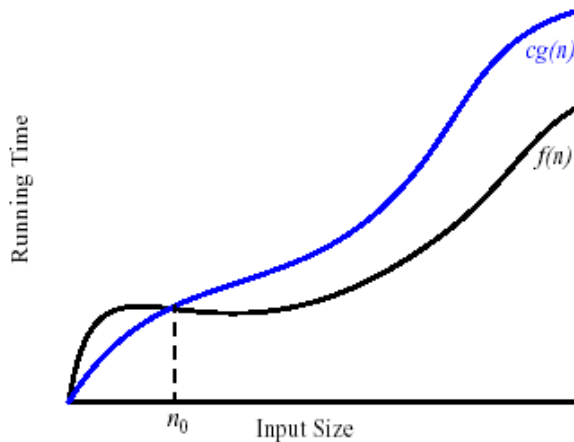


CSSE 230 Day 2

Growable Arrays Continued Big-Oh and its cousins



Answer Q1 from today's in-class quiz.

Announcements

- ▶ You will not usually need the textbook in class
- ▶ What to call me?
- ▶ Tuesday is Tie day (or “Professional Attire” day)

Agenda

- ▶ Finish course intro
- ▶ Growable Array recap
- ▶ Exponents and logs (quick)

- ▶ Big-Oh and its cousins
 - Big-Omega
 - Big-Theta

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
- **Demo:** Running the JUnit tests for test, file, package, and project

Demo: Run the Adder program

230 is Like Special Forces Training

- ▶ Pushes you to your limits
- ▶ Seems relentless
- ▶ When you are done, you are ready for anything

- ▶ But you have to work hard to get there

Be willing to squarely face any deficiencies that you may bring into the course. Don't use them as an excuse, but see them as challenges that you must overcome!

Grading

Criteria	Weight
In-class quizzes	5%
HW, programs, in-class exercises	30%
Major project	10%
Exam 1	15%
Exam 2	18%
Exam 3 (during finals week)	22%

▶ Caveats

- Must get a C on at least one exam to get a C in the course
- Must have **passing exam average** to pass course
- Must demonstrate **individual programming competence**
- **Three or more unexcused absences** may result in failure

Questions?

- ▶ About the Syllabus?
- ▶ Other administrative details?
- ▶ Written Assignment 1?
 - Due tomorrow night
 - It is substantial (in amount of work, and in course credit)
- ▶ WarmUpAndStretching?

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	$5 + 10 + 20 = 35$	$\text{sum}(i, i=5..20) = 180$
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	???

