

Happy Halloween!





15-112
Lecture 2

Week 8 Thu
Efficiency &
Object-Oriented
Programming

Instructor: Pat Virtue


Previous Poll 1

Which of these needs to visit all N elements in the list `data`, assuming $N = \text{len}(\text{data})$? 

Select ALL that apply.

$O(N)$

A. `for x in data:`
 `print(x)`

B. `for i in range(len(data)):`
 `print(x)`  $x = \text{data}[i]$

C. `if x in data:`
 `print("Found it")`

D. `x = data[i]` $\text{data}[-1]$  $O(1)$

E. `x = max(data)`

F. None of the above

$O(n)$

$\text{data.index}(x)$

Previous Poll 2

Which of these needs to visit all N elements in the **set** data, assuming $N = \text{len}(\text{data})$?

Select ALL that apply.

✓ A. for x in data:
 print(x)

$O(N)$

~~B. for i in range(len(data)):
 ~~print(x)~~~~

← $x = \text{data}[i]$

✗ C. if x in data:
 print("Found it")

$O(1)$

~~D. x = data[i]~~

✓ E. x = max(data)

$O(N)$

F. None of the above

Dictionaries

Dictionaries

Map keys to values

Keys are stored like sets

Efficiency

Counting operations


Worksheet

Counting operations

N is the size of the input data

- e.g. the length of an input list

The function $f(N)$ is a measurement or count of resources used based on N

- Often based on **computation time** needed, but can be related to other resources like **space** (memory) needed
- Measured in number of operations rather than time
 - Lots of reasons, e.g. easier to compare algorithms despite changes in computer speed
- Small details either ignored or estimated (because of big-O) 

Big O

$$\underline{N \log N} \quad \underline{N}$$

Describes asymptotic behavior of a function

Informally (for 15112):

- Ignore all lower-order terms and constants

A few examples:

- ~~$3N^2 - 2N + 25$~~ is in $O(N^2)$
- ~~$30000N^2 + 2N - 25$~~ is in $O(N^2)$
- ~~$0.000001N^2 + 123456N$~~ is in $O(N^2)$
- ~~$10N \log_{17} N + 25N - 17$~~ is in $O(\underline{N \log N})$

$3N^2$ $2N^2$
 $\frac{3}{2} N^2$

Common Function Families

Constant: $O(1)$ ←

Logarithmic: $O(\log N)$

Linear: $O(N)$ ←

Loglinear: $O(N \log N)$

Quadratic: $O(N^2)$ ←

Exponential: $O(2^N)$

Poll 3

Which of these is $O(N)$ for the dictionary d , assuming $N = \text{len}(d)$?

Note: all of these are either $O(1)$ or $O(N)$

Select ALL that apply.

✓ A. for key in d:

print(d[key])

$O(N)$

~~B. for i in range(len(d)):~~

~~print(d[i])~~

Key Error (probably)

→ C. if key in d:

print("Found it")

$O(1)$

D. x = d[key]

$O(1)$

E. d[key] = x

$O(1)$

F. None of the above

Poll 4

I'm thinking of a number between 1 and 64. After each guess, I'll tell you if you're **correct** or if my number is **higher** or **lower**.

\$100 if you win. \$0 if you lose.

How many guesses do you want to buy, \$1 each?

A: 6

B: 7

C: 32

D: 64

Guess a Number: Binary Search

I'm thinking of a number between **1 and N**. After each guess, I'll tell you if you're **correct** or if my number is **higher** or **lower**.

