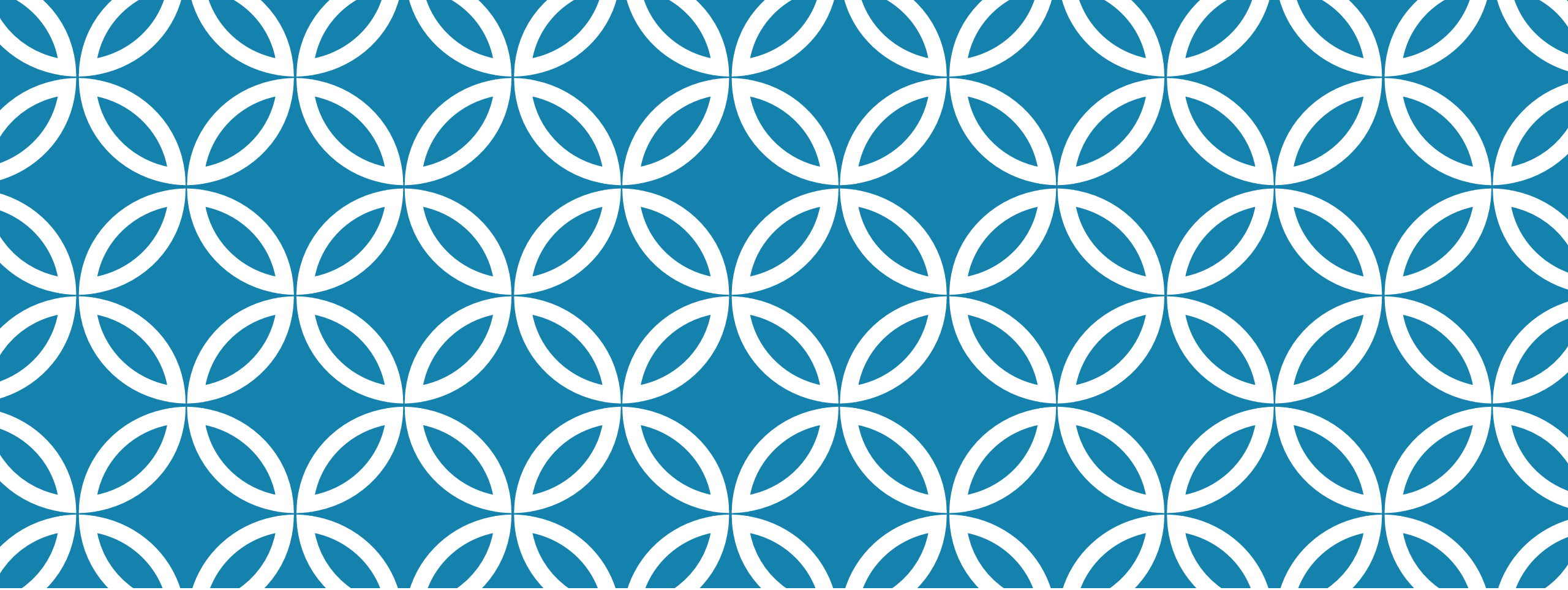


(CS2009)-DESIGN AND ANALYSIS OF ALGORITHMS

Presented By: Sandesh Kumar
Email: sandesh.kumar@nu.edu.pk

LECTURE 4 – 6 GROWTH OF FUNCTIONS & ASYMPTOTIC NOTATIONS

GROWTH OF FUNCTION

- Growth functions are used to estimate the number of steps an algorithm uses as its input grows. The largest number of steps needed to solve the given problem using an algorithm on input of specified size is worst-case complexity.

- **Asymptotic Notation:** The word **Asymptotic** means approaching a value or curve arbitrarily closely. i.e. some sort of limit is taken.

ASYMPTOTIC GROWTH RATE

- Algorithm complexity is usually very complex. The growth of the complexity functions is what is more important for the analysis and is a suitable measure for the comparison of algorithms with increasing input size n.
- Asymptotic notations like big-O, big-Omega, and big-Theta are used to compute the complexity because different implementations of algorithm may differ in efficiency.

ASYMPTOTIC GROWTH RATE

- Two reasons why we are interested in asymptotic growth rates.
- **Practical purposes:** For large problems, when we expect to have big computational requirements.
- **Theoretical purposes:** concentrating on growth rates **frees us** from some important issues.
  - **Fixed Costs:** (e.g. switching the computer on!), which may dominate for a small problem size but be largely irrelevant
  - **Machine and implementation details**
  - The growth rate will be a compact and **easy to understand the function.**

PROPERTIES OF GROWTH-RATE FUNCTION

- Example: 5n + 3
- Estimated running time for different values of n:
  - n=10 => 53 steps
  - n=100 => 503 steps
  - n=1000 => 5003 steps
  - n=1,000,000 => 5,000,003 steps
- As "n" grows the number of steps grow in linear proportion to n for this function "Sum"
- What about the "+3" and "5" in 5n+3?

