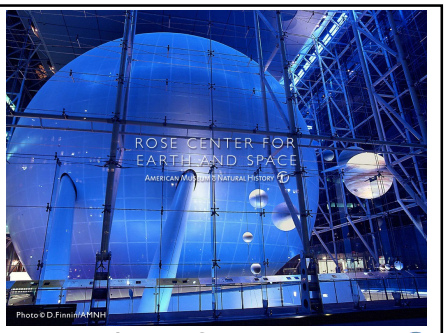
# Class 11: Golden Ages, Orders of Growth, and Astrophysics




CS150: Computer Science
University of Virginia
Computer Science

David Evans
http://www.cs.virginia.edu/evans

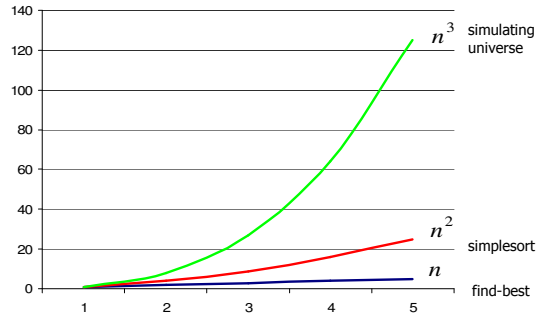
## Astrophysics

- "If you're going to use your computer to simulate some phenomenon in the universe, then it only becomes interesting if you change the scale of that phenomenon by at least a factor of 10. … For a 3D simulation, an increase by a factor of 10 in each of the three dimensions increases your volume by a factor of 1000."
- How much work is astrophysics simulation (in Θ notation)?

$\Theta(n^3)$ When we double the size of the simulation, the work octuples! (Just like oceanography octopi simulations)

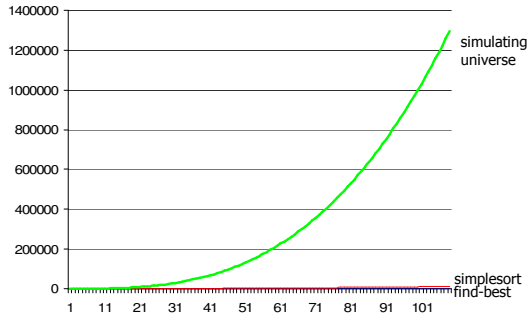


## Orders of Growth






## Orders of Growth






## Astrophysics and Moore's Law

- Simulating universe is $\Theta(n^3)$
- Moore's law: computing power doubles every 18 months
- Tyson: to understand something new about the universe, need to scale by 10x
- How long does it take to know twice as much about the universe?



## Knowledge of the Universe

```
;;; doubling every 18 months = ~1.587 * every 12 months
(define (computing-power nyears)
  (if (= nyears 0) 1
     (* 1.587 (computing-power (- nyears 1)))))

;;; Simulation is θ(n³) work
(define (simulation-work scale)
   (* scale scale scale))

(define (log10 x)  (/ (log x) (log 10))) ;;; log is base e
;;; knowledge of the universe is log 10 the scale of universe
;;; we can simulate
(define (knowledge-of-universe scale) (log10 scale))
```

# Knowledge of the Universe

```
(define (computing-power nyears)
  (if (= nyears 0) 1 (* 1.587 (computing-power (- nyears 1)))))
  ;;; doubling every 18 months = ~1.587 * every 12 months
(define (simulation-work scale) (* scale scale scale))
   ;;; Simulation is O(n^3) work
(define (log10 x)  (/ (log x) (log 10)))
   ;;; primitive log is natural (base e)
(define (knowledge-of-universe scale) (log10 scale))
  ;;; knowledge of the universe is log 10 the scale of universe we can simulate
(define (find-knowledge-of-universe nyears)
  (define (find-biggest-scale scale)
    ;;; today, can simulate size 10 universe = 1000 work
    (if (> (/ (simulation-work scale) 1000)
           (computing-power nyears))
        (- scale 1)
        (find-biggest-scale (+ scale 1))))
  (knowledge-of-universe (find-biggest-scale 1)))
```



```
> (find-knowledge-of-universe 0)
1.0
> (find-knowledge-of-universe 1)
1.041392685158225
> (find-knowledge-of-universe 2)
1.1139433523068367
> (find-knowledge-of-universe 5)
1.322219294733919
> (find-knowledge-of-universe 10)
1.6627578316815739
> (find-knowledge-of-universe 15)
2.0
> (find-knowledge-of-universe 30)
3.00560944536028
> (find-knowledge-of-universe 60)
5.0115366121349325
> (find-knowledge-of-universe 80)
6.348717927935257
```

*Only two things are infinite, the universe and human stupidity, and I'm not sure about the former.*
Albert Einstein

Will there be any mystery left in the Universe when you die?



