

CS473-Algorithms I

Lecture 2

Asymptotic Notation

O-notation (upper bounds)

- $f(n) = O(g(n))$ if \exists positive constants c, n_0 such that
$$0 \leq f(n) \leq cg(n), \forall n \geq n_0$$

e.g., $2n^2 = O(n^3)$

$$2n^2 \leq cn^3 \Rightarrow cn \geq 2 \Rightarrow c = 1 \text{ \& } n_0 = 2$$

or

$$c = 2 \text{ \& } n_0 = 1$$

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

O-notation (upper bounds)

- “=” is funny; “one-way” equality
- O-notation is sloppy, but convenient
- though sloppy, must understand what **really** means
- think of $O(g(n))$ as a set of functions:

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

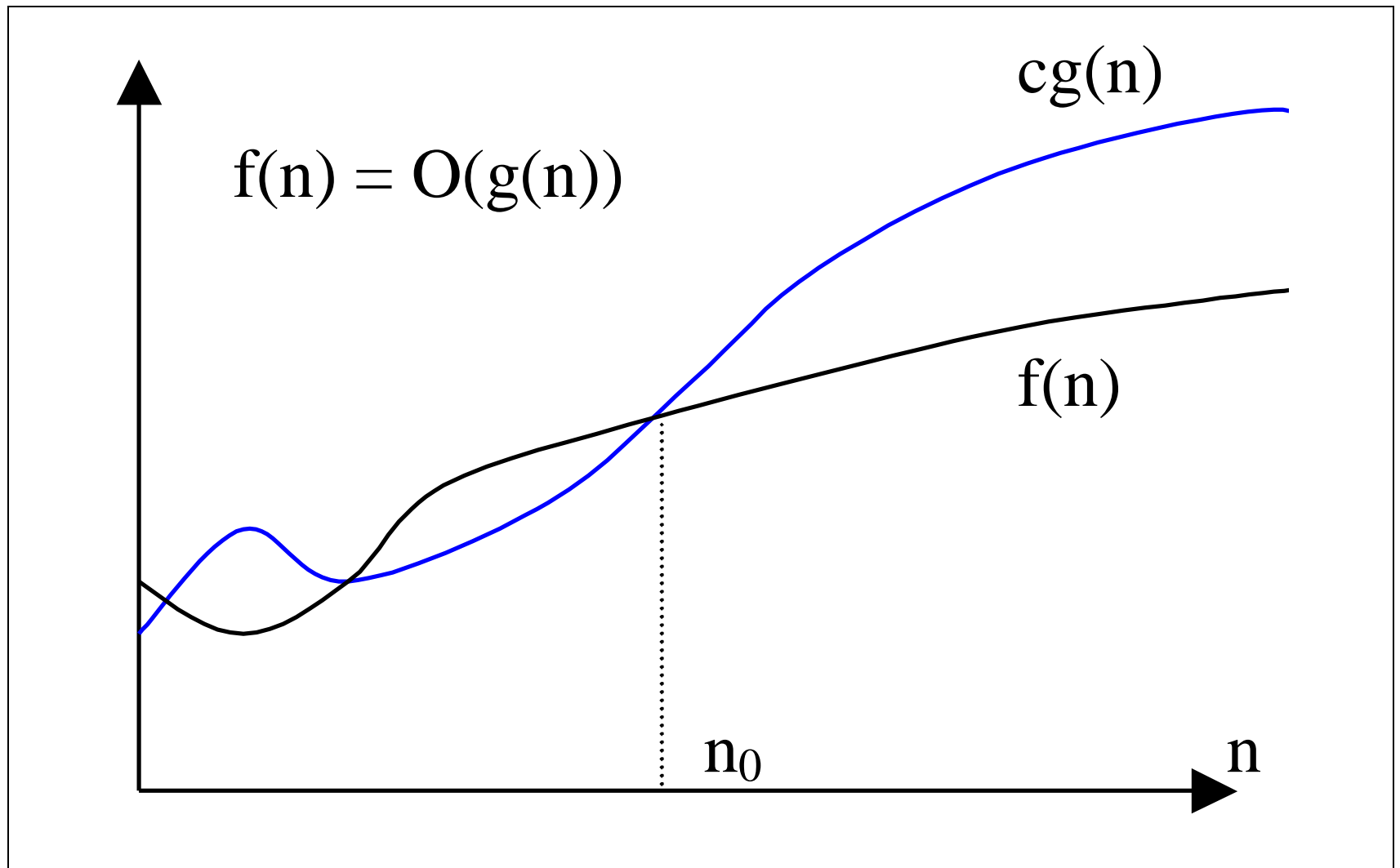
$$0 \leq f(n) \leq cg(n), \forall n \geq n_0\}$$

