CS473 - Algorithms I

Lecture 2

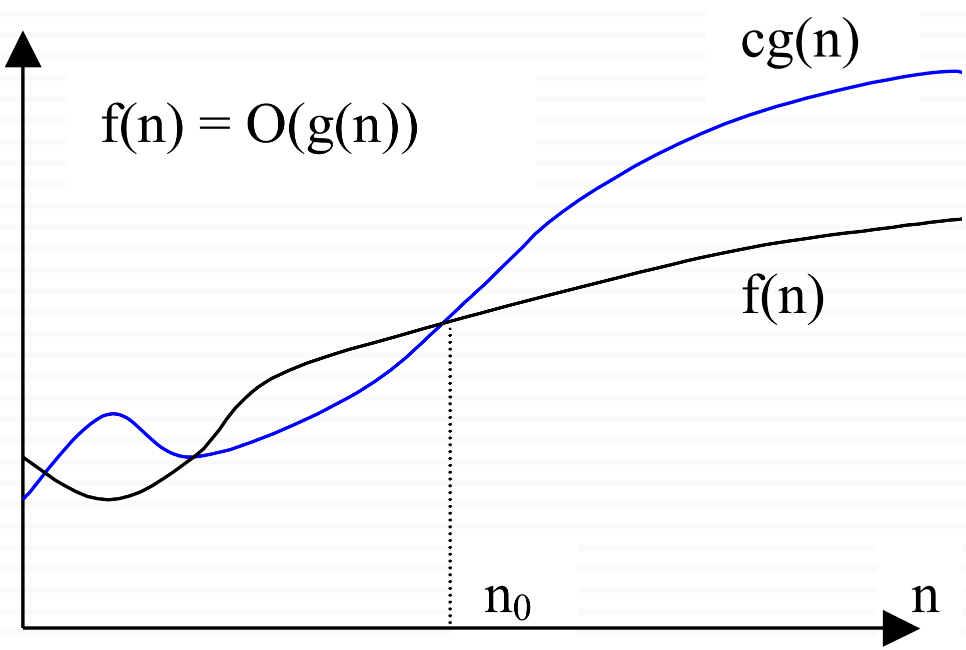
Asymptotic Notation

Cevdet Aykanat and Mustafa Ozdal
Computer Engineering Department, Bilkent University

O-notation: Asymptotic upper bound

$f(n) = O(g(n))$ if $\exists$ positive constants c, n_0 such that

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



Asymptotic running times of algorithms are usually defined by functions whose domain are N={0, 1, 2, …} (natural numbers)

Example

Show that $2n^2 = O(n^3)$

We need to find two positive constants: **c** and $\mathbf{n_0}$ such that:

$$0 \le 2n^2 \le cn^3 \quad \text{for all } n \ge n_0$$

Choose $c = 2$ and $n_0 = 1$

➔ $2n^2 \le 2n^3$ for all $n \ge 1$

Or, choose $c = 1$ and $n_0 = 2$

➔ $2n^2 \le n^3$ for all $n \ge 2$

Example

Show that $2n^2 + n = O(n^2)$

We need to find two positive constants: **c** and $\mathbf{n_0}$ such that:

$$0 \le 2n^2 + n \le cn^2 \text{ for all } n \ge n_0$$

$$2 + (1/n) \le c \text{ for all } n \ge n_0$$

Choose $c = 3$ and $n_0 = 1$

$$\Rightarrow 2n^2 + n \le 3n^2 \text{ for all } n \ge 1$$

O-notation

- What does $f(n) = O(g(n))$ really mean?
  - The notation is a little sloppy
  - One-way equation
    - e.g. $n^2 = O(n^3)$, but we cannot say $O(n^3) = n^2$
- $O(g(n))$ is in fact a set of functions:

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

O-notation

- $O(g(n)) = \{f(n): \exists$ positive constants c, n_0 such that

  $0 \le f(n) \le cg(n), \forall n \ge n_0\}$
- In other words: $O(g(n))$ is in fact:

  the set of functions that have asymptotic upper bound g(n)
- e.g. $2n^2 = O(n^3)$ *means* $2n^2 \in O(n^3)$

$2n^2$ is in the set of functions that have asymptotic upper bound n^3

True or False?

$10^9 n^2 = O(n^2)$	True	Choose $c = 10^9$ and $n_0 = 1$ $0 \le 10^9 n^2 \le 10^9 n^2$ for $n \ge 1$
$100 n^{1.9999} = O(n^2)$	True	Choose $c = 100$ and $n_0 = 1$ $0 \le 100 n^{1.9999} \le 100 n^2$ for $n \ge 1$
$10^{-9} n^{2.0001} = O(n^2)$	False	$10^{-9} n^{2.0001} \le c n^2$ for $n \ge n_0$ $10^{-9} n^{0.0001} \le c$ for $n \ge n_0$ Contradiction

O-notation

- O-notation is an upper bound notation
- What does it mean if we say:

"The runtime ($T(n)$) of Algorithm A is ***at least*** $O(n^2)$"

→ says nothing about the runtime. Why?