  As n gets large, the +3 becomes insignificant 5 is inaccurate, as different operations require varying amounts of time and also **does not have any significant importance**

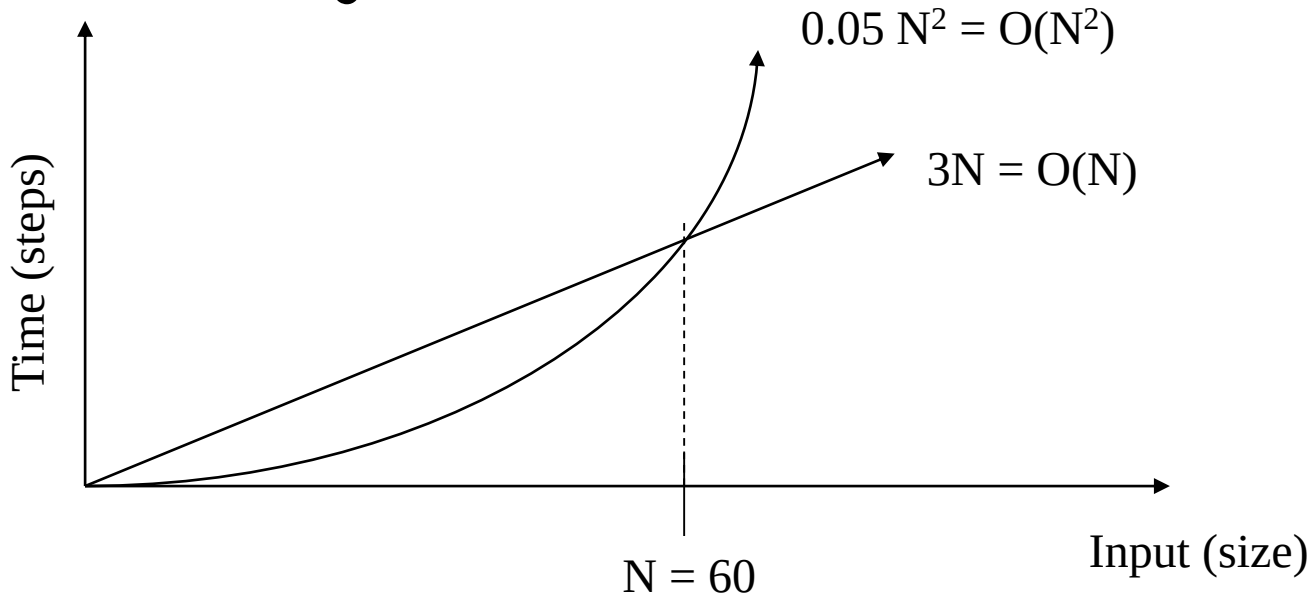
What is fundamental is that the time is *linear* in **n**

ASYMPTOTIC ALGORITHM ANALYSIS

- The asymptotic analysis of an algorithm determines the running time in big-Oh notation.
- To perform the asymptotic analysis
  - We find the worst-case number of primitive operations executed as a function of the input size n
  - We express this function with big-Oh notation
- Example: An algorithm executes T(n) = 2n2 + n elementary operations. We say that the algorithm runs in O(n2) time
- Growth rate is not affected by constant factors or lower-order terms so these terms can be dropped
- The $2n^2 + n$ time bound is said to **"grow asymptotically"** like n^2
- This gives us an approximation of the complexity of the algorithm
- Ignores lots of (machine dependent) details

ALGORITHM EFFICIENCY

- Measuring efficiency of an algorithm
  - do its analysis i.e. growth rate.
  - Compare efficiencies of different algorithms for the same problem.
- As inputs get larger, any algorithm of a smaller order will be more efficient than an algorithm of a larger order



IMPORTANT FUNCTIONS

Function	Name
c	Constant
$\log N$	Logarithmic
$\log^2 N$	Log-squared
N	Linear
$N \log N$	N log N
N^2	Quadratic
N^3	Cubic
2^N	Exponential
$N!$	
N^N	

Functions in order of increasing growth rate

A COMPARISON OF GROWTH RATE FUNCTIONS

	n					
Function	10	100	1,000	10,000	100,000	1,000,000
1	1	1	1	1	1	1
$\log_2 n$	3	6	9	13	16	19
n	10	10^2	10^3	10^4	10^5	10^6
$n * \log_2 n$	30	664	9,965	10^5	10^6	10^7
n^2	10^2	10^4	10^6	10^8	10^{10}	10^{12}
n^3	10^3	10^6	10^9	10^{12}	10^{15}	10^{18}
2^n	10^3	10^{30}	10^{301}	$10^{3,010}$	$10^{30,103}$	$10^{301,030}$

ASYMPTOTIC NOTATIONS

- Asymptotic Notations Θ, O, Ω, o, ω
- We use Θ to mean “order exactly”, (Tight Bound)
- O to mean “order at most”, (Tight Upper Bound)
- Ω to mean “order at least”, (Tight Lower Bound)
- o to mean “upper bound”,
- ω to mean “lower bound”,

BIG-OH NOTATION (O)

- If f, g: N $\rightarrow$ R^+, then we can define Big-Oh as

For a given function $g(n) \geq 0$, denoted by $O(g(n))$ the set of functions,

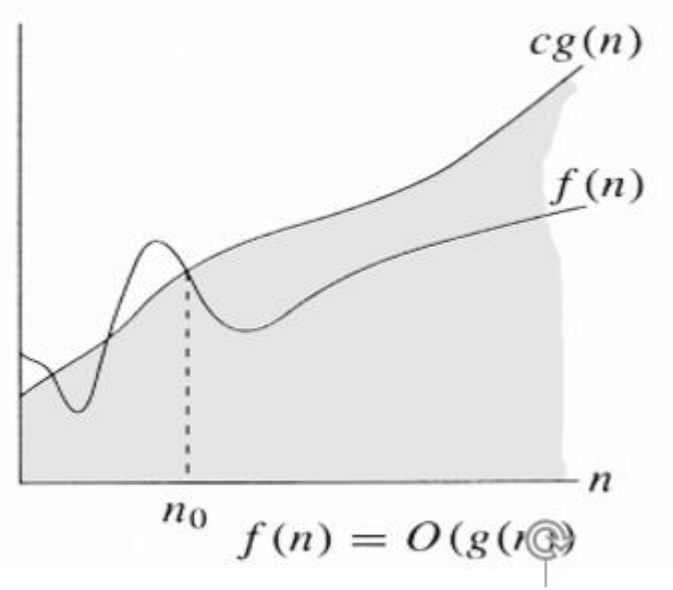
$O(g(n)) = \{ f(n)$: there exist positive constants c and n_o such that

$0 \leq f(n) \leq cg(n)$, for all $n \geq n_o \}$

$f(n) = O(g(n))$ means function $g(n)$ is an asymptotically upper bound for $f(n)$.

