Cost of Sorts and Asymptotic Growth

(and addition!)

One-Slide Summary

- **g is in O(f)** iff there exist positive constants *c* and n_0 such that $g(n) \le cf(n)$ for all $n \ge n_0$.
- If g is in O(f) we say that f is an **upper bound** for g.
- We use **Omega Ω** for lower bounds and **Theta Θ** for tight bounds.
- To **prove** that g is in O(f) you must find the constants *c* and n_0.
- We can add two numbers with electricity.

Outline

- Sorting: timing and costs
- Big Oh: upper bound
- Big Omega: lower bound
- Big Theta: tight bound

- Time Permitting:
- Adding Two Numbers With Electricity

Administrivia

- Don't forget to turn in your fractal
  - Separate form and button on adjudicator
- Late policy for PS3 Code
  - about 10% per full day
- Reading
  - In Chapter 6 of the Course Book ("Machines"), there is a running example about computing and implementing logic with a particular substance that you can pour. What was it? One word answer.

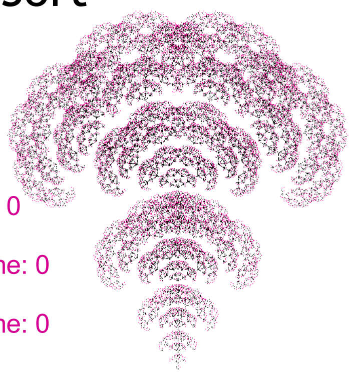
Sorting Cost

Recall "simple sorting": Find best card, take it out and set it aside, repeat.

- What grows?
  - n = the number of elements in lst
- How much work are the pieces?
  - find-best: work scales as n (increases by one)
  - delete: work scales as n (increases by one)
- How many times does sort evaluate find-best and delete? n
- Total cost: scales as n^2

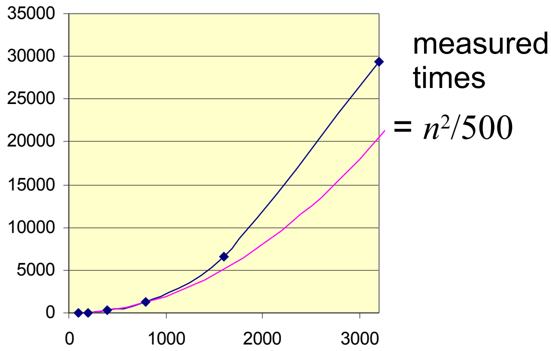
Timing Sort

```
> (time (sort < (revintsto 100)))
cpu time: 20 real time: 20 gc time: 0
> (time (sort < (revintsto 200)))
cpu time: 80 real time: 80 gc time: 0
> (time (sort < (revintsto 400)))
cpu time: 311 real time: 311 gc time: 0
> (time (sort < (revintsto 800)))
cpu time: 1362 real time: 1362 gc time: 0
> (time (sort < (revintsto 1600)))
cpu time: 6650 real time: 6650 gc time: 0
```



Cherry Blossom
by Ji Hyun Lee, Wei Wang

Timing Sort



Sorting Cost

```
(define (sort lst cf)
  (if (null? lst) lst
      (let ((best (find-best lst cf)))
        (cons best (sort (delete lst best) cf)))))
(define (find-best lst cf)
   (if (= 1 (length lst)) (car lst)
       (pick-better cf (car lst) (find-best (cdr lst) cf))))
```

If we double the length of the list, the amount of work *approximately* quadruples: there are twice as many applications of find-best, and each one takes twice as long!

Growth Notations

- $g \in O(f)$ ("Big-Oh")
  g grows **no faster than** f (f is upper bound)
- $g \in \Theta(f)$ ("Theta")
  g grows **as fast as** f (f is tight bound)
- $g \in \Omega(f)$ ("Omega")
  g grows **no slower than** f (f is lower bound)

Which one would we most like to know?

Meaning of O ("big Oh")

g is in $O(f)$ iff:

There are positive constants c and n_0 such that

$$g(n) \leq cf(n)$$

for all $n \geq n_0$.

O Examples

g is in $O(f)$ iff there are positive constants c and n_0 such that $g(n) \leq cf(n)$ for all $n \geq n_0$.

Is n in $O(n^2)$?

Is $10n$ in $O(n)$?

Is n^2 in $O(n)$?