$O(n^2)$: The set of functions with asymptotic ***upper bound*** n^2

$T(n) \geq O(n^2)$ means: $T(n) \geq h(n)$ for some $h(n) \in O(n^2)$

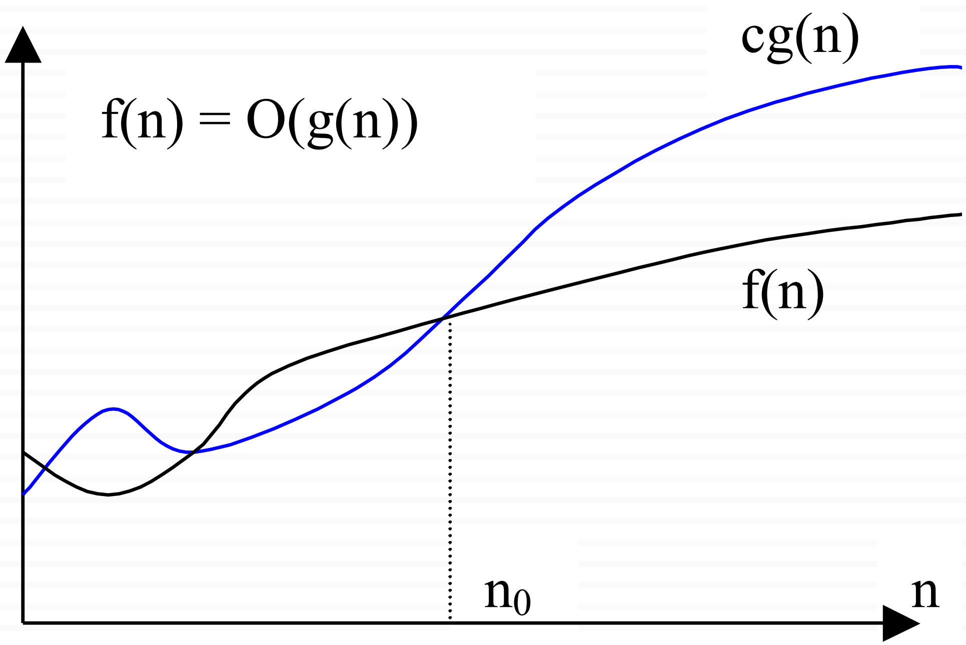
$h(n) = 0$ function is also in $O(n^2)$. Hence: $T(n) \geq 0$

runtime must be nonnegative anyway!

Summary: *O*-notation: Asymptotic upper bound

$f(n) \in O(g(n))$ if $\exists$ positive constants c, n_0 such that

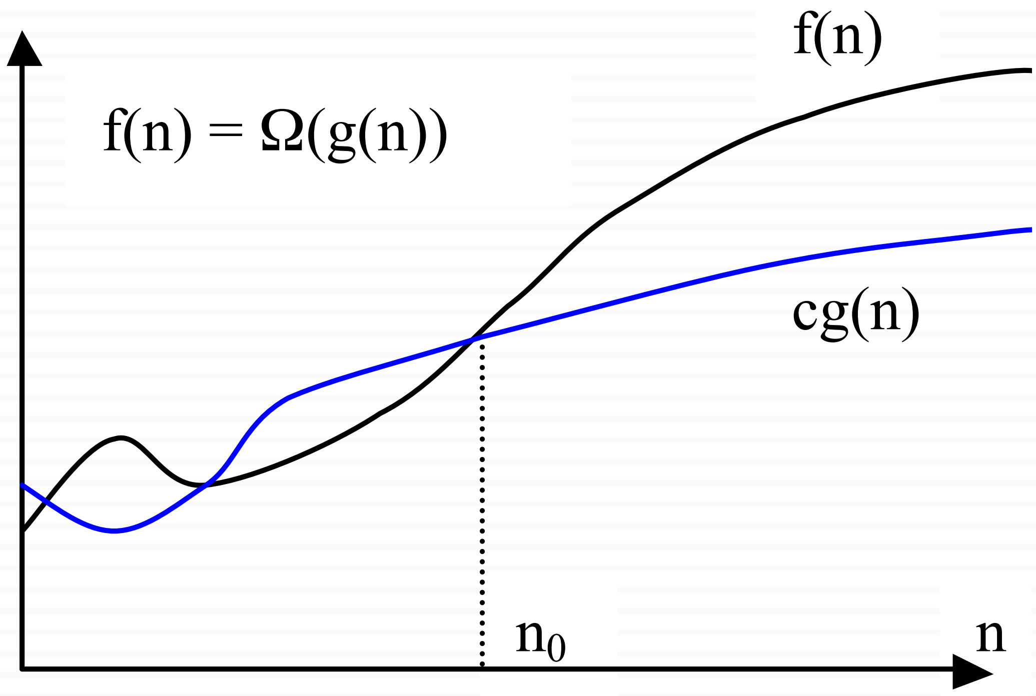
$$0 \leq f(n) \leq cg(n), \forall n \geq n_0$$



