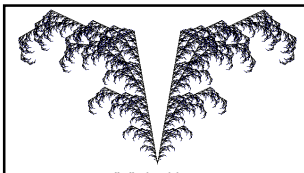
Lecture 12: Something about Sneezewort

532. Achillea Ptarmica L.
Sneezewort

From *Illustrations of the Britisch Flora* (1924) by Walter Hood Fitch
http://www.zum.de/stueber/fitch/high/IMG_8821.html

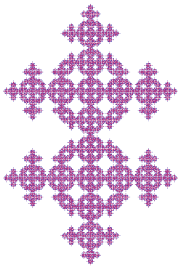
CS150: Computer Science
University of Virginia
Computer Science

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



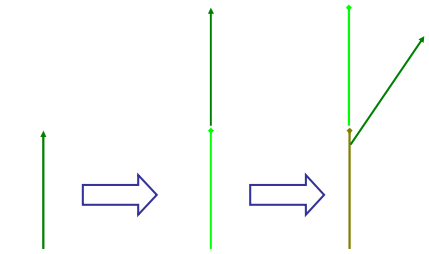
"V" shrubbery
by Andrew Jesien. Beckv Elstad

Robot Cav Man
by Jamie Jeon & Walter Borges



After the Incident
by Ben Morrison and Liz Peterson

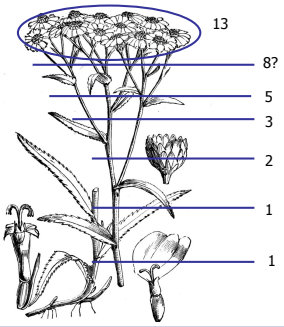
Sneezewort Growth



First Time Unit Second Time Unit Offshoot

Could we model Sneezewort with PS3 code?

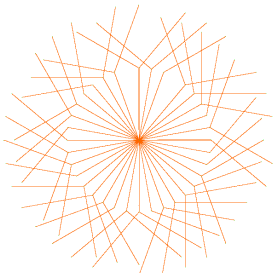
Sneezewort Numbers



pink
by Jessica Geist, Ellen Clarke

Fibo Results

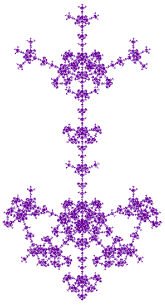
```
> (fibo 2)
1
> (fibo 3)
2
> (fibo 4)
3
> (fibo 10)
55
> (fibo 60)
```
Still working…



At least we finished.
by Dmitriy Semenov and Sara Alspaugh

Tracing Fibo

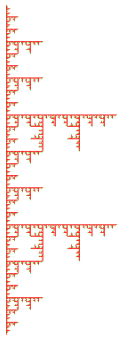
```
> (require-library "trace.ss")
> (trace fibo)
(fibo)
> (fibo 3)
|(fibo 3)
|       (fibo 2)
|       1
|       (fibo 1)
|       1
|2
2
```



Purple Arrow
by Rachel Lathbury and Andrea Yoon

```
> (fibo 5)
|(fibo 5)
| (fibo 4)
| |(fibo 3)
| | (fibo 2)
| | 1
| | (fibo 1)
| | 1
| |2
| |(fibo 2)
| |1
| 3
| (fibo 3)
| |(fibo 2)
| |1
| |(fibo 1)
| |1
| 2
|5
5
```

A right-wing Christmas - awwwwww.....
by Andrew Baker & Emily Lam



To calculate (fibo 5) we calculated:

(fibo 4)	1 time
(fibo 3)	2 times
(fibo 2)	3 times } 5 times total
(fibo 1)	2 times

= 8 calls to fibo = (fibo 6)

How many calls to calculate (fibo 60)?

fast-fibo

```
(define (fast-fibo n)
  (define (fib-helper a b left)
    (if (<= left 0)
        b
        (fib-helper b (+ a b) (- left 1))))
  (fib-helper 1 1 (- n 2)))
```


Fast-Fibo Results

```
> (fast-fibo 10)
55
> (time (fast-fibo 61))
cpu time: 0 real time: 0 gc time: 0
2504730781961
```

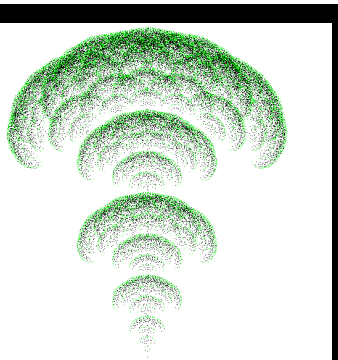
2.5 Trillion applications
2.5 GHz computer does 2.5 *Billion* simple operations per second, so 2.5 Trillion applications operations take ~1000 seconds.
Each application of fibo involves hundreds of simple operations…


```
  ;;; The Earth's mass is 6.0 x 10^24 kg
> (define mass-of-earth (* 6 (expt 10 24)))
  ;;; A typical rabbit's mass is 2.5 kilograms
> (define mass-of-rabbit 2.5)
> (/ (* mass-of-rabbit (fast-fibo 60)) mass-of-earth)
6.450036483e-013
> (/ (* mass-of-rabbit (fast-fibo 120)) mass-of-earth)
2.2326496895795693
```

According to Bonacci's model, after less than 10 years, rabbits would out-weigh the Earth!

