2. Asymptotic Notations

Dr. Swati Agarwal

Agenda

Classes of Functions

- $f(n) = 2$ constant
- $f(n) = 200$ constant
- $f(n) = 2000$ constant
- $f(n) = \log n$ logarithmic
- $f(n) = n$ linear
- $f(n) = n^2$ quadratic
- $f(n) = n^3$ cubic
- $f(n) = 2^n$ exponential

Compare classes of functions

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

$Input\ Size(n)$	$\log n$	n	$nlogn$	n^2	n^3	2^n
5	3	5	15	25	125	32
10	4	10	40	100	10^3	10^3
100	7	100	700	10^4	10^6	10^{30}
1000	10	10^3	10^4	10^6	10^9	10^{300}

Asymptotic Notations

- Mathematical notations (languages) to represent the time analysis function of an algorithm.
- Used to define the growth rate of an algorithm as the input size is increased.
- Performance of an algorithm in-terms of the input size.
- Three standard asymptotic notations:
  - Big-Oh $\mathcal{O} \rightarrow$ upper bound
  - Big-Omega $\Omega \rightarrow$ lower bound
  - Big-Theta $\Theta \rightarrow$ Average bound

Big-Oh $\mathcal{O}$

Definition

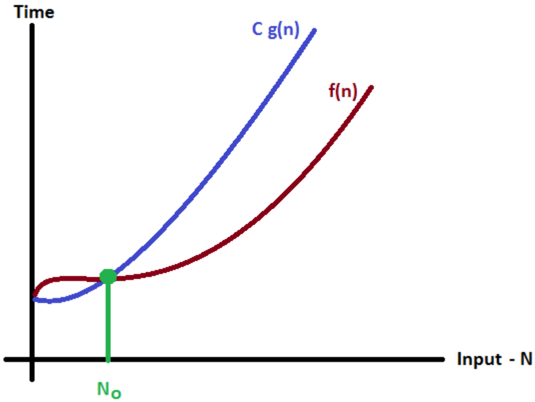
The function $f(n) = \mathcal{O}(g(n)) \iff \exists$ +ve constant c and n_0 such that $f(n) \leq c \times g(n) \quad \forall n \geq n_0$ where $n_0 \geq 1$ and $c > 0$

Example

$f(n) = 3n + 2$

select a value of c and n_0 such that $f(n)$ is always lesser than or equal to the $g(n)$.

Graphical Representation Big-Oh $\mathcal{O}$



Big-Omega Ω

Definition

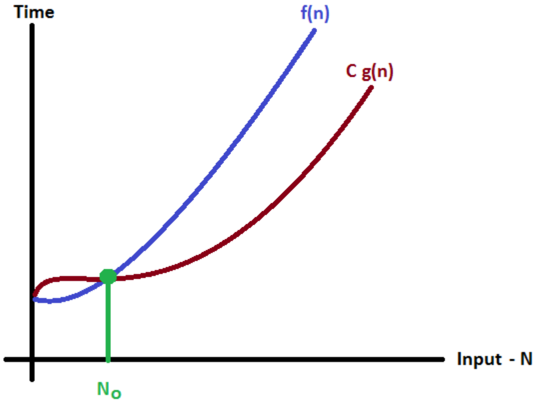
The function $f(n) = \Omega(g(n)) \iff \exists$ +ve constant c and n_0 such that $f(n) \geq c \times g(n) \quad \forall n \geq n_0$ where $n_0 \geq 1$ and $c>0$

Example

$f(n) = 3n + 2$

select a value of c and n_0 such that $f(n)$ is always greater than or equal to the $g(n)$.

Graphical Representation Big-Omega Ω



Big-Theta Θ

Definition

The function $f(n) = \Theta(g(n)) \iff \exists$ +ve constants c_1, c_2 and n_0 such that $c_1 \times g(n) \leq f(n) \leq c_2 \times g(n) \quad \forall n \geq n_0$ where $n_0 \geq 1$ and $c_1 > 0$ and $c_2 > 0$

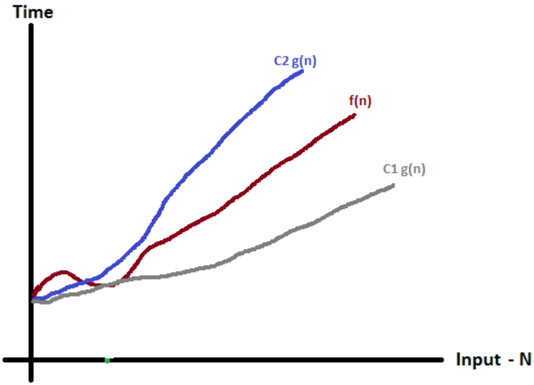
Primarily used when both upper bound and lower bound functions are asymptotically same.

Example

For $f(n) = 2n + 3$
$g(n) = 1 \times n$ for lower bound and $g(n) = 5 \times n$ for upper bound for $\forall n \geq 1$.
$1 \times n \geq 2n + 3 \geq 10 \times n$

$f(n) = \Theta(g(n))$

Graphical Representation Big-Theta Θ



Recall the classes of functions

- While computing the time function of any algorithm, always select the highest degree of polynomial
  - constant 1
  - linear n
  - quadratic n^2
  - cubic n^3
  - exponential 2^n or n^n
- While computing the bound, always select the closest value.
  - for big-oh notation, $f(n) = 2n + 3$, $g(n) = 2n^2\ \forall\ n \geq 2$, $g(n) = 2n^2 + 3n^2$ and $g(n) = 10n\ \forall\ n \geq 1$
  - best suitable $g(n)$ will be n instead of n^2.

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

lower bound

upper bound

Ω(n) f(n) O(n)

Exercise

1. $f(n) = 2n^2 + 3n + 2$ $\Theta(n^2)$
2. $f(n) = n^2 \log n + n$ $\Theta(n^2 \log n)$
3. $f(n) = n!$ $\mathcal{O}(n^n), \Omega(1)$
4. $f(n) = \log n!$ $\mathcal{O}(n \log n), \Omega(1)$

No tight/average bound for factorial functions