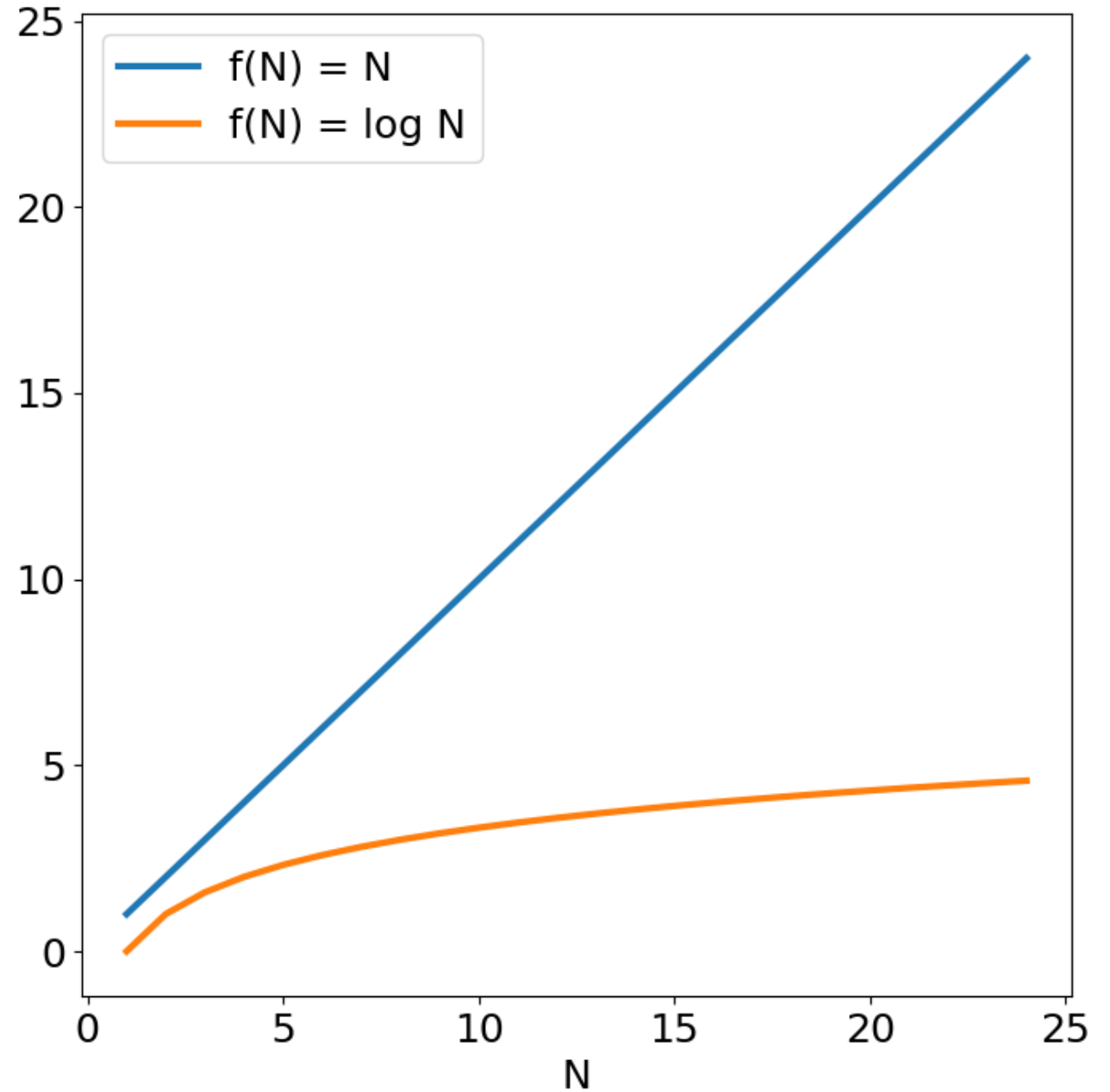
What is the maximum number of guesses you'll need to play this game?

N	10	100	1,000	10,000	100,000	1,000,000	10,000,000
$\log_2 N$	3.3	6.6	10.0	13.3	16.6	19.9	23.3
$\lfloor \log_2 N \rfloor + 1$	4.0	7.0	11.0	14.0	17.0	20.0	24.0

