Formal Definitions of
Big-Oh, Big-Omega, and Big-Theta

(i.e., get ready to do lots of math)

Slides by **Sean Szumlanski**
for **COP 3503**, Computer Science II

Fall 2021

Big-Oh (Lite)

Back in CS1, when faced with a runtime like this:

$$T(n) = \frac{1}{8}n^3 + \frac{1}{2}n^2 + 46$$

…we learned to derive Big-Oh by doing the following:

1. Look for the highest-order term
2. Drop any constants

So, you all have no trouble telling me that **T(n)** is **O(n^3)**.

(By the way, you could derive a runtime like T(n) using summations!)

Big-Oh

(The Bitter Truth)

It turns out there's a formal, mathematical definition for Big-Oh:

f(n) is **O(g(n))** *iff* $f(n) \le c_1 \cdot g(n)$ for $n \ge N_0$

…where c_1 and N_0 are constants. Here's what those constants mean:

N_0 <u>At some point</u>, although maybe not right away, $c_1 \cdot g(n)$ meets or exceeds f(n)

c_1 We can <u>multiply</u> g(n) by a constant to make it meet or exceed f(n). (Note: We require $c_1 > 0$.)

Big-Oh

(The Bitter Truth)

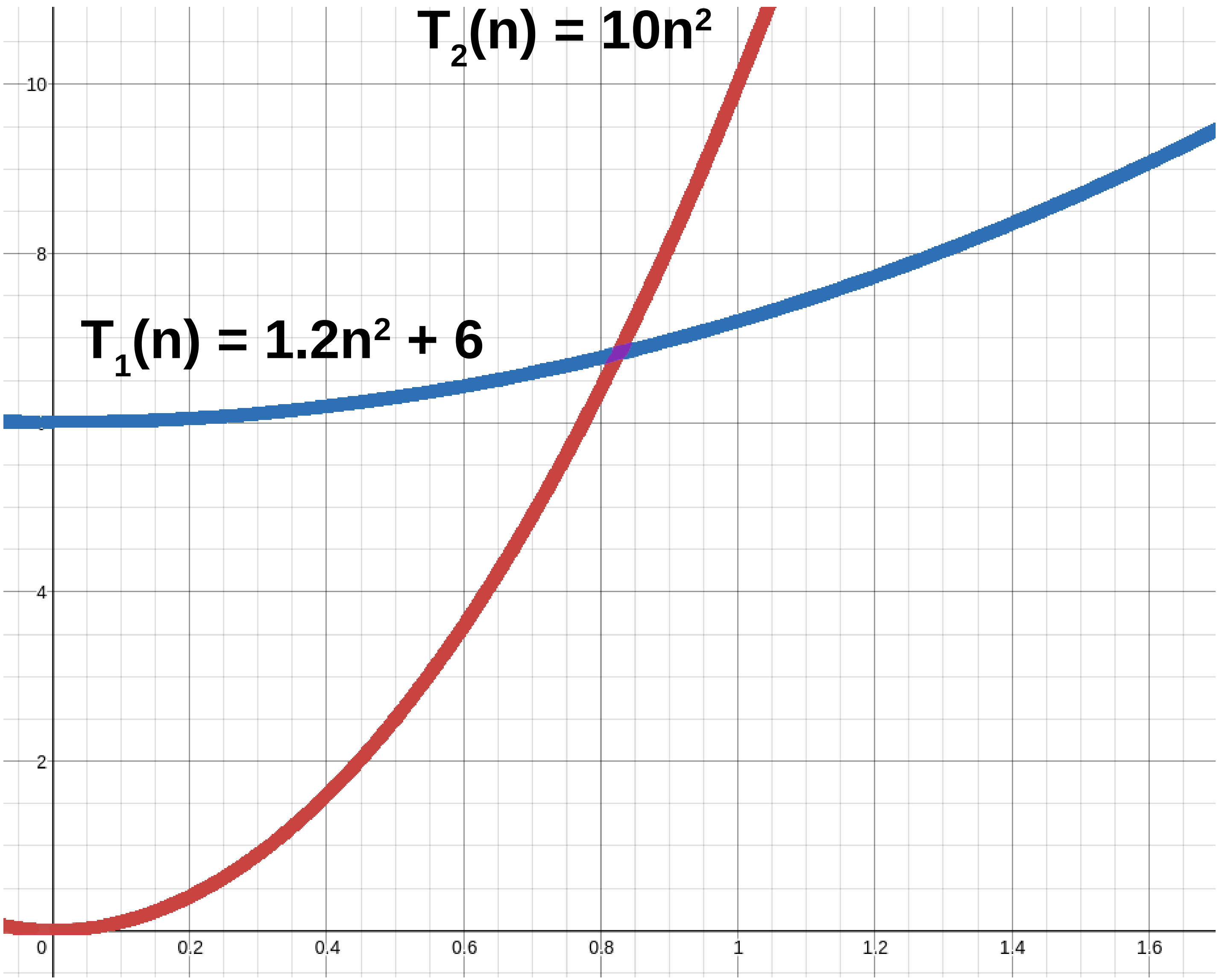
$f(n)$ is **$O(g(n))$** iff **$f(n) \le c_1 \cdot g(n)$** for **$n \ge N_0$**

