CSCI 241

Scott Wehrwein

Asymptotic Runtime Analysis: Intuition

Goals

Gain some intuition for why dropping constants is a reasonable thing to do.

Gain familiarity with some of the common runtime classes.

Where we are right now:

Me: 3. Drop constants and lower-order terms.

You:

Strategy:
1. Identify constant-time operations.
2. Determine how many times each happens.
3. Drop constants and lower-order terms.

?!?!

Reminder: constant-time operations

Key insight: a fixed number of primitive operations is itself a primitive operation.

Example:

```
i++
```

is shorthand for

```
i = i + 1
```

this is just a special case of dropping constants!

A count of 3 operations yields runtime class O(1)

But... really? *any* constant?

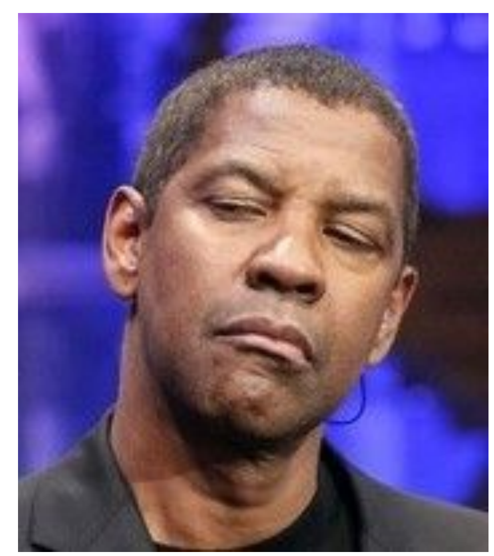
Claim: O(N) is more efficient than $O(N^2)$

The big-O could hide an arbitrarily large constant!

O(N) vs $O(N^2)$

could mean

$10,000N$ vs $N^2/8$



But... really? *any* constant?

A practical argument

FLOPs = FLoating-point Operation Per second

My MacBook Pro from 2013:

3.17 **giga**FLOPs

World's fastest supercomputer (Fugaku at RIKEN Center, Japan):

513.9 **peta**FLOPs

source: https://www.top500.org/lists/top500/2020/06/

Fugaku is **162,099,274 times faster**.

Big-O measures what counts

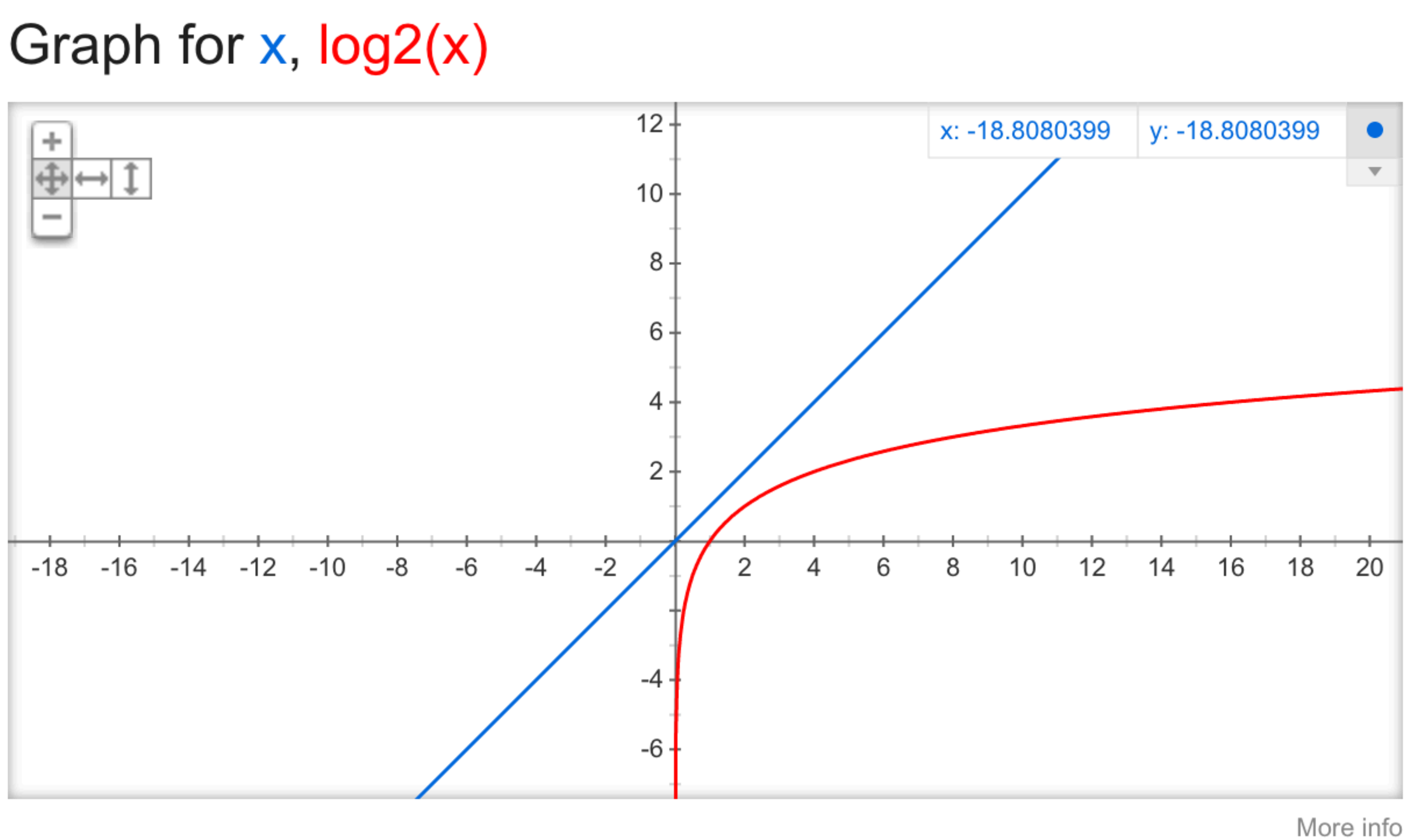
- If you could make up the difference by buying more computers or waiting a few years, call it equivalent.

