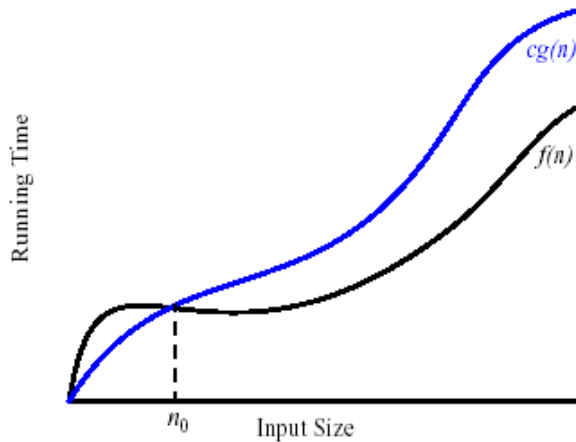


CSSE 230 Day 2

Growable Arrays Continued
Big-Oh and its cousins



Answer Q1 from today's in-class quiz.

Agenda and goals

- ▶ Finish course intro
- ▶ Growable Array recap
- ▶ Big-Oh and cousins

- ▶ After today, you'll be able to
 - Use the term *amortized* appropriately in analysis
 - explain the meaning of big-Oh, big-Omega (Ω), and big-Theta (θ)
 - apply the definition of big-Oh to prove runtimes of functions
 - use limits to show that a function is O , θ , or Ω of another function.

Announcements and FAQ

- ▶ You will not usually need the textbook in class
- ▶ Tuesday is Tie day (or “Professional Attire” day)
- ▶ Late days?
- ▶ Individual competence requirement

You must demonstrate programming competence on exams to succeed

- ▶ See syllabus for exam weighting and caveats.
- ▶ Think of every programming you write as a practice test
 - Especially HW4 and test 2a

Warm Up and Stretching thoughts

- Short but intense! ~45 lines of code total in our solutions to all but Adder
- Be sure to read the description of how it will be graded. Note how style will be graded.
- **Demo:** Running the JUnit tests for test, file, package, and project

Demo: Run the Adder program

Questions?

- ▶ About Homework 1?
 - Aim to complete tonight, since it is due tomorrow night
 - It is substantial (in amount of work, and in course credit)
- ▶ About the Syllabus?

Growable Arrays Exercise

Daring to double

Growable Arrays Table

N	E_N	Answers for problem 2
4	0	0
5	0	0
6	5	5
7	5	$5 + 6 = 11$
10	5	$5 + 6 + 7 + 8 + 9 = 35$
11	$5 + 10 = 15$	$5 + 6 + 7 + 8 + 9 + 10 = 45$
20	15	$\text{sum}(i, i=5..19) = 180$ using Maple
21	$5 + 10 + 20 = 35$	$\text{sum}(i, i=5..20) = 200$
40	35	$\text{sum}(i, i=5..39) = 770$
41	$5 + 10 + 20 + 40 = 75$	$\text{sum}(i, i=5..40) = 810$

Doubling the Size

- ▶ Doubling each time:
 - Assume that $N = 5(2^k) + 1$.
- ▶ Total # of array elements copied:

k	N	#copies
0	6	5
1	11	$5 + 10 = 15$
2	21	$5 + 10 + 20 = 35$
3	41	$5 + 10 + 20 + 40 = 75$
4	81	$5 + 10 + 20 + 40 + 80 = 155$
k	$= 5(2^k) + 1$	$5(1 + 2 + 4 + 8 + \dots + 2^k)$

Express as a closed-form expression in terms of K, then express in terms of N

Adding One Each Time

- ▶ Total # of array elements copied:

N	#copies
6	5
7	5 + 6
8	5 + 6 + 7
9	5 + 6 + 7 + 8
10	5 + 6 + 7 + 8 + 9
N	???

Express as a closed-form
expression in terms of N

Conclusions

- ▶ What's the **average** overhead cost of adding an additional string...
 - in the doubling case?
 - in the add-one case?
- ▶ So which should we use?