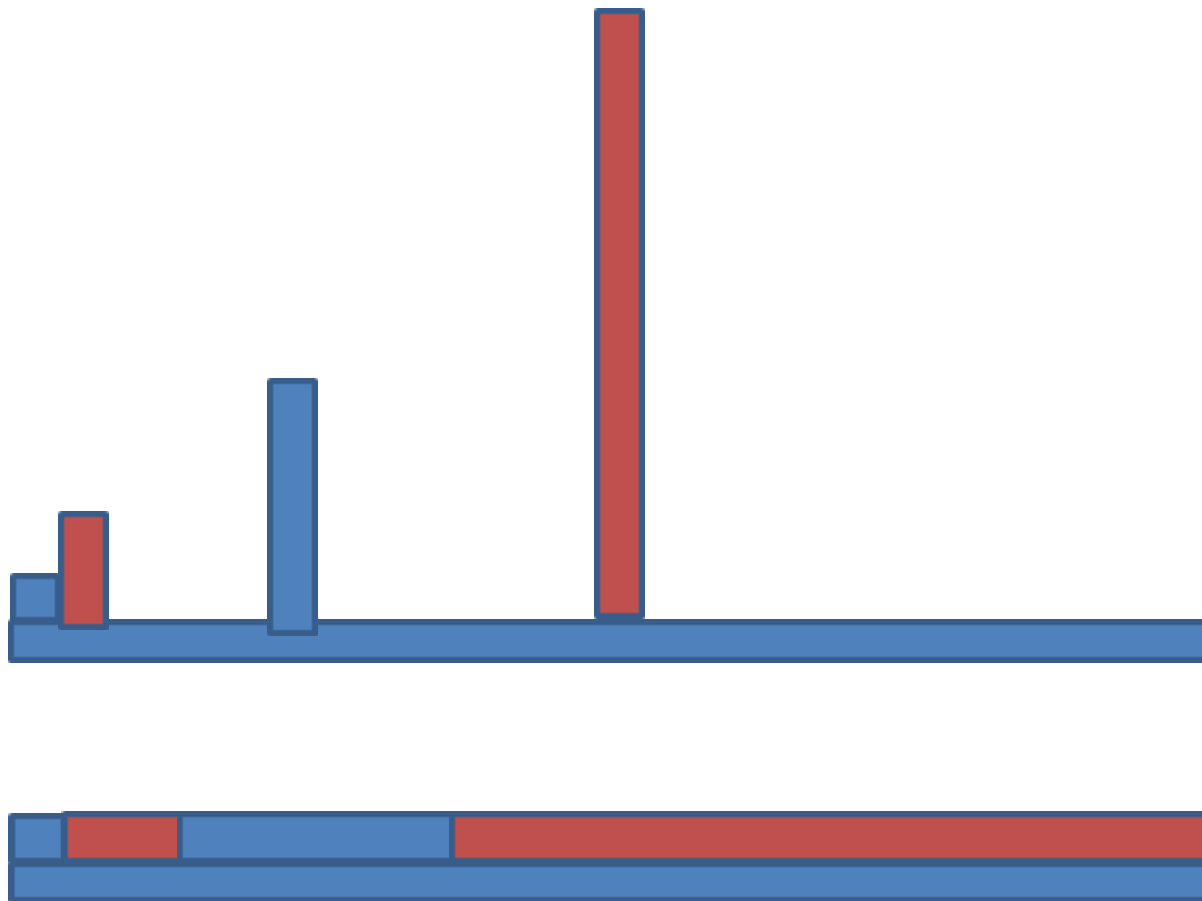
Express as a closed-form
expression in terms of N

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
sometimes
called the
amortized cost