hence, $2n^2 = O(n^3)$ **means** that $2n^2 \in O(n^3)$

O-notation

- O-notation is an upper-bound notation
- e.g., makes no sense to say “running time of an algorithm is at least $O(n^2)$ ”. Why?
 - let running time be $T(n)$
 - $T(n) \geq O(n^2)$ means
$$T(n) \geq h(n) \text{ for some } h(n) \in O(n^2)$$
 - however, this is true for any $T(n)$ since
$$h(n) = 0 \in O(n^2), \quad \& \quad \text{running time} > 0,$$
so stmt tells nothing about running time

O-notation (upper bounds)



Ω -notation (lower bounds)

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

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

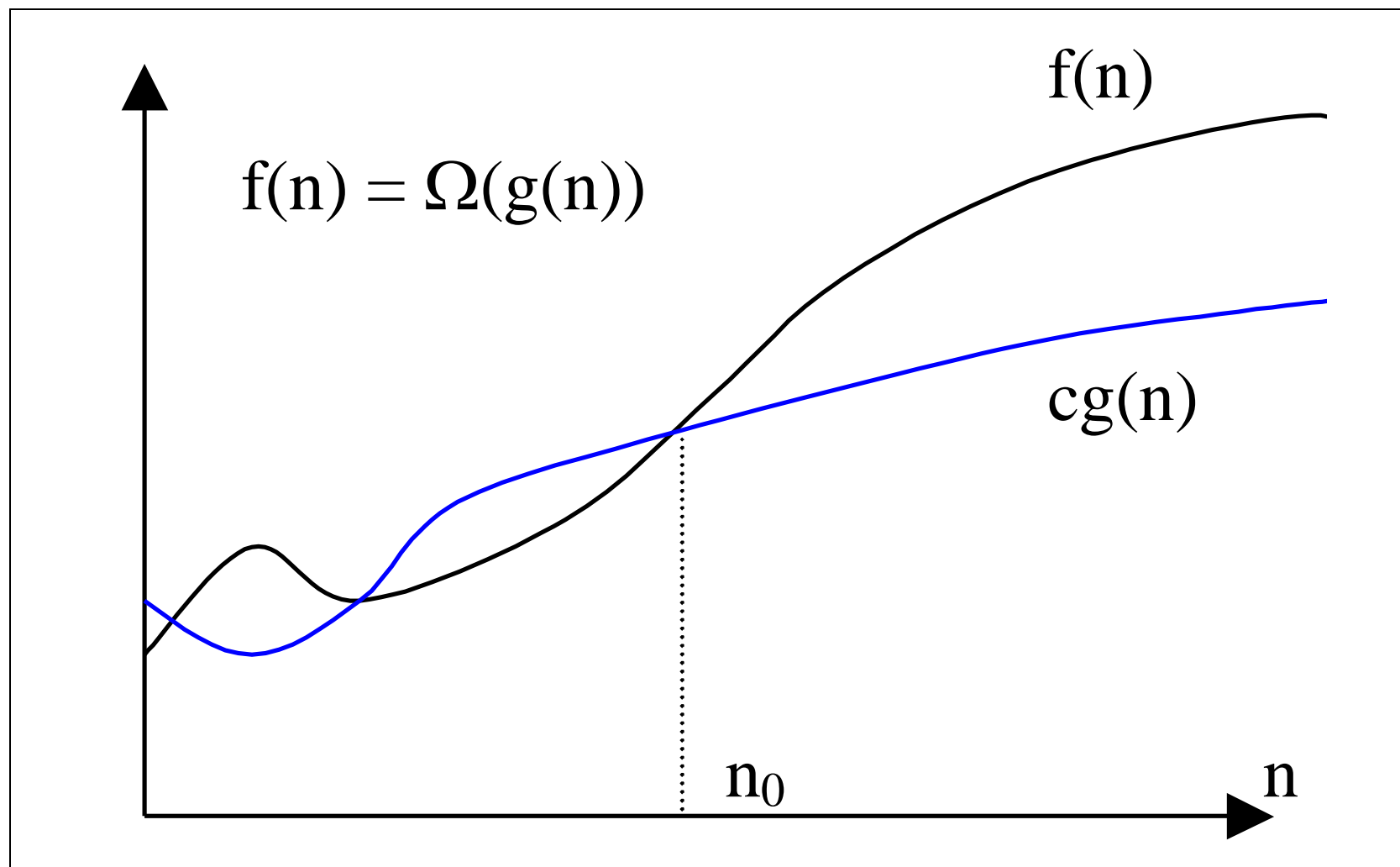
e.g., $\sqrt{n} = \Omega(\lg n)$ ($c = 1, n_0 = 16$)

