Asymptotic Notation

CSE 250 Spring 2023

Feb 8 and 10, 2023

Textbook: Ch. 7.3-7.4

When is an algorithm "fast"?

- Real world ("Wall Clock") time?

  Is 10s fast? 100ms? 10μs?
  It depends on the task!

  Do you rank the algorithm or the implementation?
  Compare Grace Hopper's implementation to yours.

  CPU Effects (e.g., ARM RK3399S vs Intel i9 vs AMD 5950)
  Different speed/capability trade-offs

  Bottlenecks: CPU vs IO vs Memory vs Network vs ...

Wall-clock time is not great for a 50k-ft view.

Growth Functions

$$f(n)$$

$\mathbf{n}$: The "size" of the input

e.g., the number of users, rows of data, etc...

$\mathbf{f(n)}$: The number of "steps" taken for an input of size $\mathbf{n}$

e.g., 20 steps per user is $20 \times n$ (with $n = |\texttt{Users}|$)

Growth Function Assumptions

Problem sizes are non-negative integers

$n \in \mathbb{Z}^+ \cup \{0\}$

We can't reverse time

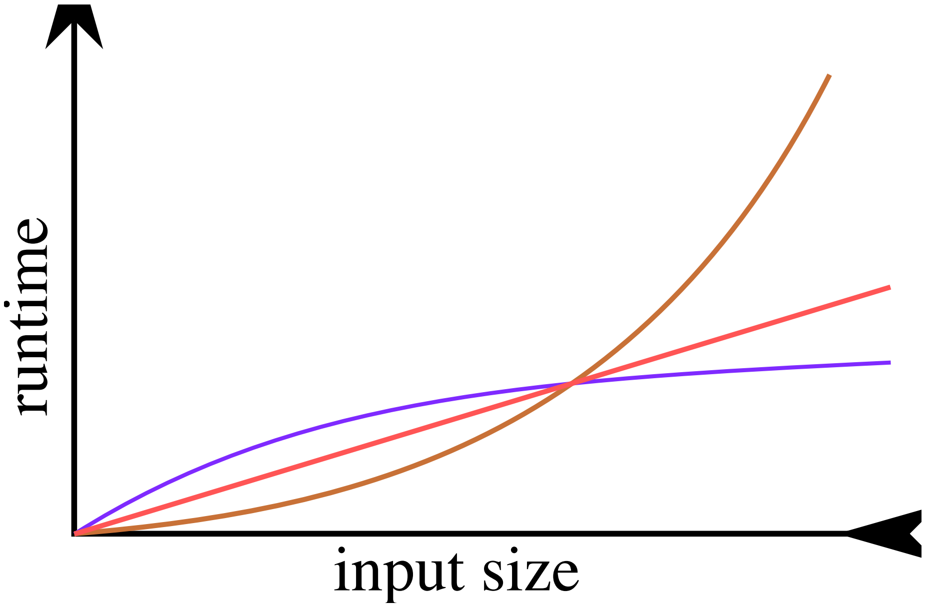
$f(n) \geq 0$

Smaller problems aren't harder than bigger problems

For any $n_1 < n_2, f(n_1) \leq f(n_2)$

To make the math simpler, we'll allow fractional steps.

... but $f_1(n) = 20n \quad \not\equiv \quad f_2(n) = 19n$



runtime
input size

Idea: Organize growth functions into complexity classes.

Asymptotic Analysis @ 5000 feet

Case 1: $lim_{n\to\infty}\frac{f(n)}{g(n)} = \infty$

($f(n)$ is "bigger"; $g(n)$ is the better runtime on larger data)

Case 2: $lim_{n\to\infty}\frac{f(n)}{g(n)} = 0$

($g(n)$ is "bigger"; $f(n)$ is the better runtime on larger data)

Case 3: $lim_{n\to\infty}\frac{f(n)}{g(n)} = \text{some constant}$

($f(n)$, $g(n)$ "behave the same" on larger data)

Big-Theta

The following are all saying the same thing

- $\lim_{n\to\infty} \frac{f(n)}{g(n)}$ = some non-zero constant.
- $f(n)$ and $g(n)$ have the same complexity.
- $f(n)$ and $g(n)$ are in the same complexity class.

Big-Theta

The following are all saying the same thing

- $\lim_{n \to \infty} \frac{f(n)}{g(n)} =$ some non-zero constant.
- f(n) and g(n) have the same complexity.
- f(n) and g(n) are in the same complexity class.
- $f(n) \in \Theta(g(n))$

Big-Theta (As a Limit)

$f(n) \in \Theta(g(n))$ iff...