Average overhead for doubling scheme



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

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

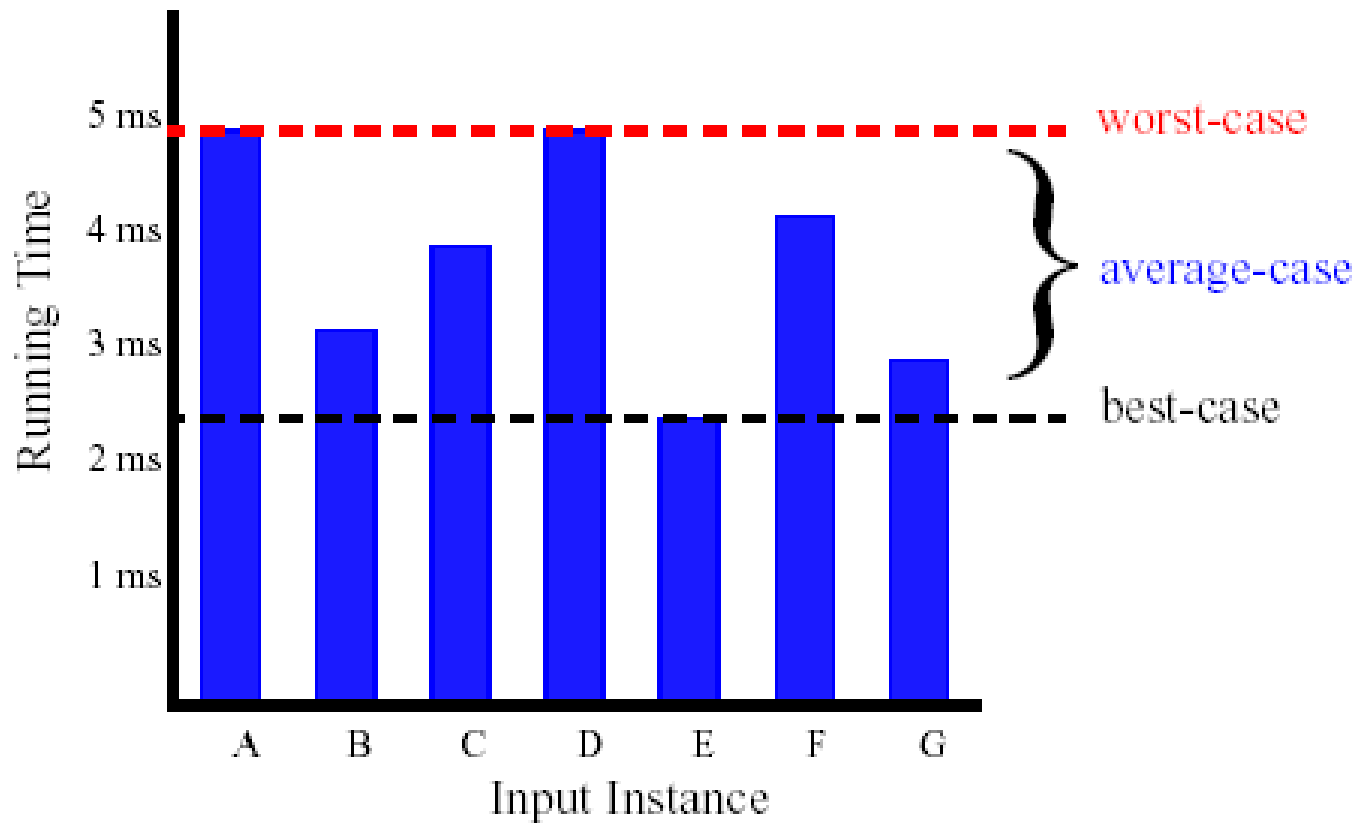
$$k= \log(n+1)/3$$

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" time complexity?
- ▶ What do we mean by "Average Case" time complexity?
- ▶ 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



