Exam 1

• Handed out at end of Friday’s class, due at the beginning of Wednesday’s class
• Open non-human resources except for Scheme interpreters but no help from other people
• Covers everything through this Wednesday including:
  — Lectures 1-16, Course Book Chapters 1-8, PS 1-4
• Sample exams from previous years: if you can do well on Spring 2009 Exam 1, you should do well on our Exam 1 (of course, questions will be different!)
• Review Session, Wednesday 6:30 in Olsson 001

Running Time Practice

From ps3:

```
(define (flatten-commands ll)
  (if (null? ll) ll
   (if (is-lsystem-command? (car ll))
       (cons (car ll) (flatten-commands (cdr ll)))
       (flat-append (car ll) (flatten-commands (cdr ll))))))
```

What is the asymptotic running time of `flatten-commands`?

First: determine running times of all the procedures applied in `flatten-commands`.

Flatten Running Time

From ps3:

```
(define (flatten-commands ll)
  (if (null? ll) ll
   (if (is-lsystem-command? (car ll))
       (cons (car ll) (flatten-commands (cdr ll)))
       (flat-append (car ll) (flatten-commands (cdr ll))))))
```

null?, car, cons, cdr — we already know there are constant time

What about `is-lsystem-command`?

```
(define (is-lsystem-command? lcommand)
  (or (is-forward? lcommand)
      (is-rotate? lcommand)
      (is-offshoot? lcommand)))
```

or is a special form:

OrExpression ::= (or MoreExpressions)

To evaluate (or Expr1 MoreExpressions):
1. Evaluate Expr1.
2. If it evaluates to a non-false value, that is the value of the or expression. None of the other sub-expressions are evaluated. Otherwise, the value of the or-expression is the value of (or MoreExpressions)

The value of (or) is false.
is-lsystem-command?

(define (is-lsystem-command? lcommand)
(or (is-forward? lcommand)
(is-rotate? lcommand)
(is-offshoot? lcommand)))

Each of these procedures has constant running time: they involve only applications of constant time procedures eq? and car.

Flatten Running Time

From ps3:

(define (flatten-commands ll)
(if (null? ll) ll
(if (is-lsystem-command? (car ll))
(cons (car ll) (flatten-commands (cdr ll)))
(flat-append (car ll) (flatten-commands (cdr ll)))))))

First: determine running times of all the procedures applied in flatten-commands.

null?, car, cons, cdr, and is-lsystem-command? are constant time

Running Time Practice

Remember: we care about the size of the input.

(define (flat-append lst ll)
(if (null? lst) ll
(if (is-lsystem-command? (car ll))
(cons (car ll) (flat-append (cdr lst) ll)))
(flat-append (car lst) (flat-append (cdr lst) ll))))

Other than the recursive call, each execution is constant time:
null?, car, cons, cdr, are constant time
How many recursive calls are there?
N₁, the number of elements in the first input
What is the running time?
The asymptotic running time of flat-append is in θ(N₁)
where N₁ is the number of elements in the first input.

Note: flat-append is the same as list-append! (Stupid to define this as a separate procedure and name it flat-append.)

Flatten Running Time

(define (flatten-commands ll)
(if (null? ll) ll
(if (is-lsystem-command? (car ll))
(cons (car ll) (flatten-commands (cdr ll)))
(flat-append (car ll) (flatten-commands (cdr ll)))))))

First: determine running times of all the procedures applied in flatten-commands.

null?, car, cons, cdr, and is-lsystem-command? are constant time

Second: determine running time for each application except for recursive call.

Need to consider both paths:
(if (is-lsystem-command? (car ll))
(cons (car ll) (flatten-commands (cdr ll)))
(flat-append (car ll) (flatten-commands (cdr ll)))))

Fractal Finalists

Teamwork by Rose Cunnion and Lucy Raper

to be continued Wednesday...
Charge

- PS4 is due Wednesday
- Exam 1 is out Friday, due next Wednesday
- Exam Review, Wednesday 6:30 in Olsson 001

Returning PS3

- Front
- Back

- abc8a ... dwa2x
- eab8d ... jsw8a
- jta9nk ... mz2h
- os9e ... wch9a