i.e., $1 \times \lg n \leq \sqrt{n} \quad \forall n \geq 16$

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

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

Ω -notation (lower bounds)



Θ -notation (tight bounds)

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

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

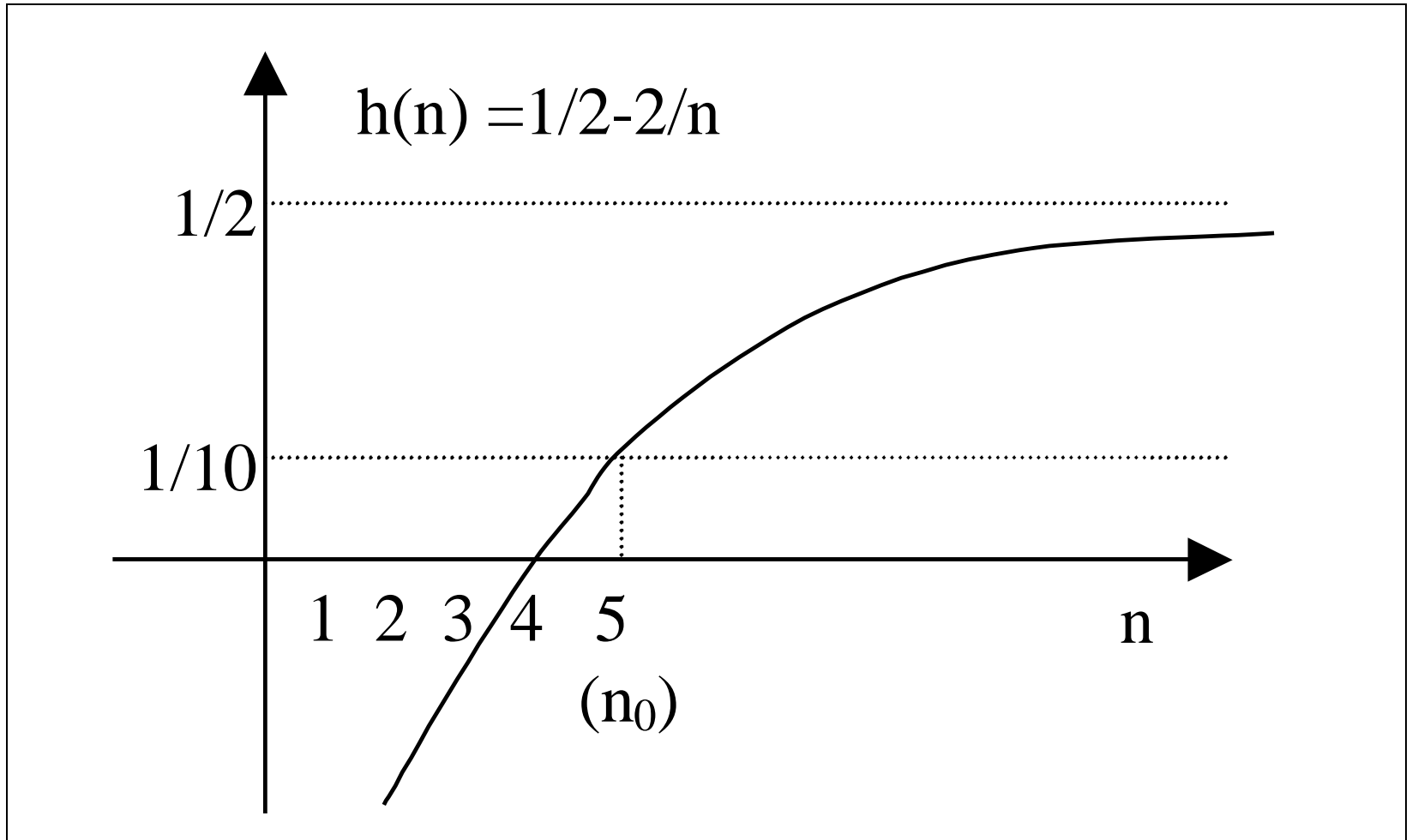
- example:

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

$$0 \leq c_1 n^2 \leq \frac{1}{2}n^2 - 2n \leq c_2 n^2$$

$$c_1 \leq \frac{1}{2} - \frac{2}{n} \leq c_2$$

Θ -notation: example ($0 < c_1 \leq h(n) \leq c_2$)



Θ -notation: example ($0 < c_1 \leq h(n) \leq c_2$)

$$h(n) = \frac{1}{2} - \frac{2}{n} \leq \frac{1}{2} = c_2, \forall n \geq 0$$

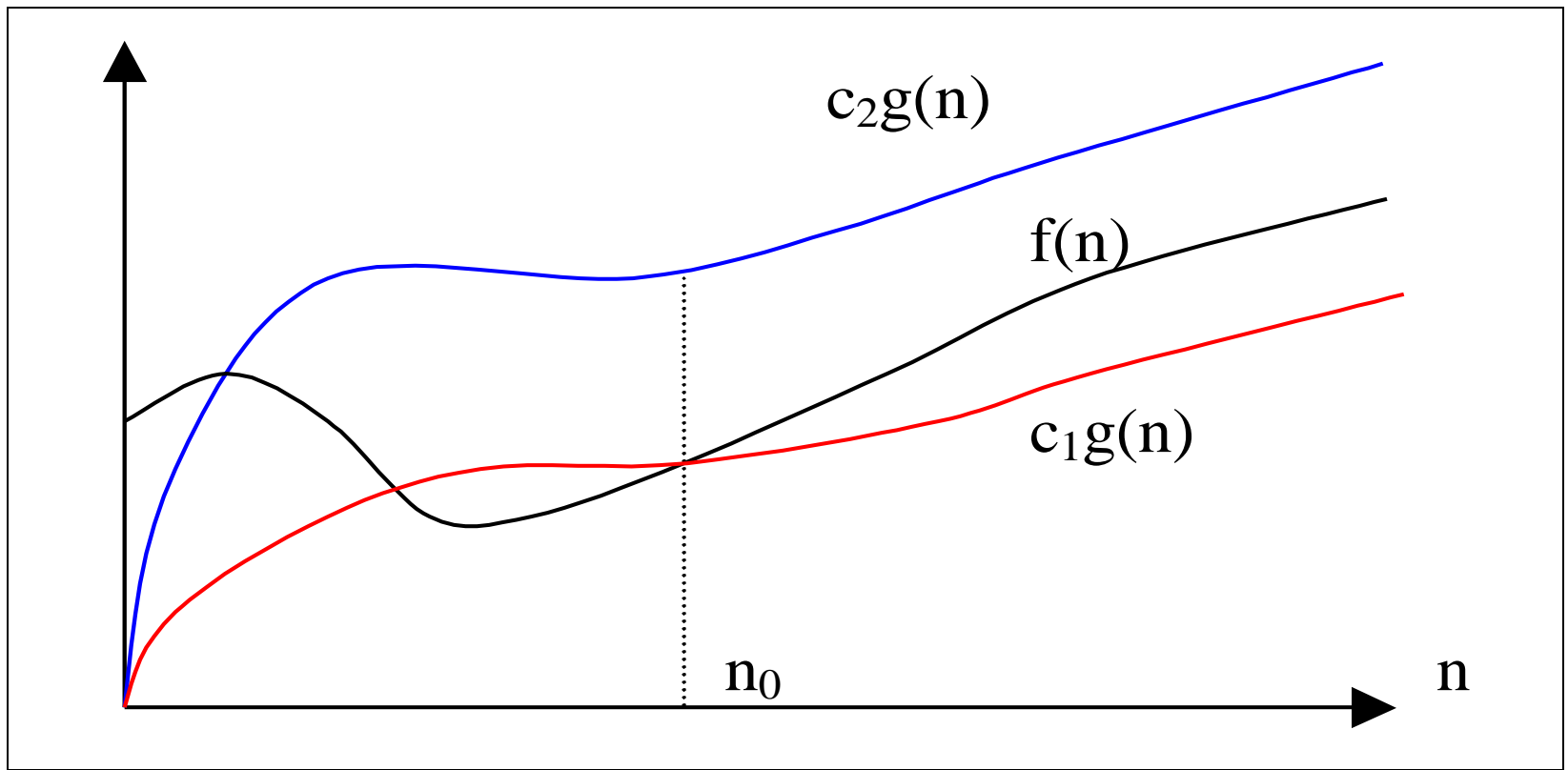
$$h(n) = \frac{1}{2} - \frac{2}{n} \geq \frac{1}{10} = c_1, \forall n \geq 5$$

therefore

$$c_1 = \frac{1}{10}, c_2 = \frac{1}{2}, n_0 = 5$$

Θ -notation (tight bounds)

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



Θ -notation (tight bounds)

- Prove that $10^{-8} n^2 \neq \Theta(n)$
 - suppose c_2, n_0 exist such that $10^{-8} n^2 \leq c_2 n, \forall n \geq n_0$
 - but then $c_2 \geq 10^{-8} n$
 - **contradiction** since c_2 is a constant
- **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

Θ -notation (tight bounds)

- **Theorem:** $(O \text{ and } \Omega) \Leftrightarrow \Theta$
 - Θ is stronger than both O and Ω
 - i.e., $\Theta(g(n)) \subseteq O(g(n))$ and
 $\Theta(g(n)) \subseteq \Omega(g(n))$

Using asymptotic notation for describing running times

O-notation

- used to bound **worst-case** running times
 - also bounds running time on **arbitrary inputs** as well
- e.g., $O(n^2)$ bound on **worst-case** running time of insertion sort also applies to its running time on **every input**

Using O-notation for describing running times

- Abuse to say “running time of insertion sort is $O(n^2)$ ”
 - for a given n , actual running time depends on 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 for describing running times

- What we really mean by “running time of insertion sort is $O(n^2)$ ”
 - worst-case running time of insertion sort is $O(n^2)$

or equivalently

- no matter what particular input of size n is chosen (for each value of) running time on that set of inputs is $O(n^2)$

Using Ω -notation for describing running times

- used to bound the **best-case** running times
 \Rightarrow also bounds the running time on **arbitrary inputs** as well
- e.g., $\Omega(n)$ bound on **best-case** running time of insertion sort
 \Rightarrow running time of insertion sort is $\Omega(n)$