Ω-notation: Asymptotic lower bound

$f(n) = \Omega(g(n))$ if $\exists$ positive constants c, n_0 such that

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



Ω: "big Omega"

Example

Show that $2n^3 = \Omega(n^2)$

We need to find two positive constants: **c** and $\mathbf{n_0}$ such that:

$$0 \leq cn^2 \leq 2n^3 \quad \text{for all } n \geq n_0$$

Choose $c = 1$ and $n_0 = 1$

$$\Rightarrow n^2 \leq 2n^3 \text{ for all } n \geq 1$$

Example

Show that $\sqrt{n} = \Omega(\lg n)$

We need to find two positive constants: **c** and $\mathbf{n_0}$ such that:

$$c \lg n \le \sqrt{n} \quad \text{for all } n \ge n_0$$

Choose $c = 1$ and $n_0 = 16$

$$\Rightarrow \lg n \le \sqrt{n} \text{ for all } n \ge 16$$

Ω-notation: Asymptotic Lower Bound

- $\Omega(g(n)) = \{f(n): \exists$ positive constants c, n_0 such that
  $0 \leq cg(n) \leq f(n), \forall n \geq n_0\}$

- In other words: Ω (g(n)) is in fact:
  the set of functions that have asymptotic lower bound g(n)

True or False?

$10^9 n^2 = \Omega(n^2)$

True

Choose $c = 10^9$ and $n_0 = 1$

$0 \leq 10^9 n^2 \leq 10^9 n^2$ for $n \geq 1$

$\cancel{100n^{1.9999} = \Omega(n^2)}$

False

$cn^2 \leq 100n^{1.9999}$ for $n \geq n_0$

$n^{0.0001} \leq (100/c)$ for $n \geq n_0$

Contradiction

$10^{-9} n^{2.0001} = \Omega(n^2)$

True

Choose $c = 10^{-9}$ and $n_0 = 1$

$0 \leq 10^{-9} n^2 \leq 10^{-9} n^{2.0001}$ for $n \geq 1$

Summary: O-notation and Ω-notation

- $O(g(n))$: The set of functions with asymptotic upper bound $g(n)$

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

  $f(n) \in O(g(n))$ if $\exists$ positive constants c, n_0 such that

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

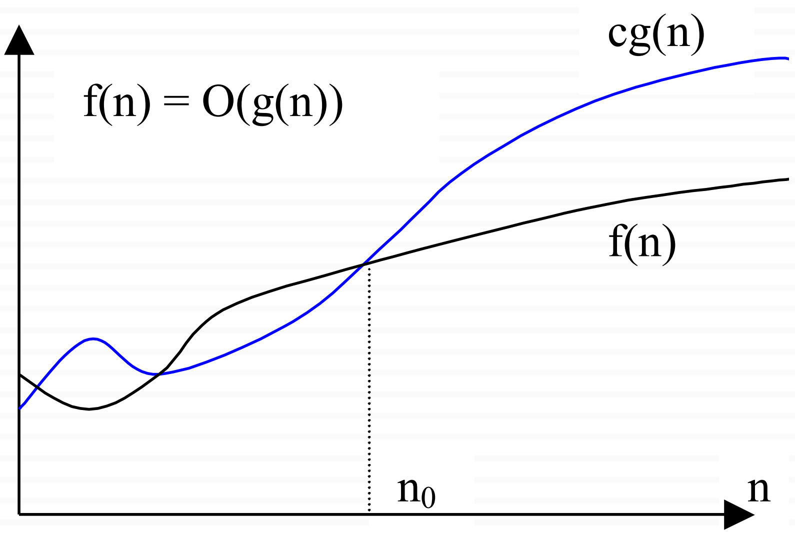
- $\Omega(g(n))$: The set of functions with asymptotic lower bound $g(n)$

