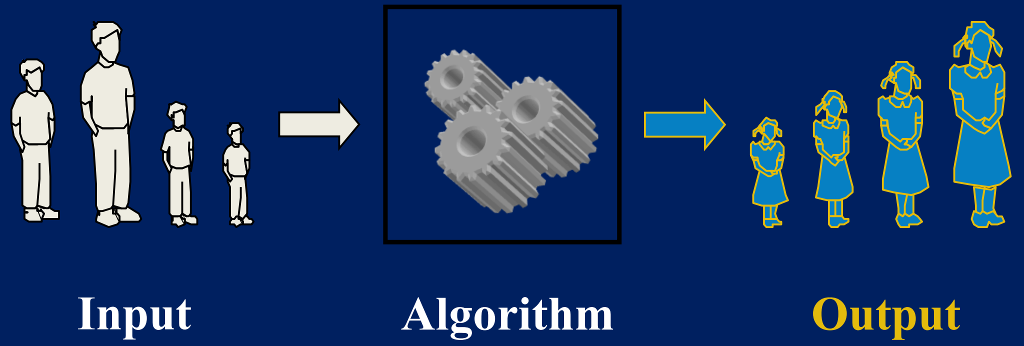
DESIGN & ANALYSIS OF ALGORITHM (BCSC0012)

Chapter 2: Asymptotic Notations



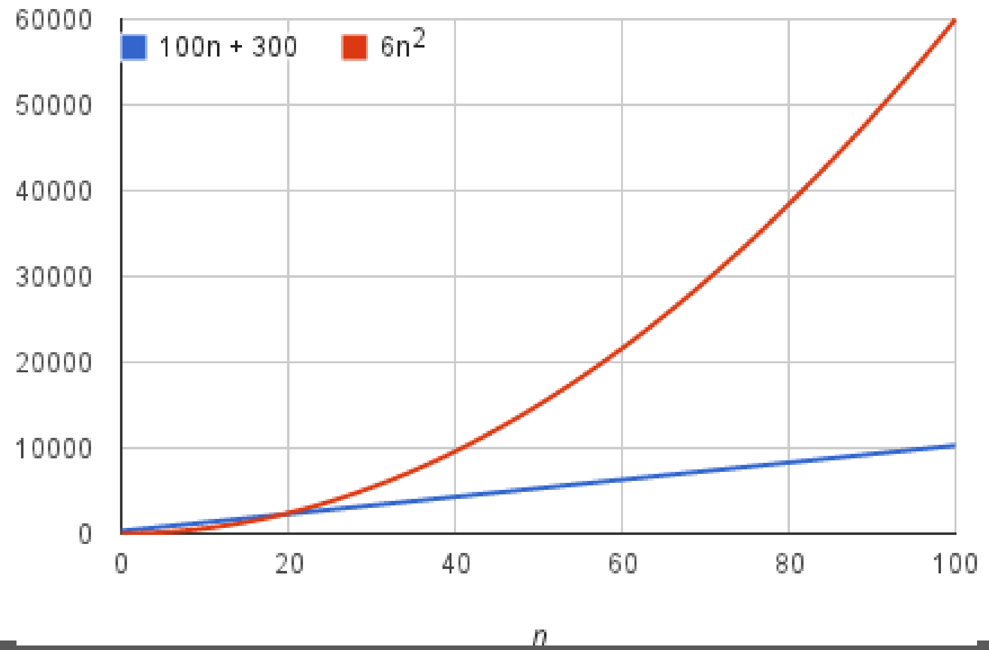
Prof. Anand Singh Jalal

Department of Computer Engineering & Applications

Rate of Growth

$6n^2 + 100n + 300$ machine instructions. The $6n^2$ term becomes larger than the remaining terms, $100n + 300$, once n becomes large enough, 20 in this case. Here's a chart showing values of $6n^2$ and $100n + 300$ for values of n from 0 to 100:

By dropping the less significant terms and the constant coefficients, we can focus on the important part of an algorithm's running time—its rate of growth