Let's examine this idea **graphically**.

Big-Oh

(The Bitter Truth)

f(n) is **O(g(n))** iff **f(n) ≤ c_1·g(n)** for **n ≥ N_0**



Initially:

T_1(n) > T_2(n)

Eventually:

T_2(n) dwarfs **T_1(n)**

(...and ever after)

The **upper bound** on the **T_1(n)** curve is formed by a **multiple** of n^2.

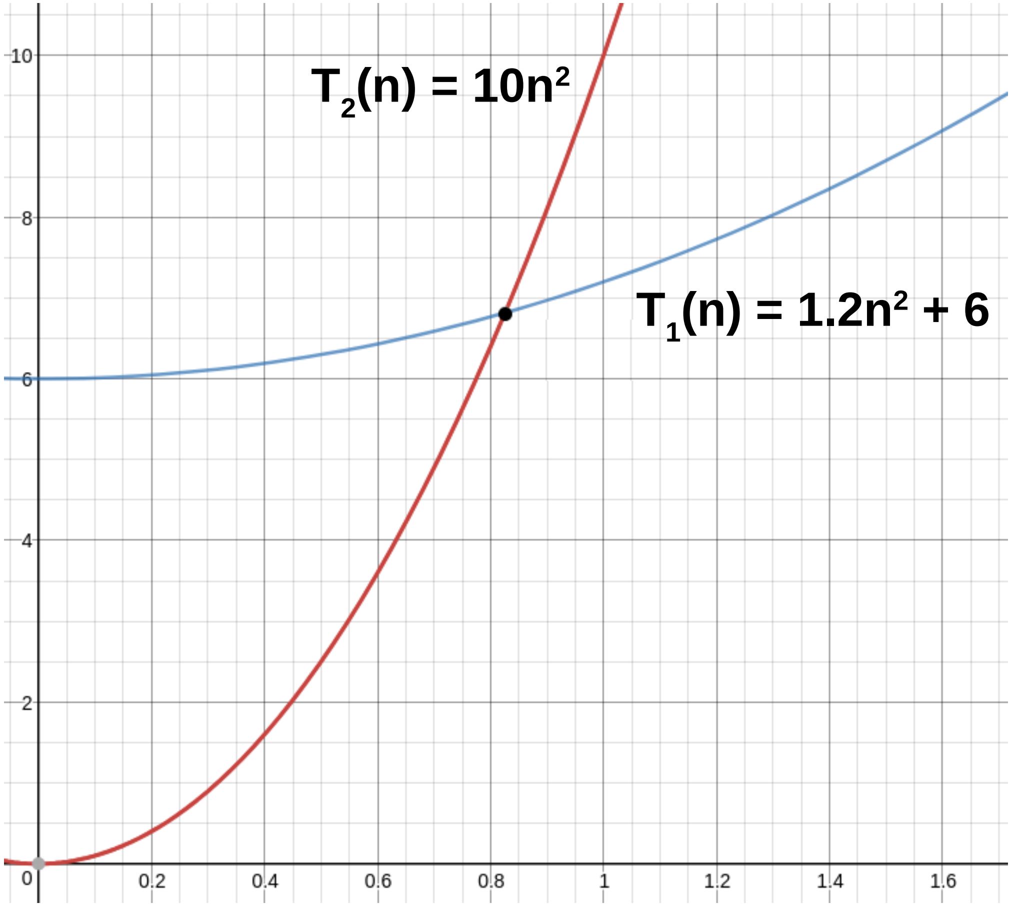
T_1(n) is **O(n^2)**.