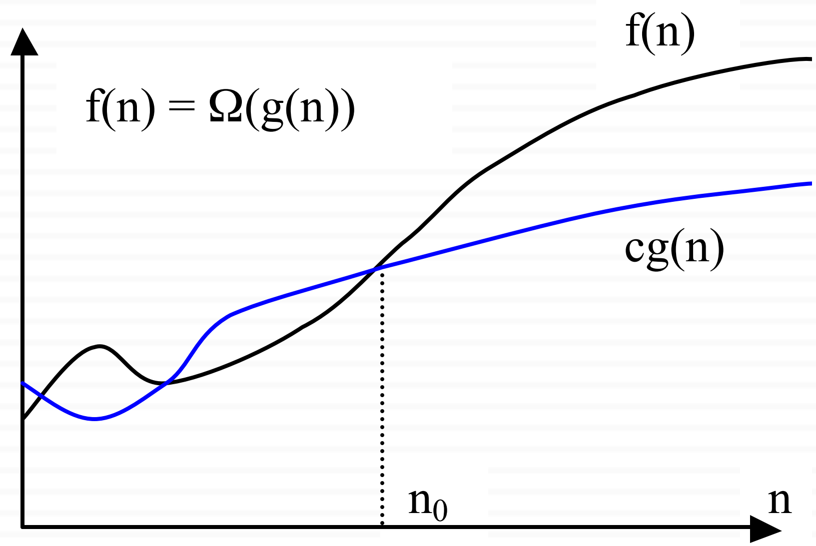
  $f(n) = \Omega(g(n))$

  $f(n) \in \Omega(g(n))$ $\exists$ positive constants c, n_0 such that

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

Summary: O-notation and Ω-notation

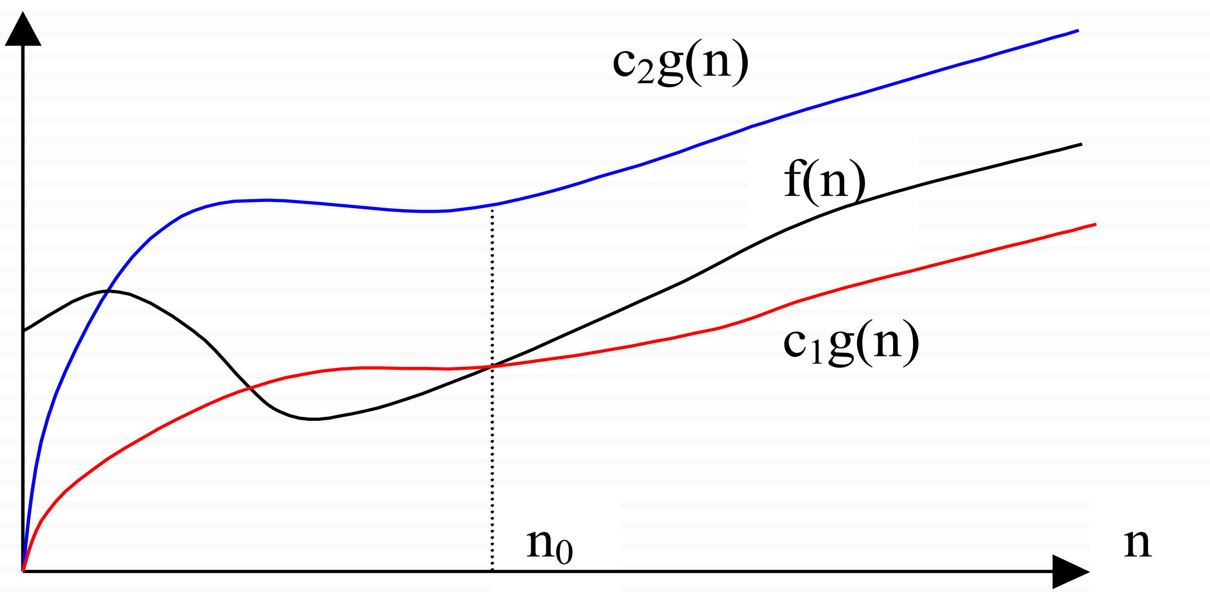




Θ-notation: Asymptotically tight bound

- $f(n)=\Theta(g(n))$ if $\exists$ positive constants c_1, c_2, n_0 such that

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



Example

Show that $2n^2 + n = \Theta(n^2)$

We need to find 3 positive constants: $\mathbf{c_1}$, $\mathbf{c_2}$ and $\mathbf{n_0}$ such that:

$0 \le c_1 n^2 \le 2n^2 + n \le c_2 n^2$ for all $n \ge n_0$

$c_1 \le 2 + (1/n) \le c_2$ for all $n \ge n_0$

Choose $c_1 = 2$, $c_2 = 3$, and $n_0 = 1$

$\Rightarrow 2n^2 \le 2n^2 + n \le 3n^2$ for all $n \ge 1$

Example

Show that $\frac{1}{2}n^2 - 2n = \Theta(n^2)$

We need to find 3 positive constants: $\mathbf{c_1}$, $\mathbf{c_2}$ and $\mathbf{n_0}$ such that:

