(\log n)$, it is faster, for sufficiently large n, than if it had taken $O(n)$. Similarly $O(n \log n)$ is better than $O(n^2)$ but no so good as $O(n)$.