Topic: Asymptotic notations

CSI-406 [3(3-0)]

Basic Concepts

An *algorithm* is a finite set of precise instructions for performing a computation or for solving a problem.

- What is the goal of analysis of algorithms?
  - To compare algorithms mainly in terms of running time but also in terms of other factors (e.g., memory requirements, programmer's effort etc.)
- What do we mean by running time analysis?
  - Determine how running time increases as the size of the problem increases.

Types of Analysis

- Worst case
  - Provides an upper bound on running time
  - An absolute guarantee that the algorithm would not run longer, no matter what the inputs are
- Best case
  - Provides a lower bound on running time
  - Input is the one for which the algorithm runs the fastest
- Average case
  - Provides a prediction about the running time
  - Assumes that the input is random

Asymptotic Analysis

- To compare two algorithms with running times *f(n)* and *g(n)*, we need a rough measure that characterizes how fast each function grows.
- Express running time as a function of the input size *n* (i.e., *f(n)*).
- Compare different functions corresponding to running times.
- Such an analysis is independent of machine time, programming style,etc.
- Compare functions in the limit, that is, **asymptotically!** (i.e., for large values of *n*)

Asymptotic notation

A way to describe the behavior of functions *in the limit* or *without bounds*.

- The notations are defined in terms of functions whose domains are the set of natural numbers $N=\{0,1,2,\ldots\}$.
- Such notations are convenient for describing the worst-case running time function $T(n)$.
- It can also be extended to the domain of real numbers.

Asymptotic notations

<u>Asymptotic growth rate</u> : -

- Big Oh (O) -notation
- Omega (Ω) -notation
- Theta (Θ) -notation
- Little Oh (o) -notation
- ω-notation

O notation :	asymptotic “less than”	: $f(n) \leq cg(n)$
Ω notation :	asymptotic “greater than”	: $f(n) \geq c\,g(n)$
Θ notation :	asymptotic “equality”	: $c_1\,g(n) \leq f(n) \leq c_2 g(n)$

"O" in asymptotic analysis

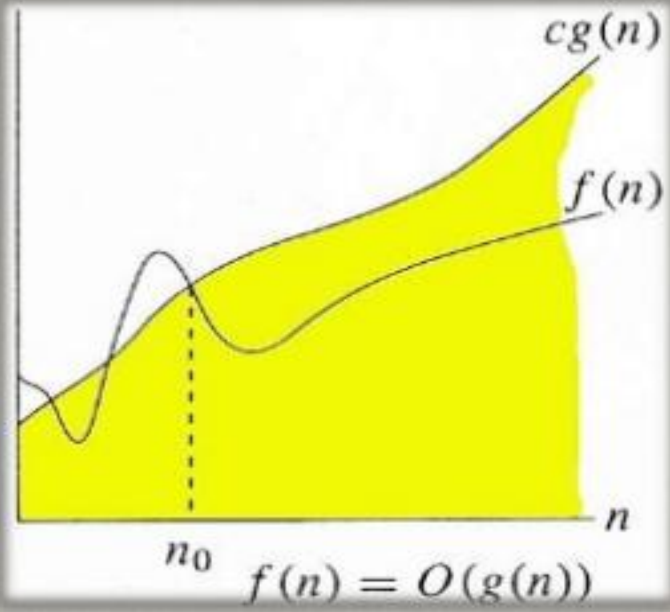
Big-O Notation (Omicron)

possibly *asymptotically* tight **upper** bound for f(n) - Cannot do worse, can do better

- n is the problem size.
- $f(n) \in O(g(n))$ where:

$O(g(n)) = \{ f(n) : \exists$ positive constants c, n_0 such that $0 \le f(n) \le cg(n), \forall n \ge n_0\}$

Meaning for all values of $n \ge n_0$ f(n) is on or below g(n).



- O(g(n)) is a set of all the functions f(n) that are less than or equal to cg(n), $\forall n \ge n_0$.

If $f(n) \le cg(n)$, $c > 0$, $\forall n \ge n_0$ then $f(n) \in O(g(n))$

"Ω" in asymptotic analysis

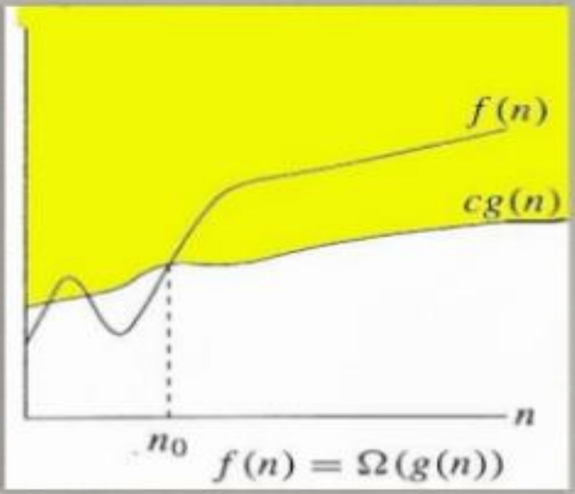
Big Omega Notation (Ω)

possibly asymptotically tight **lower** bound for f(n) - Cannot do better, can do worse
$f(n) \in \Omega(g(n))$ where:

$\Omega(g(n)) = \{f(n): \exists$ positive constants $c > 0$, n_0 such that $0 \le cg(n) \le f(n), \forall\ n \ge n_0\}$

Meaning for all values of $n \ge n_0$ $f(n)$ is on or above $g(n)$.

$\Omega(g(n))$ is a set of all the functions $f(n)$ that are greater than or equal $cg(n)$, $\forall\ n \ge n_0$.



If $cg(n) \le f(n)$, $c > 0$ and $\forall\ n \ge n_0$, then $f(n) \in \Omega(g(n))$

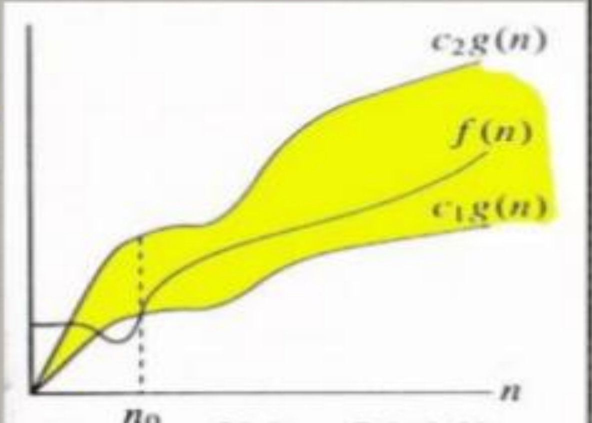
“Θ” in asymptotic analysis

Big Theta Notation(Θ)

asymptotically tight bound for f(n)
f(n) ∈ **Θ**(g(n)) where :

$\Theta(g(n)) =$
$\{f(n)$: $\exists$ positive constants c_1, c_2, n_0 such that
$0 \le c_1\, g(n) \le f(n) \le c_2 g(n),\ \forall\, n \ge n_0\}$

- Positive means greater than 0.
- **Θ**(g(n)) is a set of all the functions f(n) that are between c_1 g(n) and c_2g(n), $\forall$ n ≥ n_0.
- If f(n) is between c_1 g(n) and c_2g(n), $\forall$ n ≥ n_0, then f(n) ∈ **Θ**(g(n))



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

Relations Between Θ, Ω, O

Theorem : For any two functions $g(n)$ and $f(n)$,

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

$f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.

- I.e., $\Theta(g(n)) = O(g(n)) \cap \Omega W(g(n))$
- In practice, asymptotically tight bounds are obtained from asymptotic upper and lower bounds.