- *We may write f(n) = O(g(n)) OR f(n) $\in$ O(g(n))*
- Intuitively: Set of all functions whose *rate of growth* is the same as or lower than that of *g(n)*. *f(n)* is bounded above by *g(n)* for all sufficiently large n

BIG-OH NOTATION (O)



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

$\exists\ c > 0, \exists\ n_0 > 0$ and $\forall n \geq n_0, 0 \leq f(n) \leq c.g(n)$

BIG-OH NOTATION (O)

- The idea behind the big-O notation is to establish an **upper boundary** for the growth of a function f(n) for large n.
- This boundary is specified by a function g(n) that is usually much **simpler** than f(n).
- We accept the constant C in the requirement
  - $f(n) \leq C \cdot g(n)$ whenever $n > n_0$,
- We are only interested in large n, so it is OK if
  - $f(n) > C \cdot g(n)$ for $n \leq n_0$.
- The relationship between f and g can be expressed by stating either that g(n) is an upper bound on the value of f(n) or that in the long run , f grows at most as fast as g.

EXAMPLE

- As a simple illustrative example, we show that the function $2n^2 + 5n + 6$ is $O(n^2)$.
- For all $n \geq 1$, it is the case that
  - $2n^2 + 5n + 6 \leq 2n^2 + 5n^2 + 6n^2 = 13n^2$
- Hence, we can take $c = 13$ and $no = 1$, and the definition is satisfied.

EXAMPLE

- Prove that $2n^2 = O(n^3)$

Proof:

Assume that $f(n) = 2n^2$, and $g(n) = n^3$

$f(n) = O(g(n))$?

- Now we have to find the existence of c and n_0

$f(n) \leq c.g(n) \rightarrow 2n^2 \leq c.n^3 \rightarrow 2 \leq c.n$

if we take, $c = 1$ and $n_0 = 2$ OR

$c = 2$ and $n_0 = 1$ then

$2n^2 \leq c.n^3$

- Hence $f(n) = O(g(n))$, $c = 1$ and $n_0 = 2$

EXAMPLE

- Prove that $n^2 = O(n^2)$

Proof:

Assume that f(n) = n^2, and g(n) = n^2

f(n) = O(g(n)) ?

- Now we have to find the existence of c and n0

f(n) ≤ c.g(n) ➔ $n^2 \leq c.n^2$ ➔ 1 ≤ c

if we take, c = 1, n0= 1

Then

$n^2 \leq c.\ n^2$ for c = 1 and n ≥ 1

- Hence, $n^2 = O(n^2)$, where c = 1 and n0= 1

EXAMPLE

- Prove that $1000.n2 + 1000.n = O(n2)$

Proof:

Assume that $f(n) = 1000.n2 + 1000.n$, and $g(n) = n2$

We have to find existence of c and n0 such that

$0 \leq f(n) \leq c.g(n) \qquad \forall\ n \geq n0$

$1000.n2 + 1000.n \leq c.n2$

for $c = 1001$, $\quad 1000.n2 + 1000.n \leq 1001.n2$

➔ $1000.n \leq n2$ ➔ $\quad n2 - 1000.n \geq 0$ ➔ $n\ (n-1000) \geq 0$,

this true for $n \geq 1000$

- Hence $f(n) = O(g(n))$ for $c = 1001$ and $n0 = 1000$

EXAMPLE

- Prove that $n^3 = O(n^2)$

  Proof:

  On contrary we assume that there exist some positive constants c and n0 such that

  $0 \leq n^3 \leq c.\ n^2 \quad \forall\ n \geq n0$

  $n \leq c$

- Since c is any fixed number and n is any arbitrary constant, therefore $n \leq c$ is not possible in general.
- Hence our supposition is wrong and $n^2 \leq c.\ n^2$, for $n \geq n0$ is not true for any combination of c and n0.
- Hence, $n^3 = O(n^2)$ does not hold

EXAMPLE

- Prove that $2n + 10 = O(n)$

  Proof:

  Assume that $f(n) = 2n + 10$, and $g(n) = n$

  $f(n) = O(g(n))$?

- Now we have to find the existence of c and n0

  $f(n) \leq c.g(n) \rightarrow 2n + 10 \leq c.n \rightarrow (c - 2)\, n \geq 10 \rightarrow n \geq 10/(c - 2)$

  $c > 2$ for $n > 0$, we pick, $c = 3$, then $n0 = 10$

Then

$2n + 10 \leq c.n$ for $c = 3$ and $n \geq 10$

Hence, $2n + 10 = O(n)$, where $c = 3$ and $n0 = 10$

EXAMPLE

- Prove that : $3n^3 + 20n2 + 5 = O(n^3)$

  Proof:

  Need $c > 0$ and $n0 \geq 1$ such that

  $3n^3 + 20n2 + 5 \leq c.\ n^3$ for $n \geq n0$

  This is true for $c = 4$, $n0 = 21$

OR $c = 28$, $n0 = 1$

- Hence, $3n^3 + 20n2 + 5 = O(n^3)$, where $c = 4$ and $n0 = 21$

EXAMPLE

- Prove that : $10n + 500 = O(n)$

Proof: Function n will never be larger than the function $500 + 10$ n, no matter how large n gets.