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

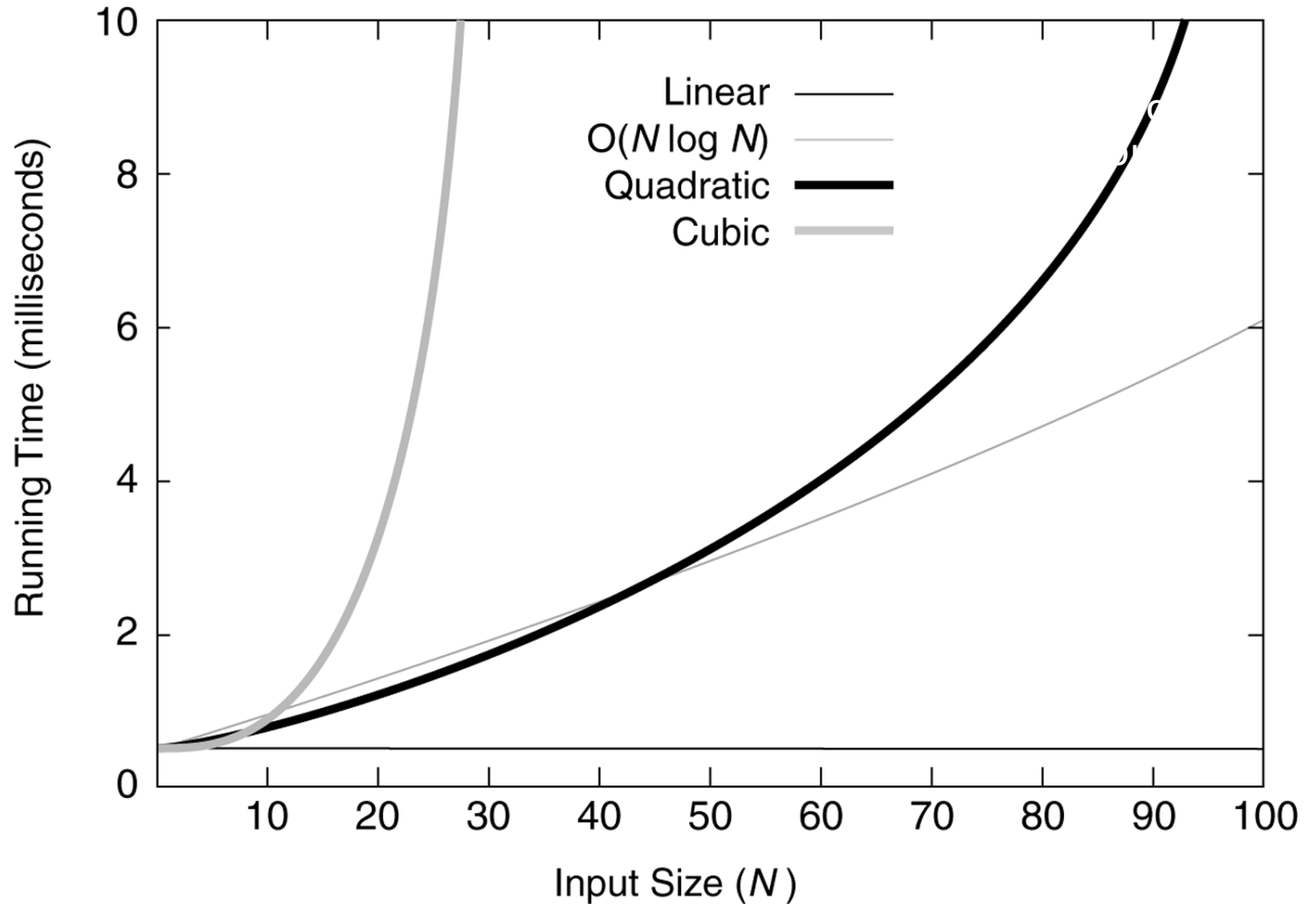


Figure 5.2

Running times for moderate inputs

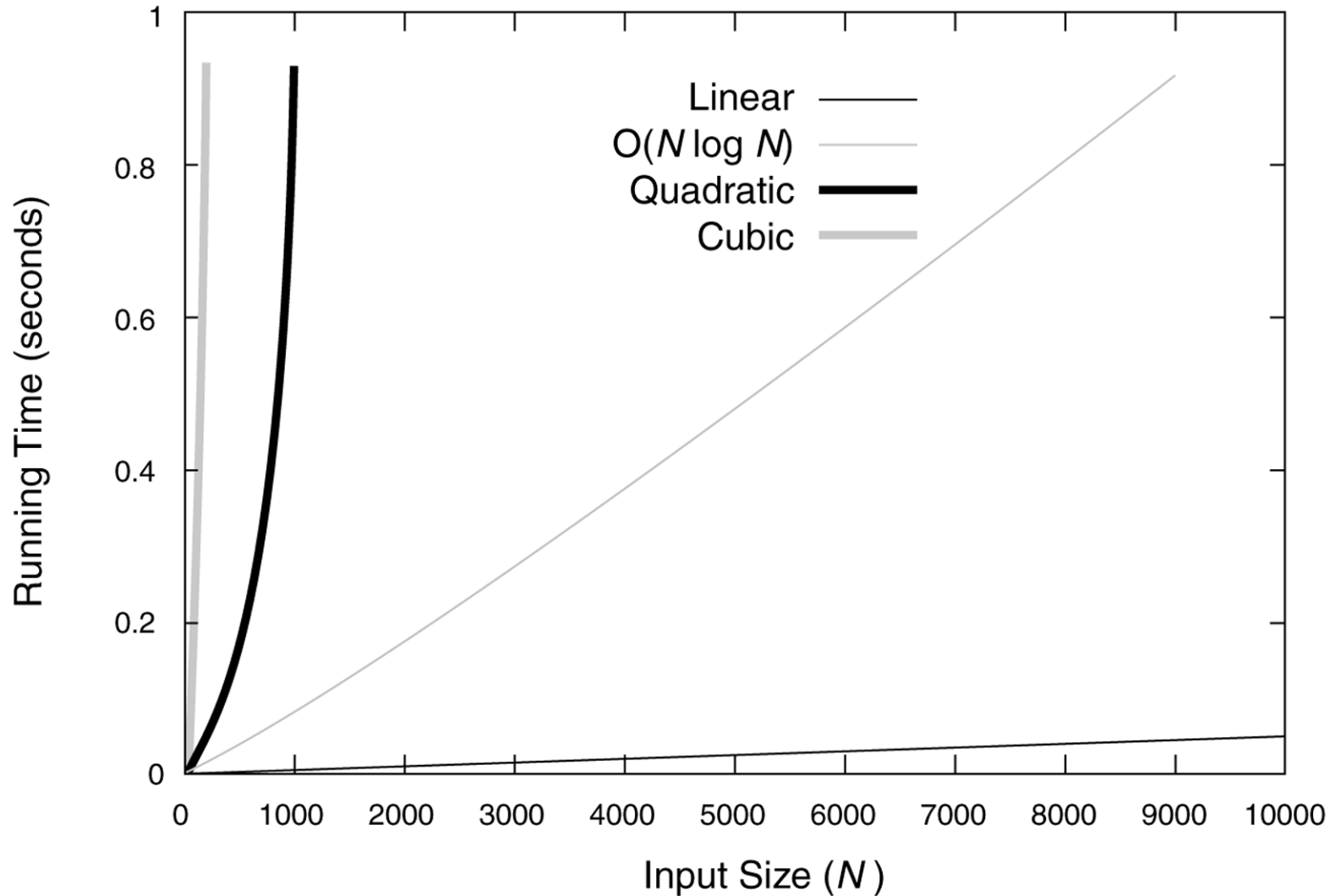


Figure 5.3

Functions in order of increasing growth rate

FUNCTION	NAME
c	Constant
$\log N$	Logarithmic