But... really? *any* constant?

A theoretical argument

O(log n) vs O(n)

But what about the constants?

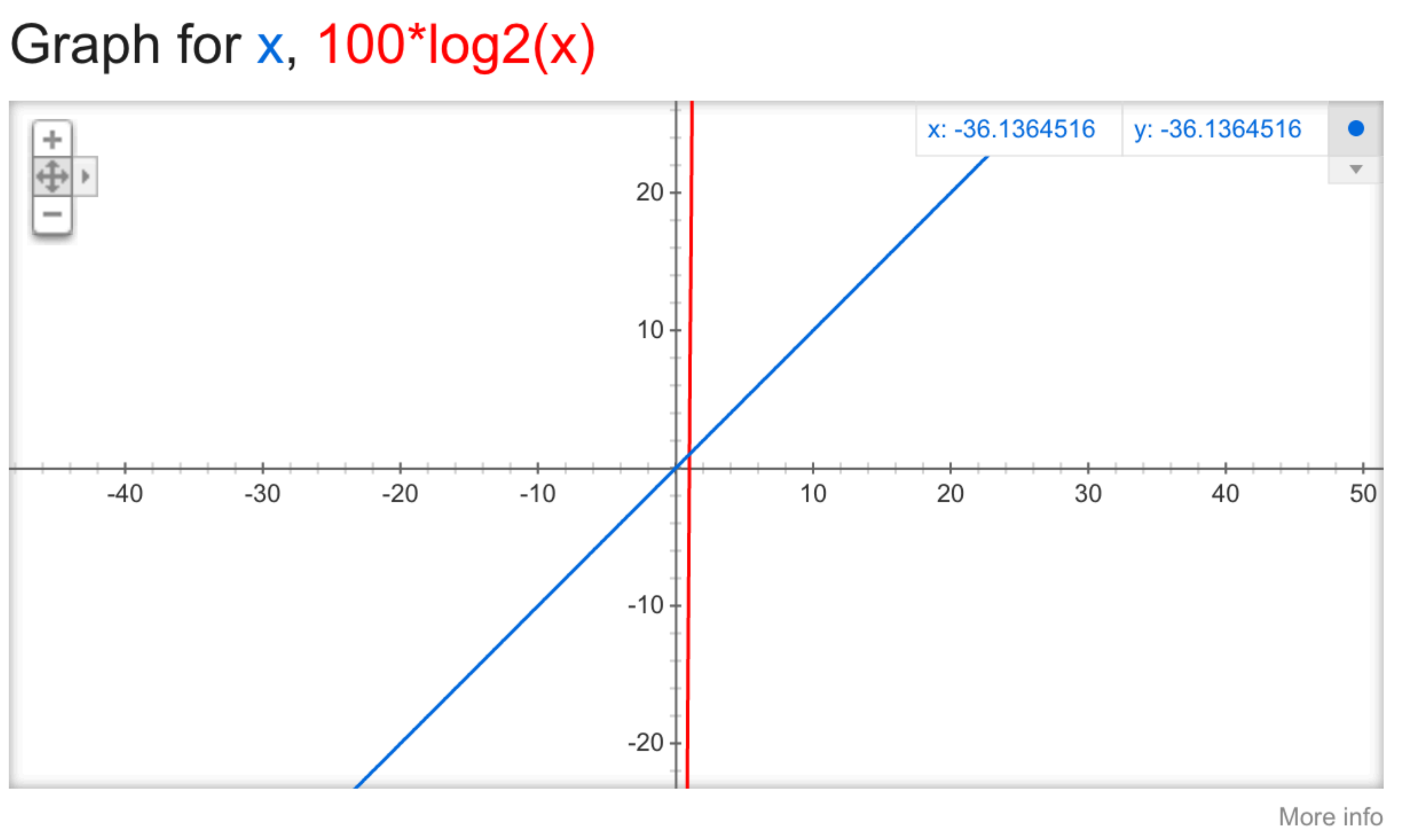


