CS61A Discussion 7:
Orders of Growth and Trees

TA: **Jerry Chen**
Email: **jerry.c@berkeley.edu**
TA Website: **jerryjrchen.com/cs61a**

Attendance

Form: **tinyurl.com/jerrydisc**

Please remember to checkin! Only ~17 from last week

For the weekly question,
please tell me your least favorite part of hw04.

(Of course, please only check in if you showed up!)

Agenda

1. Week in Review
2. Composition
3. Orders of Growth
4. Trees (with mutation)

Week In Review

Hw4

- Was challenging!

Ants - Due next Thursday

- It's ok, I haven't started either :)

Lab 7 (Recursive Objects) - Due Friday

Mt2 - **7-9pm, Wednesday** after Spring Break (3/30)

- ***Submit alternate time request ASAP!***

Hog composition

Composition

Computers don't care how "neat" your code is

Humans (coworkers, boss, future you?) **do care!**

Composition

Basic Principles

- Pretend you are reading the code for the first time
- Simple > Complex
- Pick meaningful names
- (Personal opinion) give operators room to breathe!

Composition

Composition is not as important as correctness.
In my opinion:

- Correctness > Efficiency > Composition

"Done is better than perfect" - Facebook

Composition & style guide are at: http://cs61a.org/articles/resources.html

Orders of Growth

Context change — from writing programs to **evaluating their performance**

How do we describe how fast a program is?

Orders of Growth

Why do we care?

In the news

Google's DeepMind defeats legendary Go player Lee Se-dol in historic victory

The Verge - 1 day ago

DeepMind founder Demis Hassabis expressed "huge respect for Lee Se-dol and his ...

Match 1 - Google DeepMind Challenge Match: Lee Sedol vs AlphaGo

YouTube - 1 day ago

Google's Deepmind AI beats Go world champion in first match

Engadget - 23 hours ago

More news for deepmind

Orders of Growth

One way is through orders of growth. How does a program respond to a growing input size?

Ω

θ

O

Orders of Growth

```python
def factorial(n):
    if n == 0:
        return 1
    return n * factorial(n - 1)
```

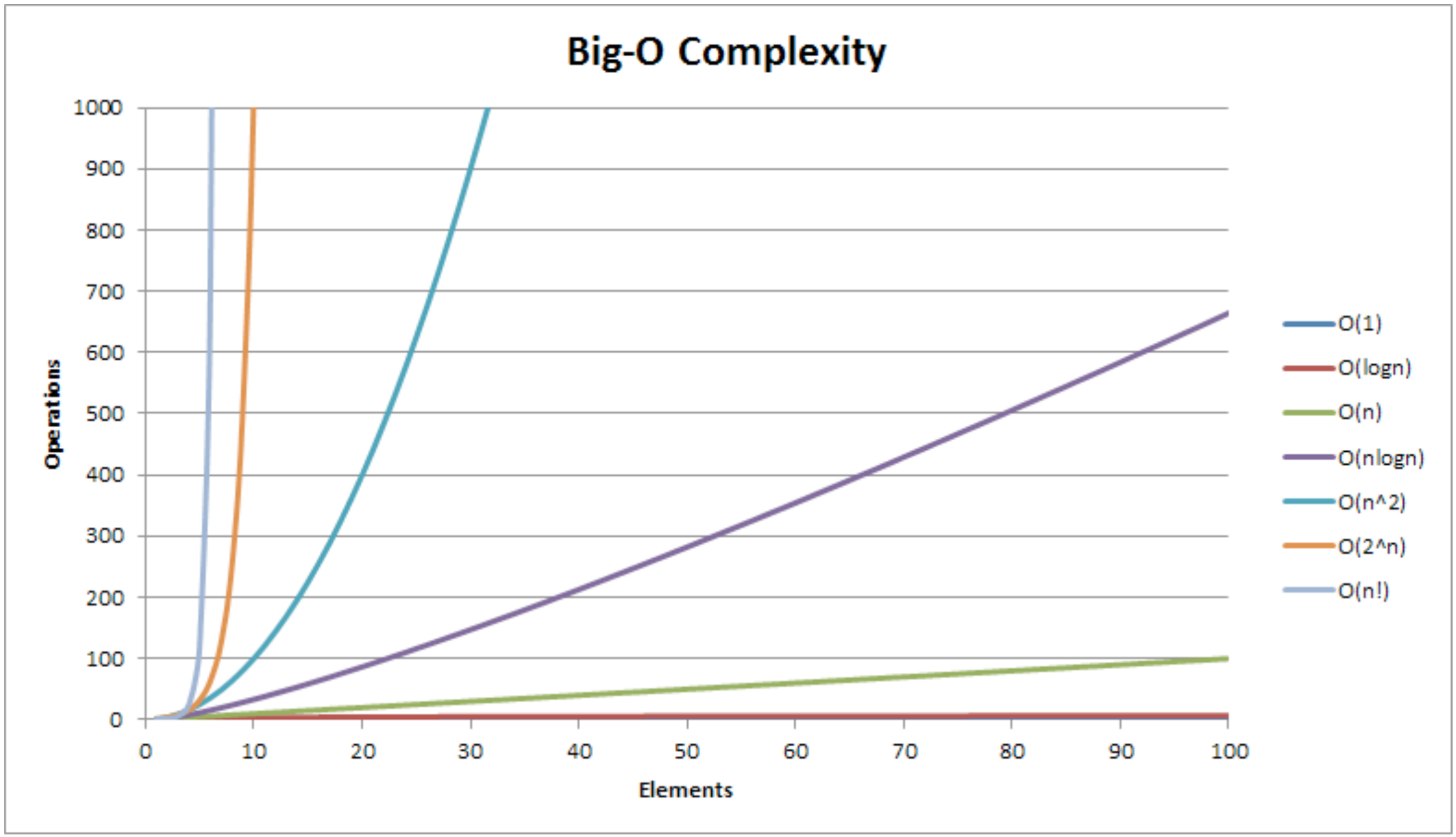
How does this scale with respect to size of input n?