However, there are constants c0 and n0 such that

$500 + 10n \leq c.n$ when $n \geq n0$.

One choice for these constants is $c = 20$ and $n0 = 50$.
Other choices for c0 and n0:

For example, any value of $c > 20$ will work for $n0 = 50$.

Therefore, $500 + 10n = O(n)$.

EXAMPLE

- Prove which of the following function is larger by order of growth? $(1/3)n$ or 17?

Let's check if

$(1/3)n = O(17)$

$(1/3)n \leq c.17$, which is true for $c=1, n0 = 1$

Let's check if

$17 = O((1/3)n)$

$17 \leq c.(1/3)n$, which is true for $c > 17.3n$

And hence can't be bounded for large n. That's why $(1/3)n$ is less in growth rate then 17.

EXAMPLE

- Prove that : $8n^2 + 2n - 3 \in O(n^2)$

Proof:

Need $c > 0$ and $n0 \geq 1$ such that

$8n^2 + 2n - 3 \leq c.\ n^2$ for $n \geq n0$

- Consider the reasoning:

$f(n) = 8n^2 + 2n - 3 \leq 8n^2 + 2n \leq 8n^2 + 2n^2 = 10n^2$

- Hence, $8n^2 + 2n - 3 \in O(n^2)$, where $c = 10$ and $n0 = 1$

BIG – OMEGA NOTATION (Ω)

- If f, g: N $\rightarrow$ R^+, then we can define Big-Omega as

For a given function $g(n)$ denote by $\Omega(g(n))$ the set of functions,

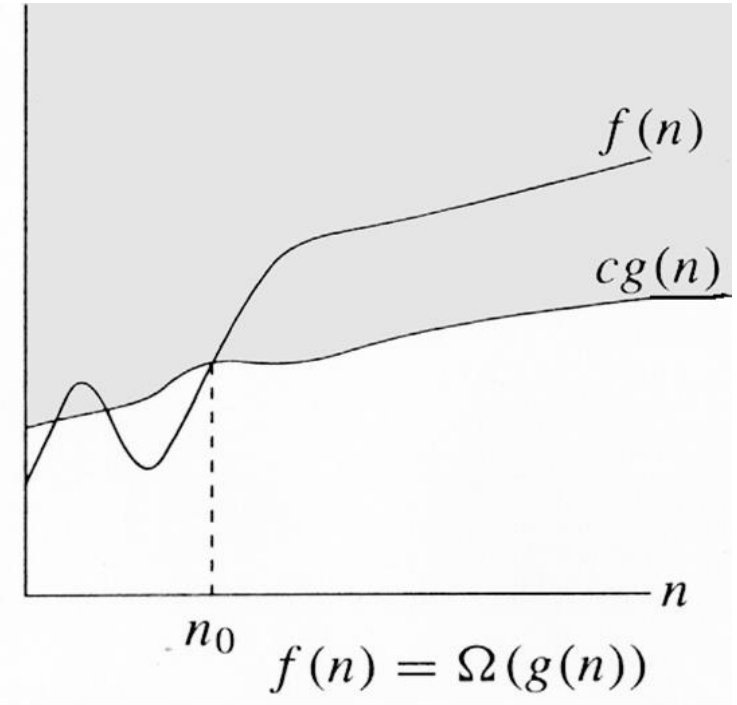
$\Omega(g(n)) = \{ f(n)$: there exist positive constants c and n_o such that

$0 \leq cg(n) \leq f(n)$ for all $n \geq n_o \}$

$f(n) = \Omega(g(n))$, means that function $g(n)$ is an asymptotically lower bound for $f(n)$.

- We may write f(n) = Ω(g(n)) OR f(n) $\in$ Ω(g(n))
- **Intuitively:** Set of all functions whose rate of growth is the same as or higher than that of g(n).

BIG – OMEGA NOTATION (Ω)



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

g(n) is an asymptotically lower bound for f(n).

$\exists\ c > 0, \exists\ n0 > 0\ , \forall n \geq n0,\ f(n) \geq c.g(n)$

Note the duality rule: $t(n) \in \Omega(f(n)) \equiv f(n) \in O(t(n))$

EXAMPLE

- Prove that $3n + 2 \in \Omega(n)$

  Proof:

  Assume that $f(n) = 3n + 2$, and $g(n) = n$

  $f(n) \in \Omega(g(n))$?

- We have to find the existence of c and n_0 such that

  $c.g(n) \leq f(n)$ for all $n \geq n_0$

  $c.n \leq 3n + 2$

At R.H.S a positive term is being added to 3n, which will make L.H.S ≤ R.H.S for all values of n, when $c = 3$.

- Hence $f(n) \in \Omega(g(n))$, for $c = 3$ and $n_0 = 1$

EXAMPLE

- Prove that $5.n^2 \in \Omega(n)$

  Proof:

  Assume that $f(n) = 5.n^2$, and $g(n) = n$

  $f(n) \in \Omega(g(n))$?

- We have to find the existence of c and n_0 such that

  $c.g(n) \leq f(n)$ for all $n \geq n_0$

  $c.n \leq 5.n^2 \Leftarrow c \leq 5.n$

if we take, $c = 5$ and $n_0 = 1$ then

$c.n \leq 5.n^2$ for all $n \geq n_0$

- Hence $f(n) \in \Omega(g(n))$, for $c = 5$ and $n_0 = 1$

EXAMPLE

- Prove that $5n^2 + 2n - 3 \in \Omega(n^2)$

  Proof:

  Assume that $f(n) = 5n^2 + 2n - 3$, and $g(n) = n^2$

  $f(n) \in \Omega(g(n))$?

