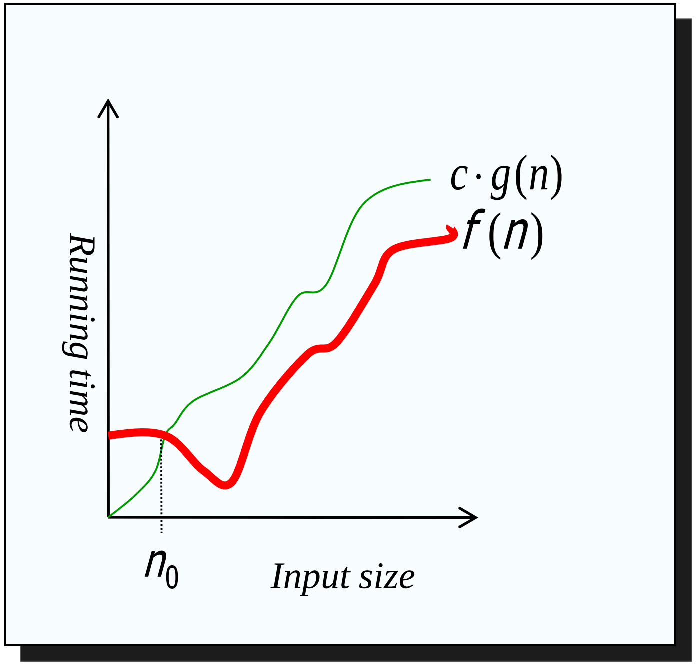
Upper Bound Notation

- InsertionSort's runtime is ***$O(n^2)$***
  - runtime is *in* $O(n^2)$
  - Read O as "Big-O"

Upper Bound Notation

- InsertionSort's runtime is ***O(n²)***
  - runtime is *in* $O(n^2)$
  - Read O as "Big-O"
- In general, a function
  - $f(n)$ is $O(g(n))$ if there exist positive constants c and n_0 such that $f(n) \leq c \cdot g(n)$ for all $n \geq n_0$



Insertion Sort Is $O(n^2)$

- Proof:
  - Use the formal definition of *O* to demonstrate that $an^2 + bn + c = O(n^2)$

$$O(g(n)) = \{f(n) : \text{there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \leq f(n) \leq cg(n) \text{ for all } n \geq n_0\} .$$

Insertion Sort Is $O(n^2)$

- Proof:
  - Use the formal definition of *O* to demonstrate that $an^2 + bn + c = O(n^2)$

$$O(g(n)) = \{f(n) : \text{ there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \le f(n) \le cg(n) \text{ for all } n \ge n_0\}.$$

  - If any of *a*, *b*, and *c* are less than 0 replace the constant with its absolute value

Insertion Sort Is $O(n^2)$

- Proof:
  - Use the formal definition of *O* to demonstrate that $an^2 + bn + c = O(n^2)$

$$O(g(n)) = \{ f(n) : \text{ there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \le f(n) \le cg(n) \text{ for all } n \ge n_0 \} .$$

  - If any of *a*, *b*, and *c* are less than 0 replace the constant with its absolute value
    - o $0 \le f(n) \le k \cdot g(n)$ for all $n \ge n_0$ *(k and n_0 must be positive)*

Insertion Sort Is O(n^2)

- Proof:
  - Use the formal definition of *O* to demonstrate that $an^2 + bn + c = O(n^2)$

$$O(g(n)) = \{ f(n) : \text{there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \le f(n) \le cg(n) \text{ for all } n \ge n_0 \} .$$

  - If any of *a*, *b*, and *c* are less than 0 replace the constant with its absolute value
    - $0 \le f(n) \le k \cdot g(n)$ for all $n \ge n_0$ *(k and n_0 must be positive)*
    - $0 \le an^2 + bn + c <= kn^2$

Insertion Sort Is $O(n^2)$

- Proof:
  - Use the formal definition of *O* to demonstrate that $an^2 + bn + c = O(n^2)$

$$O(g(n)) = \{ f(n) : \text{ there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \le f(n) \le cg(n) \text{ for all } n \ge n_0 \} .$$

