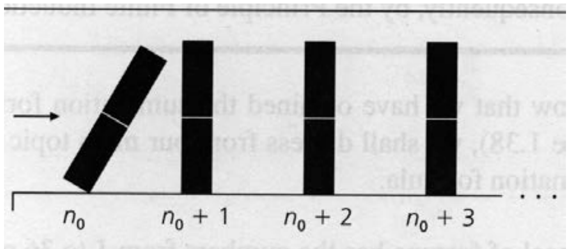
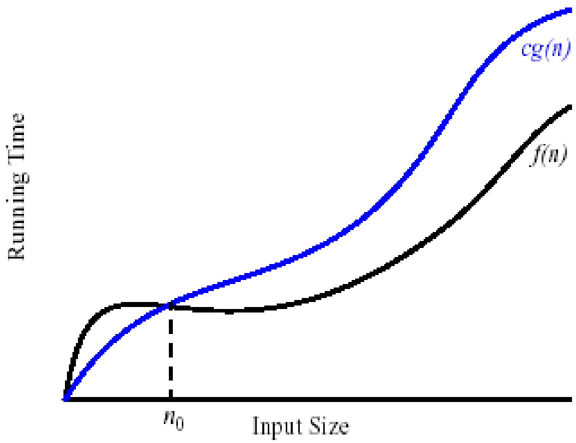
Sit with your "Growable Arrays" partner.





CSSE 230 Day 2

Growable Arrays Continued
Induction intro
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?

Agenda

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

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

Warm Up and Stretching thoughts

- Short but intense! ~35 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 tonight
  - 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	sum(i, i=5..19) = 180 **using Maple**
21	5 + 10 + 20 = 35	sum(i, i=5..20) = 180
40	35	sum(i, i=5..39) = 770
41	5 + 10 + 20 + 40 = 75	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	$= 5(2^k) + 1$	$5(1 + 2 + 4 + 8 + \ldots + 2^k)$

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

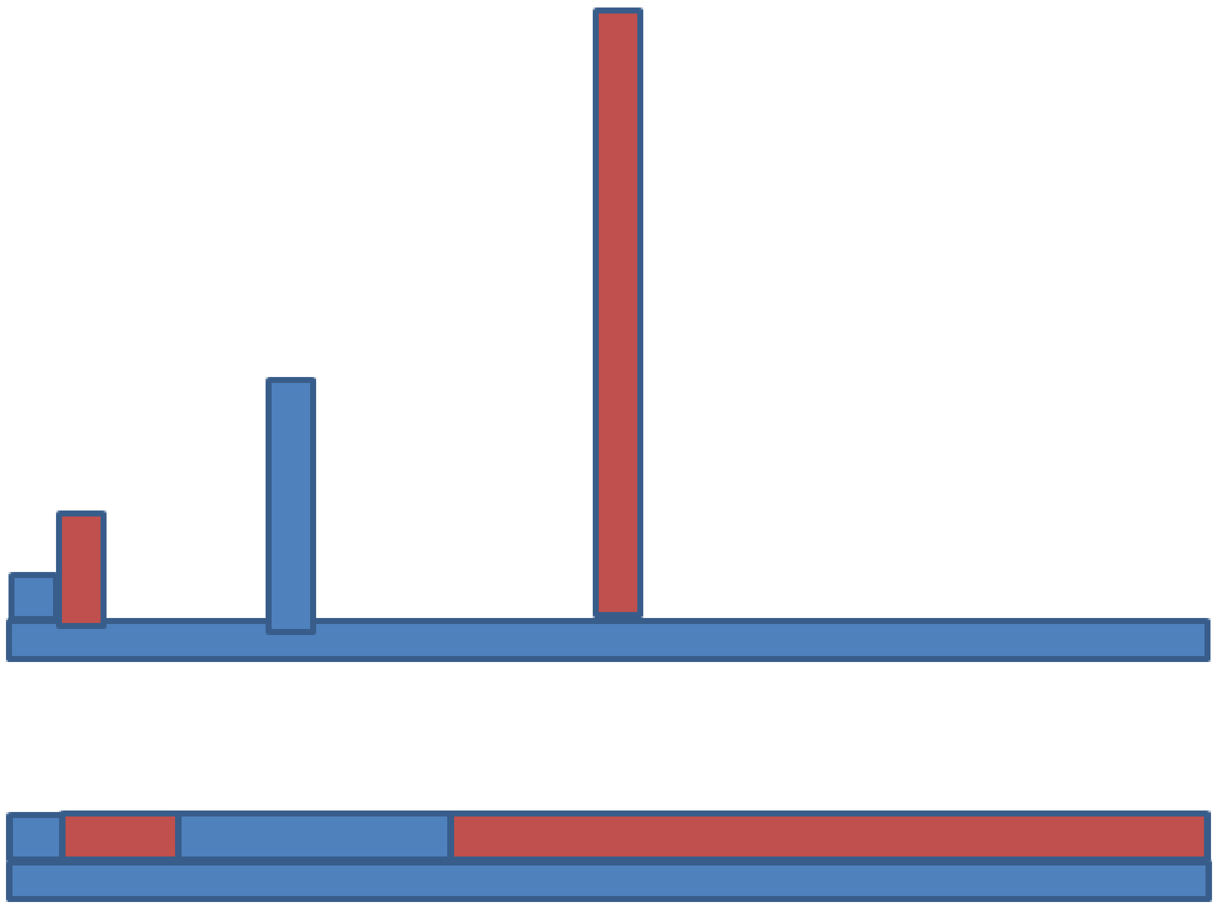
- What's the **average** overhead cost of adding an additional string…
  - in the doubling case?
  - in the add-one case?

