Asymptotic Notation: O

Use asymptotic notations to describe asymptotic efficiency of algorithms.
(Ignore constant coefficients and lower-order terms.)

Asymptotic Notation: O

Use asymptotic notations to describe asymptotic efficiency of algorithms.
(Ignore constant coefficients and lower-order terms.)

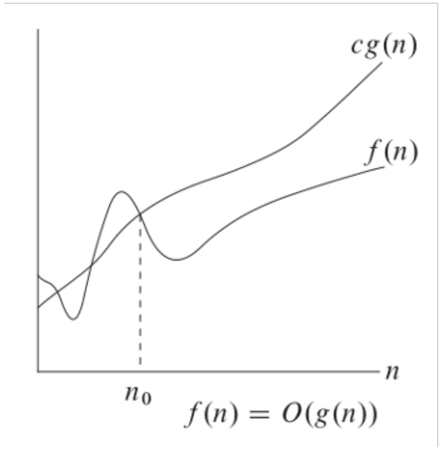
Given a function $g(n)$, we denote by $O(g(n))$ the following **set of functions**:
$\{f(n) : \text{exists } c > 0 \text{ and } n_0 > 0, \text{such that } f(n) \leq c \cdot g(n) \text{ for all } n \geq n_0\}$

Asymptotic Notation: O

Use asymptotic notations to describe asymptotic efficiency of algorithms.
(Ignore constant coefficients and lower-order terms.)

Given a function $g(n)$, we denote by $O(g(n))$ the following **set of functions**:
$\{f(n) : \text{exists } c > 0 \text{ and } n_0 > 0, \text{such that } f(n) \leq c \cdot g(n) \text{ for all } n \geq n_0\}$

$f(n) = O(g(n))$: $f(n)$ is **asymptotically smaller** than $g(n)$.



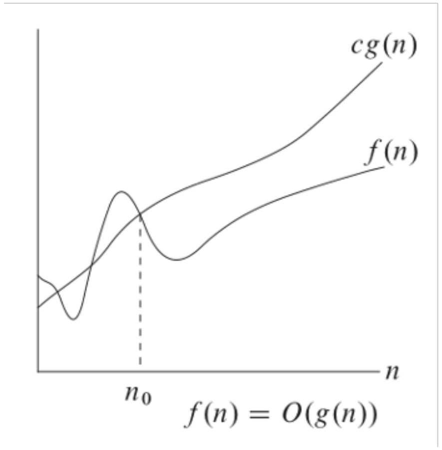
Asymptotic Notation: O

Use asymptotic notations to describe asymptotic efficiency of algorithms.
(Ignore constant coefficients and lower-order terms.)

Given a function $g(n)$, we denote by $O(g(n))$ the following **set of functions**:
$\{f(n) : \text{exists } c > 0 \text{ and } n_0 > 0, \text{such that } f(n) \leq c \cdot g(n) \text{ for all } n \geq n_0\}$

$f(n) = O(g(n))$: ~~$f(n)$ is **asymptotically smaller** than $g(n)$.~~

$f(n) \in O(g(n))$: $f(n)$ is **asymptotically at most** $g(n)$.



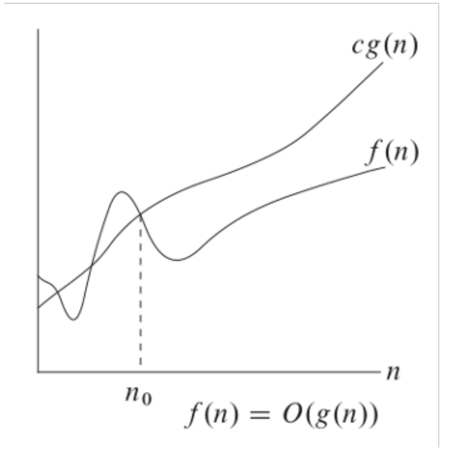
Asymptotic Notation: O

Use asymptotic notations to describe asymptotic efficiency of algorithms.
(Ignore constant coefficients and lower-order terms.)

Given a function $g(n)$, we denote by $O(g(n))$ the following **set of functions**:
$\{f(n) : \text{exists } c > 0 \text{ and } n_0 > 0, \text{such that } f(n) \leq c \cdot g(n) \text{ for all } n \geq n_0\}$

$f(n) = O(g(n))$: ~~$f(n)$ is **asymptotically smaller** than $g(n)$~~.

$f(n) \in O(g(n))$: $f(n)$ is **asymptotically at most** $g(n)$.



Insertion Sort as an $\boxed{\text{example}}$:

- Best Case: $T(n) \approx cn + c'n - c''$
- Worst case: $T(n) \approx cn + (c'/2)n^2 - c''$

Asymptotic Notation: O

Use asymptotic notations to describe asymptotic efficiency of algorithms.
(Ignore constant coefficients and lower-order terms.)