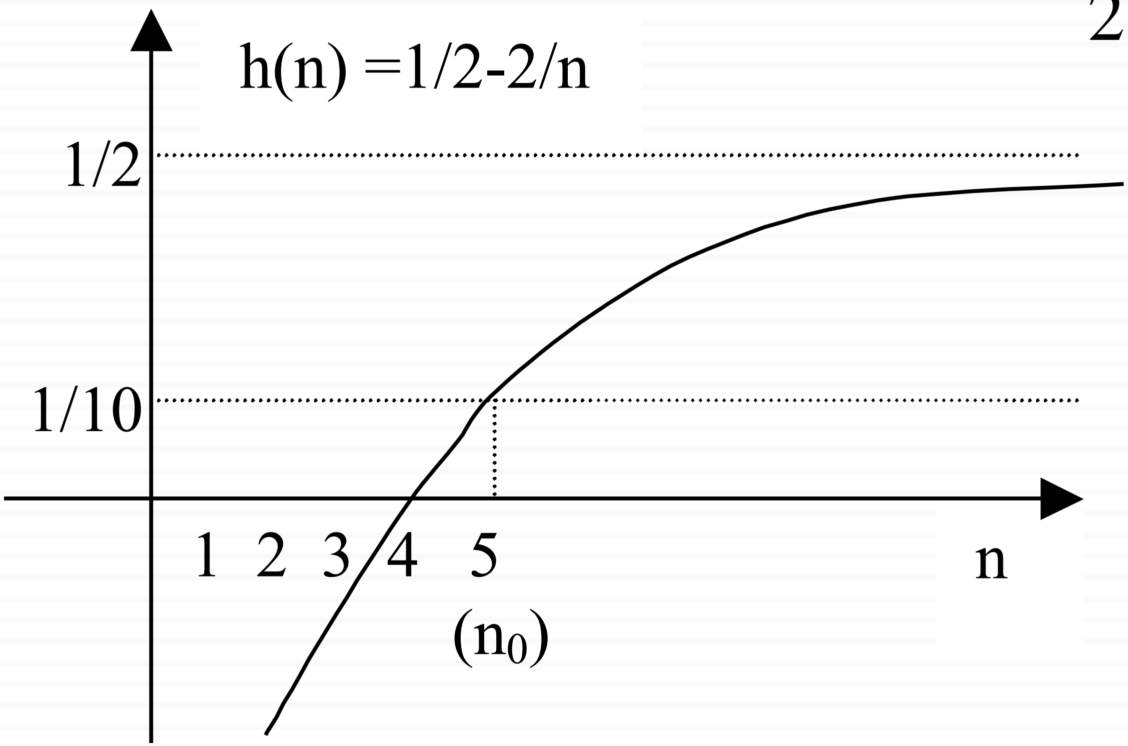
$$0 \le c_1 n^2 \le \frac{1}{2}n^2 - 2n \le c_2 n^2 \quad \text{for all } n \ge n_0$$

$$c_1 \le \frac{1}{2} - \frac{2}{n} \le c_2 \quad \text{for all } n \ge n_0$$

Example (cont'd)

- Choose 3 positive constants: c_1, c_2, n_0 that satisfy:

$$c_1 \le \frac{1}{2} - \frac{2}{n} \le c_2 \qquad \text{for all } n \ge n_0$$



$$\frac{1}{10} \le \frac{1}{2} - \frac{2}{n} \qquad \text{for } n \ge 5$$

$$\frac{1}{2} - \frac{2}{n} \le \frac{1}{2} \qquad \text{for } n \ge 0$$

Example (cont'd)

- Choose 3 constants: c_1, c_2, n_0 that satisfy:

$$c_1 \leq \frac{1}{2} - \frac{2}{n} \leq c_2 \qquad \text{for all } n \geq n_0$$

$$\frac{1}{10} \leq \frac{1}{2} - \frac{2}{n} \qquad \text{for } n \geq 5$$

$$\frac{1}{2} - \frac{2}{n} \leq \frac{1}{2} \qquad \text{for } n \geq 0$$

Therefore, we can choose:: $c_1 = \frac{1}{10}$ $\quad c_2 = \frac{1}{2}$ $\quad n_0 = 5$

Θ-notation: Asymptotically tight bound

- Theorem: leading constants & low-order terms don't matter
- Justification: can choose the leading constant large enough to make high-order term dominate other terms

True or False?