- So which should we use?

This is sometimes called the **amortized** cost

Q4-5

A way to picture the overhead



More math review

Q6

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:

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

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

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

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

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

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. $\log(2\ n \log n)$
2. $\log(n/2)$
3. $\log(\mathrm{sqrt}(n))$
4. $\log(\log(\mathrm{sqrt}(n)))$

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)

Mathematical Induction

What it is? Why is it a legitimate proof method?

What is mathematical induction?

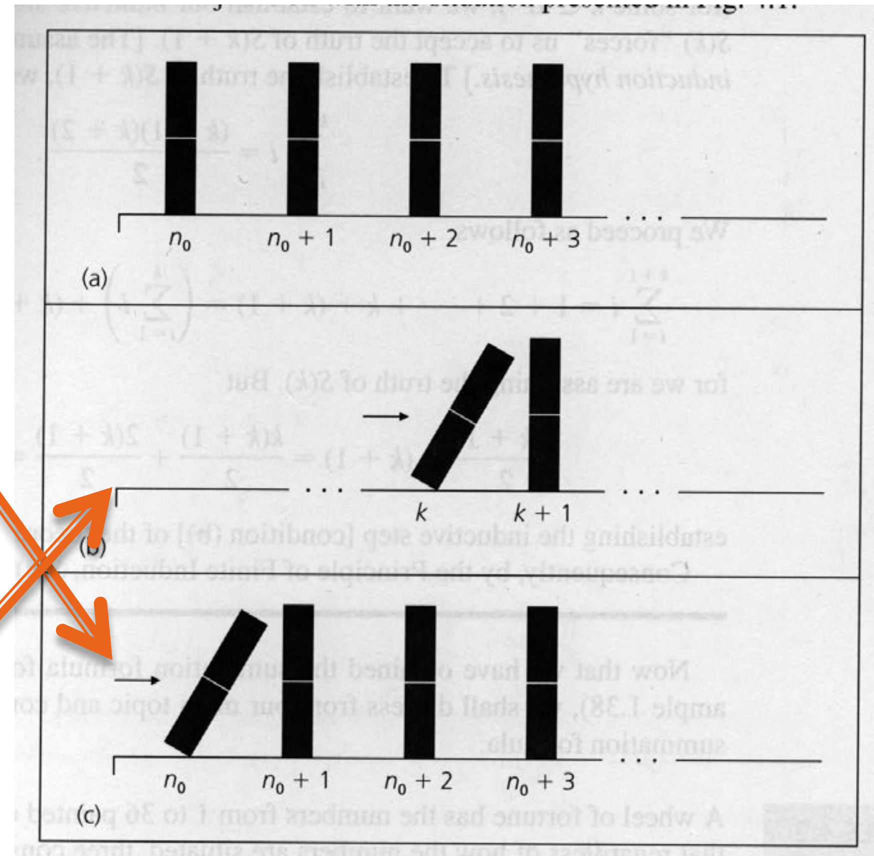
- Goal: For some boolean-valued property p(n), and some integer constant n_0, prove that **p(n)** is true for all integers $n \geq n_0$
- Technique:
  - Show that **p(n_0)** is true
  - Show that for all $k \geq n_0$, **p(k) implies p(k+1)**