Linear vs Binary Search

Linear search: $O(N)$

Binary search: $O(\log N)$



Linear vs Binary Search

Linear search: $O(N)$

$N = 40$



Binary search: $O(\log N)$



Common Function Families

✓ Constant: $O(1)$

✓ Logarithmic: $O(\log N)$

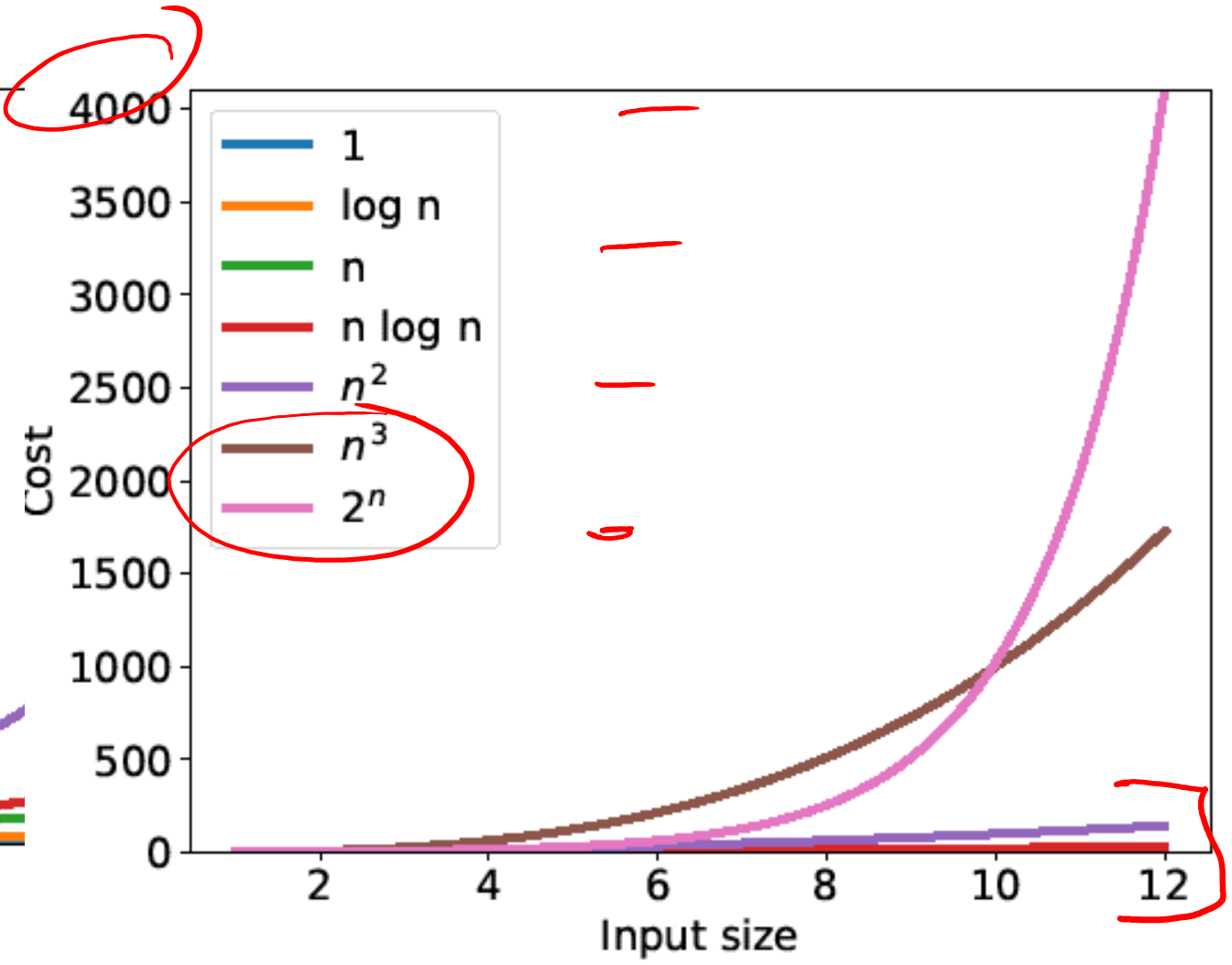
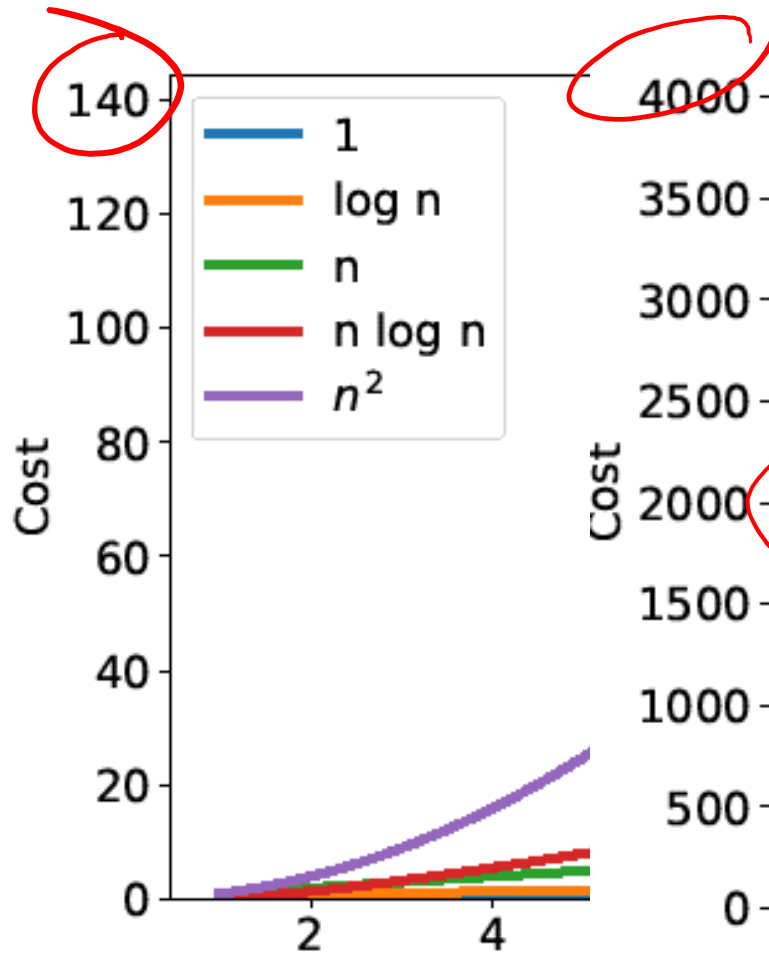
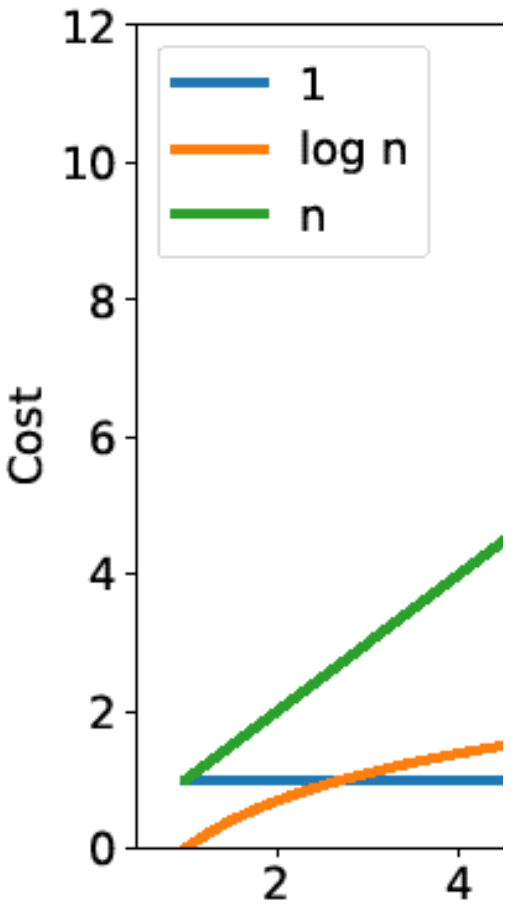
✓ Linear: $O(N)$