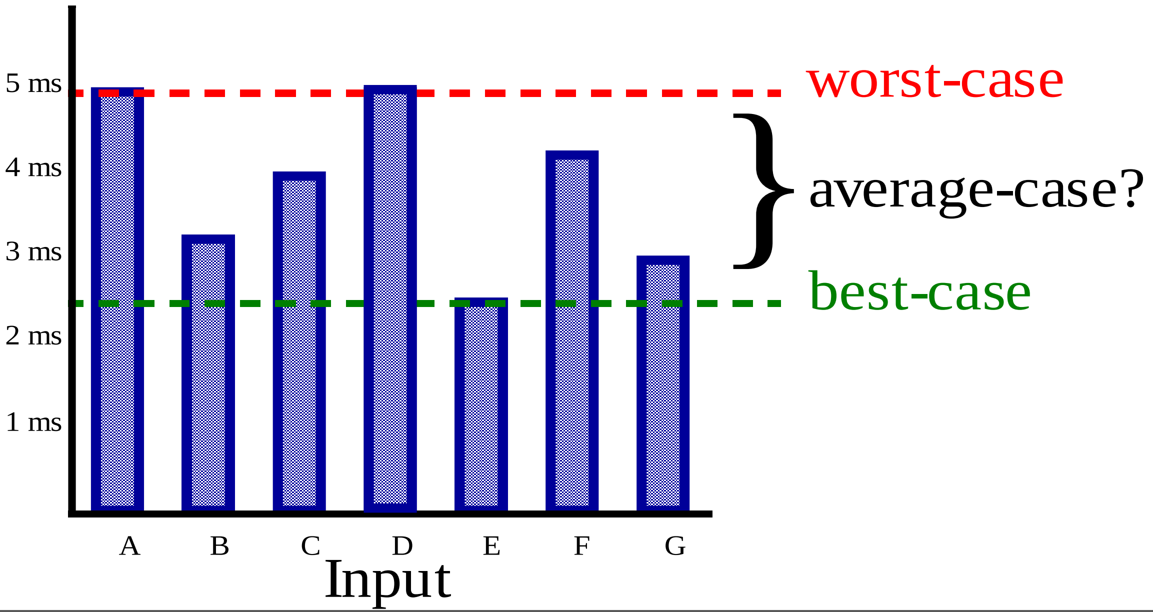


Algorithm: Introduction

An **algorithm** is a step-by-step procedure for solving a problem in a finite amount of time.

Average Case vs. Worst Case Running Time of an algorithm

- An algorithm may run faster on certain data sets than on others.
- Finding the average case can be very difficult, so typically algorithms are measured by the worst-case time complexity.
- Also, in certain application domains (e.g., air traffic control, surgery, IP lookup) knowing the worst-case time complexity is of crucial importance.



Asymptotic Notation

The main idea of asymptotic analysis is to have a **measure of efficiency of algorithms** that doesn't depend on machine specific constants, and doesn't require algorithms to be implemented and time taken by programs to be compared.

Asymptotic notations are the mathematical notations used to describe the running time (time complexity) of an algorithm when the input tends towards a particular value or a limiting value.