$10^9 n^2 = \Theta(n^2)$ True

~~$100n^{1.9999} = \Theta(n^2)$~~ False

~~$10^{-9} n^{2.0001} = \Theta(n^2)$~~ False

Θ-notation: Asymptotically tight bound

- $\Theta(g(n))=\{f(n)$: $\exists$ positive constants c_1, c_2, n_0 such that

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

- In other words: $\Theta(g(n))$ is in fact:

  the set of functions that have asymptotically tight bound g(n)

Θ-notation: Asymptotically tight bound

- Theorem:

  $f(n) = \Theta(g(n))$ if and only if

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

- In other words:

  Θ is stronger than both O and Ω

- In other words:

  $\Theta(g(n)) \subseteq O(g(n))$ and

  $\Theta(g(n)) \subseteq \Omega(g(n))$

Example

- Prove that $10^{-8} n^2 \neq \Theta(n)$

Before proof, note that $10^{-8}n^2 = \Omega(n)$ but $10^{-8}n^2 \neq O(n)$

Proof by contradiction:

Suppose positive constants c_2 and n_0 exist such that:

$$10^{-8}n^2 \leq c_2 n \quad \text{for all } n \geq n_0$$

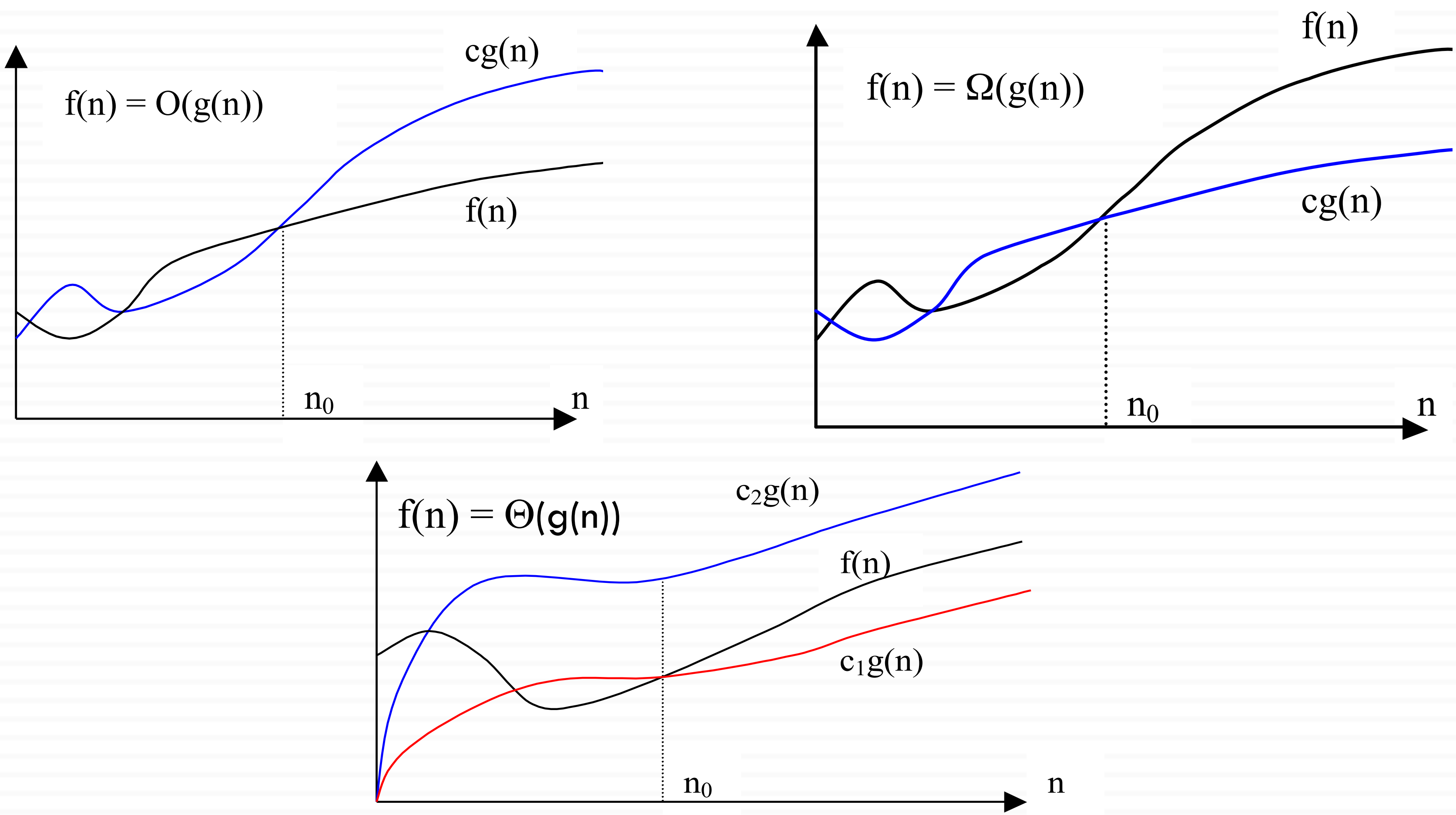
$$10^{-8}n \leq c_2 \quad \text{for all } n \geq n_0$$