# Insert Sort

```
(define (insertsort cf lst)
  (if (null? lst)
      null
      (insertel cf
         (car lst)
         (insertsort cf
                     (cdr lst)))))
```

```
(define (insertel cf el lst)
  (if (null? lst)
      (list el)
      (if (cf el (car lst))
          (cons el lst)
          (cons (car lst)
                (insertel cf el
                          (cdr lst))))))
```

insertsort is $\Theta(n^2)$



# Divide and Conquer

- Both simplesort and insertsort divide the problem of sorting a list of length $n$ into:
  - Sorting a list of length $n$-1
  - Doing the right thing with one element
- Hence, there are always n steps
  - And since each step is $\theta(n)$, they are $\theta(n^2)$
- To sort more efficiently, we need to divide the problem more evenly each step



# Can we do better?

```
(insertel < 88
          (list 1 2 3 5 6 23 63 77 89 90))
```

Suppose we had procedures
(first-half lst)
(second-half lst)
that quickly divided the list in two halves?



```
(define (insertsorth cf lst)
  (if (null? lst) null
     (insertelh cf
        (car lst)
        (insertsorth cf (cdr lst)))))
```

Same as insertsort except uses insertelh

# Insert Halves

```
(define (insertelh cf el lst) ;; assumes lst is sorted by cf
  (if (null? lst)
    (list el)
    (let ((fh (first-half lst))
          (sh (second-half lst)))
      (if (cf el (car fh))  ; before first half, put at beginning
        (append (cons el fh) sh)
        (if (null? sh) ; sh is null means fh has one element
          (append fh (list el))
          (if (cf el (car sh))
            (append (insertelh cf el fh) sh)
            (append fh (insertelh cf el sh))))))))
```

# Evaluating insertelh

```
> (insertelh < 3 (list 1 2 4 5 7))
|(insertelh #<procedure:traced-<> 3 (1 2 4 5 7))
| (< 3 1)
| #f
| (< 3 5)
| #t
| (insertelh #<procedure:traced-<> 3 (1 2 4))
| |(< 3 1)
| |#f
| |(< 3 4)
| |#t
| |(insertelh #<procedure:traced-<> 3 (1 2))
| | (< 3 1)
| | #f
| | (< 3 2)
| | #f
| | (insertelh #<procedure:traced-<> 3 (2))
| | |(< 3 2)
| | |#f
| | (2 3)
| |(1 2 3)
| (1 2 3 4)
|(1 2 3 4 5 7)
(1 2 3 4 5 7)
```

```
(define (insertelh cf el lst)
  (if (null? lst)
      (list el)
      (let ((fh (first-half lst))
            (sh (second-half lst)))
        (if (cf el (car fh))
            (append (cons el fh) sh)
            (if (null? sh)
                (append fh (list el))
                (if (cf el (car sh))
                    (append (insertelh cf el fh) sh)
                    (append fh (insertelh cf el sh))))))))
```

Every time we call insertelh, the length of the list is approximately halved!



# How much work is insertelh?

**Suppose** first-half and second-half are $\theta(1)$

Each time we call insertelh, the size of lst halves. So, doubling the size of the list only increases the number of calls by 1.

```
(define (insertelh cf el lst)
  (if (null? lst)
      (list el)
      (let ((fh (first-half lst))
            (sh (second-half lst)))
        (if (cf el (car fh))
            (append (cons el fh) sh)
            (if (null? sh)
                (append fh (list el))
                (if (cf el (car sh))
                    (append (insertelh cf el fh) sh)
                    (append fh
                            (insertelh cf el sh))))))))
```

| List Size | Number of insertelh applications |
|---|---|
| 1 | 1 |
| 2 | 2 |
| 4 | 3 |
| 8 | 4 |
| 16 | 5 |



# How much work is insertelh?

**Suppose** first-half and second-half are $\theta(1)$

Each time we call insertelh, the size of lst halves. So, doubling the size of the list only increases the number of calls by 1.

insertelh would be $\theta(\log_2 n)$

| List Size | Number of insertelh applications |
|---|---|
| 1 | 1 |
| 2 | 2 |
| 4 | 3 |
| 8 | 4 |
| 16 | 5 |