  - If any of *a*, *b*, and *c* are less than 0 replace the constant with its absolute value
    - $0 \le f(n) \le k \cdot g(n)$ for all $n \ge n_0$ *(k and n_0 must be positive)*
    - $0 \le an^2 + bn + c <= kn^2$
    - $0 \le a + b/n + c/n^2 <= k$

Insertion Sort Is $O(n^2)$

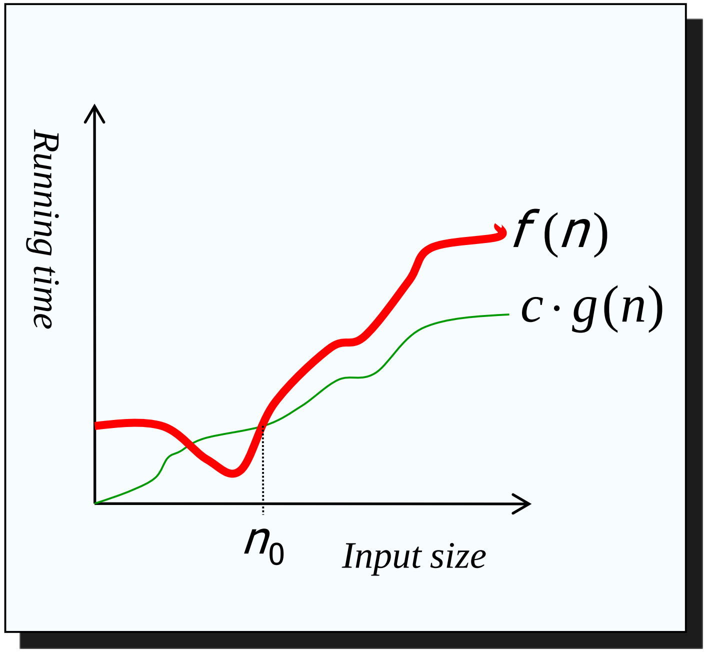
- Proof:
  - Use the formal definition of O to demonstrate that $an^2 + bn + c = O(n^2)$

$$O(g(n)) = \{f(n) : \text{there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \le f(n) \le cg(n) \text{ for all } n \ge n_0\}.$$

  - If any of *a*, *b*, and *c* are less than 0 replace the constant with its absolute value
    - o $0 \le f(n) \le k \cdot g(n)$ for all $n \ge n_0$ *(k and n_0 must be positive)*
    - o $0 \le an^2 + bn + c <= kn^2$
    - o $0 \le a + b/n + c/n^2 <= k$
- Question
  - Is InsertionSort $O(n)$?