$$0 < \lim_{n \to \infty} \frac{f(n)}{g(n)} < \infty$$

Big-Theta

$\Theta(g(n))$ is the set of functions in the same complexity class as $g(n)$

Big-Theta

$\Theta(g(n))$ is the set of functions in the same complexity class as $g(n)$

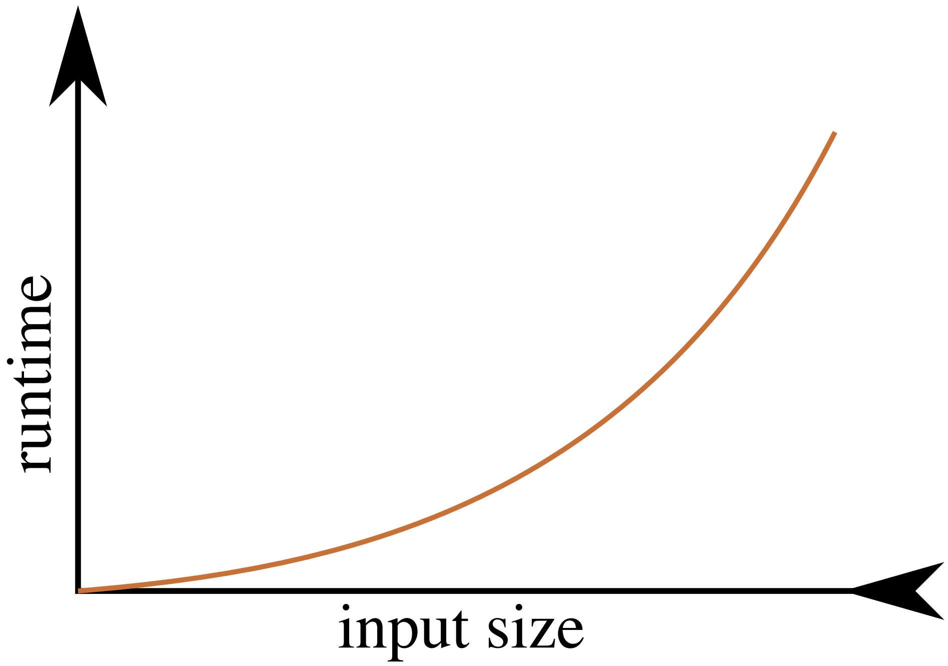
People sometimes write $f(n) = \Theta(g(n))$ when they mean $f(n) \in \Theta(g(n))$

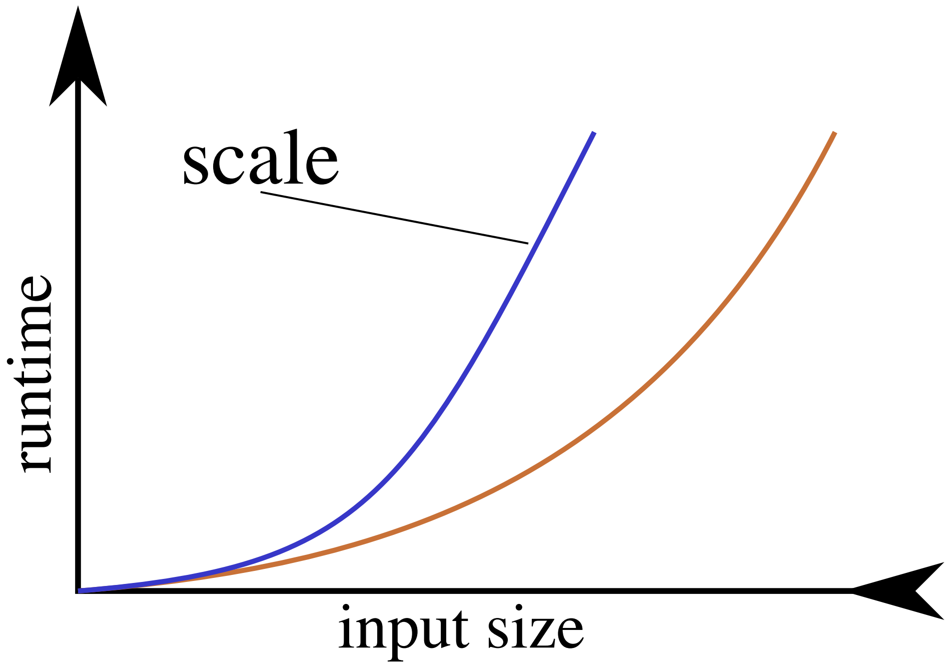
Big-Theta

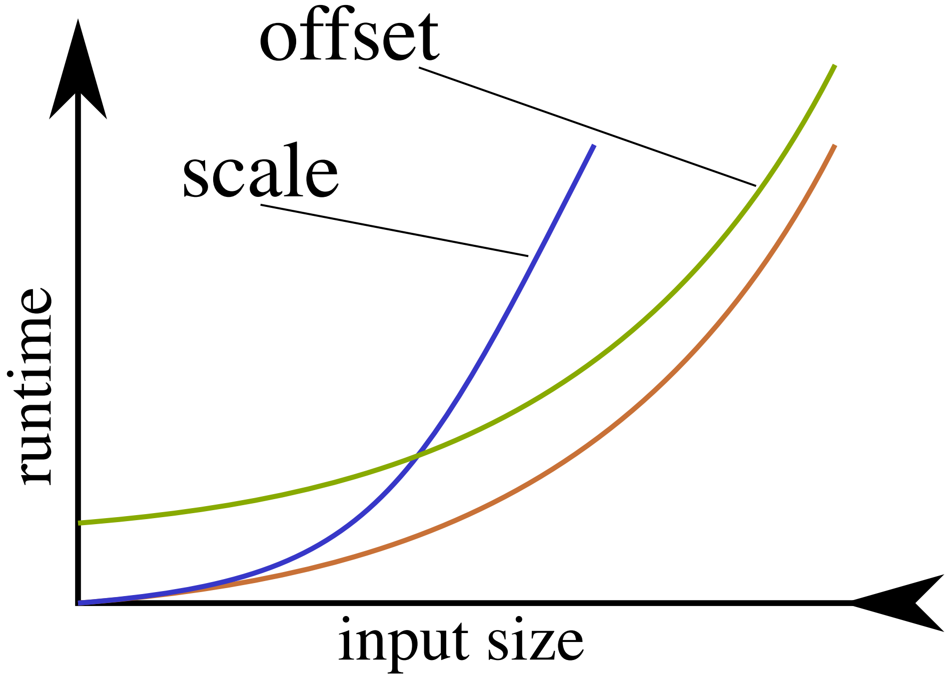
$\Theta(g(n))$ is the set of functions in the same complexity class as $g(n)$