O Examples

g is in $O(f)$ iff there are positive constants c and n_0 such that $g(n) \leq cf(n)$ for all $n \geq n_0$.

Is n in $O(n^2)$? Yes, $c = 1$ and $n_0=1$ works.

Is $10n$ in $O(n)$? Yes, $c = .09$ and $n_0=1$ works.

Is n^2 in $O(n)$? No, no matter what c we pick, $cn^2 > n$ for big enough n ($n > c$)

Ω ("Omega"): Lower Bound

g is in $O(f)$ iff there are positive constants c and n_0 such that $g(n) \le cf(n)$ for all $n \ge n_0$.

***g* is in Ω (*f*) iff** there are positive constants c and n_0 such that

$$g(n) \ge cf(n)$$

for all $n \ge n_0$.

Proof Techniques

- Theorem:

  There exists a polygon with four sides.

- Proof:

It is a polygon.
It has four sides.
QED.

What kind of proof is this?

Proof by Construction

- We can prove a "there exists an *X* with property *Y*" theorem, but showing an *X* that has property *Y*
- $O(f)$ means "**there are** positive constants c and n_0 such that $g(n) \le cf(n)$ **for all** $n \ge n_0$
- So, to prove g is in $O(f)$ we need to find c and n_0 and show that $g(n) \le cf(n)$ **for all** $n \ge n_0$

Dis-Proof by Construction

- To prove g is **not** in $O(f)$:
- $O(f)$ means: there are positive constants c and n_0 such that $g(n) \le cf(n)$ for all $n \ge n_0$
- So, to prove g is **not** in $O(f)$ we need to find a way given **any** c and n_0, to find an $n \ge n_0$ such that $g(n) > cf(n)$.

Curious why $\neg\exists c.\exists n_0.P(c,n_0) = \forall c.\forall n_0.\neg P(c,n_0)$?
Discrete math!

Liberal Arts Trivia: Archaeology

- In archaeology, this term is used to describe the exposure, processing and recording of archaeological remains. A related subconcept is stratification: relationships exist between different events in the same location or context. Sedimentary layers are deposited in a time sequence, with the oldest on the bottom and the youngest on top.

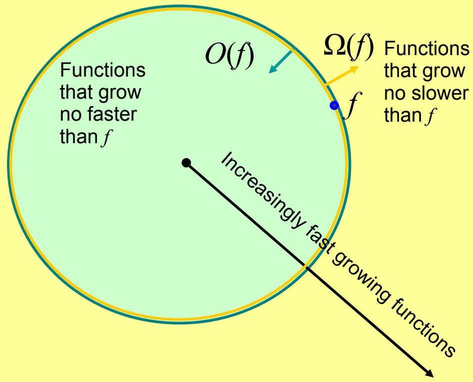
Liberal Arts Trivia: Creative Writing

- This technique is one of the four rhetorical modes of discourse, along with argumentation, description and narration. Its purpose is to inform, explain, analyze or define. When done poorly, it is sometimes referred to as a "information dump" or "plot dump".

Growth Notations

- $g \in O(f)$ ("Big-Oh")
  - g grows **no faster than** f (upper bound)
- $g \in \Theta(f)$ ("Theta")
  - g grows **as fast as** f (tight bound)
- $g \in \Omega(f)$ ("Omega")
  - g grows **no slower than** f (lower bound)

The Sets $O(f)$ and $\Omega(f)$



O and Ω Examples

g is in $O(f)$ iff there are positive constants c and n_0 such that $g(n) \leq cf(n)$ for all $n \geq n_0$.

g is in $\Omega(f)$ iff there are positive constants c and n_0 such that $g(n) \geq cf(n)$ for all $n \geq n_0$.

- n is in $\Omega(n)$
  - ?
- $10n$ is in $\Omega(n)$
  - ?
- Is n^2 in $\Omega(n)$?
  - ?
- n is in $O(n)$
  - ?
- $10n$ is in $O(n)$
  - ?
- n^2 is **not** in $O(n)$
  - ?

O and Ω Examples

g is in $O(f)$ iff there are positive constants c and n_0 such that $g(n) \leq cf(n)$ for all $n \geq n_0$.

g is in $\Omega(f)$ iff there are positive constants c and n_0 such that $g(n) \geq cf(n)$ for all $n \geq n_0$.