$\log^2 N$	Log-squared
N	Linear
$N \log N$	$N \log N$ ← a.k.a "log linear"
N^2	Quadratic
N^3	Cubic
2^N	Exponential

Simple Rule for Big-Oh

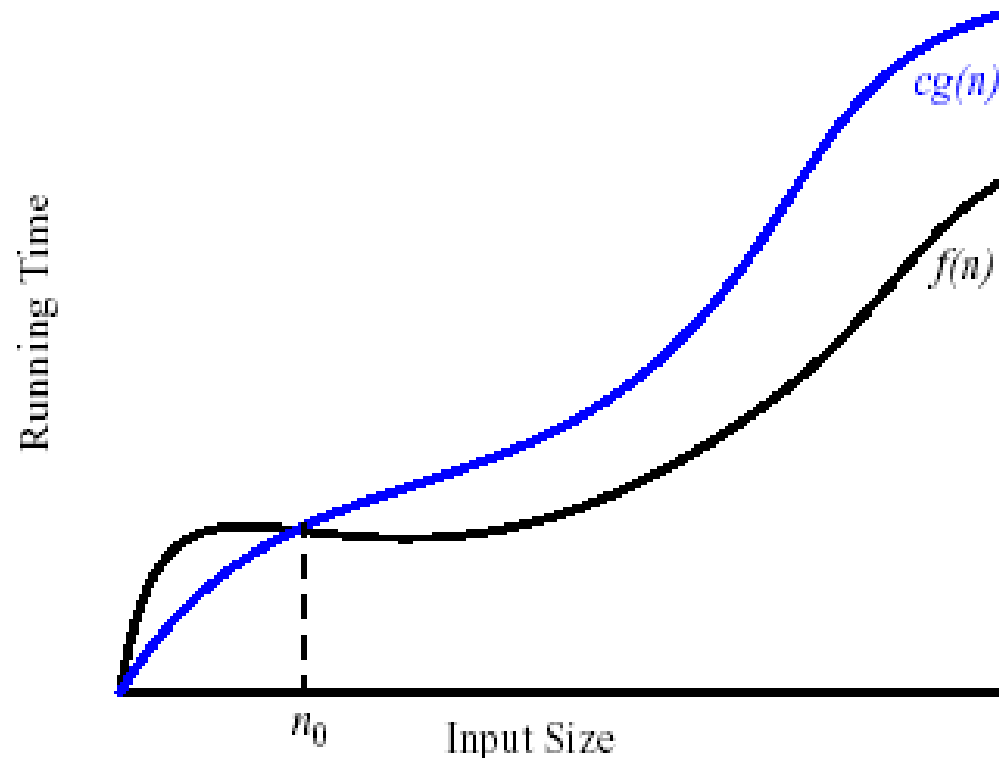
- ▶ Drop lower order terms and constant factors
- ▶ $7n - 3$ is $O(n)$
- ▶ $8n^2 \log n + 5n^2 + n$ is $O(n^2 \log n)$