Asymptotic notations show the class of a function

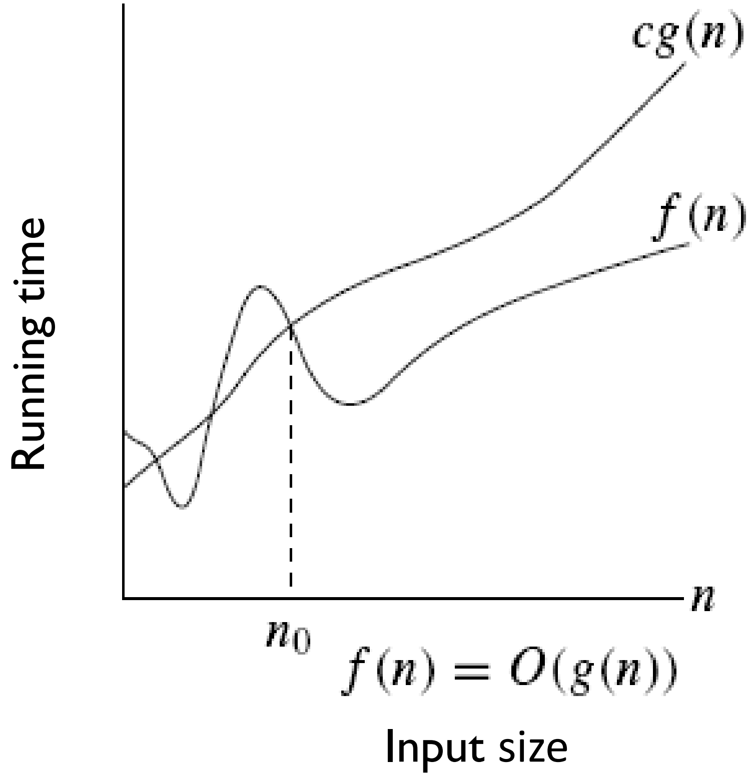
Asymptotic Notation …

Practical Significance

- **Big Omega (Ω) :**
  - Best case
  - Never achieve better than this
- **Big Theta (Θ) :**
  - Average case
- **Big -Oh (O):**
  - Worst case
  - Upper bound
  - Time must not exceed

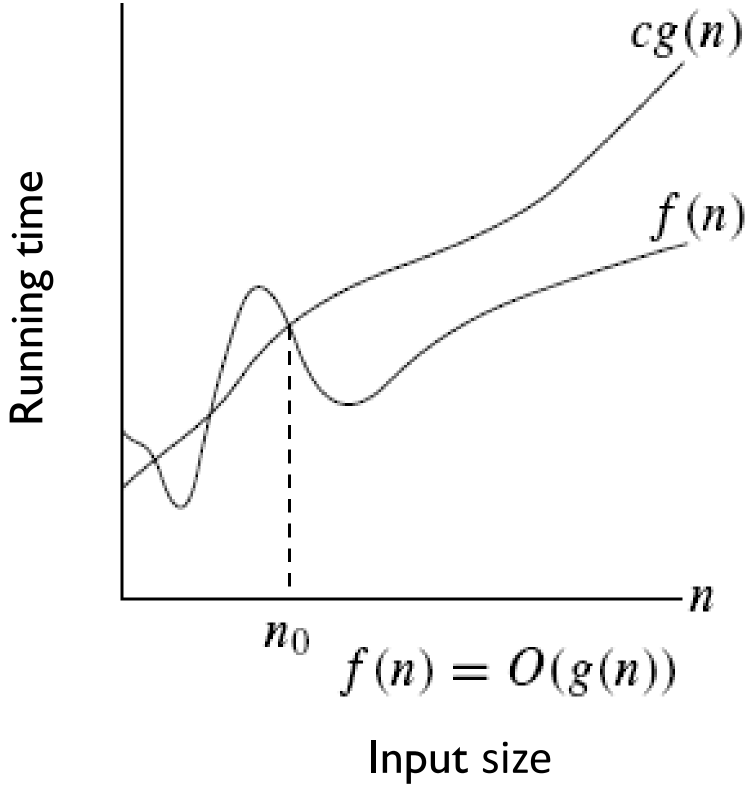
Asymptotic Notation …

- **The Big-Oh (O) Notation**
  - **Asymptotic upper bound**
  - $f(n)=O(g(n))$, if there exists constants $c > 0$ and $n_0 \geq 1$, s.t. $f(n) \leq c\, g(n)$ for $n \geq n_0$
  - $f(n)$ and $g(n)$ are functions over non-negative integers.
- **Used for worst case analysis**



Asymptotic Notation …

- **The Big-Oh (O) Notation**
  - Asymptotic upper bound
  - if f & g be the functions then
    if $\lim_{n\to\infty} f(n)/g(n) = c < \infty$
    Then
    f(n) € O(g(n))



Asymptotic Notation ...