- We have to find the existence of c and n_0 such that

  $c.g(n) \leq f(n) \quad \forall\ n \geq n_0$

  $c.n^2 \leq 5.n^2 + 2n - 3$

We can take $c = 5$, given that 2n-3 is always positive.
2n-3 is always positive for $n \geq 2$. Therefore $n_0 = 2$.

- Hence $f(n) \in \Omega(g(n))$, for $c = 5$ and $n_0 = 2$

EXAMPLE

- Prove that $100.n + 5 \notin \Omega(n2)$

  Proof:

  Let $f(n) = 100.n + 5$, and $g(n) = n2$

  Assume that $f(n) \in \Omega(g(n))$?

  Now if $f(n) \in \Omega(g(n))$ then there exist c and n0 such that

  $c.g(n) \leq f(n)$ for all $n \geq n0$

  $c.n2 \leq 100.n + 5$

For the above inequality to hold meaning f(n) grows faster than g(n).

But which means g(n) is growing faster than f(n) and hence $f(n) \nearrow \Omega(g(n))$

THETA NOTATION (Θ)

- If f, g: N $\rightarrow$ R^+, then we can define Big-Theta as

For a given function $g(n)$ denoted by $\Theta(g(n))$ the set of functions,

$\Theta(g(n)) = \{f(n)$: there exist positive constants c_1, c_2 and n_o such that

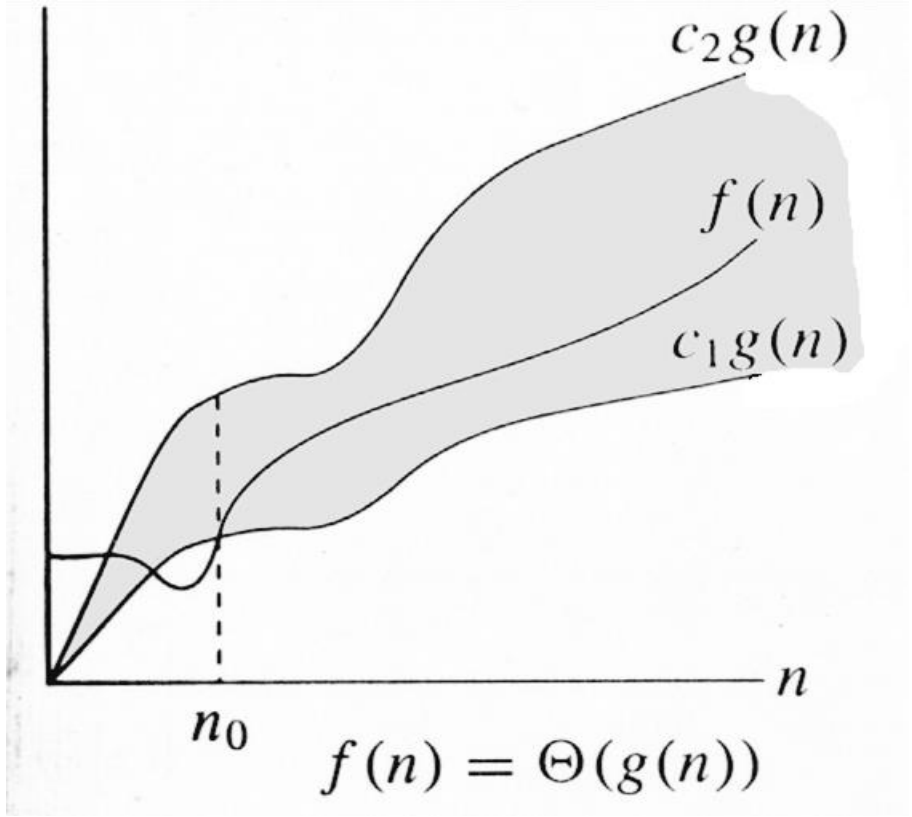
$0 \le c_1 g(n) \le f(n) \le c_2 g(n)$ for all $n \ge n_o\}$

$f(n) = \Theta(g(n))$ means function $f(n)$ is equal to $g(n)$ to within a constant factor, and $g(n)$ is an asymptotically tight bound for $f(n)$.

- We may write f(n) = Θ(g(n)) OR f(n) ∈ Θ(g(n))
- ***Intuitively***: Set of all functions that have same *rate of growth* as *g*(*n*).

When a problem is Θ(*n*), this represents both an upper and lower bound i.e. it is O(*n*) **and** Ω(*n*) (no algorithmic gap)

THETA NOTATION (Θ)



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

$\exists$ c1> 0, c2> 0, $\exists$ n0 > 0, $\forall$ n $\geq$ n0, c2.g(n) $\leq$ f(n) $\leq$ c1.g(n)

We say that g(n) is an asymptotically tight bound for f(n).

EXAMPLE

- Prove that $½.n^2 – ½.n = \Theta(n^2)$

  Proof

  Assume that $f(n) = ½.\ n^2 – ½.n$, and $g(n) = n2$

  $f(n) \in \Theta(g(n))$?

