Text Book:
Introduction to the Design and Analysis of Algorithms
Author: Anany Levitin
$2^{nd}$ Edition

Chapter 2 section2.2

Orders of growth of an algorithm's basic operation count is important

We compare order of growth of functions using asymptotic notations

## Asymptotic notations

A way of comparing functions that ignores constant factors and small input sizes

O($g$($n$)): class of functions $f$($n$) that grow ***no faster*** than $g$($n$)

Ω($g$($n$)): class of functions $f$($n$) that grow ***at least as fast*** as $g$($n$)

Θ ($g$($n$)): class of functions $f$($n$) that grow ***at same rate*** as $g$($n$)

o($g$($n$)): class of functions $f$($n$) that grow ***at slower rate*** than $g$($n$)

w($g$($n$)): class of functions $f$($n$) that grow ***at faster rate*** than $g$($n$)

## Big O notation

Formal definition

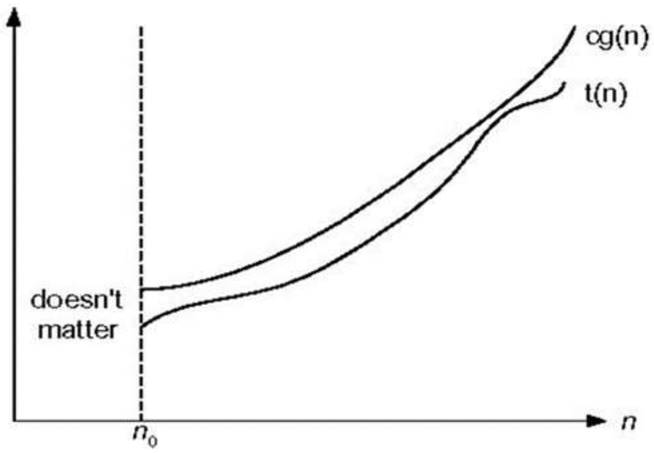
A function *t(n)* is said to be in *O(g(n)),* denoted *t(n)* $\in$*O(g(n)),*

if *t(n)* is bounded above by some 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

$$t(n) \leq cg(n) \text{ for all } n \geq n_0$$

Example: 100n+5 ∈ O(n)




## Big Omega Notation

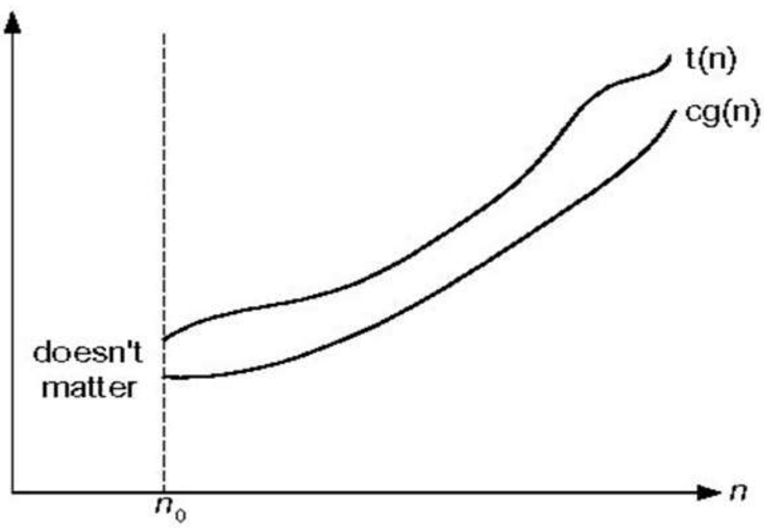
Formal definition

A function *t(n)* is said to be in $\Omega(g(n))$, denoted $t(n) \in \Omega(g(n))$, if *t(n)* is bounded below by some constant multiple of *g(n)* for all large *n*,

i.e., <u>if there exist some positive constant c and some nonnegative integer $n_0$ such that</u>

$$t(n) \geq cg(n) \text{ for all } n \geq n_0$$

Example: $10n^2 \in \Omega(n^2)$



## Theta Notation

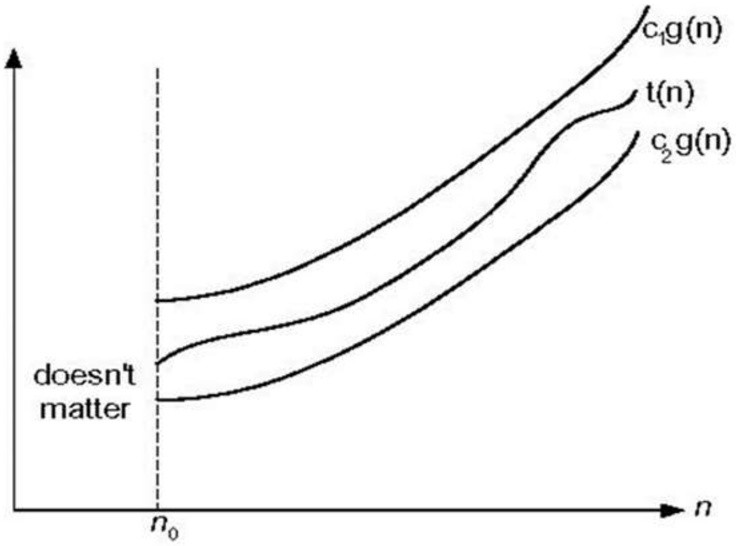
Formal definition

A function *t(n)* is said to be in $\Theta(g(n))$, denoted $t(n) \in \Theta(g(n))$, if *t(n)* is bounded both above and below by some positive constant multiples of *g(n)* for all large *n*,

i.e., <u>if there exist some positive constant $c_1$ and $c_2$ and some nonnegative integer $n_0$ such that</u>

$$c_2\, g(n) \leq t(n) \leq c_1\, g(n) \text{ for all } n \geq n_0$$

Example: $(1/2)n(n-1) \in \Theta(n^2)$




## Small o notation

Formal Definition:

A function *t(n)* is said to be in Little-o*(g(n))*, denoted $t(n) \in o(g(n))$,

<u>if for any positive constant c and some nonnegative integer $n_0$</u>

$$0 \leq t(n) < cg(n) \text{ for all } n \geq n_0$$

Example:

If $f(n) = n$ & $g(n) = n^2$,

then for any value of c>0,

$f(n) < c(n^2)$

$f(n) \in o(g(n))$

## Small omega notation

Formal Definition:

A function *t(n)* is said to be in Little- w*(g(n)),* denoted *t(n)* $\in$ w*(g(n)),*
<u>if for any positive constant c and some nonnegative integer $n_0$</u>

$t(n) > cg(n) \geq 0$ for all $n \geq n_0$

Example : If $f(n) = 3n^2 + 2$, $g(n) = n$

then for any value of c>0

f(n)> cg(n)

$f(n) \in w(n)$

## Theorems

- If $t_1(n) \in O(g_1(n))$ and $t_2(n) \in O(g_2(n))$, then

  $t_1(n) + t_2(n) \in O(\max\{g_1(n), g_2(n)\})$.

  For example,

  $5n^2 + 3n\log n \in O(n^2)$

- If $t1(n) \in \Theta(g1(n))$ and $t2(n) \in \Theta(g2(n))$, then

  $t1(n) + t2(n) \in \Theta(\max\{g1(n), g2(n)\})$

- $t1(n) \in \Omega(g1(n))$ and $t2(n) \in \Omega(g2(n))$, then

  $t1(n) + t2(n) \in \Omega(\max\{g1(n), g2(n)\})$