For example: $f(n) = 3n+2$ $g(n)=n$

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

Then $f(n) \leq c.\ g(n)$

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

$c=4\ ,\ n_0 = 2$

$$\therefore\ f(n) = O(n)$$

$3n+2 \leq 3n+2n$ $\forall n \geq 1$

$3n+2 \leq 5n$

or $3n+2 \leq 5n^2$

$$\therefore\ f(n) = O(n^2)$$

Since O(n) is closest bound so we will take O(n)

$$1 < \log_2 n < \sqrt{n} < n < n\log_2 n < n^2 < n^3 < \ldots\ldots\ldots < 2^n < 3^n \ldots < n^n$$

Asymptotic Notation …

- **For example: $f(n) = 3n+2 \quad g(n)=n$**
- $f(n)= O(g(n))$

$$\lim_{n\to\infty} f(n)/g(n) = c < \infty$$
$$= \lim_{n\to\infty} (3n+2)/n$$
$$= \lim_{n\to\infty} (3+2/n)$$
$$= \lim_{n\to\infty} (3+2/\infty)$$
$$= 3+ 2/(1/0)$$
$$= 3+ 2*0/1$$
$$= 3+0$$
$$= 3 < \infty$$

Hence, we can say $f(n)=O(g(n))$

$$f(n)=O(n)$$

Big-Oh and Growth Rate

- The big-Oh notation gives an **upper bound on the growth rate** of a function
- The statement "$f(n)$ is $O(g(n))$" means that the growth rate of $f(n)$ is no more than the growth rate of $g(n)$
- We can use the big-Oh notation to rank functions according to their growth rate

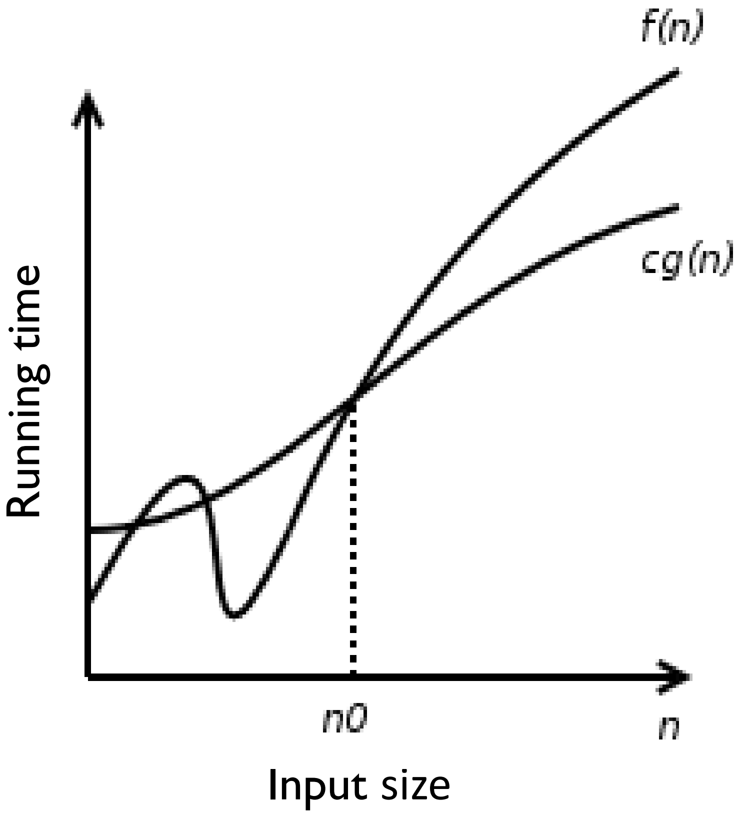
	$f(n)$ is $O(g(n))$	$g(n)$ is $O(f(n))$
$g(n)$ grows more	Yes	No
$f(n)$ grows more	No	Yes
Same growth	Yes	Yes

Big-Oh Rules

- If $f(n)$ is a polynomial of degree d, then $f(n)$ is $O(n^d)$, i.e.,
  1. Drop lower-order terms
  2. Drop constant factors
- Use the smallest possible class of functions
  - Say "$2n$ is $O(n)$" instead of "$2n$ is $O(n^2)$"
- Use the simplest expression of the class
  - Say "$3n + 5$ is $O(n)$" instead of "$3n + 5$ is $O(3n)$"

Asymptotic Notation ...

- **The Big-Omega (Ω) Notation**
  - Asymptotic lower bound
  - $f(n) = \Omega(g(n))$, if there exists constants $c > 0$ and $n_0 \geq 1$, s.t. $f(n) \geq c\,g(n) \geq 0$ for $n \geq n_0$
  - $f(n)$ and $g(n)$ are functions over non-negative integers.