- n is in $\Omega(n)$
  - Yes, pick $c = 1$
- $10n$ is in $\Omega(n)$
  - Yes, pick $c = 1$
- Is n^2 in $\Omega(n)$?
  - Yes! (pick $c = 1$)
- n is in $O(n)$
  - Yes, pick $c = 1$
- $10n$ is in $O(n)$
  - Yes, pick $c = 10$
- n^2 is **not** in $O(n)$
  - Pick $n > c$

Example

- Is n in $\Omega(n^2)$?

$n \geq cn^2$ for all $n \geq n_0$

$1 \geq cn$ for all $n \geq n_0$

No matter what c is, I can make this false by using $n = (1/c + 1)$

g is in $\Omega(f)$ iff there are positive constants c and n_0 such that $g(n) \geq cf(n)$ for all $n \geq n_0$.

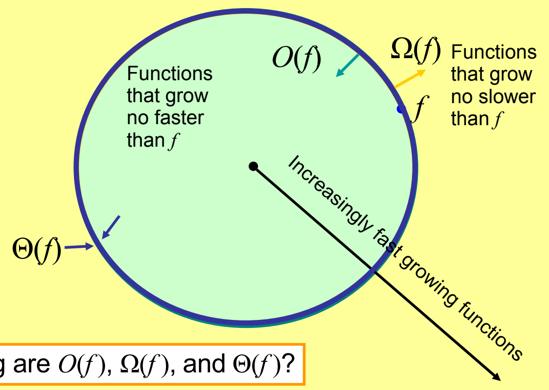
Θ ("Theta"): Tight Bound

g is $\Theta(f)$ iff

g is in $O(f)$

and g is in $\Omega(f)$

The Sets $O(f)$, $\Omega(f)$, and $\Theta(f)$



How big are $O(f)$, $\Omega(f)$, and $\Theta(f)$?

Θ Examples

- Is $10n$ in $\Theta(n)$?
- Is n^2 in $\Theta(n)$?
- Is n in $\Theta(n^2)$?

Θ Examples

- Is $10n$ in $\Theta(n)$?
  - **Yes**, since $10n$ is $\Omega(n)$ and $10n$ is in $O(n)$
    - Doesn't matter that you choose different c values for each part; they are independent
- Is n^2 in $\Theta(n)$?
  - **No**, since n^2 is not in $O(n)$
- Is n in $\Theta(n^2)$?
  - **No**, since n^2 is not in $\Omega(n)$

Recall: Sorting Cost

- What grows?
  - n = the number of elements in lst
- How much work are the pieces?
  - find-best: work scales as n (increases by one)
  - delete: work scales as n (increases by one)
- How many times does sort evaluate find-best and delete? n
- Total cost: scales as n^2

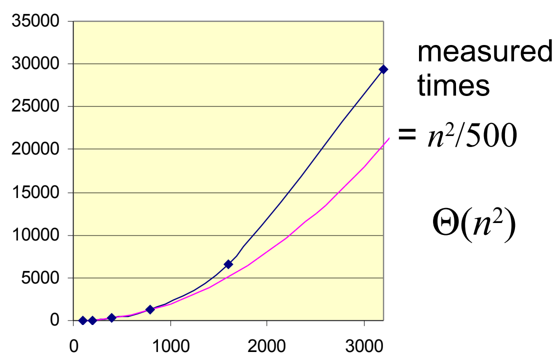
Sorting Cost

```
(define (sort lst cf)
  (if (null? lst) lst
      (let ((best (find-best lst cf)))
        (cons best (sort (delete lst best) cf)))))
(define (find-best lst cf)
   (if (= 1 (length lst)) (car lst)
        (pick-better cf (car lst) (find-best (cdr lst) cf))))
```

If we double the length of the list, the amount of work *approximately* quadruples: there are twice as many applications of find-best, and each one takes twice as long.

The running time of this sort procedure is in $\Theta(n^2)$

Timing Sort



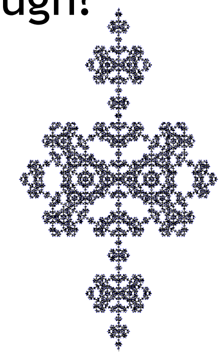
Is our sort good enough?

Takes over 1 second to sort 1000-length list. How long would it take to sort 1 million items?