  That is, show that whenever p(k) is true, then p(k+1) is also true.

Q7

Why does induction work?

- Goal: prove that p(n) is true for all n ≥ n0
- Technique:
  - Show that p(n0) is true
  - Show that for all $k \geq n_0$, p(k) implies p(k+1)



From Ralph Grimaldi's discrete math book.

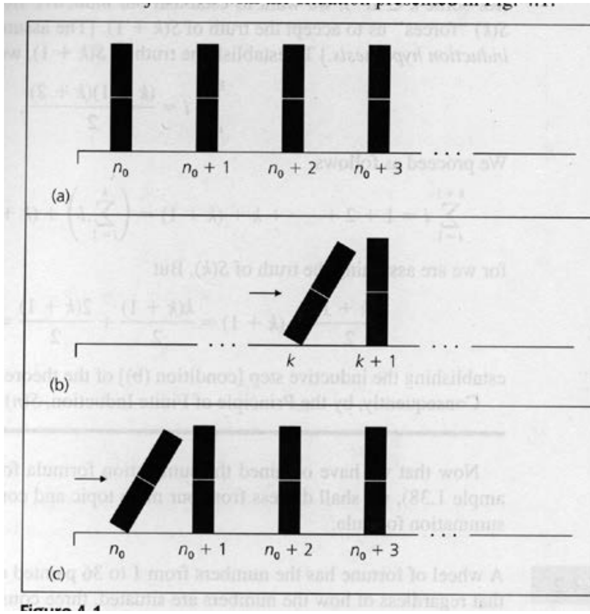
dominoes video

More formally

- We can prove induction works using one assumption: the Well-Ordering Principle
- The Well-Ordering Principle says
  - Every non-empty set of non-negative integers has a smallest element

Note: This slide and the next two are no longer part of CSSE 230. They are included in case you are interested. You will not be required to know or understand their contents

Sketch of Proof that Induction works



- Given:
  - $p(n_0)$ is true, and
  - For all $k \geq n_0$, $p(k)$ implies $p(k+1)$
- Then:
  - Let $S = \{n \geq n_0 : p(n) \text{ is false}\}$. Intuitively, S is dominoes that don't fall down
  - Assume S isn't empty and show that it leads to a contradiction.
  - By WOP, S has a minimum element, call it n_{min}
  - $n_{min} > n_0$ by first "given" and definition of S
  - So $n_{min} - 1 \geq n_0$ and $p(n_{min} - 1)$ is true
    - $p(n_{min} - 1)$ is true or else $n_{min} - 1$ would be in S, and so n_{min} would not be the smallest element
  - By second "given", $p(n_{min} - 1 + 1) = p(n_{min})$ is true
  - Ack! It's both true and false! So S must actually be empty

Proof that induction works

In case you want more details than we did in class.