But... really? *any* constant?

A theoretical argument

O(log n) vs O(n)

But what about the constants?



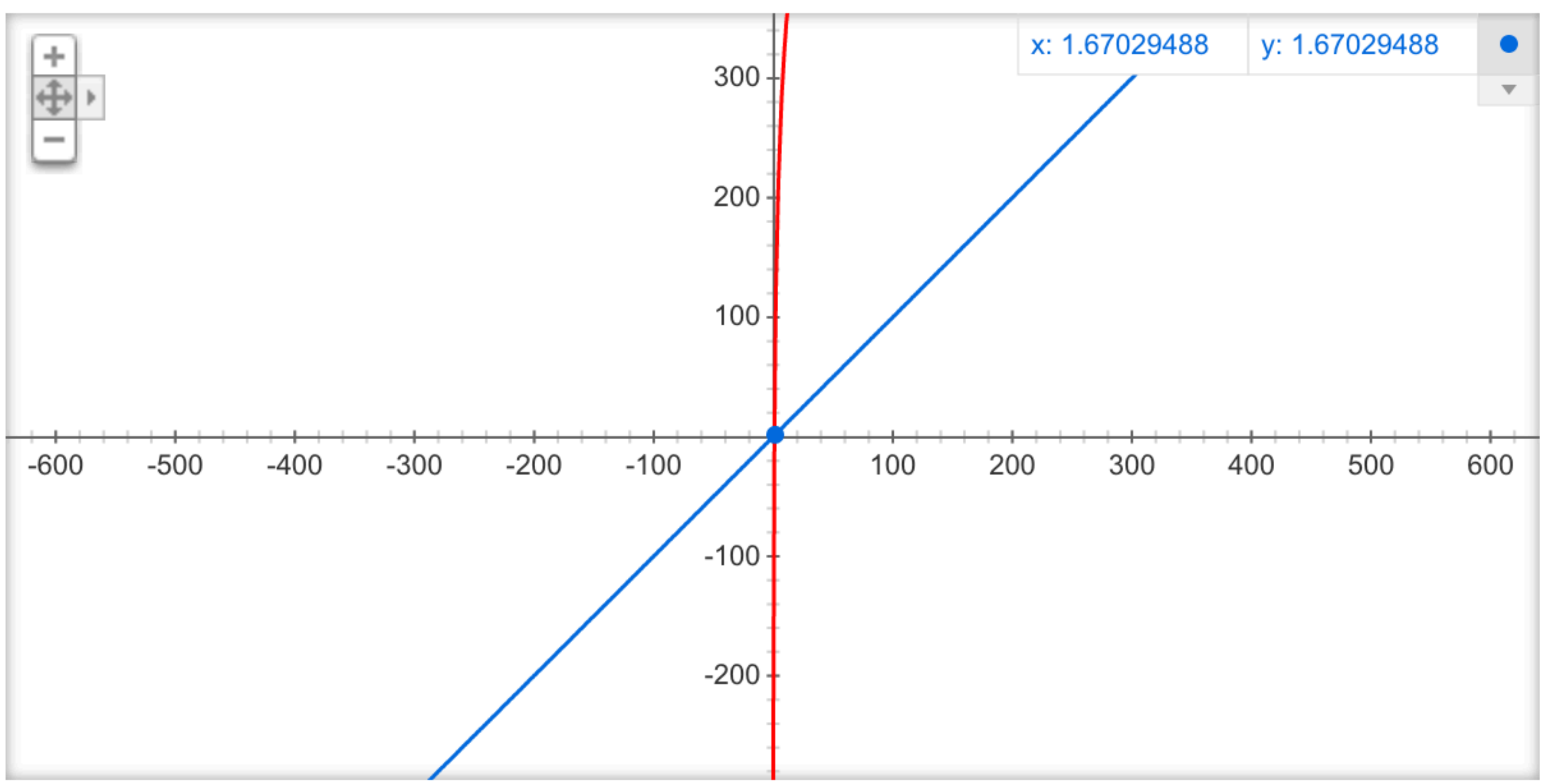
But… really? *any* constant?