  We have to find the existence of c1, c2 and n0 such that

  $c1.g(n) \leq f(n) \leq c2.g(n)$ for all $n \geq n0$

Since, $½\ n^2$ - ½ n ≤ $½\ n^2$ $\forall n \geq 0$ if c2= ½ and

$½\ n^2$ - ½ n ≥ $½\ n^2$ - ½ n . ½ n ($\forall n \geq 2$) = $¼\ n^2$, c1= ¼

Hence $½\ n^2$ - ½ n ≤ $½\ n^2$ ≤ $½\ n^2$ - ½ n

c1.g(n) ≤ f(n) ≤ c2.g(n) $\forall n \geq 2$, c1= ¼, c2 = ½

Hence $f(n) \in \Theta(g(n)) \Rightarrow ½.n2 – ½.n = \Theta(n2)$

EXAMPLE

- Prove that $2.n^2 + 3.n + 6 \nearrow \Theta(n^3)$

  Proof: Let f(n) = $2.\ n^2 + 3.n + 6$, and g(n) = n^3

  we have to show that f(n) $\nearrow \Theta$(g(n))

  On contrary assume that f(n) $\in \Theta$(g(n)) i.e. there exist some positive constants c1, c2 and n0 such that: c1.g(n) ≤ f(n) ≤ c2.g(n)

  Solve for c2: f(n) ≤ c2.g(n) ➲ $2n^2 + 3n + 6 \leq 2n^2 + 3n^2 + 6n^2 \leq$ c2. n^3 ➲ c = 11 and n0= 1

  Solve for c1: c1.g(n) ≤ f(n) ➲ c1$n^3 \leq 2n^2 + 3n + 6$ ➲ c1$n^3 \leq 2n^2 \leq 2n^2 + 3n + 6$

c1.n ≤ 2, for large n this is not possible

Hence f(n) $\nearrow \Theta$(g(n)) $\Rightarrow$ 2.n^2 + 3.n + 6 $\nearrow \Theta(n^3)$

EXAMPLE

- Prove that $3.n + 2 = \Theta(n)$

  Proof: Let $f(n) = 3.n + 2$, and $g(n) = n$

  Assume that $f(n) \in \Theta(g(n))$ i.e. there exist some positive constants c1, c2 and n0 such that:

$$c1.g(n) \leq f(n) \leq c2.g(n)$$

$$c1.n \leq 3.n + 2 \leq c2.\ n$$

Take $c1 = 3$, as $3n \leq 3n + 2$ with $n0 = 1$

$3n + 2 \leq 3n + 2n \leq c2.\ n$ ➔ $5n \leq c2.\ n$ ➔ $5 \leq c2$

$c1 = 3, c2 = 5, n0 = 1$

Hence $f(n) \nearrow \Theta(g(n)) \Rightarrow 3.n + 2 = \Theta(n)$

EXAMPLE

- Prove that $½.n^2 – 3.n = \Theta(n^2)$

Proof

Let f(n) = $½. n^2$ – 3.n, and g(n) = n^2 f(n) $\in \Theta$(g(n))?

We have to find the existence of c1, c2 and n0 such that

c1.g(n) ≤ f(n) ≤ c2.g(n) $\forall$ n ≥ n0

c1. $n^2 \leq ½. n^2 – 3.n \leq$ c2. n^2

Since, ½ n^2 - 3 n ≤ ½ n^2 $\forall$n ≥ 1 if c2= ½ and

½ n^2 - 3 n ≥ 1/4 n^2 ($\forall$n ≥ 6), c1= ¼

c1.g(n) ≤ f(n) ≤ c2.g(n) $\forall$n ≥ 6, c1= ¼, c2 = ½

Hence f(n) $\in \Theta$(g(n)) $\Rightarrow$ ½.n2 – 3.n = Θ(n2)

LITTLE-OH NOTATION

- o-notation is used to denote a upper bound that is not asymptotically tight.

For a given function $g(n) \geq 0$, denoted by $o(g(n))$ the set of functions,

$$o(g(n)) = \begin{Bmatrix} f(n): \text{for any positive constants } c, \text{ there exists a constant } n_o \\ \text{such that } 0 \leq f(n) < cg(n) \text{ for all } n \geq n_o \end{Bmatrix}$$

- f(n) becomes insignificant relative to g(n) as n approaches infinity. g(n) is an upper bound for f(n), not asymptotically tight

e.g., $2n = o(n^2)$ but $2n^2 \neq o(n^2)$. $\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$

EXAMPLE

- Prove that $2n2 \in o(n3)$

  Proof:

  Assume that $f(n) = 2n2$, and $g(n) = n3$

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

Now we have to find the existence n0 for any c $f(n) < c.g(n)$ this is true

$\Leftrightarrow 2n2 < c.n3 \Leftrightarrow 2 < c.n$

- This is true for any c, because for any arbitrary c we can choose n0 such that the above inequality holds.

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

EXAMPLE

- Prove that $2n^2 \in o(n^3)$

  Proof:

  Assume that f(n) $= 2n^2$, and g(n) $= n^3$

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

Now we have to find the existence n0 for any c f(n) < c.g(n) this is true

⬅ $2n^2 <$ c. n^3 ⬅ 2 < c.n

- This is true for any c, because for any arbitrary c we can choose n0 such that the above inequality holds.

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

EXAMPLE

- Prove that $n^2 \notin o(n^2)$

  Proof:

  Assume that f(n) = n^2 , and g(n) = n^2

  Now we have to show that f(n) $\notin$ o(g(n))

  Since f(n) < c.g(n) ⬅ $n^2 < c.\ n^2$ ⬅ 1 ≤ c,

- In our definition of small o, it was required to prove for any c but here there is a constraint over c .

Hence, $n^2 \notin o(n^2)$, where c = 1 and n0= 1

LITTLE-OMEGA NOTATION

- Little-ω notation is used to denote a lower bound that is not asymptotically tight.