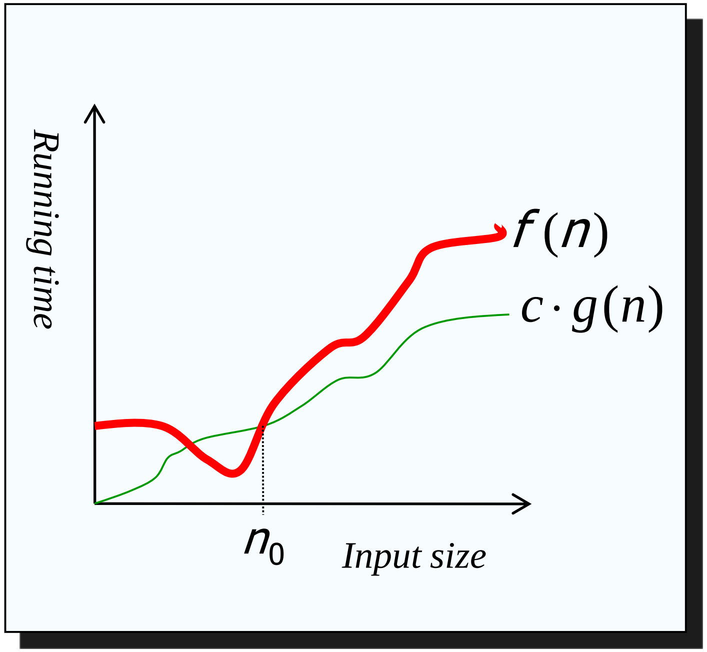
Lower Bound Notation

- InsertionSort's runtime is $\Omega(n)$



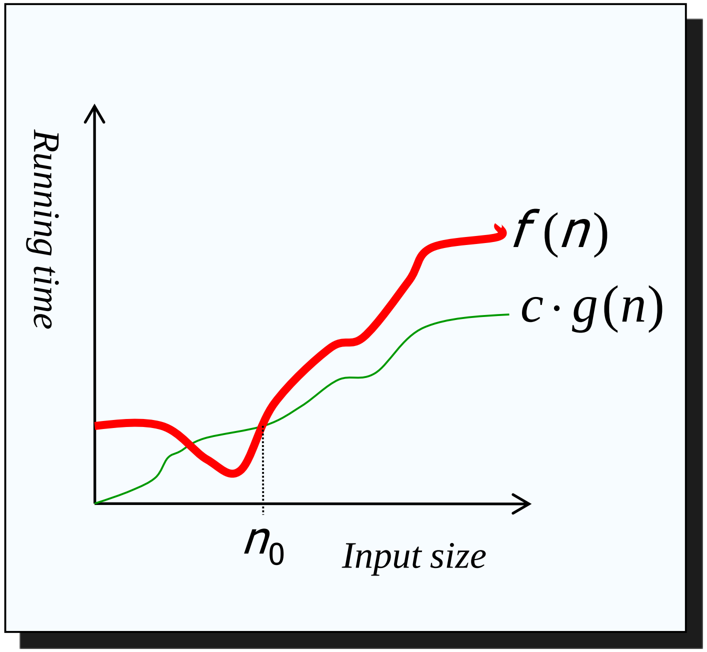
Lower Bound Notation

- InsertionSort's runtime is $\Omega(n)$
- In general, a function
  - $f(n)$ is $\Omega(g(n))$ if there exist positive constants c and n_0 such that $0 \leq c \cdot g(n) \leq f(n) \quad \forall n \geq n_0$



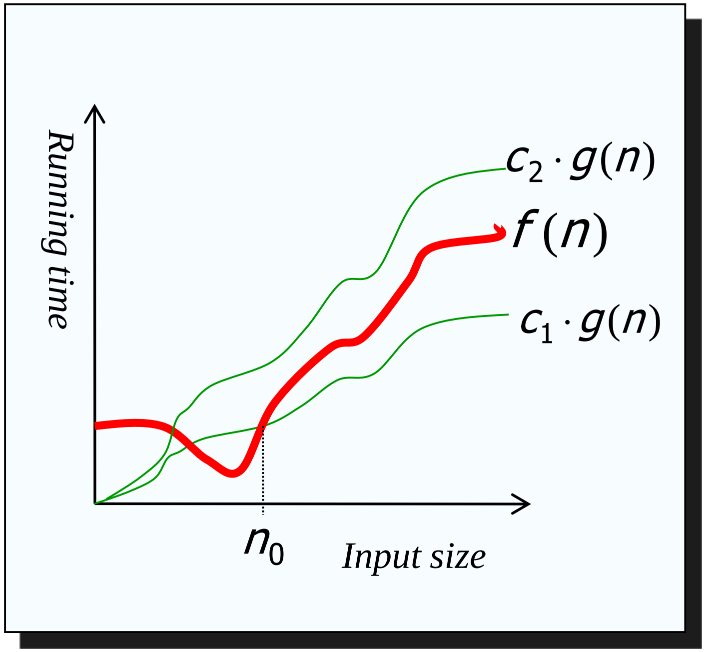
Lower Bound Notation

- InsertionSort's runtime is $\Omega(n)$
- In general, a function
  - $f(n)$ is $\Omega(g(n))$ if there exist positive constants c and n_0 such that $0 \le c \cdot g(n) \le f(n) \;\; \forall\, n \ge n_0$
- Proof:
  - Suppose runtime is an + b
    - o $0 \le cn \le an + b$
    - o $0 \le c \le a + b/n$