This is called the **amortized** cost

More math review

Review these as needed

- Logarithms and Exponents

- properties of **logarithms**:

$$\log_b(xy) = \log_b x + \log_b y$$

$$\log_b(x/y) = \log_b x - \log_b y$$

$$\log_b x^\alpha = \alpha \log_b x$$

$$\log_b x = \frac{\log_a x}{\log_a b}$$

- properties of **exponentials**:

$$a^{(b+c)} = a^b a^c$$

$$a^{bc} = (a^b)^c$$

$$a^b / a^c = a^{(b-c)}$$

$$b = a^{\log_a b}$$

$$b^c = a^{c \cdot \log_a b}$$

Practice with exponentials and logs

(Do these with a friend after class, not to turn in)

Simplify: Note that $\log n$ (without a specified) base means $\log_2 n$. Also, $\log n$ is an abbreviation for $\log(n)$.

1. $\log(2n \log n)$

2. $\log(n/2)$

3. $\log(\sqrt{n})$

4. $\log(\log(\sqrt{n}))$

5. $\log_4 n$

6. $2^{2 \log n}$

7. if $n=2^{3k} - 1$, solve for k .

Where do logs come from in algorithm analysis?

Solutions

No peeking!

Simplify: Note that $\log n$ (without a specified) base means $\log_2 n$.
Also, $\log n$ is an abbreviation for $\log(n)$.