Big-Oh

(The Bitter Truth)

f(n) is **O(g(n))** iff **f(n) ≤ c_1·g(n)** for **n ≥ N_0**

In a bit more detail for when you read through this at home:



Here, we see graphically what the definition of Big-Oh intends to capture.

Initially, $T_1(n) > T_2(n)$, but there comes a point where $T_2(n)$ dwarfs $T_1(n)$. That's the idea behind N_0; the inequality holds – not always, but rather, **after some point**.

Also, we see that there's some multiple of n^2 that forms an upper bound on the $T_1(n)$ curve. Notice that $T_2(n)$ is simply some constant (10) times our Big-Oh (n^2).

So, it looks like $T_1(n)$ is $O(n^2)$.

Big-Oh

(The Bitter Truth)

f(n) is **O(g(n))** iff **$f(n) \leq c_1 \cdot g(n)$** for **$n \geq N_0$**

If it's true that **$1.2n^2 + 6$ is $O(n^2)$**,

we should be able to **prove it mathematically**.

Big-Oh

(The Bitter Truth)

$f(n)$ is **$O(g(n))$** iff **$f(n) \le c_1 \cdot g(n)$** for **$n \ge N_0$**

Example: Let $f(n) = 1.2n^2 + 6$. Prove $f(n)$ is $O(n^2)$.

We must find c_1 and N_0 such that $f(n) \le c_1 \cdot n^2$ for $n \ge N_0$

Proof: $f(n) = 1.2n^2 + 6 \le 1.2n^2 + 6n^2 = 7.2n^2$ (for $n \ge 1$)

^ (this inequality holds because $6n^2 \ge 6$ when $n \ge 1$)

That's it! I showed $f(n) \le c_1 \cdot g(n)$ for $n \ge N_0$. (**c_1** $= 7.2$, **$g(n)$** $= n^2$, and **N_0** $= 1$)

Notice that I needed some **constant** times **n^2** on the right-hands side, so I established an inequality in which all the **lower-order terms were converted to n^2 terms**. That's generally how we'll approach these problems.

Big-Oh

(The Bitter Truth)

$f(n)$ is **$O(g(n))$** iff **$f(n) \leq c_1 \cdot g(n)$** for **$n \geq N_0$**

If it’s true that **$3n^2 + 4$ is $O(n^2)$**,

we should be able to **prove it mathematically**.

Big-Oh

(The Bitter Truth)

$f(n)$ is **$O(g(n))$** iff **$f(n) \leq c_1 \cdot g(n)$** for **$n \geq N_0$**

Example: Let $f(n) = 3n^2 + 4$. Prove $f(n)$ is $O(n^2)$.

We must find c_1 and N_0 such that $f(n) \leq c_1 \cdot n^2$ for $n \geq N_0$

Proof: $f(n) = 3n^2 + 4 \leq 3n^2 + 4n^2 = 7n^2$ (for $n \geq 1$)

^ (this inequality holds because $4n^2 \geq 4$ when $n \geq 1$)

That's it! I showed $f(n) \leq c_1 \cdot g(n)$ for $n \geq N_0$. (**c_1** $= 7$, **$g(n)$** $= n^2$, and **N_0** $= 1$)

Notice that I needed some **constant** times **n^2** on the right-hands side, so I established an inequality in which all the **lower-order terms were converted to n^2 terms**. That's generally how we'll approach these problems.

Big-Oh

(The Bitter Truth)

$f(n)$ is **$O(g(n))$** iff **$f(n) \leq c_1 \cdot g(n)$** for **$n \geq N_0$**

Let's prove that **$3n^2 + 10$ is $O(n^2)$**,

using a slight twist on this approach.

Big-Oh

(The Bitter Truth)

$f(n)$ is **$O(g(n))$** iff **$f(n) \le c_1 \cdot g(n)$** for **$n \ge N_0$**

Another example: Let $f(n) = 3n^2 + 10$. Prove $f(n)$ is $O(n^2)$.