A theoretical argument

O(log n) vs O(n)

But what about the constants?

Graph for x, 100*log2(x)



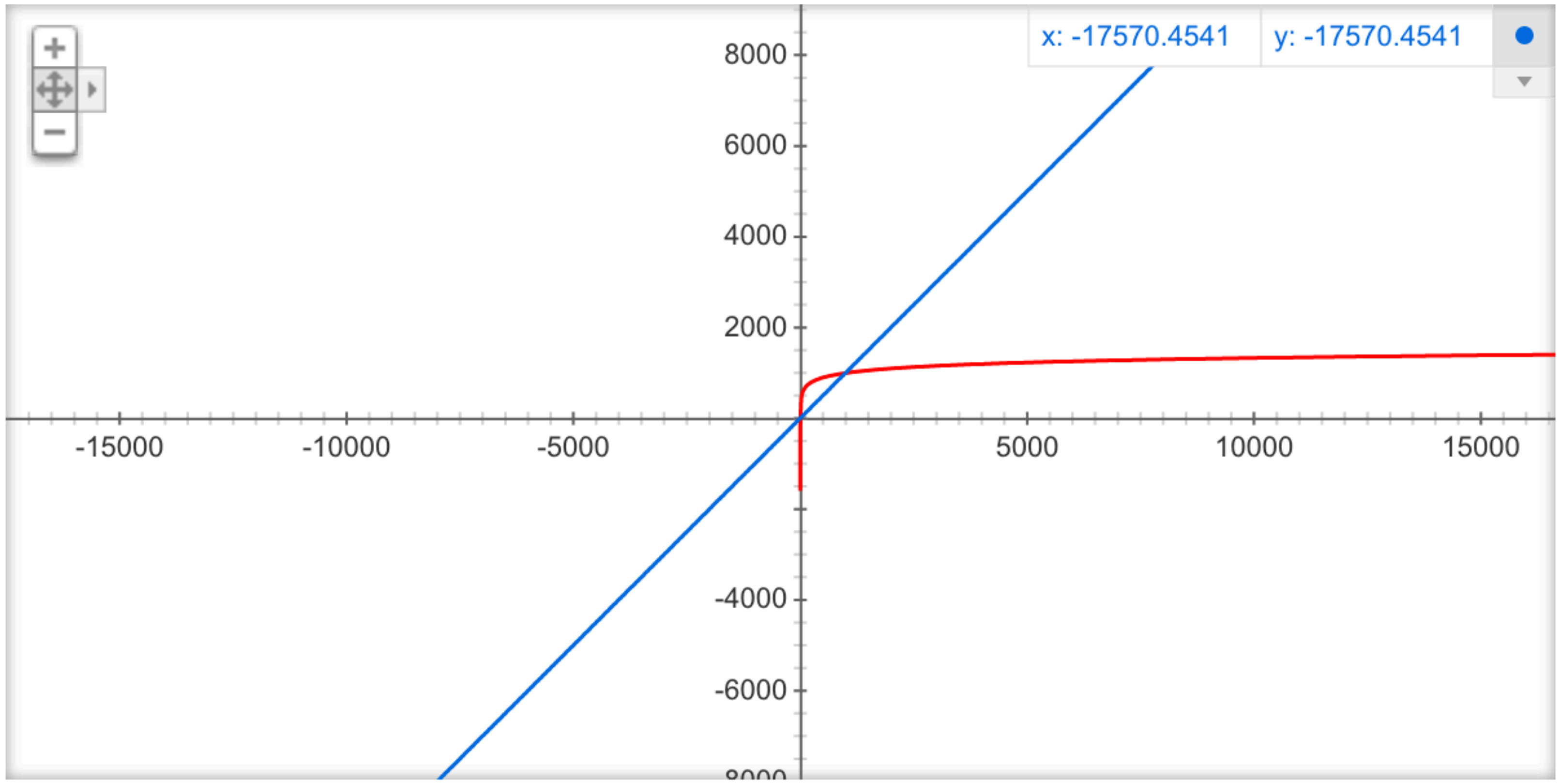
But... really? *any* constant?

A theoretical argument

O(log n) vs O(n)

But what about the constants?

Graph for x, 100*log2(x)



Big-O measures what counts

- If you could make up the difference by buying more computers or waiting a few years, call it equivalent.
- If N is small, it'll finish quickly regardless of which algorithm you use, so focus on performance for large N.

Common Complexity Classes