- Hypothesis:
  a) $p(n_0)$ is true.
  b) For all $k \geq n_0$, $p(k)$ implies $p(k+1)$.

  Desired **Conclusion**: If n is any integer with $n \geq n_0$, then $p(n)$ is true. **If this conclusion is true, induction is a legitimate proof method.**

- **Proof**: Assume a) and b). Let S be the set $\{n \geq n_0 : p(n) \text{ is false}\}$. **We want to show that S is empty**; we do it by contradiction.
  - **Assume that S is non-empty.** Then the well-ordering principle says that S has a smallest element (call it s_{min}). We try to show that this leads to a contradiction.
  - Note that $p(s_{min})$ has to be false. **Why?**
  - s_{min} cannot be n_0, by hypothesis (a). Thus s_{min} must be $> n_0$. **Why?**
  - Thus $s_{min}-1 \geq n_0$. Since s_{min} is the smallest element of S, $s_{min} - 1$ cannot be an element of S. **What does this say about $p(s_{min} - 1)$?**
    - $p(s_{min} - 1)$ is true.
  - By hypothesis (b), using the $k = s_{min} - 1$ case, $p(s_{min})$ is also true. This **contradicts** the previous statement that $p(s_{min})$ is false.
  - Thus **the assumption that led to this contradiction** (S is nonempty) **must be false**.
  - Therefore S is empty, and $p(n)$ is true for all $n \geq n_0$.

A simple proof by induction

- P(n): $1 + 2 + 3 + \ldots + n = n(n+1)/2$.