We must find c_1 and N_0 such that $f(n) \le c_1 \cdot n^2$ for $n \ge N_0$

Proof: $f(n) = 3n^2 + 10 \le 3n^2 + n^2 = 4n^2$ (for $n \ge \sqrt{10}$)

^ (this inequality holds because $n^2 \ge 10$ when $n \ge \sqrt{10}$)

That's it! I showed $f(n) \le c_1 \cdot g(n)$ for $n \ge N_0$. (**c_1** $= 4$, **$g(n)$** $= n^2$, and **N_0** $= \sqrt{10}$)

Notice that I approached this a bit differently from the previous slide. Instead of ***multiplying*** an existing constant by n^2, I ***replaced*** a constant with n^2! So, there are different ways to lock down values for these constants!

Big-Oh

(The Bitter Truth)

$f(n)$ is **$O(g(n))$** iff **$f(n) \leq c_1 \cdot g(n)$** for **$n \geq N_0$**

Let's prove that **$3n^2 + 10$ is $O(n^3)$**.

This might be a bit jarring, but it's true.

Big-Oh

(The Bitter Truth)

$f(n)$ is **$O(g(n))$** iff **$f(n) \le c_1 \cdot g(n)$** for **$n \ge N_0$**

Example: Let $f(n) = 3n^2 + 10$. Prove $f(n)$ is $O(n^3)$.

We must find c_1 and N_0 such that $f(n) \le c_1 \cdot n^3$ for $n \ge N_0$

Proof: $f(n) = 3n^2 + 10 \le 3n^2 + n^2 = 4n^2 \le 4n^3$ (for $n \ge \sqrt{10}$)

^ (this inequality holds because $n^2 \ge 10$ when $n \ge \sqrt{10}$)

That's it! I showed $f(n) \le c_1 \cdot g(n)$ for $n \ge N_0$. (**c_1** $= 4$, **$g(n)$** $= n^3$, and **N_0** $= \sqrt{10}$)

- It might be a bit jarring to you to see that **$f(n) = 3n^2 + 10$** is **$O(n^3)$**. From CS1, we're used to taking only the **highest-order term** and locking it down as our Big-Oh.
- What we're seeing, however, is that Big-Oh is actually a sort of **upper bound**. And upper bounds can get **arbitrarily large** and still be upper bounds. Note: See terminology in Webcourses: "upper bound," "asymptotic upper bound," "tight bound," and so on.

Big-Omega

(Very similar to Big-Oh)

This is our formal, mathematical definition for Big-Oh:

f(n) is **O(g(n))** *iff* $f(n) \leq c_1 \cdot g(n)$ for $n \geq N_0$

There's a related concept, Big-Omega, whose definition is:

f(n) is **Ω(g(n))** *iff* $f(n) \geq c_1 \cdot g(n)$ for $n \geq N_0$

We use Big-Oh to articulate an **upper bound** on a function.

We use Big-Omega to articulate a **lower bound** on a function.

Note: For both definitions, we require $c_1 > 0$.

Big-Omega
(Lower Bound)

f(n) is **Ω(g(n))** iff **f(n) $\geq$ $c_1 \cdot$g(n)** for **n $\geq$ N_0**

Let's prove that **$3n^2 + 4$ is $\Omega(n^2)$**.

(Can you see intuitively that this is true?)

Big-Omega

(Lower Bound)

$f(n)$ is **$\Omega(g(n))$** iff **$f(n) \geq c_1 \cdot g(n)$** for **$n \geq N_0$**

Example: Let $f(n) = 3n^2 + 4$. Prove $f(n)$ is $\Omega(n^2)$.

We must find c_1 and N_0 such that $f(n) \geq c_1 \cdot n^2$ for $n \geq N_0$

Proof: $f(n) = 3n^2 + 4 \geq 3n^2$ (for $n \geq 1$)

^ (do you agree that this inequality holds?)

That's it! I showed $f(n) \geq c_1 \cdot g(n)$ for $n \geq N_0$. (**c_1** = 3, **$g(n)$** = n^2, and **N_0** = 1)

Big-Omega
(Lower Bound)

f(n) is **Ω(g(n))** iff **f(n) ≥ c_1·g(n)** for **n ≥ N_0**

Let's prove that **$3n^2 + 4$ is Ω(1)**.