1. $1 + \log n + \log \log n$

2. $\log n - 1$

3. $\frac{1}{2} \log n$

4. $-1 + \log \log n$

5. $(\log n) / 2$

6. n^2

7. $n+1=2^{3k}$

$$\log(n+1)=3k$$

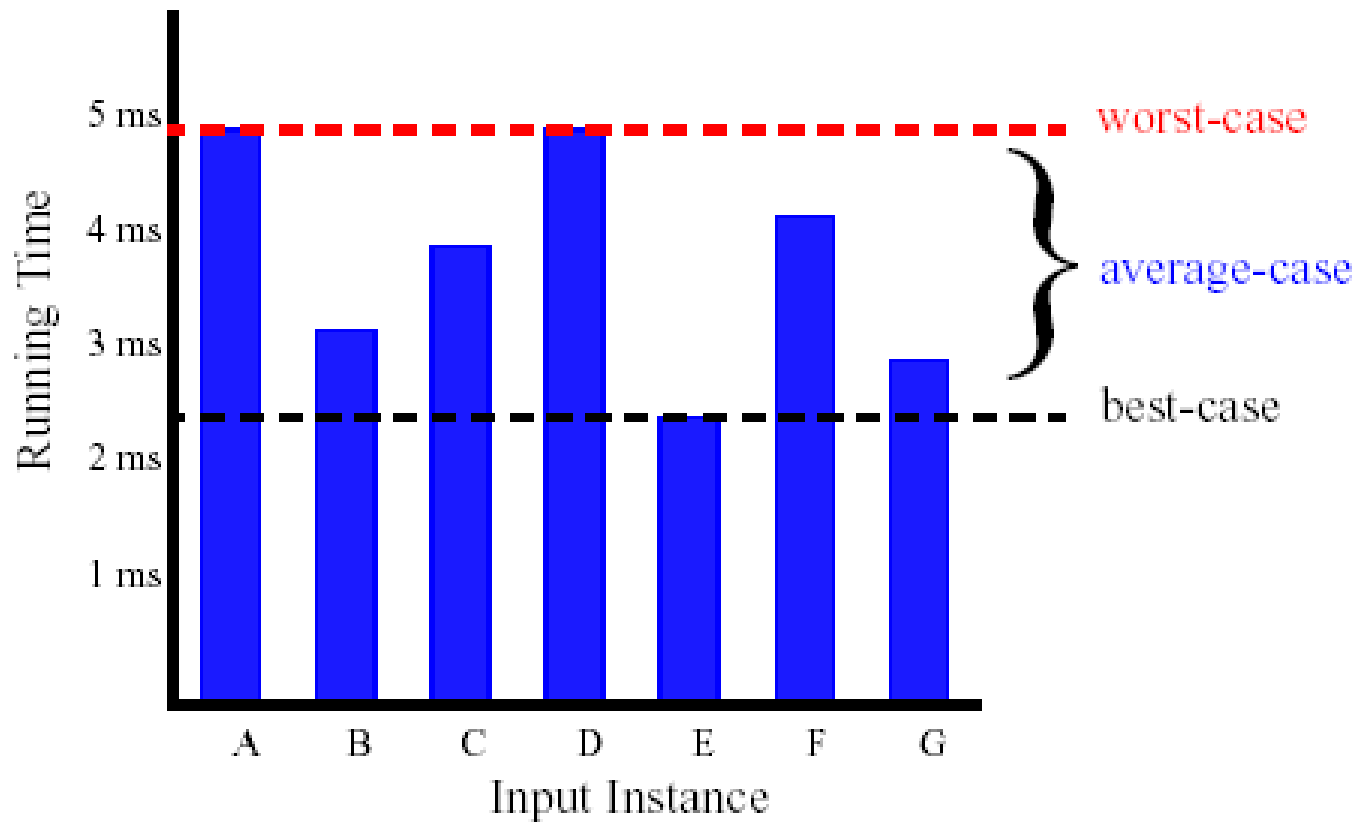
$$k = \log(n+1)/3$$

A: Any time we cut things in half at each step
(like binary search or mergesort)

Running Times

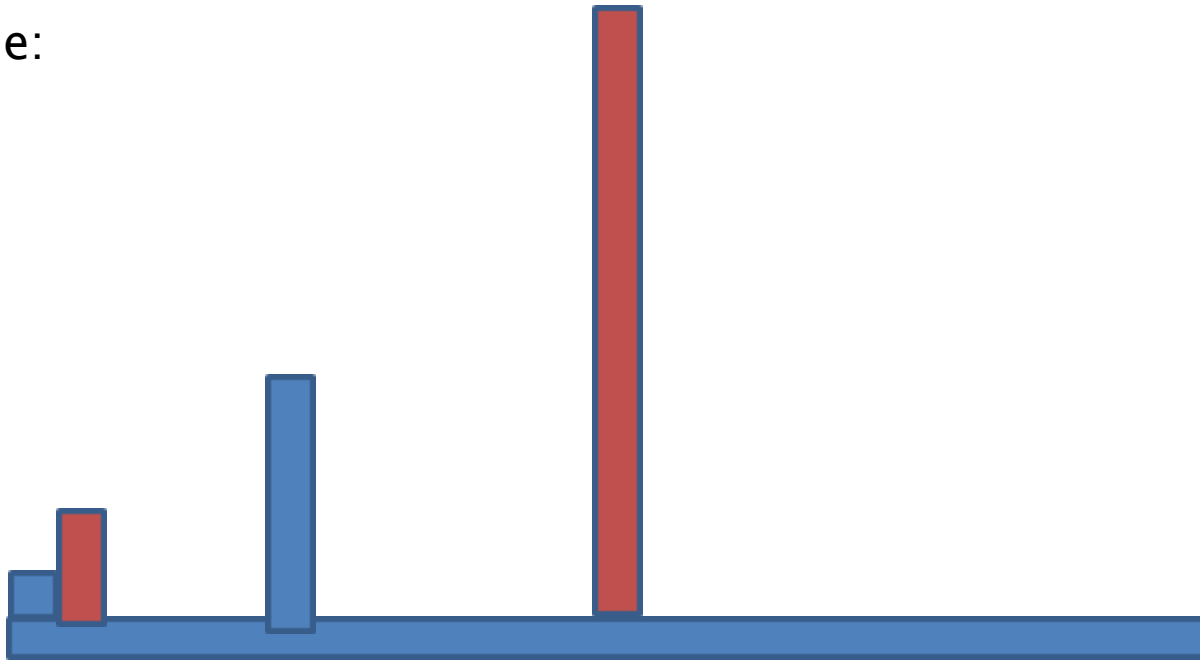
- ▶ Algorithms may have different *time complexity* on different data sets
- ▶ What do we mean by "Worst Case"?
- ▶ What do we mean by "Average Case"?
- ▶ What are some application domains where knowing the Worst Case time complexity would be important?
- ▶ <http://cacm.acm.org/magazines/2013/2/160173-the-tail-at-scale/fulltext>