Asymptotic Tight Bound

- A function $f(n)$ is $\Theta(g(n))$ if there exist positive constants c_1, c_2, and n_0 such that $0 \le c_1\, g(n) \le f(n) \le c_2\, g(n)$ *for all* $n \ge n_0$
- Theorem
  - $f(n)$ is $\Theta(g(n))$ *iff* f(n) is both $O(g(n))$ and $\Omega(g(n))$



Exercise: Asymptotic Notation

- Use the formal definition of Θ

$$\Theta(g(n)) = \{ f(n) : \text{there exist positive constants } c_1, c_2, \text{ and } n_0 \text{ such that } 0 \le c_1 g(n) \le f(n) \le c_2 g(n) \text{ for all } n \ge n_0 \} .^{1}$$

to demonstrate that $\frac{1}{2}n^2 - 3n = \Theta(n^2)$

[1]Within set notation, a colon means “such that”

Exercise: Asymptotic Notation

- Use the formal definition of Θ

$$\Theta(g(n)) = \{ f(n) : \text{there exist positive constants } c_1, c_2, \text{ and } n_0 \text{ such that } 0 \le c_1 g(n) \le f(n) \le c_2 g(n) \text{ for all } n \ge n_0 \} .^1$$

to demonstrate that $\frac{1}{2}n^2 - 3n = \Theta(n^2)$

Solution:

$0 \le c_1 n^2 \le \frac{1}{2} n^2 - 3n \le c_2 n^2$ for all $n \ge n_0$

Note that c_1 e c_2 must be **positive constants**

[1]Within set notation, a colon means “such that”

Exercise: Asymptotic Notation

- Use the formal definition of Θ

$$\Theta(g(n)) = \{ f(n) : \text{there exist positive constants } c_1, c_2, \text{ and } n_0 \text{ such that } 0 \le c_1 g(n) \le f(n) \le c_2 g(n) \text{ for all } n \ge n_0 \} .^1$$

to demonstrate that $\frac{1}{2}n^2 - 3n = \Theta(n^2)$

Solution:

$0 \leq c_1 n^2 \leq \frac{1}{2} n^2 - 3n \leq c_2 n^2$ for all $n \geq n_0$

Note that c_1 e c_2 must be **positive constants**

$\frac{1}{2} - 3/n \leq c2$

$\boldsymbol{\frac{1}{2} \leq c_2}$

For sufficiently large *n*, the term ½ is kept

[1]Within set notation, a colon means “such that”

Exercise: Asymptotic Notation

- Use the formal definition of Θ

$$\Theta(g(n)) = \{f(n) : \text{there exist positive constants } c_1, c_2, \text{ and } n_0 \text{ such that } 0 \le c_1 g(n) \le f(n) \le c_2 g(n) \text{ for all } n \ge n_0\}\ .^1$$

to demonstrate that $\frac{1}{2}n^2 - 3n = \Theta(n^2)$

Solution:

$0 \le c_1 n^2 \le \frac{1}{2} n^2 - 3n \le c_2 n^2$ for all $n \ge n_0$

Note that c_1 e c_2 must be **positive constants**

$\frac{1}{2} - 3/n \le c2$

$\boldsymbol{\frac{1}{2} \le c_2}$

For sufficiently large *n*, the term ½ is kept

$c1 \le \frac{1}{2} - 3/n$

$\boldsymbol{c_1 \le 1/14 \text{ for } n \ge 7}$

n=7 is the smallest value for c_1 to be a positive constant

[1]Within set notation, a colon means “such that”