- Base case
- Induction hypothesis
- Induction step

Q7

Interlude

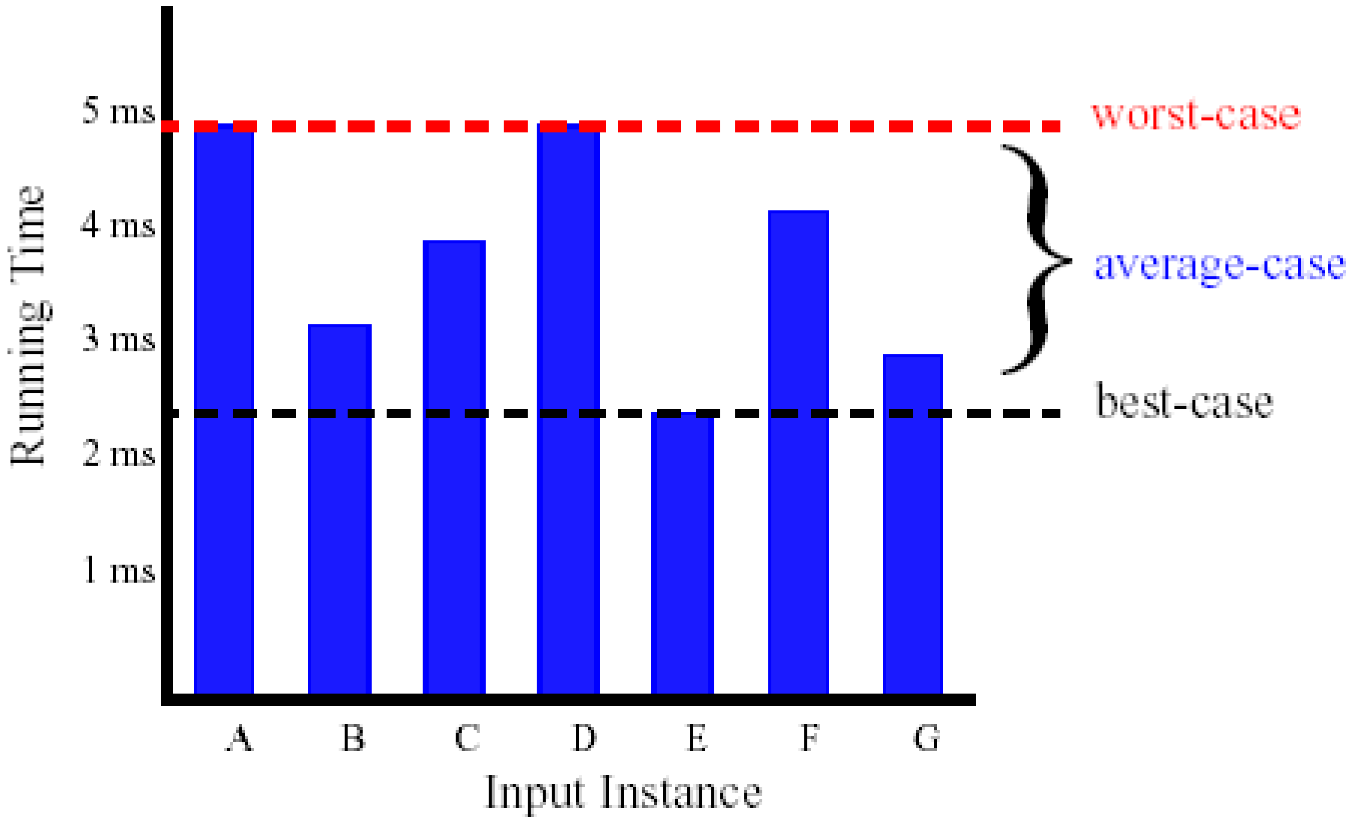
On Liquor Production
by David M. Smith

A friend who's in liquor production
Owns a still of astounding construction.
The alcohol boils
Through old magnetic coils;
She says that it's "proof by induction."