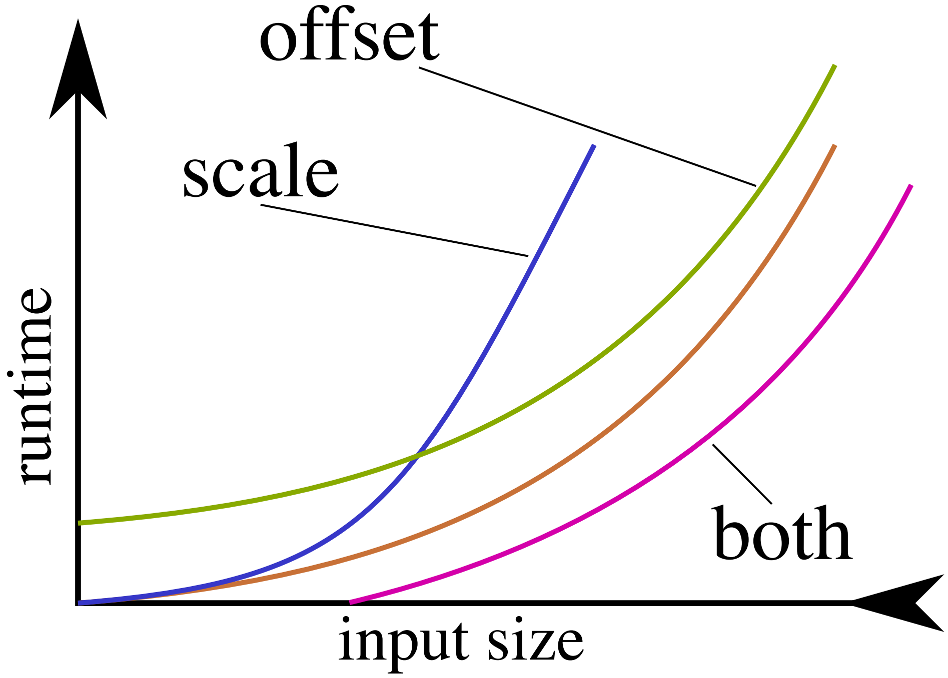
People sometimes write $f(n) = \Theta(g(n))$ when they mean $f(n) \in \Theta(g(n))$

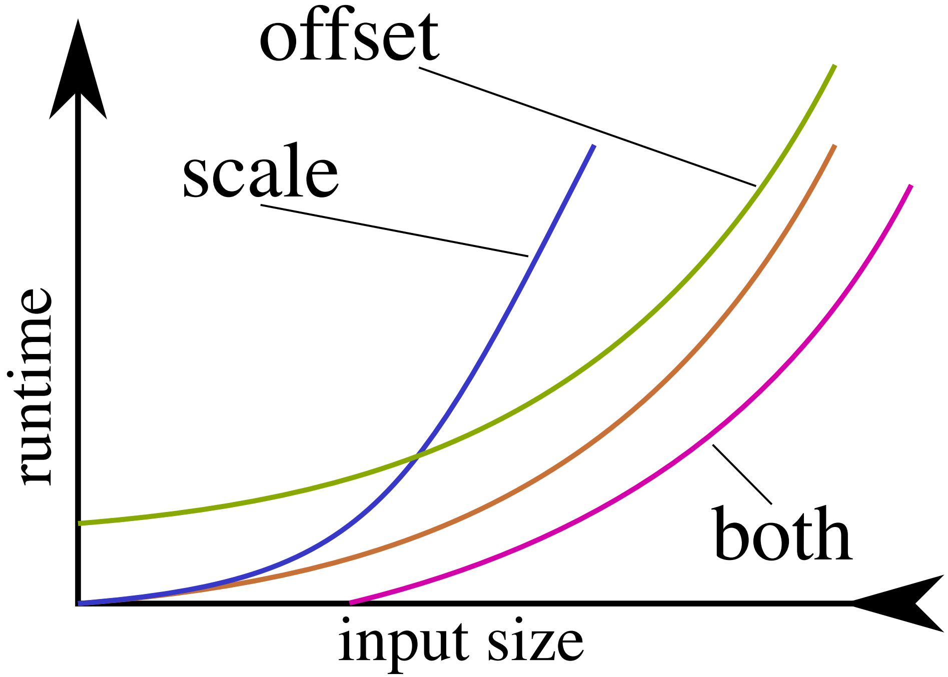
Symmetric: $f(n) \in \Theta(g(n))$ is the same as $g(n) \in \Theta(f(n))$


