Asymptotic Analysis

Session Learning Outcome-SLO

- Estimate algorithmic complexity
- Learn approximation tool
- Specify the behaviour of algorithm

Asymptotic Analysis

- Analysis of a given algorithm with larger values of input data
- Theory of approximation.
- Asymptote of a curve is a line that closely approximates a curve but does not touch the curve at any point of time
- Used by paul bachman in number theory
- Specify the behaviour of the algorithm when the input size increases

Asymptotic Notation

- The order of growth of the running time of an algorithms, gives a simple characterization of algorithms efficiency.
- It also allows to compare relative performance of alternative algorithms.
- ***Asymptotic order*** is concerned with how the running time of an algorithm increases with the size of the input, if input increases from small value to large values

1. Big-Oh notation (O)
2. Big-Omega notation (Ω)
3. Theta notation (θ)
4. Little-oh notation (o)
5. Little-omega notation (ω)

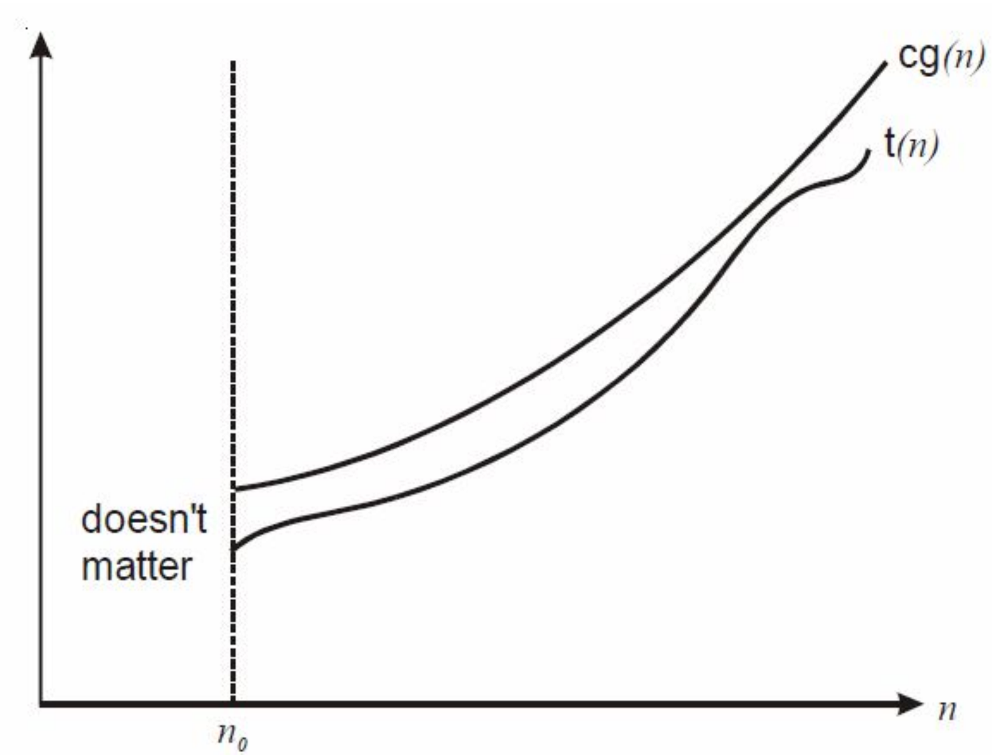
Big-Oh Notation (O)

- Big-oh notation is used to define the worst-case running time of an algorithm and concerned with large values of n.
- ***Definition:*** A function $t(n)$ is said to be in $O(g(n))$, denoted as $t(n) \in O(g(n))$, 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 non-negative integer $n0$ such that

$$t(n) \leq cg(n) \text{ for all } n \geq n0$$

- **O(*g*(*n*)):** Class of functions $t(n)$ that grow no faster than $g(n)$.
- Big-oh puts asymptotic ***upper bound*** on a function.