For a given function $g(n)$, denote by $\omega(g(n))$ the set of all functions.

$\omega(g(n)) = \{f(n)$: for any positive constants c, there exists a constant n_o such that $0 \le cg(n) < f(n)$ for all $n \ge n_o\}$

- $f(n)$ becomes arbitrarily large relative to $g(n)$ as n approaches infinity

e.g., $\frac{n^2}{2} = \omega(n)$ but $\frac{n^2}{2} \neq \omega(n^2)$.

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

EXAMPLE

- Prove that $5.n^2 \in \omega(n)$

Proof:

Assume that $f(n) = 5.n^2$, and $g(n) = n$

$f(n) \in \Omega(g(n))$?

We have to prove that for any c there exists n0 such that

$c.g(n) < f(n) \quad \forall\ n \geq n0$

$c.n < 5.n^2 \Leftarrow c < 5.n$

This is true for any c, because for any arbitrary c e.g. $c = 1000000$, we can choose $n0 = 1000000/5 = 200000$ and the above inequality does hold.

And hence $f(n) \in \omega(g(n))$

EXAMPLE

- Prove that $5.n + 10 \notin \omega(n)$

  Proof:

  Assume that $f(n) = 5.n + 10$, and $g(n) = n$

  $f(n) \notin \Omega(g(n))$?

We have to find the existence n0 for any c, such that

$c.g(n) < f(n) \quad \forall\ n \geq n0$

$c.n < 5.n + 10$, if we take $c = 16$ then

$16.n < 5.n + 10 \Leftrightarrow 11.n < 10$ is not true for any positive integer.

Hence $f(n) \notin \omega(g(n))$

EXAMPLE

- Prove that 100.n $\notin \omega(n^2)$

  Proof:

  Let f(n) = 100.n, and g(n) = n^2

  Assume that f(n) $\in \omega$(g(n))

  Now if f(n) $\in \omega$(g(n)) then there n0 for any c such that

  c.g(n) < f(n) $\quad \forall$ n $\geq$ n0

  ⇔ c. n^2 < 100.n ⇔ c.n < 100

  If we take c = 100, n < 1, not possible

  Hence f(n) ↗ ω(g(n)) i.e. 100.n $\notin \omega$(n2)

ASYMPTOTIC FUNCTIONS SUMMARY

- If $f(n) = \Theta(g(n))$ we say that f(n) and g(n) grow at the same rate, asymptotically

- If $f(n) = O(g(n))$ and $f(n) \neq \Omega(g(n))$, then we say that f(n) is asymptotically slower growing than g(n).

- If $f(n) = \Omega(g(n))$ and $f(n) \neq O(g(n))$, then we say that f(n) is asymptotically faster growing than g(n).

USEFULNESS OF NOTATIONS

- It is not always possible to determine behaviour of an algorithm using Θ-notation.
- For example, given a problem with n inputs, we may have an algorithm to solve it in $a.n^2$ time when n is even and c.n time when n is odd. OR
- We may prove that an algorithm never uses more than $e.n^2$ time and never less than f.n time.
- In either case we can neither claim $\Theta(n)$ nor $\Theta(n^2)$ to be the order of the time usage of the algorithm.
- Big O and Ω notation will allow us to give at least partial information

USEFULNESS OF NOTATIONS

- To express the efficiency of our algorithms which of the three notations should we use?
- As computer scientist we generally like to express our algorithms as big O since we would like to know the upper bounds of our algorithms.

Why?

- If we know the worse case then we can aim to improve it and/or avoid it.

USEFULNESS OF NOTATIONS

Even though it is correct to say "7n - 3 is $O(n^3)$", a better statement is "7n - 3 is O(n)", that is, one should make the approximation as tight as possible

Simple Rule:

Drop lower order terms and constant factors

7n-3 is O(n)

8n2log n + 5n2 + n is O(n2log n)

Strictly speaking this use of the equals sign is incorrect

- the relationship is a set inclusion, not an equality
- $f(n) \in O(g(n))$ is better

BIG OH DOES NOT TELL THE WHOLE STORY

Question?

- If two algorithms A and B have the same asymptotic complexity, say $O(n^2)$, will the execution time of the two algorithms always be same?
- How to select between the two algorithms having the same asymptotic performance?

Answer:

- They may not be the same. There is this small matter of the constant of proportionality. Suppose that A does ten operations for each data item, but algorithm B only does three.
- It is reasonable to expect B to be faster than A even though both have the same asymptotic performance. The reason is that asymptotic analysis ignores constants of proportionality.

BIG OH DOES NOT TELL THE WHOLE STORY

```
Algorithm_A {
    set up the algorithm; /*taking 50 time units*/
    read in n elements into array A; /* 3 units per element */
    for (i = 0; i < n; i++) {
        do operation1 on A[i]; /* takes 10 units */
        do operation2 on A[i]; /* takes 5 units */
        do operation3 on A[i]; /* takes 15 units */
    }
}
```