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

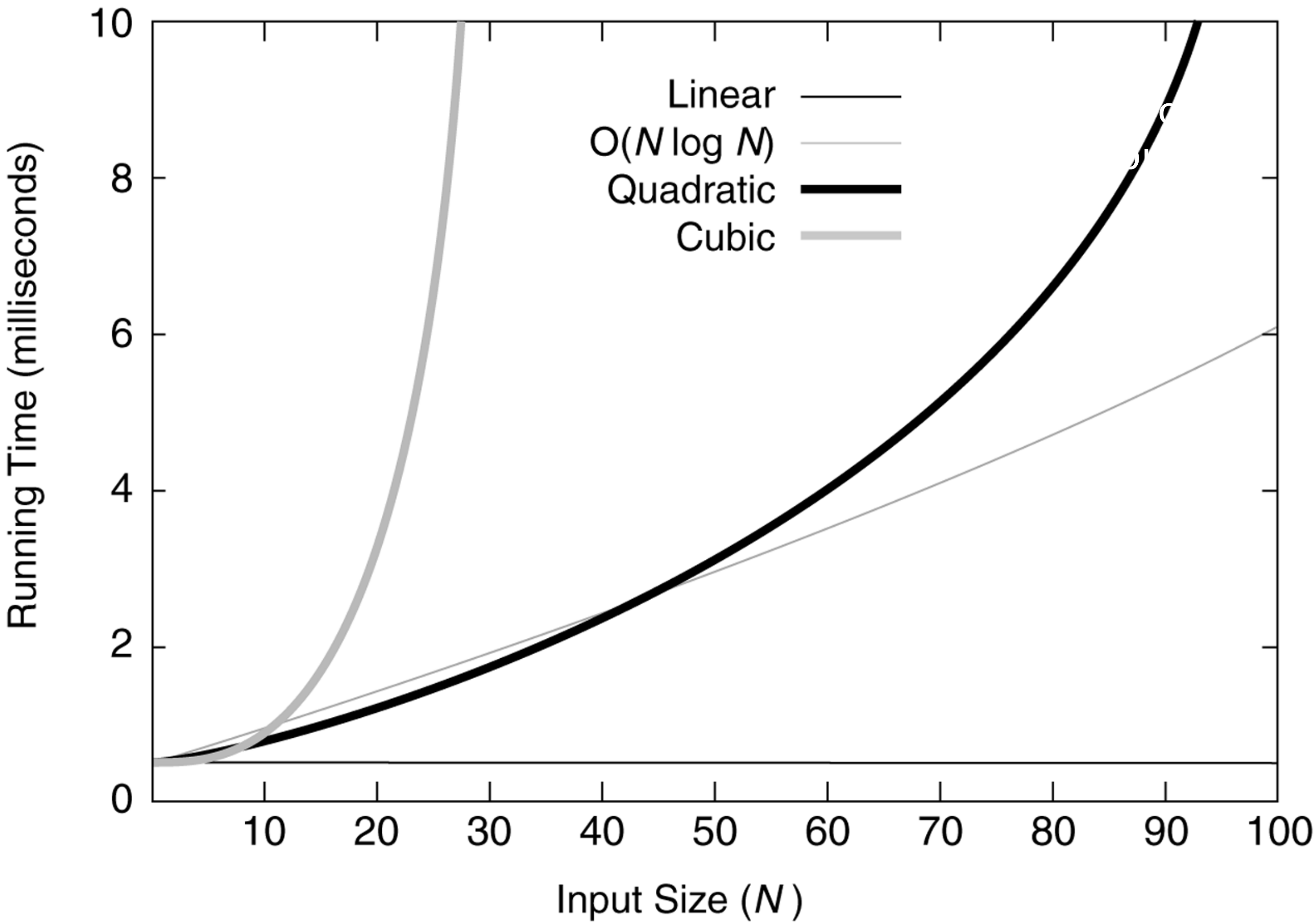


Figure 5.2

Running times for moderate inputs

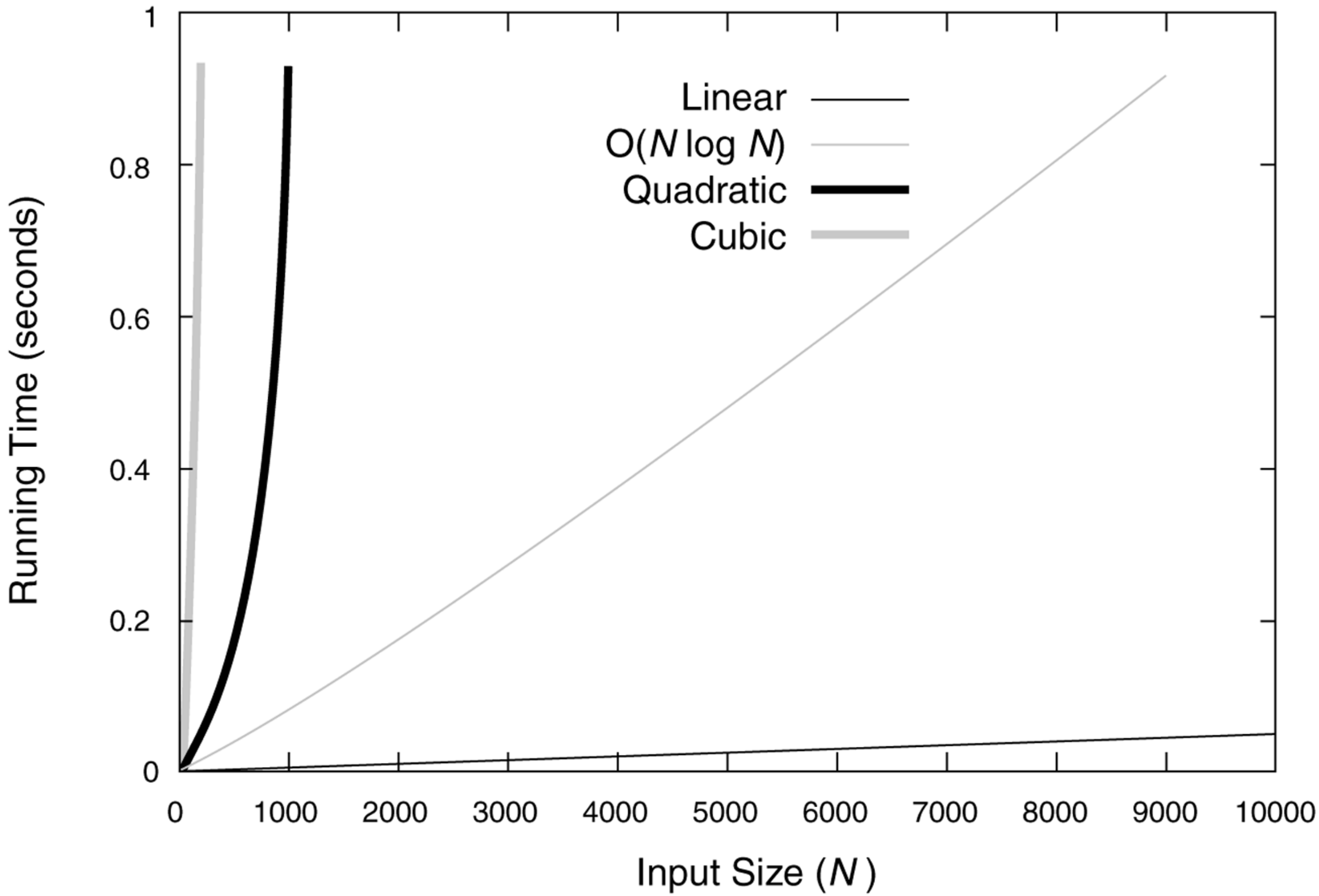