Beware the Bunnies!!

Beware the Sneezewort!!

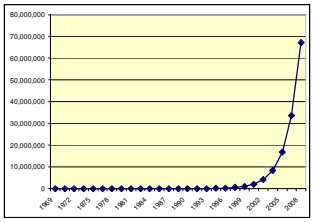


Broccoli Fallout by Paul DiOrio, Rachel Phillips

Evaluation Cost

Actual running times vary according to:

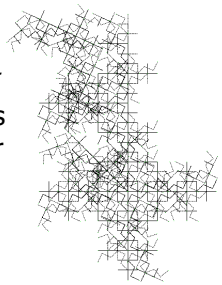
- How fast a processor you have
- How much memory you have
- Where data is located in memory
- How hot it is
- What else is running
- etc...



Moore's "Law" – computing power doubles every 18 months

Measuring Cost

- How does the cost scale with the *size of the input*
- If the input size increases by one, how much longer will it take?
- If the input size *doubles*, how much longer will it take?



Nokomis McCaskill
Chris Hooe

Cost of Fibonacci Procedures

```
(define (fibo n)
        (if (or (= n 1) (= n 2))
    1 ;;; base case
    (+ (fibo (- n 1))
       (fibo (- n 2)))))
```

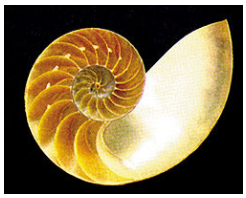
```
(define (fast-fibo n)
  (define (fib-helper a b left)
    (if (= left 0)
        b
        (fib-helper b (+ a b) (- left 1))))
  (fib-helper 1 1 (- n 2)))
```

Input	fibo	fast-fibo
m	q	$z = mk$
$m+1$	$q*\Phi$	$(m+1)k$
$m+2$	at least q^2	$(m+2)k$

Φ = (/ (+ 1 (sqrt 5)) 2) = "The Golden Ratio" ~ 1.618033988749895...
~ (/ (fast-fibo 61) (fast-fibo 60)) = 1.618033988749895

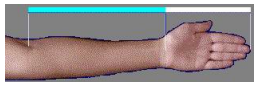
The Golden Ratio

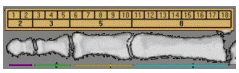
Parthenon



Nautilus Shell

More Golden Ratios





http://www.fenkefeng.org/essaysm18004.html
by Oleksiy Stakhov

PS2 Question

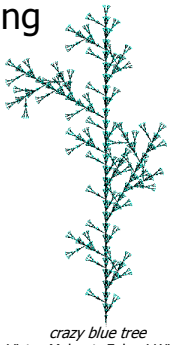
```
(define (find-best-hand hands)
  (car (sort hands higher-hand?)))
```

```
(define (find-best lst cf)
  (if (= 1 (length lst)) (car lst)
      (pick-better cf (car lst) (find-best (cdr lst) cf))))
(define (pick-better cf num1 num2)
  (if (cf num1 num2) num1 num2))
(define (find-best-hand hands)
  (find-best hands higher-hand?))
```

Which is better and by how much?

Simple Sorting

- Can we use find-best to implement sort?
- Use (find-best lst) to find the best
- Remove it from the list
- Repeat until the list is empty



crazy blue tree
by Victor Malaret, Folami Williams

Simple Sort

```
(define (sort lst cf)
  (if (null? lst) lst
      (let ((best (find-best lst cf)))
        (cons
          best
          (sort (delete lst best) cf)))))
```


Sorting Hands

```
(define (sort lst cf)
  (if (null? lst) lst
      (let ((best (find-best lst cf)))
        (cons
          best
          (sort (delete lst best) cf)))))

(define (sort-hands lst)
  (sort lst higher-hand?))
```


Sorting

```
(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))))
(define (pick-better cf num1 num2)
  (if (cf num1 num2) num1 num2))
```

How much work is sort?

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

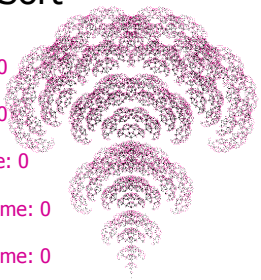
Sorting Cost

```
(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

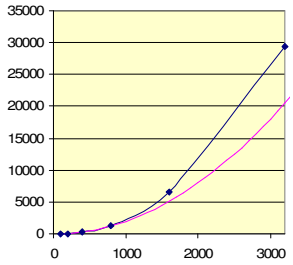
Timing Sort

```
> (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

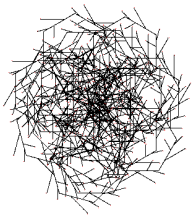


measured times

$= n^2/500$

Charge

- Read Chapter 6: formal notations we will use for this type of analysis
- PS4 out now: you know everything you need for the programming parts; we will cover more on analysis Wednesday and Friday
- Beware the Bunnies and Sneezewort!



The Mask
by Zachary Pruckowski, Kristen Henderson