Contradiction: c_2 is a constant

Summary: O, Ω, and Θ notations

- $O(g(n))$: The set of functions with asymptotic upper bound $g(n)$
- $\Omega(g(n))$: The set of functions with asymptotic lower bound $g(n)$
- $\Theta(g(n))$: The set of functions with asymptotically tight bound $g(n)$
- $f(n) = \Theta(g(n))$ if and only if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$

Summary: O, Ω, and Θ notations



o (“small o”) Notation
Asymptotic upper bound that is not tight

Reminder: Upper bound provided by O (“big O”) notation can be tight or not tight:

e.g. $2n^2 = O(n^2)$ is asymptotically tight
$2n = O(n^2)$ is not asymptotically tight

both true

o-Notation: An upper bound that is not asymptotically tight

o ("small o") Notation
Asymptotic upper bound that is <u>not tight</u>

- $o(g(n)) = \{f(n)$: for **<u>any</u>** constant $c > 0$,

  $\exists$ a constant $n_0 > 0$, such that

  $0 \le f(n) < cg(n), \forall n \ge n_0\}$

- Intuitively: $$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$$

  - e.g., $2n = o(n^2)$, any positive c satisfies

    but $2n^2 \ne o(n^2)$, $c = 2$ does not satisfy

ω ("small omega") Notation
Asymptotic lower bound that is <u>not tight</u>

- $\omega(g(n))$ = {$f(n)$: for **any** constant $c > 0$,
  $\exists$ a constant $n_0 > 0$, such that
  $0 \le cg(n) < f(n), \forall n \ge n_0$}

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

- e.g., $n^2/2 = \omega(n)$, any positive c satisfies
  but $n^2/2 \neq \omega(n^2)$, $c = 1/2$ does not satisfy

Analogy to the comparison of two real numbers

- $f(n) = O(g(n)) \leftrightarrow a \leq b$
- $f(n) = \Omega(g(n)) \leftrightarrow a \geq b$
- $f(n) = \Theta(g(n)) \leftrightarrow a = b$

- $f(n) = o(g(n)) \leftrightarrow a < b$
- $f(n) = \omega(g(n)) \leftrightarrow a > b$

True or False?

$5n^2 = O(n^2)$	True	$n^2 \lg n = O(n^2)$	False
$5n^2 = \Omega(n^2)$	True	$n^2 \lg n = \Omega(n^2)$	True
$5n^2 = \Theta(n^2)$	True	$n^2 \lg n = \Theta(n^2)$	False
$5n^2 = o(n^2)$	False	$n^2 \lg n = o(n^2)$	False
$5n^2 = \omega(n^2)$	False	$n^2 \lg n = \omega(n^2)$	True

$2^n = O(3^n)$	True		
$2^n = \Omega(3^n)$	False	$2^n = o(3^n)$	True
$2^n = \Theta(3^n)$	False	$2^n = \omega(3^n)$	False

Analogy to comparison of two real numbers

- Trichotomy property for real numbers:

  For any two real numbers a and b,

  we have ***either*** $a < b$, ***or*** $a = b$, ***or*** $a > b$

- Trichotomy property *does not hold* for asymptotic notation

  For two functions $f(n)$ & $g(n)$, it may be the case that

  neither $f(n) = O(g(n))$ ***nor*** $f(n) = \Omega(g(n))$ ***holds***

  e.g. n and $n^{1+\sin(n)}$ *cannot be compared asymptotically*

Asymptotic Comparison of Functions

(Similar to the relational properties of real numbers)

Transitivity: holds for all

e.g., $f(n) = \Theta(g(n))$ & $g(n) = \Theta(h(n)) \Rightarrow f(n) = \Theta(h(n))$

Reflexivity: holds for Θ, O, Ω

e.g., $f(n) = O(f(n))$

Symmetry: holds only for Θ

e.g., $f(n) = \Theta(g(n)) \Leftrightarrow g(n) = \Theta(f(n))$

Transpose symmetry: holds for $(O \leftrightarrow \Omega)$ and $(o \leftrightarrow \omega)$

e.g., $f(n) = O(g(n)) \Leftrightarrow g(n) = \Omega(f(n))$

Using O-Notation to Describe Running Times

- Used to bound worst-case running times
  - Implies an upper bound runtime for arbitrary inputs as well

- Example:

  "Insertion sort has worst-case runtime of $O(n^2)$"

  Note: This $O(n^2)$ upper bound also applies to its running time on every input.