- **Used for best case running time or lower bound of algorithmic problem.**

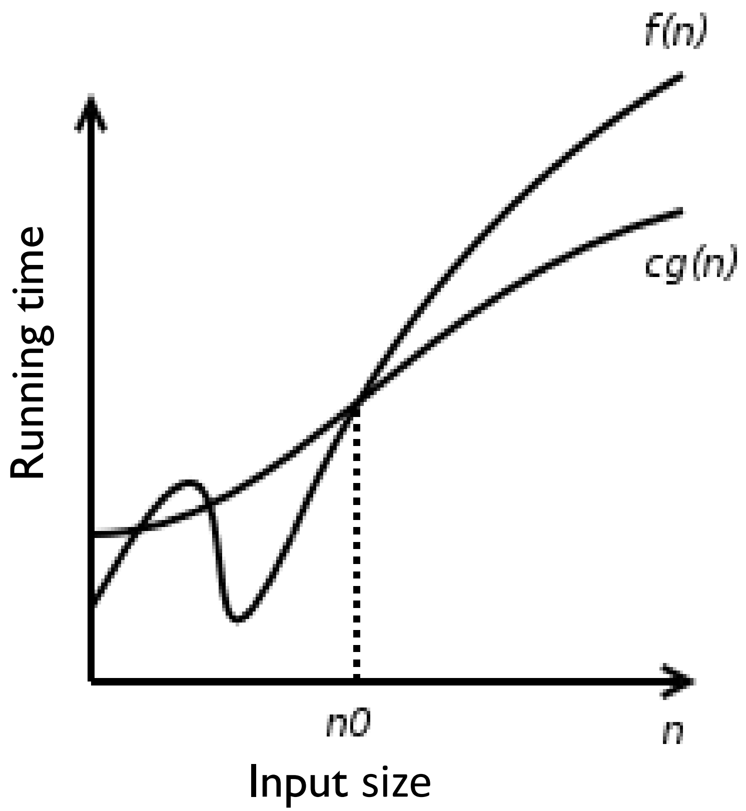
Asymptotic Notation …

- **The Big-Omega (Ω) Notation**
  - Asymptotic lower bound
  - if f & g be the functions then