Average Case and Worst Case



Worst-case vs amortized cost for adding an element to an array using the doubling scheme

Worst-case:
 $O(n)$



amortized:
 $O(1)$



Asymptotics: The “Big” Three

Big-Oh

Big-Omega

Big-Theta

Asymptotic Analysis

- ▶ We only care what happens when N gets large
- ▶ Is the function linear? quadratic?
exponential?

Figure 5.1

Running times for small inputs

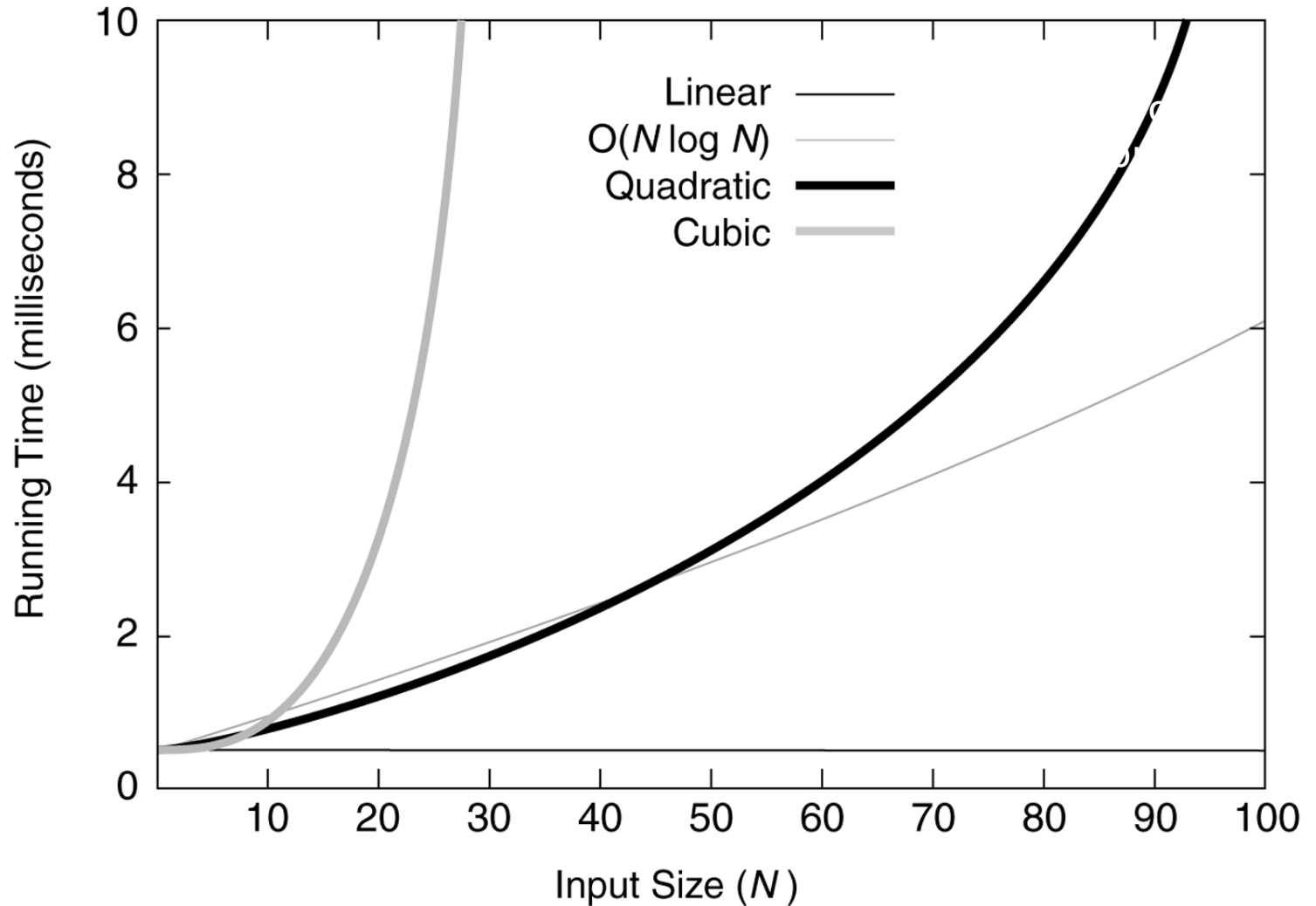


Figure 5.2

Running times for moderate inputs

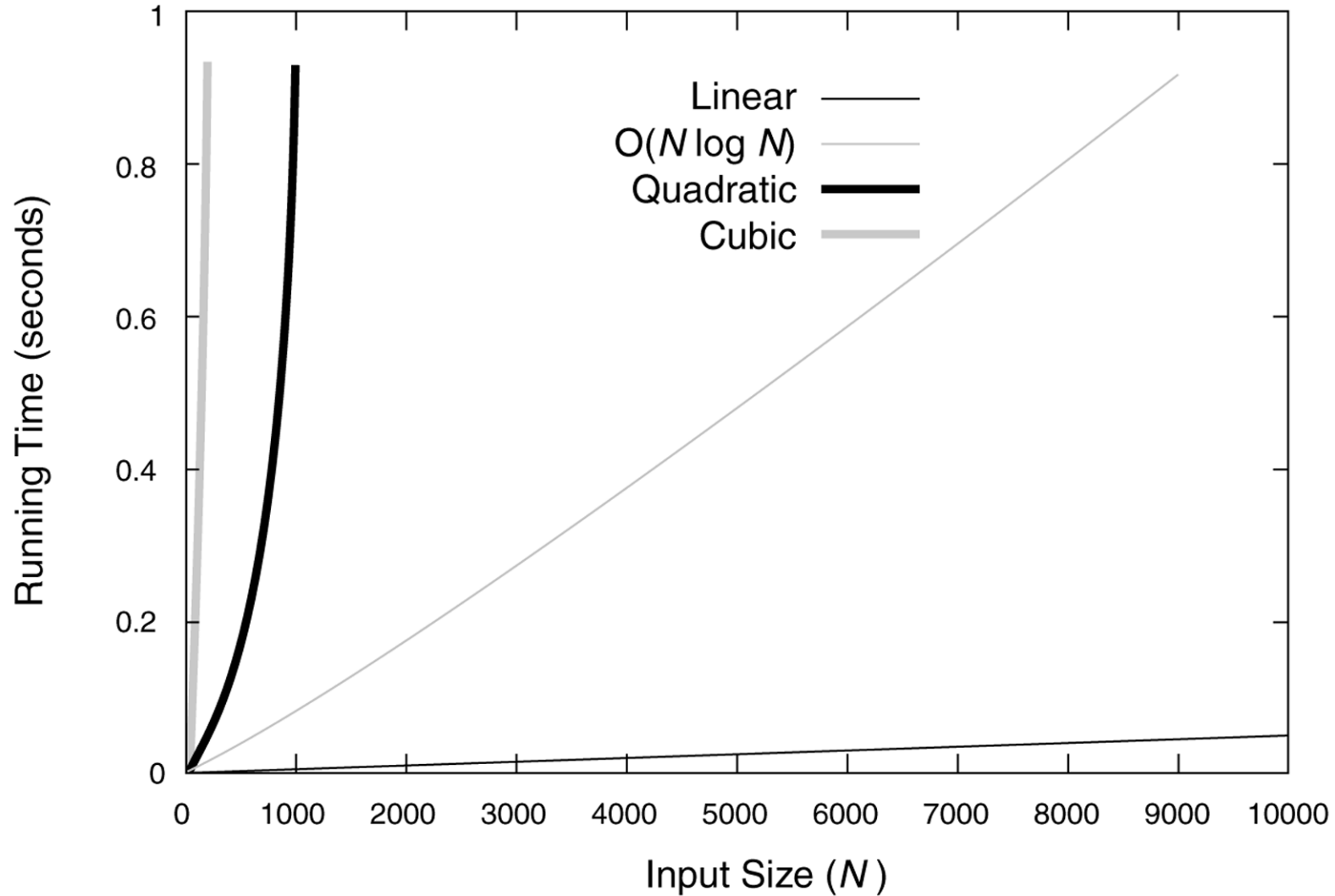


Figure 5.3

Functions in order of increasing growth rate

The answer to most big-Oh questions is one of these functions

FUNCTION	NAME
c	Constant
$\log N$	Logarithmic
$\log^2 N$	Log-squared
N	Linear
$N \log N$	$N \log N$
N^2	Quadratic