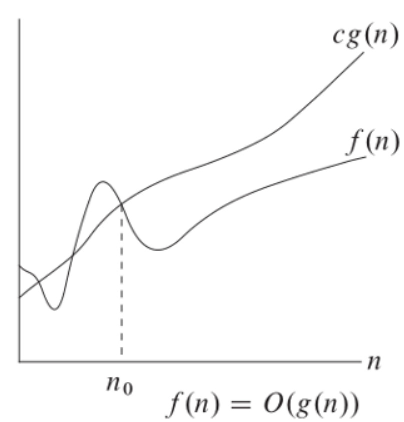
Given a function $g(n)$, we denote by $O(g(n))$ the following **set of functions**:
$\{f(n) : \text{exists } c > 0 \text{ and } n_0 > 0, \text{such that } f(n) \leq c \cdot g(n) \text{ for all } n \geq n_0\}$

$f(n) = O(g(n))$: ~~$f(n)$ is **asymptotically smaller** than $g(n)$~~.

$f(n) \in O(g(n))$: $f(n)$ is **asymptotically at most** $g(n)$.

Insertion Sort as an example:

- Best Case: $T(n) \approx cn + c'n - c''$ $\quad T(n) = O(n)$
- Worst case: $T(n) \approx cn + (c'/2)n^2 - c''$ $\quad T(n) = O(n)$



Asymptotic Notation: O

Use asymptotic notations to describe asymptotic efficiency of algorithms.
(Ignore constant coefficients and lower-order terms.)

Given a function $g(n)$, we denote by $O(g(n))$ the following **set of functions**:
$\{f(n) : \text{exists } c > 0 \text{ and } n_0 > 0, \text{such that } f(n) \leq c \cdot g(n) \text{ for all } n \geq n_0\}$

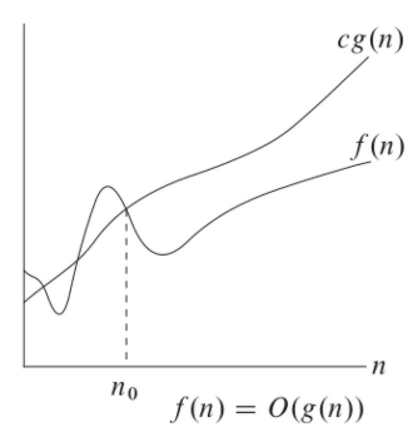
$f(n) = O(g(n))$: ~~$f(n)$ is **asymptotically smaller** than $g(n)$.~~

$f(n) \in O(g(n))$: $f(n)$ is **asymptotically at most** $g(n)$.

Insertion Sort as an example:

- Best Case: $T(n) \approx cn + c'n - c''$ $\quad T(n) = O(n)$
- Worst case: $T(n) \approx cn + (c'/2)n^2 - c''$ $\quad T(n) = O(n)$

Q: Is $n^3 = O(n^2)$? How to prove your answer?



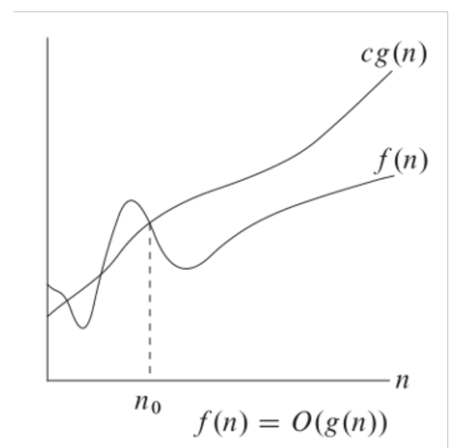
Asymptotic Notation: O

Use asymptotic notations to describe asymptotic efficiency of algorithms.
(Ignore constant coefficients and lower-order terms.)

Given a function $g(n)$, we denote by $O(g(n))$ the following **set of functions**:
$\{f(n) : \text{exists } c > 0 \text{ and } n_0 > 0, \text{such that } f(n) \leq c \cdot g(n) \text{ for all } n \geq n_0\}$

$f(n) = O(g(n))$: ~~$f(n)$ is **asymptotically smaller** than $g(n)$.~~

$f(n) \in O(g(n))$: $f(n)$ is **asymptotically at most** $g(n)$.



Insertion Sort as an $\boxed{\text{example}}$:

- Best Case: $T(n) \approx cn + c'n - c''$ $T(n) = O(n)$
- Worst case: $T(n) \approx cn + (c'/2)n^2 - c''$ $T(n) = O(n)$

Q: Is $n^3 = O(n^2)$? How to prove your answer?

No.

Show that the equation $\left(\frac{3}{2}\right)^n \geq c$
has infinitely many solutions for n.