runtime
input size


scale
runtime
input size

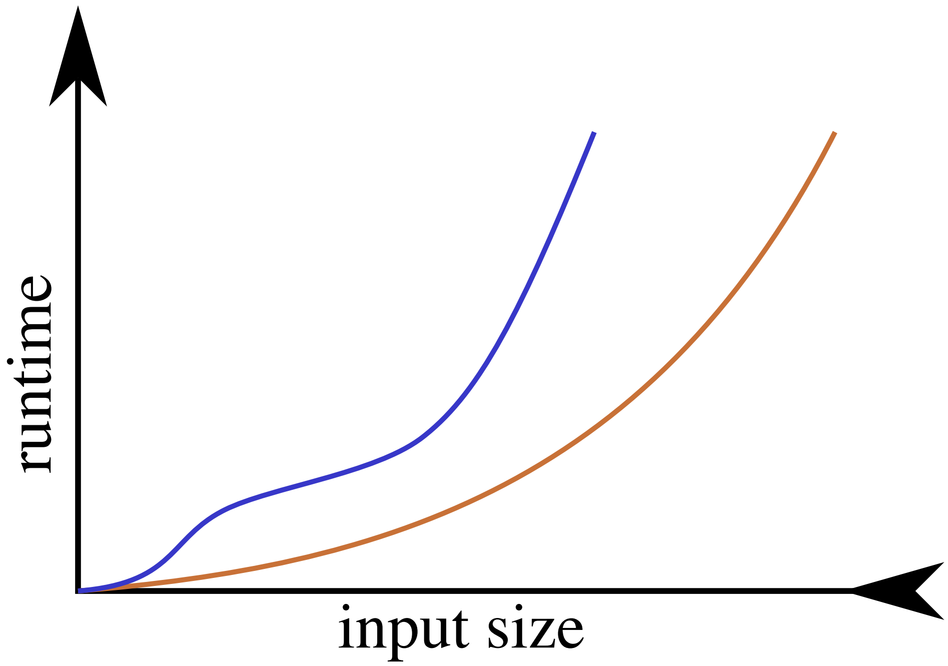

offset
scale
runtime
input size


offset
scale
runtime
both
input size



all in the same complexity cla

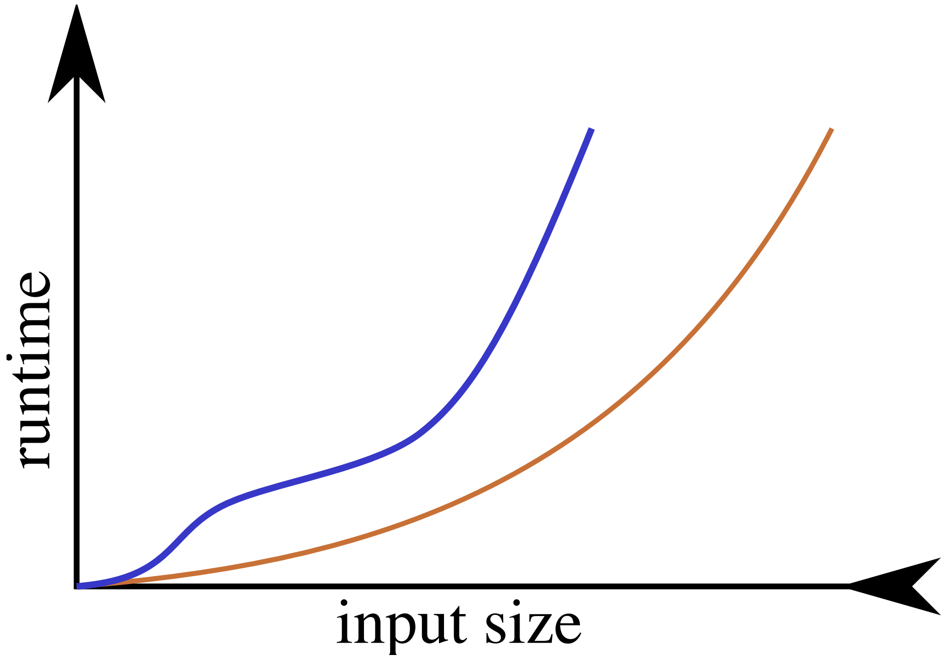
If you can shift/stretch $g(n)$ into $f(n)$, they're in the same class.