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



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.
- ▶ $f(n) = n + 12$, $g(n) = ???$.
- ▶ $f(n) = n + \sin(n)$, $g(n) = ???$
- ▶ $f(n) = n^2 + \text{sqrt}(n)$, $g(n) = ???$

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.

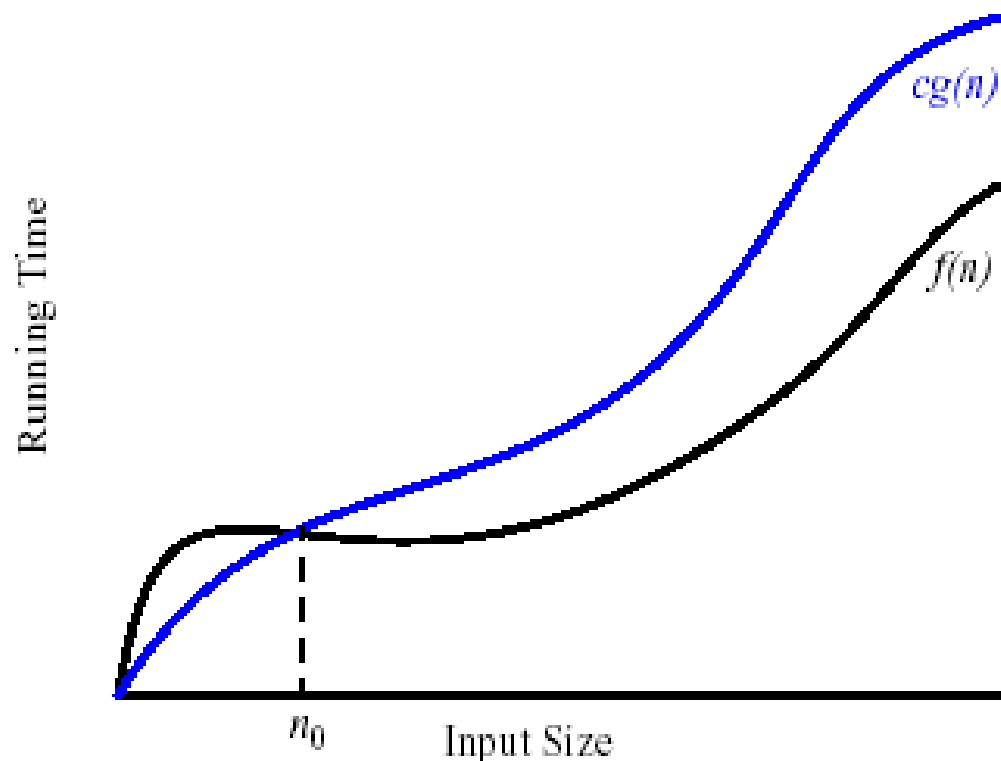
$\Omega?$

$\Theta?$

- 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



Q7b,c

Theta is a tighter bound

- ▶ If we ask “true or false: $3n+2$ is $O(n^3)$ ”, what’s the answer?
 - True!
- ▶ And if we ask “true or false: $3n+2$ is $\Theta(n^3)$ ”, what’s the answer?
 - False

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 say: $3n+2$ is $O(5n-3\log(n) + 17)$
 - And it would be technically correct...
 - It would also be poor taste ... and put me in a bad mood.

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

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
2. $\log n$ and n (on these questions and solutions ONLY, let $\log n$ mean natural log)
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,

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