Loglinear: $O(N \log N)$ ←

✓ Quadratic: $O(N^2)$

Exponential: $O(2^N)$

Complexity Classes



Efficiency of Sorting Algorithms

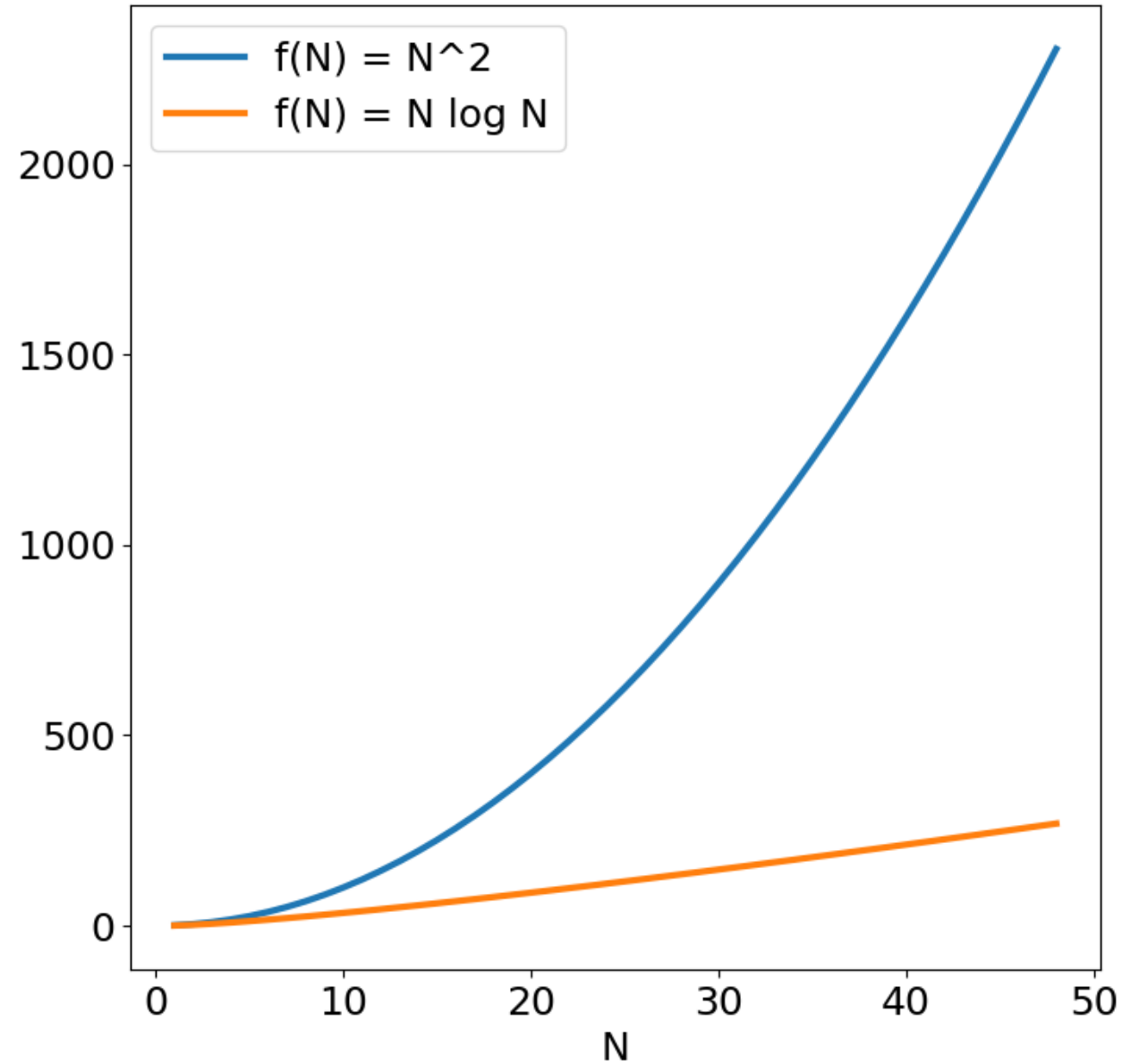
Sorting algorithms

xSortLab

<http://math.hws.edu/eck/js/sorting/xSortLab.html>

Selection sort: $O(N^2)$

Merge sort: $O(N \log N)$



Sorting algorithms

Selection sort: $O(N^2)$

Loop

- Find max in unsorted region
- Swap max with value at the end of the unsorted region
- Shrink unsorted region by 1

Merge sort: $O(N \log N)$

Sorting algorithms

Merge sort: $O(N \log N)$

Merge concept:

Assume you had two piles that were already independently sorted.

Could you shuffle them together into one sorted pile in $O(N)$?

Obejct-oriented Programming

Term Project Preview