(Can you see intuitively that this is true?)

Big-Omega
(Lower Bound)

f(n) is **Ω(g(n))** iff **$f(n) \geq c_1 \cdot g(n)$** for **$n \geq N_0$**

Example: Let $f(n) = 3n^2 + 4$. Prove f(n) is Ω(1).

We must find c_1 and N_0 such that $f(n) \geq c_1 \cdot 1$ for $n \geq N_0$

Proof: $f(n) = 3n^2 + 4 \geq 4$ (for $n \geq 1$)

^ (do you agree that this inequality holds?)

That's it! I showed $f(n) \geq c_1 \cdot g(n)$ for $n \geq N_0$. (**c_1** = 4, **g(n)** = 1, and **N_0** = 1)

Big-Omega
(Lower Bound)

f(n) is **Ω(g(n))** iff **$f(n) \geq c_1 \cdot g(n)$** for **$n \geq N_0$**

Food for Thought:

- Why might a **lower bound** on a runtime be useful?
- When might it be useful to use bounds that are not **tight**?
- These definitions apply to all mathematical functions (not just **runtimes**).
- Big-Oh is **not** akin to worst-case runtime.
- Big-Omega is **not** akin to best-case runtime.

Big-Theta

(Very similar to our old conception of Big-Oh from CS1!)

$$\mathbf{f(n)} \text{ is } \mathbf{\Theta(g(n))} \quad \textit{iff}: \begin{cases} \mathbf{f(n)} \text{ is } \mathbf{O(g(n))} & \leftarrow \textbf{upper} \text{ bound} \\ \text{— AND —} & \\ \mathbf{f(n)} \text{ is } \mathbf{\Omega(g(n))} & \leftarrow \textbf{lower} \text{ bound} \end{cases}$$

- With big-theta, **f(n)** is sandwiched between **g(n) curves** (upper and lower).
- The **g(n) curves** are each multiplied by some constant, of course.
- We say this **g(n)** function forms a **tight bound** on f(n).
- We effectively have an **f(n) sandwich** on **g(n) bread**.
- There are many **delicious sandwiches** to be made in Webcourses.

Big-Omega	**Big-Oh**	**Big-Theta**
(Lower Bound)	(Upper Bound)	(Sandwich Bound)

Suppose we have some function with:

- **Best-Case Runtime:** $4n^2 + n$
- **Worst-Case Runtime:** $3n^3 + 2$

Can we say the **best-case** runtime is...

$\Omega(n^2)$ YES	$\Omega(1)$ YES	$\Omega(n^3)$ NO
$O(n^2)$ YES	$O(1)$ NO	$O(n^3)$ YES
$\Theta(n^2)$ YES	$\Theta(1)$ NO	$\Theta(n^3)$ NO

Big-Omega	**Big-Oh**	**Big-Theta**
(Lower Bound)	(Upper Bound)	(Sandwich Bound)

Suppose we have some function with:

- **Best-Case Runtime:** $4n^2 + n$
- **Worst-Case Runtime:** $3n^3 + 2$

Can we say the **worst-case** runtime is...

$\Omega(n^2)$ YES	**$\Omega(1)$** YES	**$\Omega(n^3)$** YES
$O(n^2)$ NO	**$O(1)$** NO	**$O(n^3)$** YES
$\Theta(n^2)$ NO	**$\Theta(1)$** NO	**$\Theta(n^3)$** YES

Big-Omega	**Big-Oh**	**Big-Theta**
(Lower Bound)	(Upper Bound)	(Sandwich Bound)

Suppose we have some function with:

Best-Case Runtime: $4n^2 + n$

Worst-Case Runtime: $3n^3 + 2$

<u>In general</u>, can we say the runtime for this function is...

$\Omega(n^2)$ YES	$\Omega(1)$ YES	$\Omega(n^3)$ NO
$O(n^2)$ NO	$O(1)$ NO	$O(n^3)$ YES
$\Theta(n^2)$ NO	$\Theta(1)$ NO	$\Theta(n^3)$ NO