1s = time to sort 1000
4s ~ time to sort 2000

1M is 1000 * 1000

Sorting time is n^2
so, sorting 1000 times as many items will take 1000^2 times as long = 1 million seconds ~ 11 days



Eyes
by John Devor
and Eric Montgomery

Note: there are 800 Million VISA cards in circulation.
It would take 20,000 years to process a VISA transaction at this rate.

Which of these is true?

- Our sort procedure is too slow for VISA because its running time is in $O(n^2)$
- Our sort procedure is too slow for VISA because its running time is in $\Omega(n^2)$
- Our sort procedure is too slow for VISA because its running time is in $\Theta(n^2)$

Which of these is true?

- ~~Our sort procedure is too slow for VISA because its running time is in $O(n^2)$~~
- Our sort procedure is too slow for VISA because its running time is in $\Omega(n^2)$
- Our sort procedure is too slow for VISA because its running time is in $\Theta(n^2)$

Knowing a running time is in $O(f)$ tells you the running time is not *worse* than f. This can *only* be good news. It doesn't tell you anything about how bad it is. *(Lots of people and books get this wrong.)*

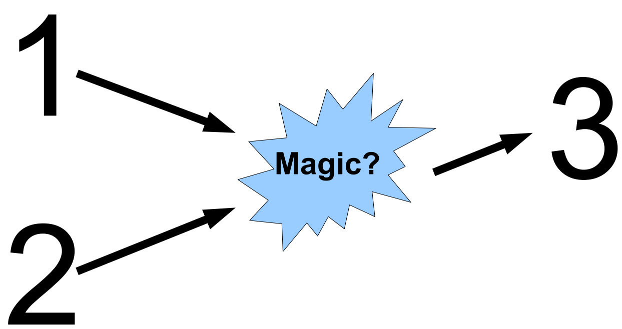
Liberal Arts Trivia: Dance

- This four wall line dance was created in 1976 by American dancer Ric Silver. It was popularized by Marcia Griffiths and remains a perennial wedding favorite. Steps: 1-4 grapevine right (tap and clap on 4), 5-8 grapevine left (tap and clap on 8), 9-12 walk back (tap and clap on 12), etc. The lyrics include "I'll teach you the ..."

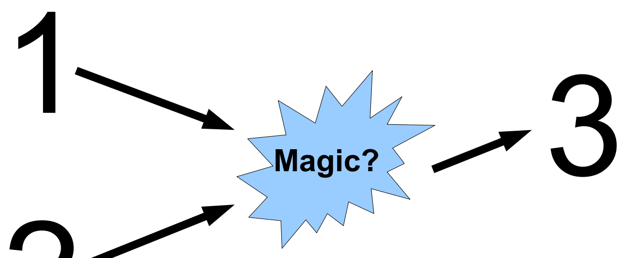
Liberal Arts Trivia: Medieval Studies

- This son of Pippin the Short was King of the Franks from 768 to his death and is known as the "father of Europe": his empire united most of Western Europe for the first time since the Romans. His rule is associated with the Carolingian Renaissance, a revival of art, religion and culture. The word for *king* in various Slavic languages (e.g., Russian, Plish, Czech) was coined after his name.

How To Add Two Numbers With Electricity

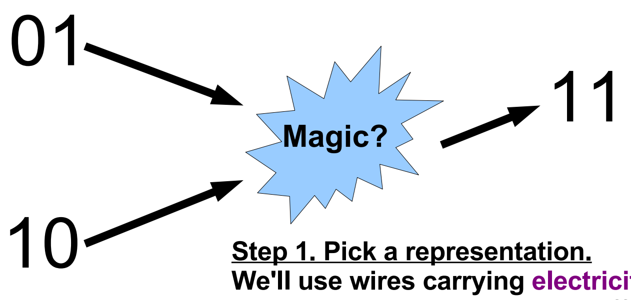


How To Add Two Numbers With Electricity



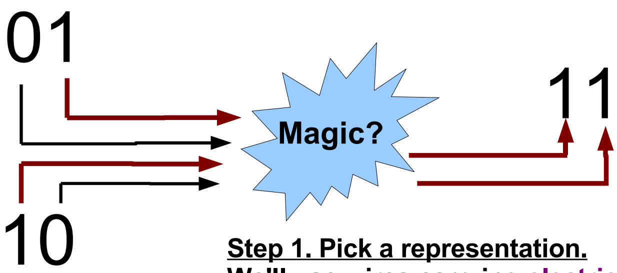
<u>Step 1. Pick a representation.</u>
We'll use wires carrying electricity.
Each wire will either be on or off.
Each wire will thus encode one bit.
We'll represent our numbers in binary.