Big-Oh Notation (O)



Big-Oh Notation (O)

$1 < \log n < \sqrt{n} < n < n \log n < n^2 < n^3 < \ldots\ldots\ldots\ldots\ldots < 2^n < 3^n < \ldots . < n^n$

- Let $t(n) = 2n + 3$ upper bound

  $2n + 3 \leq$ ______??

  $2n + 3 \leq 5n \quad n \geq 1$

  here $c = 5$ and $g(n) = n$

  $t(n) = O(n)$

  $2n + 3 \leq 5n^2 \quad n \geq 1$

  here $c = 5$ and $g(n) = n^2$

  $t(n) = O(n^2)$

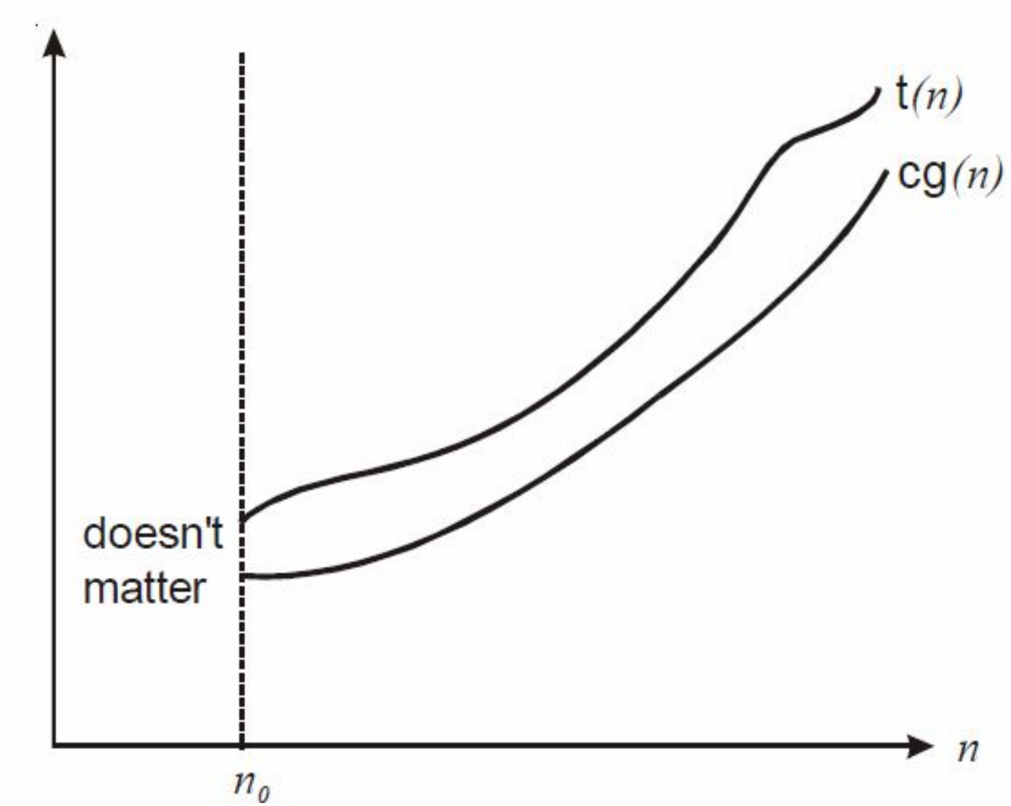
Big-Omega notation (Ω)

- This notation is used to describe the best case running time of algorithms and concerned with large values of n.
- ***Definition:*** A function $t(n)$ is said to be in $\Omega(g(n))$, denoted as $t(n) \in \Omega(g(n))$, if $t(n)$ is bounded below by some positive constant multiple of $g(n)$ for all large n. i.e., there exist some positive constant c and some non-negative integer $n0$. Such that

$$t(n) \geq cg(n) \text{ for all } n \geq n0$$

- It represents the ***lower bound*** of the resources required to solve a problem.

Big-Omega notation (Ω)



Big-Omega notation (Ω)

$1 < \log n < \sqrt{n} < n < n \log n < n^2 < n^3 < \ldots\ldots\ldots\ldots\ldots < 2^n < 3^n < \ldots < n^n$

- Let $t(n) = 2n + 3$ lower bound

  $2n + 3 \geq$ ______??