Linear - O(n)

Orders of Growth

Growth Rate (Big-O)	How does it feel?
1 (“constant”)	Great!
log N (“logarithmic”)	Still really good!
N (“linear”)	Not bad
N^2, N^3 (“poly time”)	Acceptable
2^N (“exponential”)	Ugh… “intractable” growth
N! (~~“evil”~~ “factorial”)	Same as above… but worse?

Orders of Growth



http://bigocheatsheet.com/img/big-o-complexity.png

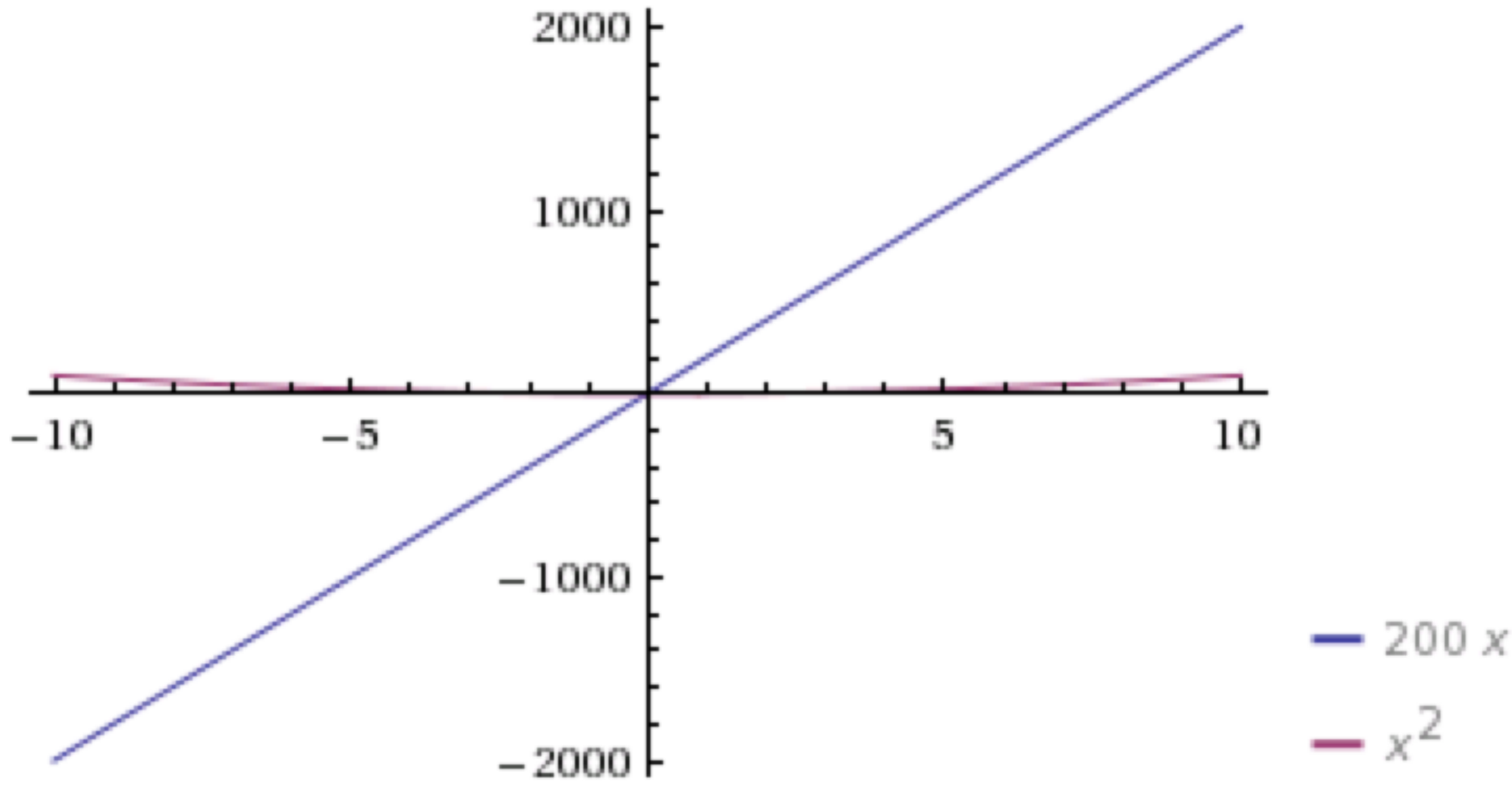
Orders of Growth

Time (μsec) for problem size N	Max N Possible in 1 second	1 hour	1 month	1 century
$\lg N$	10^{300000}	$10^{1000000000}$	$10^{8 \cdot 10^{11}}$	$10^{9 \cdot 10^{14}}$
N	10^6	$3.6 \cdot 10^9$	$2.7 \cdot 10^{12}$	$3.2 \cdot 10^{15}$
$N \lg N$	63000	$1.3 \cdot 10^8$	$7.4 \cdot 10^{10}$	$6.9 \cdot 10^{13}$
N^2	1000	60000	$1.6 \cdot 10^6$	$5.6 \cdot 10^7$
N^3	100	1500	14000	150000
2^N	20	32	41	51

Orders of Growth

Fantastic tool, but it has limitations:

Plot:

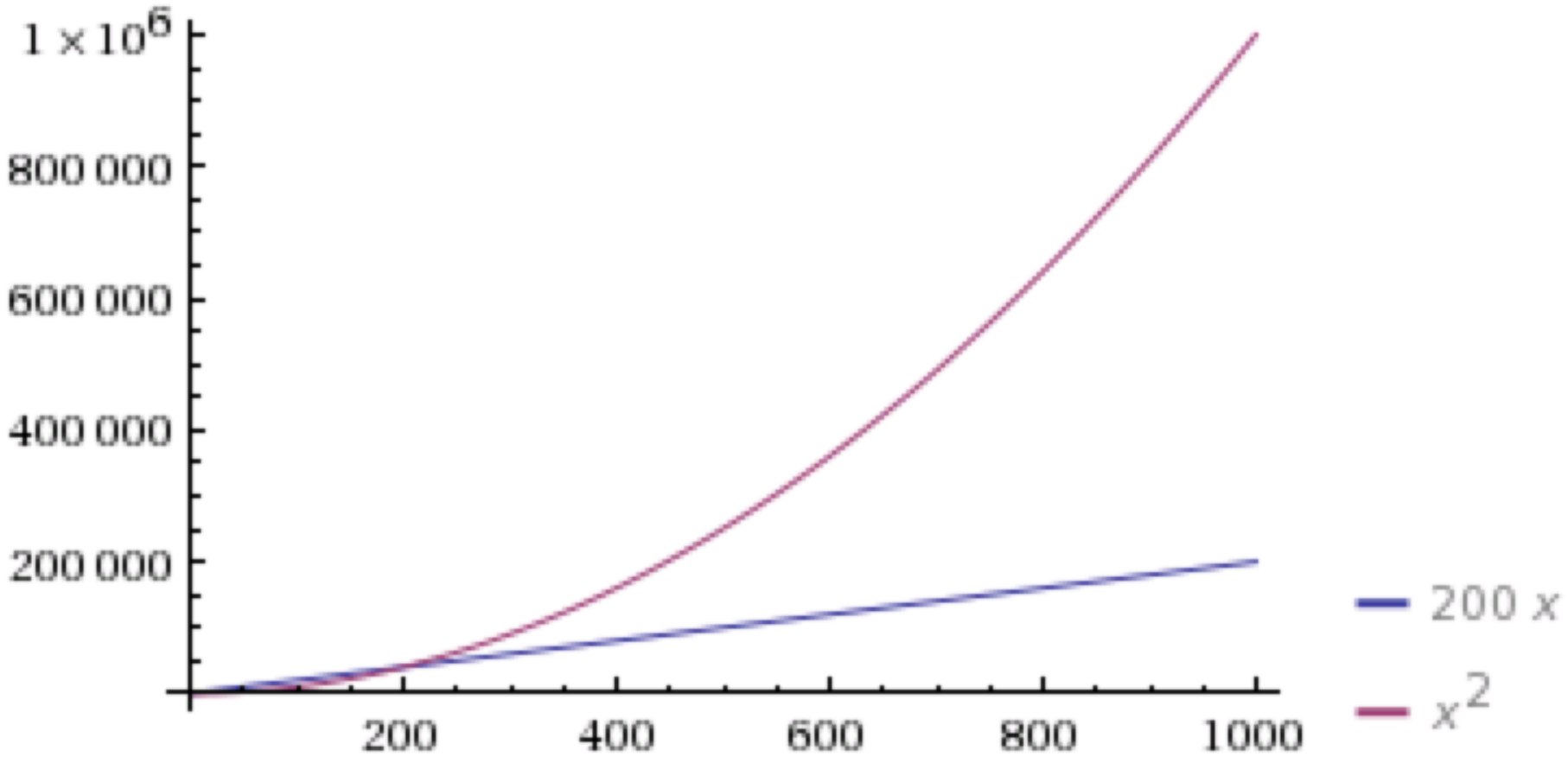


Graphs generated by WolframAlpha

Orders of Growth

But Big-O still wins in the end!

Plot:



Orders of Growth

Simplify	Answer
O(3n)	O(n) — ignore const factors
$O(n^3 + 1000n^2)$	$O(n^3)$ — larger term dominates
O(log n + n)	O(n) — larger term dominates
O(n log n + n)	O(n log n) — larger term dominates

Caveat — **these are NOT mathematically precise ways of describing growth relationships!

Orders of Growth

Question**	Answer
Is `factorial` O(n!)? O(log n)?	Yes, No. n! greatly upper bounds it. log n is not a sufficient upper bound.
$O(\log_2 n) > O(\log_{10} n)$	No! Use change of base formula.
$O(n \log(n^8)) > O(n^2 \log(n^3))$	No — use log rules to get O(n log n) vs $O(n^2 \log n)$
$O(n \log n) < O(\log n^{\log n})$	Yes — RHS is $n^{\log \log n}$ (try to introduce an exponent)

Mutable Trees

https://www.makemymerch.in/image/cache/data/groot-artwork-612x459.png

Mutable Trees

Can still build trees in much the same way:

```
Tree(<label>, [Tree(…), Tree(…), …])
```

Selectors now also allow assignment:

```
t = Tree(1)

t.label = 10
```

Mutable Trees

Interface:

```
Tree(label, [list_of_children])

Tree.label

Tree.branches (or Tree.children)

Tree.is_leaf(self)
```