Exercise: Asymptotic Notation

- Use the formal definition of Θ

$$\Theta(g(n)) = \{ f(n) : \text{there exist positive constants } c_1, c_2, \text{ and } n_0 \text{ such that } 0 \le c_1 g(n) \le f(n) \le c_2 g(n) \text{ for all } n \ge n_0 \} .^1$$

to demonstrate that $6n^3 \neq \Theta(n^2)$

Exercise: Asymptotic Notation

- Use the formal definition of Θ

$$\Theta(g(n)) = \{ f(n) : \text{there exist positive constants } c_1, c_2, \text{ and } n_0 \text{ such that } 0 \le c_1 g(n) \le f(n) \le c_2 g(n) \text{ for all } n \ge n_0 \}\ .^{1}$$

to demonstrate that $6n^3 \neq \Theta(n^2)$

Solution:

$c_1 n^2 \le 6n^3 \le c_2 n^2$ *for all* $n \ge n_0$

$$6n \le c_2 \therefore n \le \frac{c_2}{6}$$

This can not be true for sufficiently large n
since c_2 must be a constant

Other Asymptotic Notations

- A function $f(n)$ is $o(g(n))$ if for any positive constant $c > 0$, $\exists$ a constant $n_0 > 0$ such that

  $0 \leq f(n) < c\, g(n) \;\forall\, n \geq n_0$

Other Asymptotic Notations

- A function $f(n)$ is $o(g(n))$ if for any positive constant $c > 0$, $\exists$ a constant $n_0 > 0$ such that

  $$0 \leq f(n) < c\, g(n) \;\forall\; n \geq n_0$$

- A function $f(n)$ is $\omega(g(n))$ if for any positive constant $c > 0$, $\exists$ a constant $n_0 > 0$ such that

  $$0 \leq c\, g(n) < f(n) \;\forall\; n \geq n_0$$

Other Asymptotic Notations

- A function $f(n)$ is $o(g(n))$ if for any positive constant $c > 0$, $\exists$ a constant $n_0 > 0$ such that
  $0 \leq f(n) < c\, g(n) \;\forall\; n \geq n_0$
- A function $f(n)$ is $\omega(g(n))$ if for any positive constant $c > 0$, $\exists$ a constant $n_0 > 0$ such that
  $0 \leq c\, g(n) < f(n) \;\forall\; n \geq n_0$
- Intuitively,
  - o() is like <
  - ω() is like >
  - Θ() is like =
  - O() is like $\leq$
  - Ω() is like $\geq$

Asymptotic Comparisons

- We can draw an analogy between the asymptotic comparison of **two functions f and g** and the comparison of **two real numbers a and b**

$$
\begin{array}{lll}
f(n) = O(g(n)) & \text{is like} & a \le b \,, \\
f(n) = \Omega(g(n)) & \text{is like} & a \ge b \,, \\
f(n) = \Theta(g(n)) & \text{is like} & a = b \,, \\
f(n) = o(g(n)) & \text{is like} & a < b \,, \\
f(n) = \omega(g(n)) & \text{is like} & a > b \,.
\end{array}
$$

Asymptotic Comparisons

- We can draw an analogy between the asymptotic comparison of **two functions f and g** and the comparison of **two real numbers a and b**

$$
\begin{aligned}
f(n) &= O(g(n)) && \text{is like} && a \leq b\,, \\
f(n) &= \Omega(g(n)) && \text{is like} && a \geq b\,, \\
f(n) &= \Theta(g(n)) && \text{is like} && a = b\,, \\
f(n) &= o(g(n)) && \text{is like} && a < b\,, \\
f(n) &= \omega(g(n)) && \text{is like} && a > b\,.
\end{aligned}
$$

- Abuse of notation:
  - *f(n) = O(g(n))* indicates that *f(n)* $\in$ *O(g(n))*

Summary

- Analyse the running time used by an algorithm via asymptotic analysis
  - asymptotic (O, Ω, Θ, *o*, ω) notations
  - use a generic uniprocessor random-access machine
  - Time and space complexity (input size)
  - Best, average and worst-case