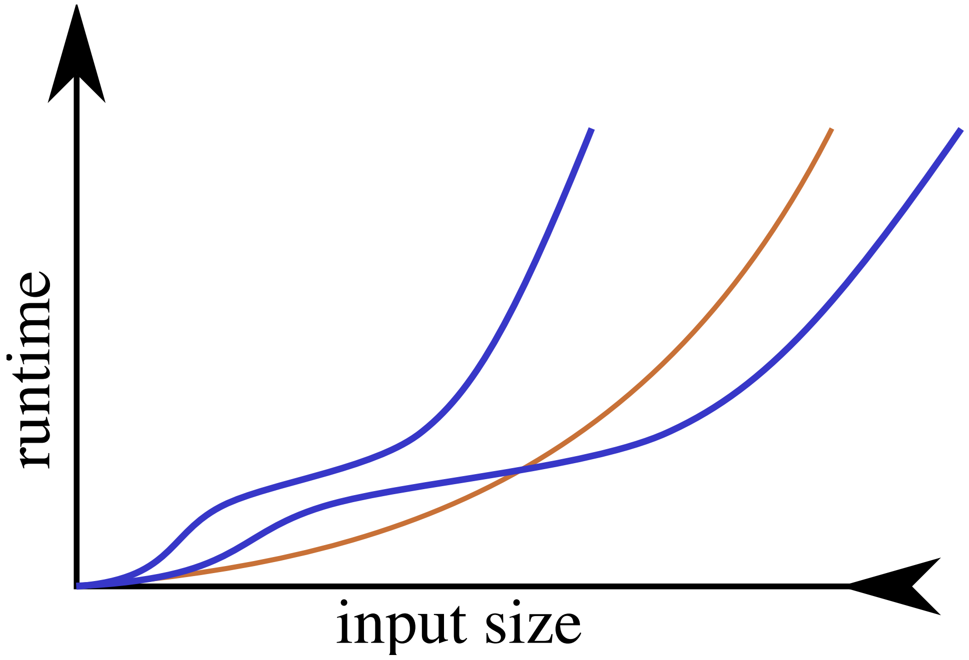

runtime
input size

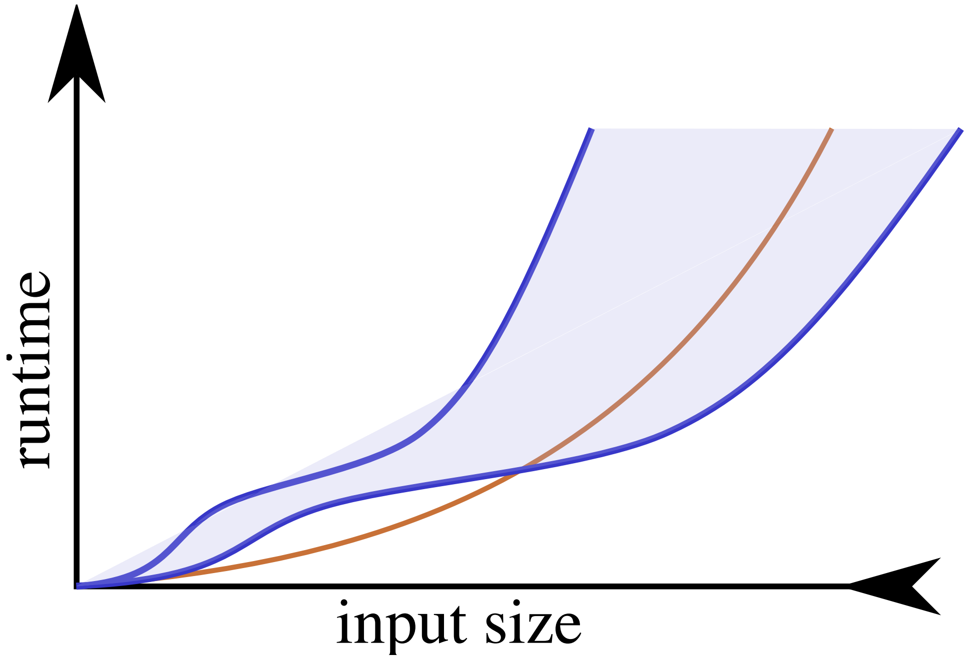
... Instead, think of $g(n)$ as a bound.

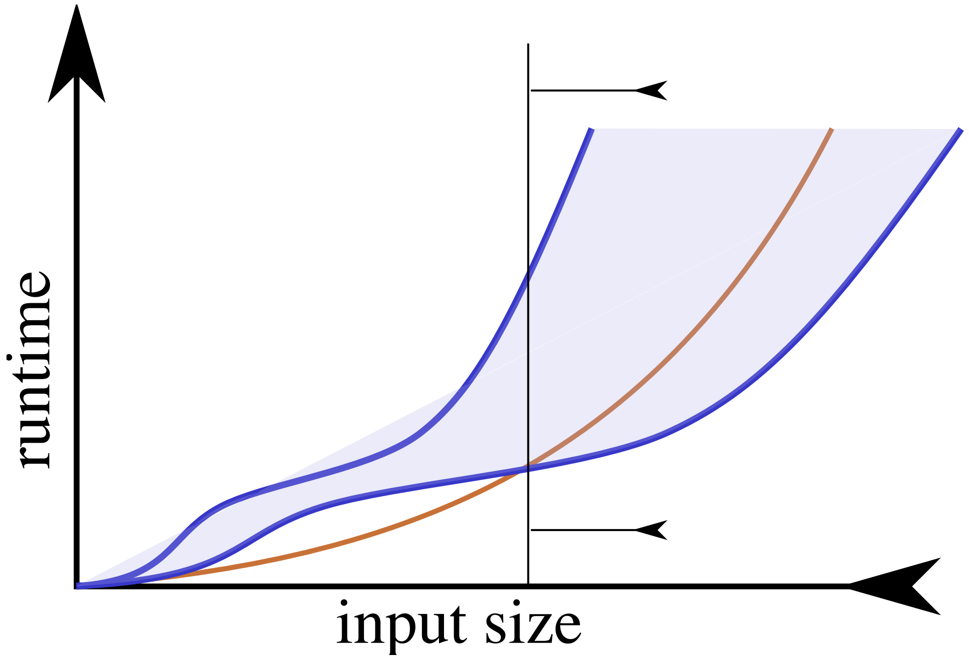
... Instead, think of g(n) as a bound.