Figure 5.3

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
N^2	Quadratic
N^3	Cubic
2^N	Exponential

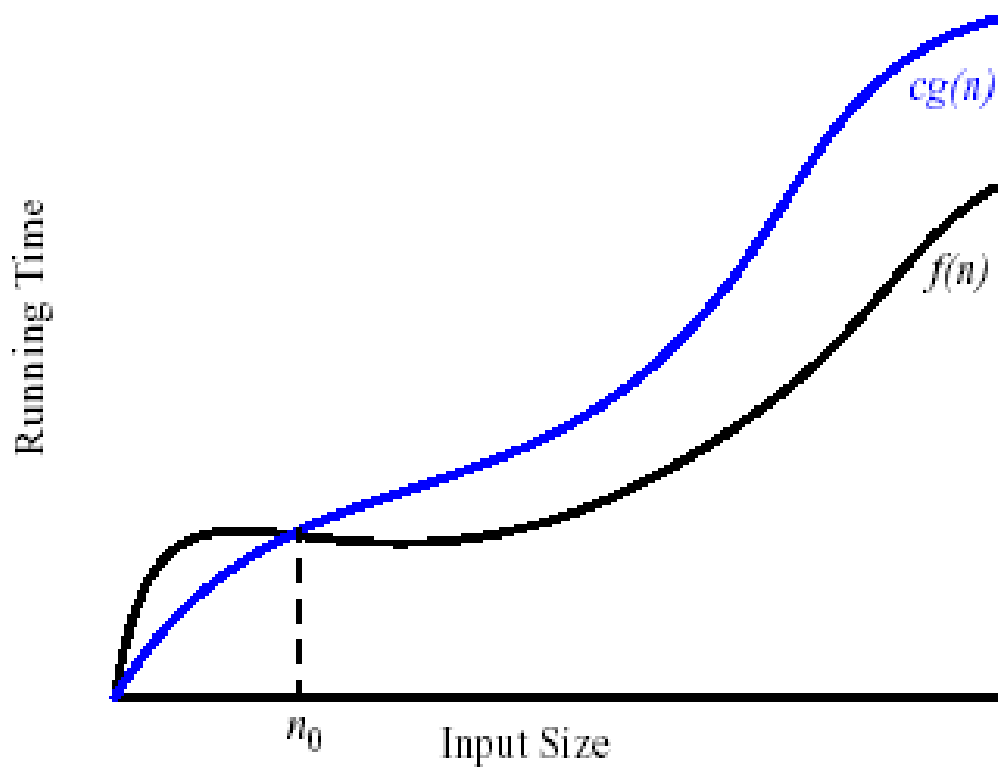
a.k.a "log linear"