N^3	Cubic
2^N	Exponential

a.k.a "log linear"

Simple Rule for Big-Oh

- ▶ Drop lower order terms and constant factors
- ▶ $7n - 3$ is $O(n)$
- ▶ $8n^2 \log n + 5n^2 + n$ is $O(n^2 \log n)$

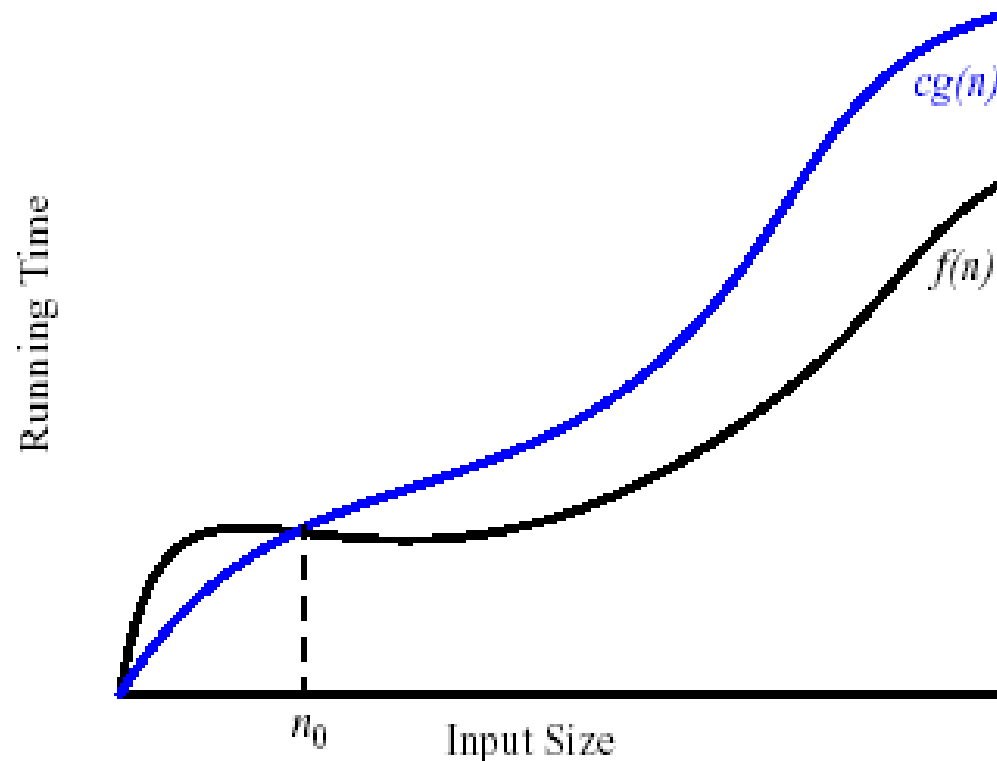
O

- The “Big-Oh” Notation

- given functions $f(n)$ and $g(n)$, we say that $f(n)$ is $O(g(n))$ if and only if $f(n) \leq c g(n)$ for $n \geq n_0$