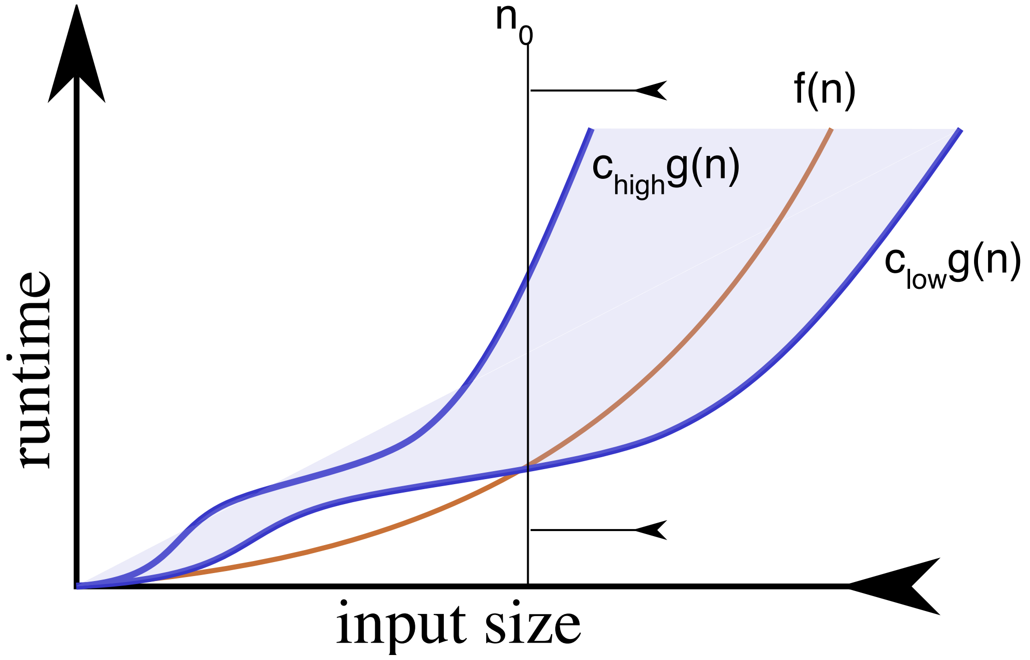
Can you bound f(n) by shift/stretching g(n)?


runtime
input size


runtime
input size


runtime
input size


runtime
input size



n0
f(n)
chighg(n)
clowg(n)
runtime
input size

Big-Theta

The following are all saying the same thing

- $\lim_{n\to\infty} \frac{f(n)}{g(n)}$ = some non-zero constant.
- $f(n)$ and $g(n)$ have the same complexity.
- $f(n)$ and $g(n)$ are in the same complexity class.
- $f(n) \in \Theta(g(n))$
- $f(n)$ is bounded from above and below by $g(n)$

Big-Theta (As a Bound)

$f(n) \in \Theta(g(n))$ iff...

$$\exists c_{low}, n_0 \text{ s.t. } \forall n > n_0, f(n) \geq c_{low} \cdot g(n)$$

$$\exists c_{high}, n_0 \text{ s.t. } \forall n > n_0, f(n) \leq c_{high} \cdot g(n)$$

Big-Theta (As a Bound)

$f(n) \in \Theta(g(n))$ iff...

$\exists c_{low}, n_0$ s.t. $\forall n > n_0$, $f(n) \geq c_{low} \cdot g(n)$

There is some c_{low} that we can multiply $g(n)$ by so that $f(n)$ is always bigger than $c_{low} g(n)$ for values of n above some n_0

$\exists c_{high}, n_0$ s.t. $\forall n > n_0$, $f(n) \leq c_{high} \cdot g(n)$

Big-Theta (As a Bound)

$f(n) \in \Theta(g(n))$ iff...

$\exists c_{low}, n_0$ s.t. $\forall n > n_0$, $f(n) \geq c_{low} \cdot g(n)$

There is some c_{low} that we can multiply $g(n)$ by so that $f(n)$ is <u>always</u> bigger than $c_{low}g(n)$ for values of n above some n_0

$\exists c_{high}, n_0$ s.t. $\forall n > n_0$, $f(n) \leq c_{high} \cdot g(n)$

There is some c_{high} that we can multiply $g(n)$ by so that $f(n)$ is <u>always</u> smaller than $c_{high}g(n)$ for values of n above some n_0

Proving Big-Theta (Without Limits)

1. Assume $f(n) \geq c_{low}g(n)$.
2. Rewrite the above formula to find a c_{low} for which it holds (for big enough n).
3. Assume $f(n) \leq c_{high}g(n)$.
4. Rewrite the above formula to find a c_{high} for which it holds (for big enough n).

