Lecture 12: Something about Sneezewort

From Illustrations of the British Flora (1924) by Walter Hood Fitch
http://www.sark.de/Kunst/1924/BritishFlora_AW1.nsf

CS150: Computer Science
University of Virginia
Computer Science

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

Sneezewort Growth

Could we model Sneezewort with PS3 code?

Sneezewort Numbers

Fibo Results

> (fibo 2) 1
> (fibo 3) 2
> (fibo 4) 3
> (fibo 10) 55
> (fibo 60) 88795

At least we finished.
by Dmitriy Semenov and Sara Alspaugh

Tracing Fibo

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

Purple Arrow
by Rachel Lathbury and Andrea Yoon
Lecture 12: Sneezewort

Fast-fibo

```scheme
(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

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

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
Lecture 12: Sneezewort

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?

Cost of Fibonacci Procedures

\[
\begin{align*}
\text{Input} & \quad \text{fibo} & \quad \text{fast-fibo} \\
\text{m} & \quad q & \quad z = mk \\
m+1 & \quad q^2 \Phi & \quad (m+1)k \\
m+2 & \quad \text{at least } q^2 & \quad (m+2)k \\
\end{align*}
\]

\[\Phi = \left(\frac{1}{\sqrt{5}}\right) = "The Golden Ratio" \approx 1.618033988749895...\]

Parthenon

Nautilus Shell

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

PS2 Question

\[
\begin{align*}
\text{(define (find-best-hand hands)} \\
\text{(car (sort hands higher-hand?)}}))
\end{align*}
\]

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

\[
\begin{align*}
\text{crazy blue tree} \\
\text{by Victor Malaret, Folami Williams}
\end{align*}
\]
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 \( n^2 \)

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

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

measured times

\[= \frac{n^2}{500}\]

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!