TA(n) = 50 + 3n + (10 + 5 + 15)*n

= 50 + 33*n

BIG OH DOES NOT TELL THE WHOLE STORY

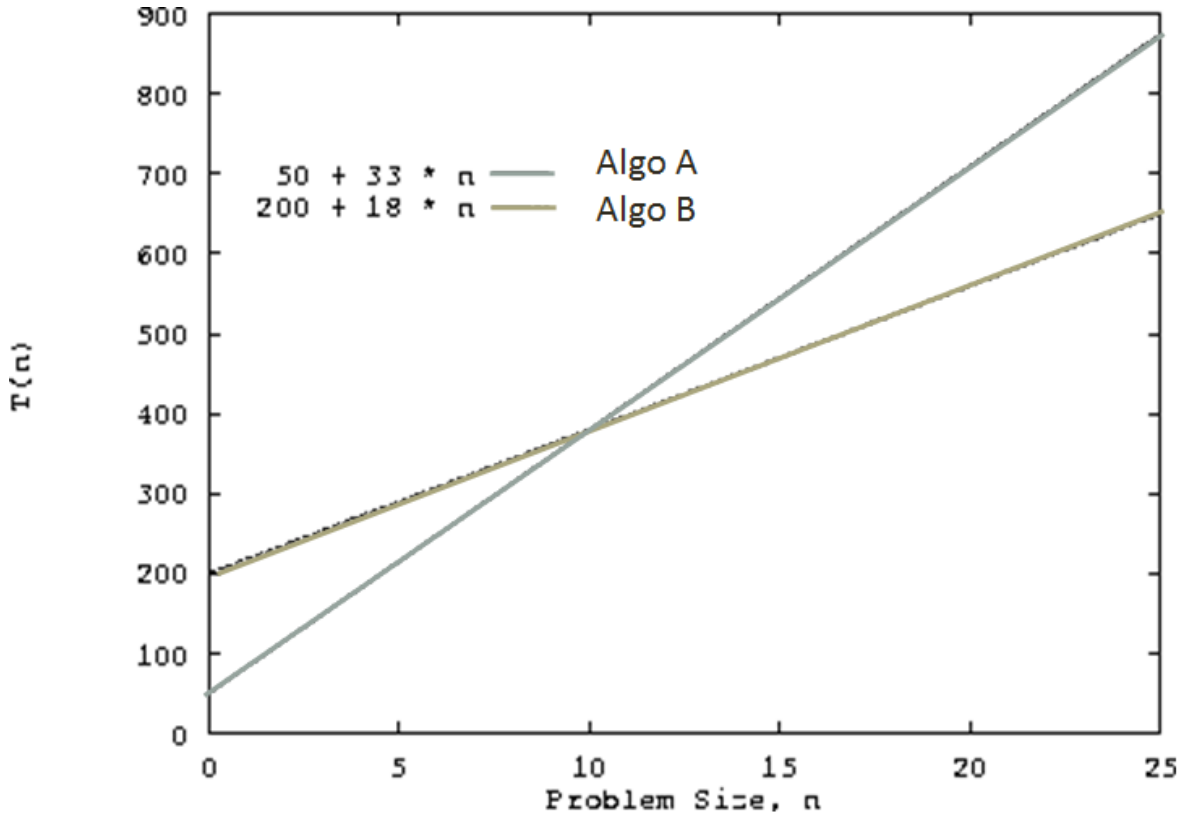
```
Algorithm_B {
        set up the algorithm; /*taking 200 time units*/
        read in n elements into array A; /* 3 units per element */
        for (i = 0; i < n; i++) {
           do operation1 on A[i]; /* takes 10 units */
           do operation2 on A[i]; /* takes 5 units */
         }
}
```

TB(n) = 200 + 3n + (10 + 5)*n

= 200 + 18*n

BIG OH DOES NOT TELL THE WHOLE STORY

- Both algorithms have time complexity O(n).
- Algorithm A sets up faster than B, but does more operations on the data Algorithm A is the better choice for small values of n. For values of n > 10, algorithm B is the better choice



A MISCONCEPTION

- A common misconception is that worst case running time is somehow defined by big-Oh, and that best case is defined by big-Omega.
- There is no formal relationship like this.
- However, worst case and big-Oh are commonly used together, because they are both techniques for finding an upper bound on running time.

RELATIONS OVER ASYMPTOTIC NOTATIONS

- Maximum rule: $O(f(n)+g(n)) = O(\max(f(n),g(n)))$
- Additive and Multiplicative Property:

  If $e(n) = O(g(n))$ & $f(n) = O(h(n))$ then $e(n) + f(n) = O(g(n) + h(n))$

  If $e(n) = O(g(n))$ & $f(n) = O(h(n))$ then $e(n) \cdot f(n) = O(g(n) \cdot h(n))$

- Dichotomy Property: If $f(n) = O(g(n))$ & $g(n) = O(f(n))$ then $f(n) = \Theta(g(n))$

  If $f(n) = \Omega(g(n))$ and $g(n) = \Omega(f(n))$ then $f(n) = \Theta(g(n))$

RELATIONS OVER ASYMPTOTIC NOTATIONS

- Reflexive Property: If $f(n) = O(f(n))$ and $f(n) = \Omega(f(n))$ and $f(n) = \Theta(f(n))$

$$f(n) \neq o(f(n)) \text{ and } f(n) \neq \omega(f(n))$$

- Symmetry over Θ: $f(n) = \Theta(g(n)) \Leftrightarrow g(n) = \Theta(f(n))$
- Transitivity Property:

$$f(n) = \Theta(g(n)) \;\&\; g(n) = \Theta(h(n)) \Rightarrow f(n) = \Theta(h(n))$$
$$f(n) = O(g(n)) \;\&\; g(n) = O(h(n)) \Rightarrow f(n) = O(h(n))$$
$$f(n) = \Omega(g(n)) \;\&\; g(n) = \Omega(h(n)) \Rightarrow f(n) = \Omega(h(n))$$
$$f(n) = o\,(g(n)) \;\&\; g(n) = o\,(h(n)) \Rightarrow f(n) = o\,(h(n))$$
$$f(n) = \omega(g(n)) \;\&\; g(n) = \omega(h(n)) \Rightarrow f(n) = \omega(h(n))$$