Tricks

If $f(n) \geq g'(n)$ and $g'(n) \geq g(n)$ then $f(n) \geq g'(n)$

Tricks

If $f(n) \geq g'(n)$ and $g'(n) \geq g(n)$ then $f(n) \geq g'(n)$

Lesson: To show $f(n) \geq cg(n)$, you can instead show:

Tricks

If $f(n) \geq g'(n)$ and $g'(n) \geq g(n)$ then $f(n) \geq g'(n)$

Lesson: To show $f(n) \geq cg(n)$, you can instead show:

1. $f(n) \geq cg'(n)$

Tricks

If $f(n) \geq g'(n)$ and $g'(n) \geq g(n)$ then $f(n) \geq g'(n)$

Lesson: To show $f(n) \geq cg(n)$, you can instead show:

1. $f(n) \geq cg'(n)$
2. $cg'(n) \geq cg(n)$

Tricks

If $f(n) \geq g(n)$ and $f'(n) \geq g'(n)$ then

$$f(n) + f'(n) \geq g(n) + g'(n)$$

Tricks

If $f(n) \geq g(n)$ and $f'(n) \geq g'(n)$ then $f(n) + f'(n) \geq g(n) + g'(n)$

Lesson: To show $f(n) + f'(n) \geq cg(n) + c'g'(n)$, you can instead show:

Tricks

If $f(n) \geq g(n)$ and $f'(n) \geq g'(n)$ then $f(n) + f'(n) \geq g(n) + g'(n)$

Lesson: To show $f(n) + f'(n) \geq cg(n) + c'g'(n)$, you can instead show:

1. $f(n) \geq cg(n)$

Tricks

If $f(n) \geq g(n)$ and $f'(n) \geq g'(n)$ then $f(n) + f'(n) \geq g(n) + g'(n)$

Lesson: To show $f(n) + f'(n) \geq cg(n) + c'g'(n)$, you can instead show:

1. $f(n) \geq cg(n)$
2. $f'(n) \geq c'g'(n)$

Tricks

- $\log(n) \geq c$ (for any $n \geq 2^c$)
- $n \geq \log(n)$ for any $n \geq 0$
- $n^2 \geq n$ for any $n \geq 1$
- $2^n \geq n^c$ for sufficiently large n

Examples

$$2^n + 4n \in \Theta(n^2)?$$

Examples

$$2^n + 4n \in \Theta(n^2)\,?$$

$$2^n + 4n \in \Theta(n)\,?$$

Examples

$$2^n + 4n \in \Theta(n^2)\,?$$

$$2^n + 4n \in \Theta(n)\,?$$

$$1000n\log(n) + 5n \in \Theta(n\log(n))\,?$$

Shortcut: Find the dominant term being summed, and remove constants.

Asymptotic Runtime

We write $T(n)$ to mean a runtime growth function.

In data structures, n is usually the number of elements in a collection.

Examples

What is the asymptotic runtime of...

Examples

What is the asymptotic runtime of...

- ...find x in a Linked List?

Examples

What is the asymptotic runtime of...

- ...find x in a Linked List?
- ...counting the number of times x appears in a Linked List?

Examples

What is the asymptotic runtime of...

- ...find x in a Linked List?
- ...counting the number of times x appears in a Linked List?
- ...using multiplication to compute Factorial?

Common Runtimes

Constant Time: $\Theta(1)$

e.g., $T(n) = c$ (runtime is independent of n)

Logarithmic Time: $\Theta(\log(n))$

e.g., $T(n) = c\log(n)$ (for some constant c)

Linear Time: $\Theta(n)$

e.g., $T(n) = c_1 n + c_0$ (for some constants c_0, c_1)

Quadratic Time: $\Theta(n^2)$

e.g., $T(n) = c_2 n^2 + c_1 n + c_0$

Polynomial Time: $\Theta(n^k)$ (for some $k \in \mathbb{Z}^+$)

e.g., $T(n) = c_k n^k + \ldots + c_1 n + c_0$

Exponential Time: $\Theta(c^n)$ (for some $c \geq 1$)

Big-O, Big-Ω

What is the asymptotic runtime of...

- ...looking up an element in an Array?

What is the asymptotic runtime of...

- ...looking up an element in an Array?

The runtime depends on where the item is in the list.

```
for(i <- 0 until data.size)
{
  if( data(i) == target ){ return i }
}
return NOT_FOUND
```

```
for(i <- 0 until data.size)
{
  if( data(i) == target ){ return i }
}
return NOT_FOUND
```

What is the runtime growth function?

$$
T(n) = \begin{cases} \ell & \textbf{if } \mathit{data}(0) == \mathit{target} \\ 2\ell & \textbf{if } \mathit{data}(1) == \mathit{target} \\ 3\ell & \textbf{if } \mathit{data}(2) == \mathit{target} \\ \ldots & \ldots \\ (n-1)\ell & \textbf{if } \mathit{data}(n-1) == \mathit{target} \\ n\ell & \textbf{otherwise} \end{cases}
$$

Aside: No general, meaningful notion of limit for $T(n)$s like this.

$$T(n) \in \Theta(n) \text{ ?}$$

If we choose $c = \ell$, we can show $T(n) \leq c \cdot n$ (for any n)

$T(n) \in \Theta(n)$?

If we choose $c = \ell$, we can show $T(n) \leq c \cdot n$ (for any n)

... but there is no c s.t. $T(n) \geq c \cdot n$ always!

