# CST 311
# Algorithm Analysis & Design

Al Lake
Oregon Institute of Technology

Chapter 3
Growth of Functions

# Asymptotic Notation

- The notations used to describe the asymptotic running time of an algorithm 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 $T(n)$, which is usually defined only on integer input sizes.

# Θ-notation

- The worst-case running time of insertion sort is:
  - $T(n) = \Theta(n^2)$
- $\Theta(g(n)) = (f(n) :$ $\in$ positive contants $c_1$, $c_2$, and $n_0$ $\exists$

  $$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \;\forall\; n \geq n_0$$

- $\in$ there exists
- $\exists$ such that
- $\forall$ for all

# O-notation

- **O-notation is used to give an upper bound on a function within a constant factor.**
- **O-notation is used when there is only an asymptotic upper bound.**

- **O(g(n)) = (f(n) : $\in$ positive contants c and $n_0$ $\exists$**

  $$0 \leq f(n) \leq cg(n) \; \forall \; n \geq n_0$$

- $\in$ **there exists**
- $\exists$ **such that**
- $\forall$ **for all**

# Ω-notation

- **Ω-notation provides an asymptotic lower bound.**
- **For a given function *g(n)*, Ω*(g(n))* (pronounced "big omega of g of n")**
- **$\Omega(g(n))$ = (f(n) : ∈ positive contants c and $n_0$ ∃**

  $$0 \leq cg(n) \leq f(n) \; \forall \; n \geq n_0$$

- **∈ there exists**
- **∃ such that**
- **∀ for all**

# Comparing Notations

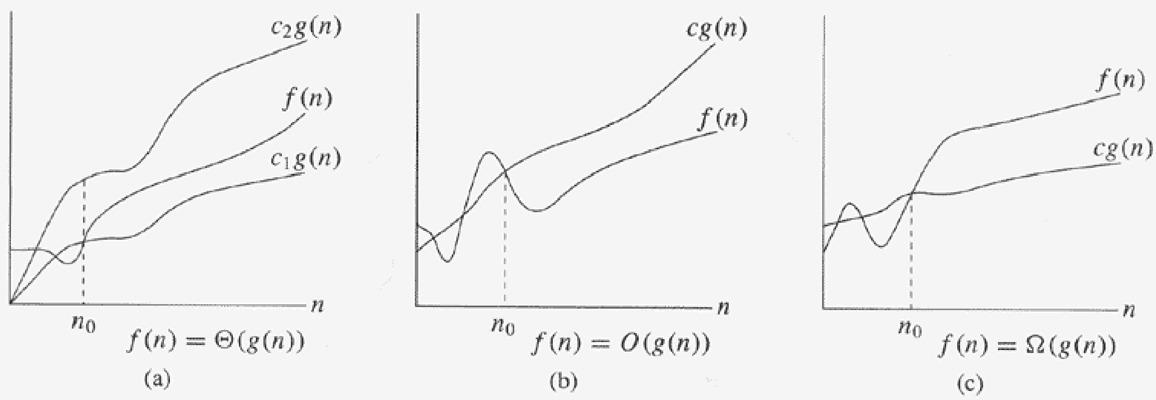


**Figure 3.1** Graphic examples of the $\Theta$, $O$, and $\Omega$ notations. In each part, the value of $n_0$ shown is the minimum possible value; any greater value would also work. **(a)** $\Theta$-notation bounds a function to within constant factors. We write $f(n) = \Theta(g(n))$ if there exist positive constants $n_0$, $c_1$, and $c_2$ such that to the right of $n_0$, the value of $f(n)$ always lies between $c_1g(n)$ and $c_2g(n)$ inclusive. **(b)** $O$-notation gives an upper bound for a function to within a constant factor. We write $f(n) = O(g(n))$ if there are positive constants $n_0$ and $c$ such that to the right of $n_0$, the value of $f(n)$ always lies on or below $cg(n)$. **(c)** $\Omega$-notation gives a lower bound for a function to within a constant factor. We write $f(n) = \Omega(g(n))$ if there are positive constants $n_0$ and $c$ such that to the right of $n_0$, the value of $f(n)$ always lies on or above $cg(n)$.

# o-notation

- **The asymptotic upper bound provided by O-notation may or may not be asymptotically tight.**
- **For example:**
  - **$2n^2 = O(n^2)$ is asymptotically tight**
  - **$2n = O(n^2)$ is not asymptotically tight**

- **o-notation is used to denote an upper bound on a function that is not asymptotically tight.**

- **o(g(n)) = (f(n) : $\in$ positive contants c and $n_0$ $\exists$ $0 \leq f(n) \leq cg(n) \ \forall \ n \geq n_0$**

- $\in$ **there exists**
- $\exists$ **such that**
- $\forall$ **for all**

# Comparison of functions

- Transitivity
- Reflexivity
- Symmetry
- Transpose symmetry

# Standard Notations

- Monotonicity
- Floors and ceilings
- Modular arithmetic
- Polynomials
- Exponentials
- Logarithms
- Factorials
- Functional iteration
- Fibonacci numbers