- c and n_0 are constants, $f(n)$ and $g(n)$ are functions over non-negative integers

$c > 0$, $n_0 \geq 0$ and an integer



Big Oh examples

- ▶ A function $f(n)$ is (in) $O(g(n))$ if there exist two positive constants c and n_0 such that for all $n \geq n_0$, $f(n) \leq c g(n)$
- ▶ So all we must do to prove that $f(n)$ is $O(g(n))$ is produce two such constants.
- ▶ $f(n) = 4n + 15$, $g(n) = ???$.
- ▶ $f(n) = n + \sin(n)$, $g(n) = ???$

Assume that all functions have non-negative values, and that we only care about $n \geq 0$. For any function $g(n)$, $O(g(n))$ is a set of functions.

Big-Oh, Big-Omega and Big-Theta

$O()$

$\Omega()$

$\theta()$

- ▶ $f(n)$ is $O(g(n))$ if $f(n) \leq cg(n)$ for all $n \geq n_0$
 - So big-Oh (O) gives an upper bound
- ▶ $f(n)$ is $\Omega(g(n))$ if $f(n) \geq cg(n)$ for all $n \geq n_0$
 - So big-omega (Ω) gives a lower bound
- ▶ $f(n)$ is $\theta(g(n))$ if it is both $O(g(n))$ and $\Omega(g(n))$
Or equivalently:
- ▶ $f(n)$ is $\theta(g(n))$ if $c_1g(n) \leq f(n) \leq c_2g(n)$ for all $n \geq n_0$
 - So big-theta (θ) gives a tight bound

We usually show algorithms (in code) are $\theta(g(n))$. Tomorrow, we'll also discuss how to show **problems** are $\theta(g(n))$.

- ▶ True or false: $3n+2$ is $O(n^3)$
- ▶ True or false: $3n+2$ is $\Theta(n^3)$

Big-Oh Style

- ▶ Give tightest bound you can
 - Saying $3n+2$ is $O(n^3)$ is true, but not as useful as saying it's $O(n)$
 - On a test, we'll ask for Θ to be clear.
- ▶ Simplify:
 - You could also say: $3n+2$ is $O(5n-3\log(n) + 17)$
 - And it would be technically correct...
 - It would also be poor taste ... and your grade will reflect that.

On homework 2...

- ▶ Suppose $T_1(N)$ is $O(f(N))$ and $T_2(N)$ is $O(f(N))$.
Prove that $T_1(N) + T_2(N)$ is $O(f(N))$
- ▶ Hint: Constants c_1 and c_2 must exist for $T_1(N)$ and $T_2(N)$ to be $O(f(N))$
 - How can you use them?
- ▶ Try it before next class

Limitations of big-Oh

- ▶ There are times when one might choose a higher-order algorithm over a lower-order one.
- ▶ Brainstorm some ideas to share with the class

C.A.R. Hoare, inventor of quicksort, wrote:
Premature optimization is the root of all evil.

Limits and Asymptotics

- ▶ Consider the limit

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)}$$

- ▶ What does it say about asymptotic relationship between f and g if this limit is...
 - 0?
 - finite and non-zero?
 - infinite?

Apply this limit property to the following pairs of functions

1. n and n^2

on these questions and solutions ONLY, let $\log n$ mean natural log

2. $\log n$ and n

3. $n \log n$ and n^2

4. $\log_a n$ and $\log_b n$ ($a < b$)

5. n^a and a^n ($a > 1$)

6. a^n and b^n ($a < b$)

Recall l'Hôpital's rule:
under appropriate conditions,

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \lim_{n \rightarrow \infty} \frac{f'(n)}{g'(n)}$$