    if $\lim_{n\to\infty} f(n)/g(n) > 0$

    Then

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



Asymptotic Notation …

- **For example: $f(n) = 3n+2 \quad g(n)=n$**
- $f(n)= \Omega(g(n))$

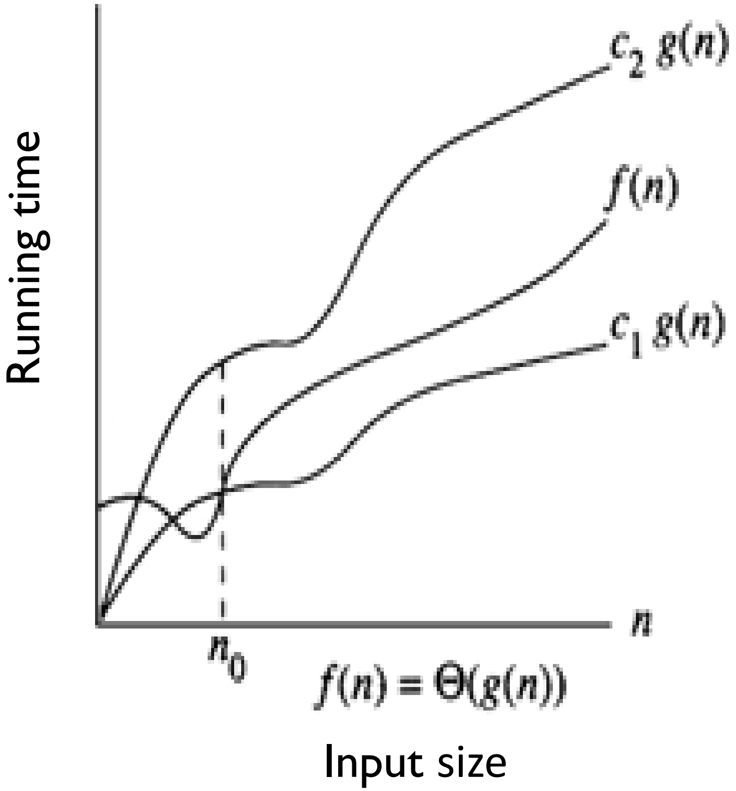
  $f(n) \geq c.\ g(n)$

  $3n+2 \geq c\ .\ n$

  $c=1,\ n_0 = 1 \quad n \geq 1$

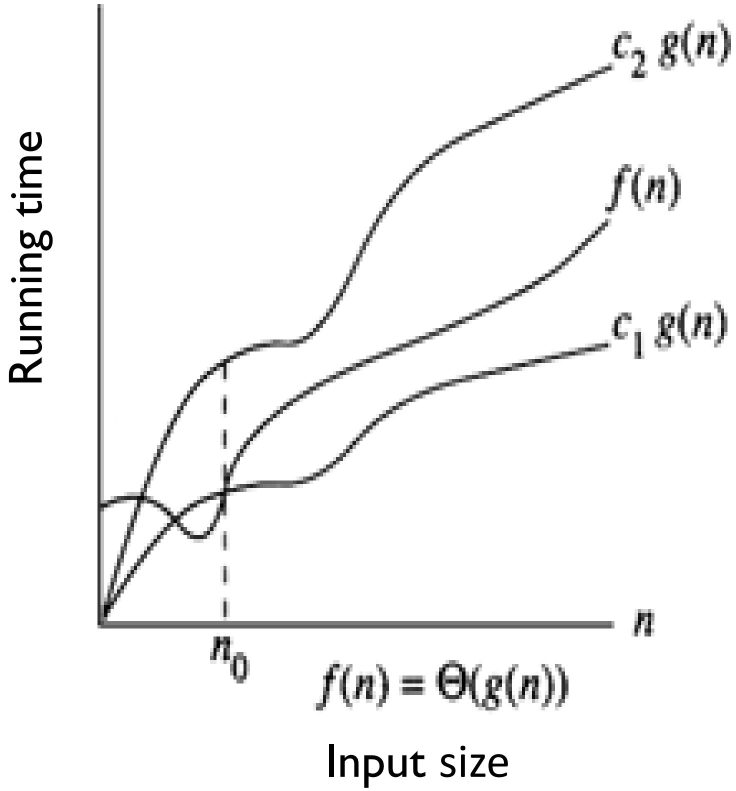
Asymptotic Notation …

- **The Big-Theta (Θ) Notation**
  - Asymptotically tight bound
  - $f(n) = \Theta(g(n))$, if there exists constants c1 & c2 and n_0 , s.t. $c1.\ g(n) \le f(n) \le c2.\ g(n)$ for $n \ge n_0$
  - f(n) and g(n) are functions over non-negative integers.
- **Used for average case running time**



Asymptotic Notation …

- **The Big-Theta (Θ) Notation**
  - Asymptotically tight bound
  - if f & g be the functions then
    if $\lim_{n\to\infty} f(n)/g(n) < \infty$
    Then,
    $f(n) \in \Theta(g(n))$



Asymptotic Notation ...

- **For example: $f(n) = 3n+2 \quad g(n)=n$**
- $f(n) = \Theta\ (g(n))$
- $c1.g(n) \le f(n) \le c2.\ g(n)$

$f(n) \le c2\ .\ g(n)$

$3n+2 \le c2\ .\ n$

$c2=4\ ,\ n_0 \ge 1$

$f(n) \ge c1.\ g(n)$

$3n+2 \ge c1.\ n$

$c1=1,\ n_0 \ge 1$

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

Asymptotic Notation …

Match each function with an equivalent function, in terms of their Θ. Only match a function if $f(n) = \Theta(g(n))$.

$f(n)$	$g(n)$
$n+30$	n^2+3n
$n^2+2n-10$	n^4
$n^3 * 3n$	$\log_2 2x$
$\log_2 x$	$3n-1$

$f(n)$	$g(n)$
$n+30$	$3n-1$
$n^2+2n-10$	n^2+3n
$n^3 * 3n$	n^4
$\log_2 x$	$\log_2 2x$

Any Questions ?

Dr. Anand Singh Jalal
Professor
Email: asjalal@gla.ac.in