Using Ω -notation for describing running times

- “running time of an algorithm is $\Omega(g(n))$ ” means
 - no matter what **particular** input of size **n** is chosen (for any **n**), running time on that set of inputs is at least a constant times **$g(n)$** , for sufficiently large **n**
 - however, it is not contradictory to say “**worst-case** running time of insertion sort is **$\Omega(n^2)$** ” since there exists an input that causes algorithm to take **$\Omega(n^2)$** time

Using Θ -notation for describing running times

- 1) used to bound **worst-case & best-case** running times of an algorithm if they **are not** asymptotically **equal**
- 2) used to bound running time of an algorithm if its **worst & best case** running times are asymptotically **equal**

Using Θ -notation for describing running times

Case (1):

- a Θ -bound on worst-/best-case running time does not apply to its running time on arbitrary inputs
- e.g., $\Theta(n^2)$ bound on worst-case running time of insertion sort does not imply a $\Theta(n^2)$ bound on running time of insertion sort on every input
since $T(n) = O(n^2)$ & $T(n) = \Omega(n)$ for insertion sort

Using Θ -notation for describing running times

Case (2):

- implies a Θ -bound on every input
 - e.g., merge sort

$$\left. \begin{array}{l} T(n) = O(n \lg n) \\ T(n) = \Omega(n \lg n) \end{array} \right\} T(n) = \Theta(n \lg n)$$

Asymptotic notation in equations

- Asymptotic notation appears alone on RHS of an equation
 - means set membership
 - e.g., $n = O(n^2)$ means $n \in O(n^2)$
- Asymptotic notation appears on RHS of an equation
 - stands for **some** anonymous function in the set
 - e.g., $2n^2 + 3n + 1 = 2n^2 + \Theta(n)$ means that
$$2n^2 + 3n + 1 = 2n^2 + h(n), \text{ for some } h(n) \in \Theta(n)$$
i.e., $h(n) = 3n + 1$

Asymptotic notation appears on LHS of an equation

- stands for **any** anonymous function in the set
 - e.g., $2n^2 + \Theta(n) = \Theta(n^2)$ means that
for **any** function $g(n) \in \Theta(n)$
 \exists **some** function $h(n) \in \Theta(n^2)$
such that $2n^2 + g(n) = h(n), \forall n$
- **RHS** provides **coarser** level of detail than **LHS**

Other asymptotic notations

o-notation

- upper bound provided by **O-notation** may or may not be **tight**
 - e.g., bound $2n^2 = O(n^2)$ is **asymptotically tight**
bound $2n = O(n^2)$ is **not asymptotically tight**
- **o-notation** denotes an **upper bound** that is **not asymptotically tight**

o-notation

- $o(g(n)) = \{f(n): \text{for any constant } c > 0,$
 $\exists \text{ a constant } n_0 > 0$
 such that $0 \leq f(n) < cg(n), \forall n \geq n_0\}$
- Intuitively, $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$
 - e.g., $2n = o(n^2)$, any positive c satisfies
 - but $2n^2 \neq o(n^2)$, $c = 2$ does not satisfy

ω -notation

- denotes a **lower bound** that is **not asymptotically tight**
- $\omega(g(n)) = \{f(n): \text{for any constant } c > 0,$
 $\exists \text{ a constant } n_0 > 0$
such that $0 \leq cg(n) < f(n), \forall n \geq n_0\}$
- Intuitively $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \infty$
 - e.g., $n^2 / 2 = \omega(n)$, any c satisfies
 - **but** $n^2 / 2 \neq \omega(n^2)$, $c=1/2$ does not satisfy

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**: (**$O \leftrightarrow \Omega$** and **$o \leftrightarrow \omega$**)
e.g., $f(n) = O(g(n)) \Leftrightarrow g(n) = \Omega(f(n))$

Analogy to the comparison of two real numbers

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

Analogy to the comparison of two real numbers

- Trichotomy property of real numbers does not hold for asymptotic notation
 - i.e., for any two real numbers a and b ,
we have either $a < b$, or $a = b$, or $a > b$
 - i.e., 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