$\log_2 a = b$ *means* $2^b = a$



# insertsorth

Same as insertsort, except uses insertelh

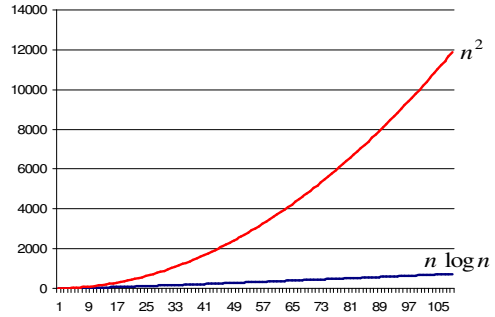
```
(define (insertsorth cf lst)
  (if (null? lst)
      null
      (insertelh cf
        (car lst)
        (insertsorth
          cf
          (cdr lst)))))
```

```
(define (insertelh cf el lst)
  (if (null? lst)
      (list el)
      (let ((fh (first-half lst))
            (sh (second-half lst)))
        (if (cf el (car fh))
            (append (cons el fh) sh)
            (if (null? sh)
                (append fh (list el))
                (if (cf el (car sh))
                    (append (insertelh cf el fh) sh)
                    (append fh
                            (insertelh cf el sh))))))))
```

insertsorth would be $\Theta(n \log_2 n)$ ***if*** we have fast first-half/second-half



# Orders of Growth






# Is there a fast first-half procedure?

- No!
- To produce the first half of a list length $n$, we need to cdr down the first $n/2$ elements
- So:
  - first-half is $\theta(n)$
  - insertelh calls first-half every time…so
    - insertelh is $\theta(n) * \theta(\log_2 n) = \theta(n \log_2 n)$
    - insertsorth is $\theta(n) * \theta(n \log_2 n) = \theta(n^2 \log_2 n)$

**Yikes! We've done all this work, and its still worse than our simplesort!**

We'll figure out how to make a fast first-half-like procedure Monday…




## The Endless Golden Age

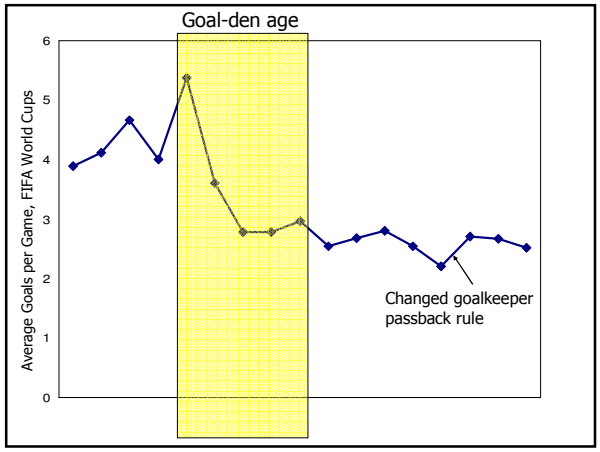
- Golden Age – period in which knowledge/quality of something doubles quickly
- At any point in history, half of what is known about astrophysics was discovered in the previous 15 years!
- Moore's law today, but other advances previously: telescopes, photocopiers, clocks, etc.



## Short Golden Ages

- Golden Age – period in which knowledge/quality of something doubles quickly
- Endless golden age: at any point in history, the amount known is twice what was known 15 years ago
- Short golden age: knowledge doubles during a short, "golden" period, but only improves gradually most of the time






## Endless Golden Age and "Grade Inflation"

- Average student gets twice as smart and well-prepared every 15 years
  - You had grade school teachers (maybe even parents) who went to college!
- If average GPA in 1970 is 2.00 what should it be today (if grading standards didn't change)?



## Grade Inflation or Deflation?

| | |
|---|---|
| 2.00 | average GPA in 1970 ("gentleman's C"?) |
| * 2 | better students 1970-1988 |
| * 2 | better students 1988-2005 |
| * 3 | admitting women, non-whites (1971) |
| * 1.54 | population increase |
| * 0.58 | increase in enrollment |

| | |
|---|---|
| Virginia 1970 | 4,648,494 |
| Virginia 2000 | 7,078,515 |
| Students 1970 | 11,000 |
| Students 2002 | 18,848 (12,595 UG) |

Average GPA today should be:

21.4 CS150 has only the best of the best students, and only the best 31/34 of them stayed in the course after PS1, so the average grade in CS150 should be 21.4*2*2*34/31 = **93.9**

## The Real Golden Rule?

Why do fields like astrophysics, medicine, biology and computer science (?) have "endless golden ages", but fields like

- music (1775-1825)
- rock n' roll (1962-1973, or whatever was popular when you were 16)
- philosophy (400BC-350BC?)
- art (1875-1925?)
- soccer (1950-1974)
- baseball (1925-1950)
- movies (1920-1940)

have short golden ages?

Thanks to Leah Nylen for correcting this (previously I had only 1930-1940, but that is only true for Hollywood movies).



## Charge

- PS3 due Monday
- Understanding the universe is $\Theta(n^3)$
  - Are there any harder problems?
- If you want to be famous pick a major that has a short golden age from 2005-2020
- Our Constitution Day recognition will be in Monday's class