  $2n + 3 \geq 1n \quad n \geq 1$

  here $c = 5$ and $g(n) = n$

  $t(n) = \Omega(n)$

  $2n + 3 \geq 1\log n \quad n \geq 1$

  here $c = 5$ and $g(n) = \log n$

  $t(n) = \Omega(\log n)$

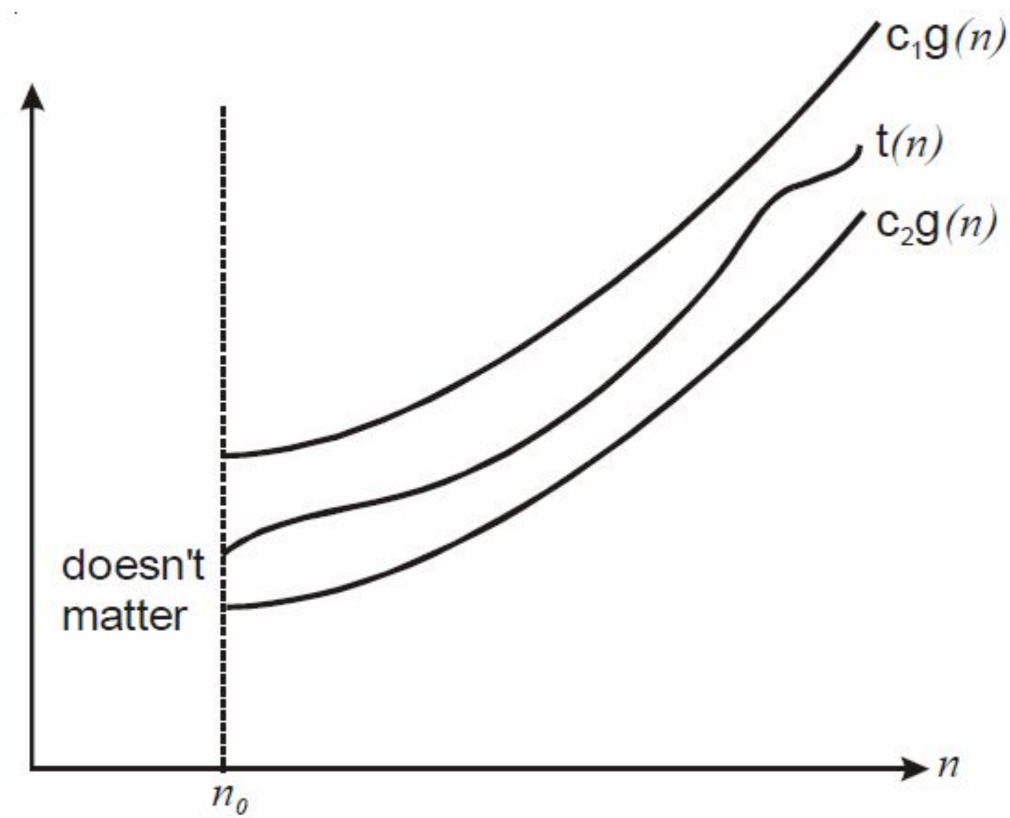
Theta notation (θ)

- ***Definition:*** A function $t(n)$ is said to be in θ ($g(n)$), denoted $t(n) \epsilon \theta(g(n))$, if $t(n)$ is bounded both above and below by some positive constant multiples of $g(n)$ for all large n. i.e., if there exist some positive constant $c1$ and $c2$ and some non-negative integer $n0$ such that

$$c2g(n) \leq t(n) \leq c1g(n) \text{ for all } n > n0$$

$$\theta(g(n)) = O(g(n)) \cap \Omega(g(n))$$

Theta notation (θ)



Theta notation (θ)

$1 < \log n < \sqrt{n} < n < n \log n < n^2 < n^3 < \ldots\ldots\ldots\ldots\ldots < 2^n < 3^n < \ldots. < n^n$

- Let $t(n) = 2n + 3$ average bound

  $\_\_\_\_\_ \leq 2n + 3 \leq \_\_\_\_\_\_$??

