Week 10 - Day 2 (Streams)

Mar 24, 2016

Announcement (We might go outside for class on Tuesday)

Audio 0:01:29

```
(define (ints-from n)
        (cons-stream n (ints-from (+ n 1)))
)

(define (sdisplay s n)
        (cond
                ((= n 8) (print "..."))
                (else
                        (print (stream-car s) " ")
                        (sdisplay (stream-cdr s) (- n 1))
                )
        )
)

(sdisplay (ints-from 2) 100)
;@ Audio 0:06:39 VM woes
(sdisplay s 100)

(define (add-streams s t)
        (cons-stream
                (+ (stream-car s) (stream-car t))
                (add-streams (stream-cdr s) (stream-cdr t))
        )
)

(sdisplay (add-streams s s) 10)

;@ Audio 0:09:07
```

```
(define (make-ones)
        (s-cons 1 (make-ones))
)

;@ Audio 0:11:40
;@ How did he do it?
(define ones (scons 1 ones))

(sdisplay (ones) 10)
```

Wowza we can make recursive calls.

Audio 0:13:00 How do we make a stream of integers starting at one and incrementing?

Ex:

1 1 1 1 1 1 1 ...

+ 1 2 3

----

1 2 3 4 5 ...

Audio 0:15:50

```
(define ints (scons 1 (add-streams ones ints)))
(sdisplay ints 100)
```

Audio 0:17:00

Fibinocci

	0	1	1	2	3	5
+	1	1	2	3	5	5
0	1	1	2	3	5	8	13

```
(define fibs (scons 0 (scons 1 (add-streams fibs (scdr fibs)))))
```

Audio 0:21:00 ## Sieve of Eratostheones

```
;@ Audio 0:23:00
;@ stream remove
```

```
(define (sremove p? s)
        (if (p? (scar s))
                (sremove p? (scdr s))
                (scons (scar s) (sremove p? (scdr s)))
        )
)

(define (divides? x y) (= (% x y) 0))

;@ Audio 0:26:30
;@ Prints integers which are not divisible by 3
;@ Hint for project3 # 8 ;)
(sdisplay (sremove (lambda (x) (divides? x 3)) ints) 20)
```

Sieve Function

Audio 0:29:00

```
(define (sieve s)
        (scons
                (scar s)
                (sieve (sremove (lambda (x) (divides? x (scar s))) (s
        )
)
;@ Audio 0:32:50
(sdisplay (sieve (scdr ints)) 20)
```

Book Problem pi/4

Audio 0:34:00

PI/4 = 1 - (1/3)+(1/5)-(1/7)+…

Audio 0:36:57

```
(define (div-streams s t)
        (cons-stream
                (/ (stream-car s) (* 1.0 (stream-car t)))
                (div-streams (stream-cdr s) (stream-cdr t))
        )
```

```
)
(define alt-ones (scons 1 (scons -1 alt-ones)))
(define odds (scons 1 (add-streams twos odds)))
(define twos (scons 2 twos))
;@ Audio 0:40:00

(define pi-stream (div-streams alt-ones odds))
(sdisplay pi-stream 10)
```

Audio 0:44:00

How do you sum an infinite amount of things?

```
(define (psum s)
        ;@ Audio 0:49:00
        (scons (scar s) (psum (scons (+ (scar s) (scar (scdr s))) (sc
;@ Audio 0:53:15
(sdisplay (psum pi-stream) 10)
(define (sscale s x)
        (scons (* x (scar s)) (sscale (scdr s) x))
)

(sdisplay (sscale (psum pi-stream) 4) 10)
```

Euler's Transform

Audio 0:56:00

This will speed up the convergence of the pi stream to being closer to accurate

```
(define (et s)
        (define s0 (scar s))
        (define s1 (scar (scdr s)))
        (define s2 (scar (scdr (scdr s))))
        ;@ Audio 0:59:30
        (scons
                (- s2 (/ (^ (- s2 s1) 2) (+ s0 (* -2 s1) s2)))
                (et (scdr s))
        )
)
```

Let’s do it again!!!

Audio 1:03:40

We need a stream of streams of Euler Transforms

```
(define (tableau s) (scons s (tableau (et s))))
```

Audio 1:05:00

Lusth is describing the streams

We can get an everbetter approximation