Using O-Notation to Describe Running Times

- Abuse to say "running time of insertion sort is $O(n^2)$"
- For a given n, the actual running time *<u>depends on</u> the particular input* of size n
  - i.e., running time is not only a function of n
- However, worst-case running time is only a function of n

Using O-Notation to Describe Running Times

- When we say:

  "*Running time of insertion sort is $O(n^2)$*",

  what we really mean is:

  "*Worst-case running time of insertion sort is $O(n^2)$*"

  or equivalently:

  "No matter what particular input of size n is chosen, the running time on that set of inputs is $O(n^2)$"

Cevdet Aykanat and Mustafa Ozdal
Computer Engineering Department, Bilkent University

Using Ω-Notation to Describe Running Times

- Used to bound best-case running times
  - Implies a lower bound runtime for arbitrary inputs as well

- Example:

  "Insertion sort has best-case runtime of $\Omega(n)$"

  Note: This $\Omega(n)$ lower bound also applies to its running time on every input.

Cevdet Aykanat and Mustafa Ozdal
Computer Engineering Department, Bilkent University

Using Ω-Notation to Describe Running Times

- When we say:

  "*Running time of algorithm A is $\Omega(g(n))$*",

  what we mean is:

  "For any input of size n, the runtime of A is at least a constant times $g(n)$ for sufficiently large n"

Using Ω-Notation to Describe Running Times

- *Note*: It's not contradictory to say:

  "*worst-case running time of insertion sort is $\Omega(n^2)$*"

  because there exists an input that causes the algorithm to take $\Omega(n^2)$.

Using Θ-Notation to Describe Running Times

- Consider 2 cases about the runtime of an algorithm:

- Case 1: Worst-case and best-case not asymptotically equal
  - ➔ Use Θ-notation to bound worst-case and best-case runtimes separately

- Case 2: Worst-case and best-case asymptotically equal
  - ➔ Use Θ-notation to bound the runtime for any input

Using Θ-Notation to Describe Running Times Case 1

- Case 1: Worst-case and best-case not asymptotically equal
  → Use Θ-notation to bound the worst-case and best-case runtimes separately

- We can say:
  - "The worst-case runtime of insertion sort is $\Theta(n^2)$"
  - "The best-case runtime of insertion sort is $\Theta(n)$"

- But, we can't say:
  - "The runtime of insertion sort is $\Theta(n^2)$ for every input"

- A Θ-bound on worst-/best-case running time does not apply to its running time on arbitrary inputs

Using Θ-Notation to Describe Running Times Case 2

- Case 2: Worst-case and best-case asymptotically equal
  - → Use Θ-notation to bound the runtime for any input

- e.g. For merge-sort, we have:

$$\left.\begin{aligned} T(n) &= O(n\lg n) \\ T(n) &= \Omega(n\lg n) \end{aligned}\right\} \quad T(n) = \Theta(n\lg n)$$

Using Asymptotic Notation to Describe Runtimes Summary

- "The <u>worst case</u> runtime of Insertion Sort is $O(n^2)$"
  - Also implies: "The runtime of Insertion Sort is $O(n^2)$"

- "The <u>best-case</u> runtime of Insertion Sort is $\Omega(n)$"
  - Also implies: "The runtime of Insertion Sort is $\Omega(n)$"

- "The <u>worst case</u> runtime of Insertion Sort is $\Theta(n^2)$"
  - But: "The runtime of Insertion Sort is not $\Theta(n^2)$"

- "The <u>best case</u> runtime of Insertion Sort is $\Theta(n)$"
  - But: "The runtime of Insertion Sort is not $\Theta(n)$"

Using Asymptotic Notation to Describe Runtimes Summary

- "The worst case runtime of Merge Sort is $\Theta(n\lg n)$"
- "The best case runtime of Merge Sort is $\Theta(n\lg n)$"
- "The runtime of Merge Sort is $\Theta(n\lg n)$"
  - *This is true, because the best and worst case runtimes have asymptotically the same tight bound* $\Theta(n\lg n)$

Asymptotic Notation in Equations

- Asymptotic notation appears alone on the RHS of an equation:
  - implies set membership

    e.g., $n = O(n^2)$ means $n \in O(n^2)$

- Asymptotic notation appears on the RHS of an equation
  - stands for some anonymous function in the set

    e.g., $2n^2 + 3n + 1 = 2n^2 + \Theta(n)$ means:

    $2n^2 + 3n + 1 = 2n^2 + h(n)$, for some $h(n) \in \Theta(n)$

    i.e., $h(n) = 3n + 1$

Asymptotic Notation in Equations

- Asymptotic notation appears on the LHS of an equation:
  - stands for any anonymous function in the set

    e.g., $2n^2 + \Theta(n) = \Theta(n^2)$ means:

    for any function $g(n) \in \Theta(n)$

    $\exists$ some function $h(n) \in \Theta(n^2)$

    such that $2n^2+g(n) = h(n)$

- RHS provides coarser level of detail than LHS