$T(n) \in \Theta(n)$?

If we choose $c = \ell$, we can show $T(n) \leq c \cdot n$ (for any n)

... but there is no c s.t. $T(n) \geq c \cdot n$ always!

... $T(1000000)$ could be as small as ℓ,
so $T(1000000) \not\geq 1000000\ell$

$$T(n) \in \Theta(1) \text{ ?}$$

If we choose $c = \ell$, we can show $T(n) \geq c \cdot 1$ (for any n)

$T(n) \in \Theta(1)$?

If we choose $c = \ell$, we can show $T(n) \geq c \cdot 1$ (for any n)

... but there is no c s.t. $T(n) \leq c \cdot 1$ always!

$T(n) \in \Theta(1)$?

If we choose $c = \ell$, we can show $T(n) \geq c \cdot 1$ (for any n)

... but there is no c s.t. $T(n) \leq c \cdot 1$ always!

... $T(1000000)$ could be as big as 1000000ℓ,
so $T(1000000) \not\leq \ell$

Problem: What if $g(n)$ doesn't bound $f(n)$ from **both** above and below?

```
if input = 1:
  /* take 1 step */
else:
  /* take n steps */
```

```
if input = 1:
  /* take 1 step */
else:
  /* take n steps */
```

Schroedinger's Code: Simultaneously behaves like $f_1(n) = 1$ and $f_2(n) = n$ (can't tell until runtime)

Upper, Lower Bounds

"Worst-Case Complexity"

$O(g(n))$ is the set of functions that are in $g(n)$'s complexity class, or a "smaller" class.

"Best-Case Complexity"

$\Omega(g(n))$ is the set of functions that are in $g(n)$'s complexity class, or a "bigger" class.

Big-O

$f(n) \in O(g(n))$ iff...

$\exists c_{high}, n_0$ s.t. $\forall n > n_0, f(n) \leq c_{high} \cdot g(n)$

Big-O

$f(n) \in O(g(n))$ iff...

$\exists c_{high}, n_0$ s.t. $\forall n > n_0$, $f(n) \leq c_{high} \cdot g(n)$

There is some c_{high} that we can multiply $g(n)$ by so that $f(n)$ is <u>always</u> smaller than $c_{high}g(n)$ for values of n above some n_0

Examples

$$2^n + 4n \in O(n^2)?$$

Examples

$$2^n + 4n \in O(n^2)\ ?$$

$$2^n + 4n \in O(n^4 + 8n^3)\ ?$$

Examples

$$2^n + 4n \in O(n^2)\ ?$$

$$2^n + 4n \in O(n^4 + 8n^3)\ ?$$

$$n\log(n) + 5n \in O(n^2 + 5n)\ ?$$

Big-Ω

$f(n) \in \Omega(g(n))$ iff...

$\exists c_{low}, n_0$ s.t. $\forall n > n_0, f(n) \geq c_{low} \cdot g(n)$

Big-Ω

$f(n) \in \Omega(g(n))$ iff...

$\exists c_{low}, n_0$ s.t. $\forall n > n_0, f(n) \geq c_{low} \cdot g(n)$

There is some c_{low} that we can multiply $g(n)$ by so that $f(n)$ is <u>always</u> smaller than $c_{low}g(n)$ for values of n above some n_0

Examples

$$2^n + 4n \in \Omega(n^2 + 5)?$$

Examples

$$2^n + 4n \in \Omega(n^2 + 5)\,?$$

$$2^n + 4n \in \Omega(\log(n))\,?$$

Examples

$$2^n + 4n \in \Omega(n^2 + 5)\ ?$$

$$2^n + 4n \in \Omega(\log(n))\ ?$$

$$n\log(n) + 5n \in \Omega(n\log(n))\ ?$$

Recap

Big-O: "Worst Case" bound
$O(g(n))$ is the functions that $g(n)$ bounds from above

Big-Ω: "Best Case" bound
$\Omega(g(n))$ is the functions that $g(n)$ bounds from below

Big-Θ: "Tight" bound
$\Theta(g(n))$ is the functions that $g(n)$ bounds from **both** above and below

Recap

Big-O: "Worst Case" bound
$O(g(n))$ is the functions that $g(n)$ bounds from above

Big-Ω: "Best Case" bound
$\Omega(g(n))$ is the functions that $g(n)$ bounds from below

Big-Θ: "Tight" bound
$\Theta(g(n))$ is the functions that $g(n)$ bounds from **both** above and below

$$f(n) \in \Theta(g(n)) \quad \leftrightarrow \quad f(n) \in O(g(n)) \text{ and } f(n) \in \Omega(g(n))$$

Recap

Big-O: "Worst Case" bound
If $T(n) \in O(g(n))$, then the runtime is no worse than $g(n)$

Big-Ω: "Best Case" bound
If $T(n) \in \Omega(g(n))$, then the runtime is no better than $g(n)$

Big-ϴ: "Tight" bound
If $T(n) \in \Theta(g(n))$, then the runtime is always $g(n)$

$$f(n) \in \Theta(g(n)) \quad \leftrightarrow \quad f(n) \in O(g(n)) \text{ and } f(n) \in \Omega(g(n))$$