  $1n \leq 2n + 3 \leq 5n \quad n \geq 1$

  here $c1 = 5$, $c2 = 1$ and $g(n) = n$

  $t(n) = \theta(n)$

Little-oh notation (o)

- This notation is used to describe the worst case analysis of algorithms and concerned with small values of n.
- ***Definition :*** A function $t(n)$ is said to be in o($g(n)$), denoted $t(n)$ ϵ o($g(n)$), if there exist some positive constant c and some non-negative integer such that

$$t(n) \leq cg(n)$$

$$\lim_{n \to \infty} \frac{t(n)}{g(n)} = 0$$

Little-omega notation (ω)

- This notation is used to describe the best case analysis of algorithms and concerned with small values of *n*.
- The function $t(n) = \omega(g(n))$ iff

$$\lim_{n\to\infty}\frac{f(n)}{g(n)}=\infty \quad \text{(or)} \quad \lim_{n\to\infty}\frac{g(n)}{f(n)}=0$$

Properties of O, Ω and θ

General property:

If t(n) is O(g(n)) then a * t(n) is O(g(n)). Similar for Ω and θ

Transitive Property :

If $f\ (n)$ ϵ O($g(n)$) and $g(n)$ ϵ O($h(n)$), then $f\ (n)$ ϵ O($h(n)$); that is O is transitive. Also Ω, θ, o and ω are transitive.

Reflexive Property

If f(n) is given then f(n) is O(f(n))

Symmetric Property

If f(n) is θ(g(n)) then g(n) is θ(f(n))

Transpose Property

If f(n) = O(g(n)) then g(n) is Ω(f(n))

Asymptotic Notation

Notation	What it means	In terms of limit	Representation	Mathematically equivalent to
Big oh (O)	Growth of t(n) is ≤ the growth of g(n)	$\lim_{n\to\infty} \frac{t(n)}{g(n)} = c. c \geq 0$	t(n) = O(g(n))	≤
Big omega (Ω)	Growth of t(n) is ≥ the growth of g(n)	$\lim_{n\to\infty} \frac{t(n)}{g(n)} \neq 0$	t(n) = Ω(g(n))	≥
Theta notation (θ)	Growth of t(n) is ≈ the growth of g(n)	$\lim_{n\to\infty} \frac{t(n)}{g(n)} = c, c > 0$	t(n) = θ(g(n))	≈
Little oh (o)	Growth of t(n) is << the growth of g(n)	$\lim_{n\to\infty} \frac{t(n)}{g(n)} = 0$	t(n) = o(g(n))	<
Little omega (ω)	Growth of t(n) is >> the growth of g(n)	$\lim_{n\to\infty} \frac{t(n)}{g(n)} = \infty$	t(n) = ω(g(n))	>

Activity

- Find the upper bound, lower bound and tight bound range for the following functions
  - $2n + 5$
  - $3n + 2$
  - $3n + 3$
  - $n^2 \log n$
  - $10 n^2 + 4 n + 2$
  - $20 n^2 + 80 n + 10$
  - $n!$
  - $\log n!$

Order of growth of functions

10^{10} operations / sec typical CPU speeds

n	$\log n$	n	$n\log n$	n^2	n^3	2^n	$n!$
10	3	10	30	100	10^3	10^3	10^6
100	7	100	700	10^4	10^6	10^{30}	10^{157}
1000	10	1000	10^4	10^6	10^9		
10^4	13	10^4	10^5	10^8	10^{12}		
10^6				10^{12}			
10^{10}	33		10^{11}				

Summary

- Asymptotic analysis estimate an algorithmic complexity
- Based on theory of approximation.
- Effective in specifying the behaviour of algorithm when the input size increases
- Big – Oh notation – upper bound
- Big – Omega notation – lower bound
- Little – oh notation – tight bound