How To Add Two Numbers With Electricity



<u>Step 1. Pick a representation.</u>
We'll use wires carrying electricity.
Each wire will either be on or off.
Each wire will thus encode one bit.
We'll represent our numbers in binary.

How To Add Two Numbers With Electricity



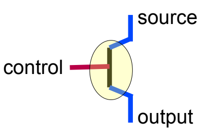
<u>Step 1. Pick a representation.</u>
We'll use wires carrying electricity.
Each wire will either be on or off.
Each wire will thus encode one bit.
We'll represent our numbers in binary.

Still Adding Numbers

- What does it mean for a wire to be "on"?
  - We'll use voltage.
  - Ex: bit 0 is 0V to 0.8V and bit 1 is 2V to 5V
- Great. So how do I combine and manipulate voltages?
  - Example: 0+0 = 0
  - Example: 0+1 = 1
  - Example: 1+0 = 1
  - Somehow I need the output to be "on" if either of the inputs is "on". How do I do it?

The Transistor

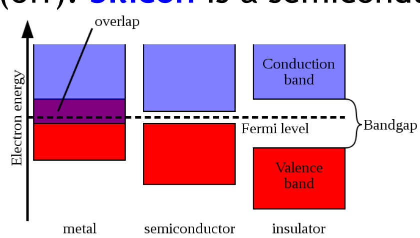
- A **transistor** is a device used to amplify or switch electronic signals.
- A transistor used as a switch has three connections to the outside world:
  - source
  - control
  - output



- If the control is "on", the source flows to the output. Otherwise, the output is "off".
  - A transistor is like a faucet.

The Transistor Continued

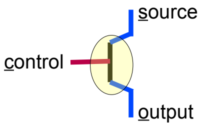
- A **transistor** is made of a solid piece of semiconductor material.
- A **semiconductor** is a material that has electrical conductivity that varies dynamically between that of a conductor (on) or an insulator (off). **Silicon** is a semiconductor.



The Transistor

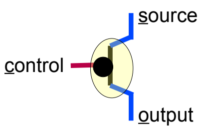
- With transistors it is possible to make two switches: normal control, and inverted control.
  - The black dot means inverted.
- Exhaustive Listing:

S	C	O
1	1	1
1	0	0
0	1	0
0	0	0



What logical operation is this?

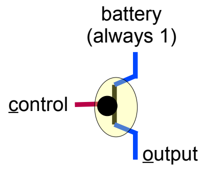
S	C	O
1	1	0
1	0	1
0	1	0
0	0	0



The Notty Transistor

- One Trick: what if we wire the source of an inverted control switch up to a battery that is always on?
- Exhaustive Listing:

C	O
1	0
0	1



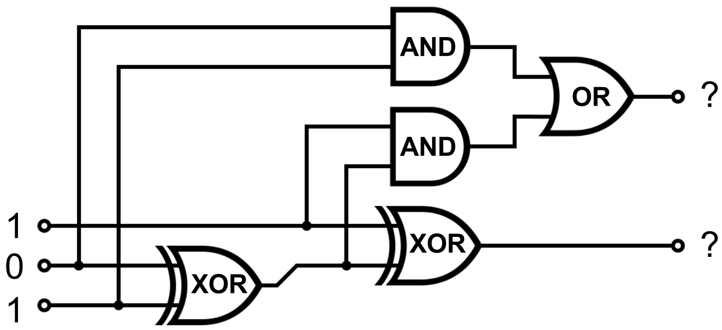
What logical operation is this?

Boolean Logic

- So we have (**and** X Y) and (**not** X) for bits.
- Also (**or** X Y) = (not (and (not X) (not Y)))
- Also (**xor** X Y) = (and (or x y) (not (and x y)))
- An electronic circuit that operates on bits and implements basic boolean logic is called a **gate**.
- So far we have and, or, xor and not gates.
- That's all we need to add numbers!