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



Big Oh examples

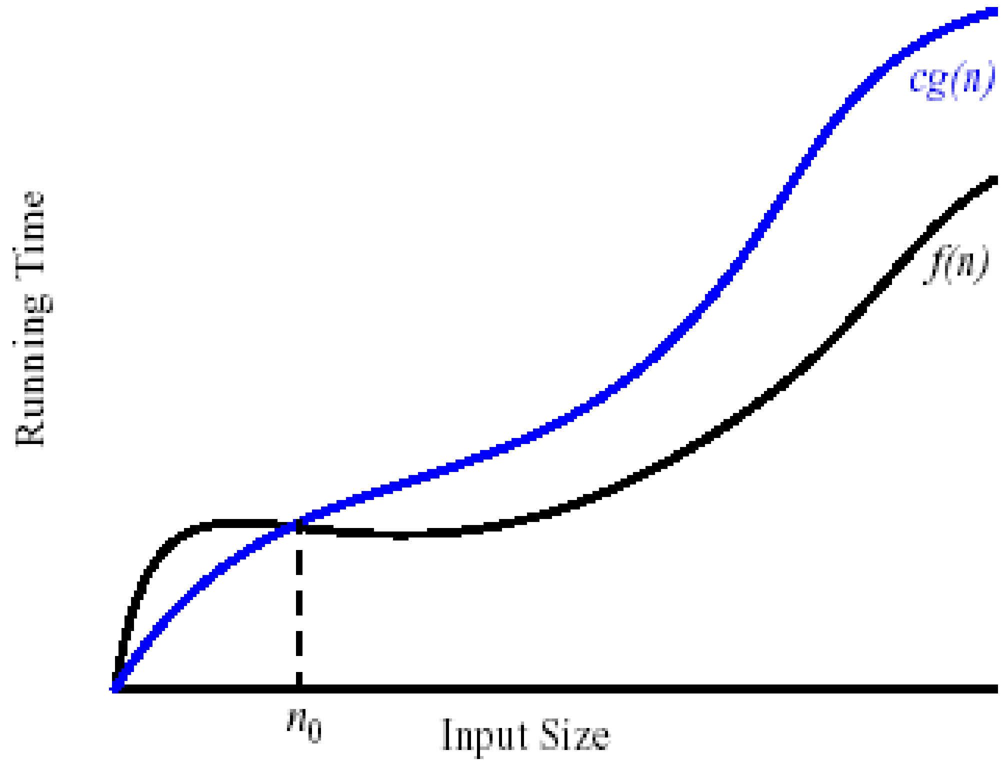
- A function f(n) is (in) O(g(n)) if there exist two positive constants **c** and $\mathbf{n_0}$ such that for all $n \geq n_0$, $f(n) \leq c\ g(n)$
- **So all we must do to prove that f(n) is O(g(n)) is produce two such constants.**
- $f(n) = n + 12$, $g(n) =$???.
- $f(n) = n + \sin(n)$, $g(n) =$???
- $f(n) = n^2 + \mathrm{sqrt}(n)$, $g(n) =$???

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.

Q8-10

Ω?
Θ?

- 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



Q11

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 $O(n)$
- 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.
- But… if I ask "true or false: $3n+2$ is $O(n^3)$", what's the answer?
  - True!

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

Q12

Limits and Asymptotics

- Consider the limit

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

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

Q13

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,

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

and:

$$\text{If } f(x) = \log x \text{ then } f'(x) = 1/x$$