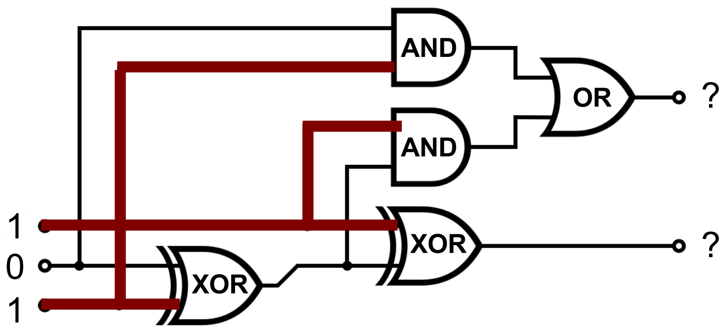
Adding Numbers!

- 1 + 0 + 1 = 10



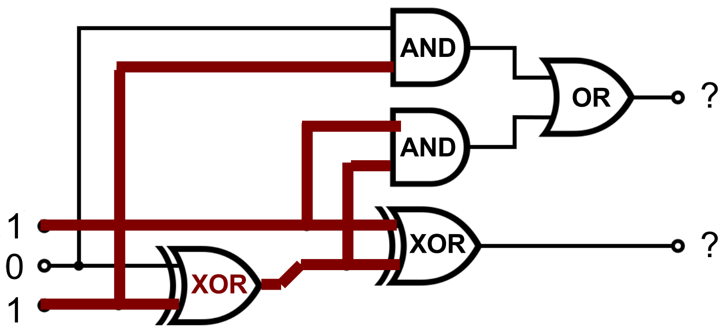
Adding Numbers!

- 1 + 0 + 1 = 10



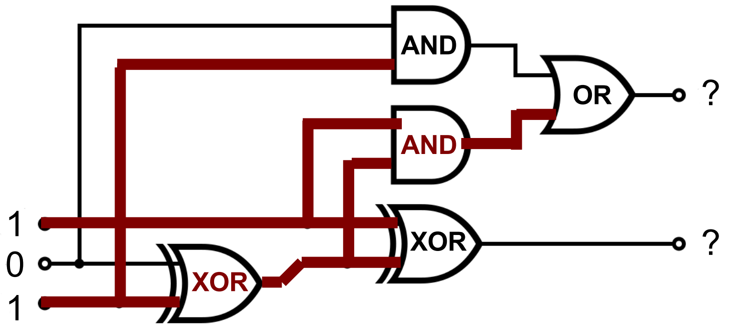
Adding Numbers!

- 1 + 0 + 1 = 10



Adding Numbers!

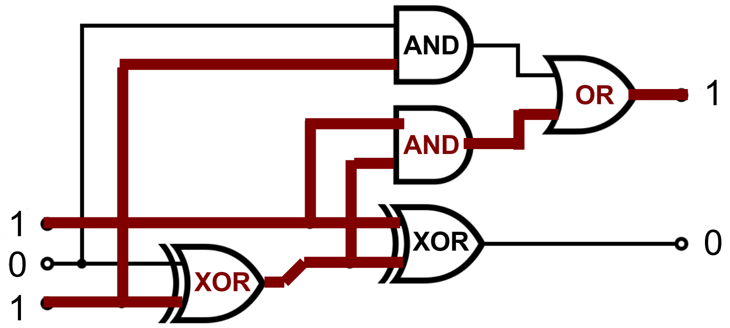
- 1 + 0 + 1 = 10



#49

Adding Numbers!

- 1 + 0 + 1 = 10



#50

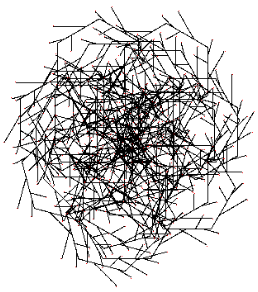
Electronic Computers

- By using *semiconductors*
  - which work using physical properties of silicon
- We can build *transistors*
  - which are like switches or faucets
- To manipulate electrical *voltages*
  - which represent bits
- Through logical *gates*
  - which encode and, or, not, etc.
- To add (and subtract, etc.) numbers!
  - In O(1) time. This is the **basis** of our cost model.

#51

Homework

- Read Couse Book Chp 7
  - You've already done so!
- Problem Set 3 Midnight
- Problem Set 4 Out
  - Start now!
  - Due Next Wednesday
  - It's for Exam Prep!



The Mask
by Zachary Pruckowski, Kristen Henderson

#52