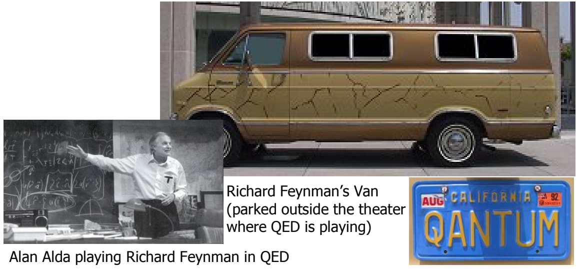
Lecture 9: Recursing Recursively



Richard Feynman's Van (parked outside the theater where QED is playing)

Alan Alda playing Richard Feynman in QED

CS150: Computer Science
University of Virginia
Computer Science

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

Menu

- Recursive Procedures
- GEB Chapter V
  - –Fibonacci
  - –RTNs
  - –Music and Recursion

Example

Define a procedure find-closest-number that takes two inputs, a goal number, and a list of numbers, and produces the number in the list numbers list that is closest to goal:

```
> (find-closest-number 150 (list 101 110 120 157 340 588))
157
> (find-closest-number 12 (list 1 11 21))
11
> (find-closest-number 12 (list 95))
95
```

Find Closest Number

Be optimistic!

Assume you can define:

(find-closest-number goal numbers)

that finds the closest number to goal from the list of numbers.

What if there is one more number?

Can you write a function that finds the closest number to match from new-number and numbers?

Finding the Closest

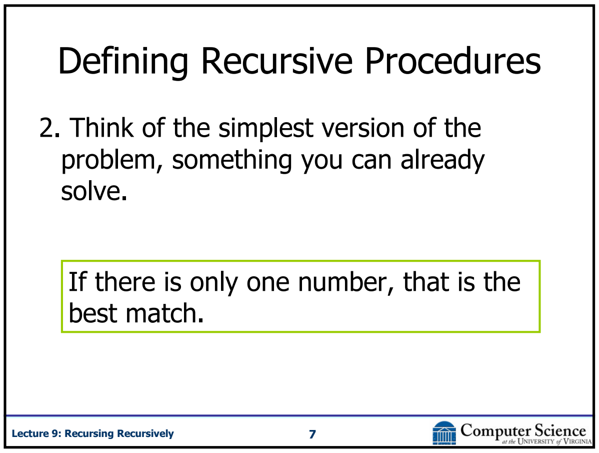
Strategy:

If the first number is closer than the closest number of the rest of the numbers, use the first number.

Otherwise, use the closet number of the rest of the numbers.

Optimistic Function

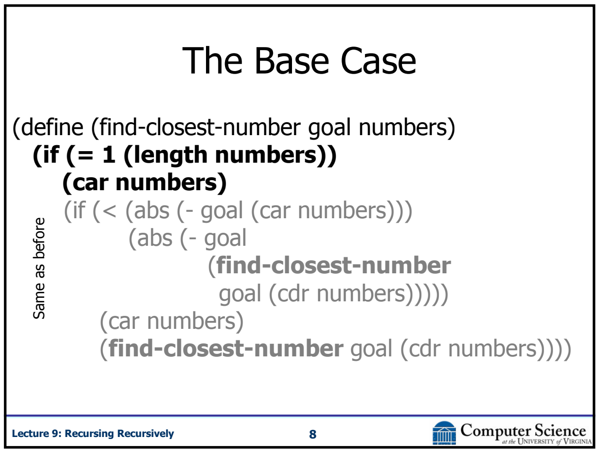
```
(define (find-closest goal numbers)
  (if (< (abs (- goal (car numbers)))
         (abs (- goal
                 (find-closest-number
                   goal (cdr numbers)))))
       (car numbers)
       (find-closest-number
        goal (cdr numbers))))
```



Defining Recursive Procedures

2. Think of the simplest version of the problem, something you can already solve.

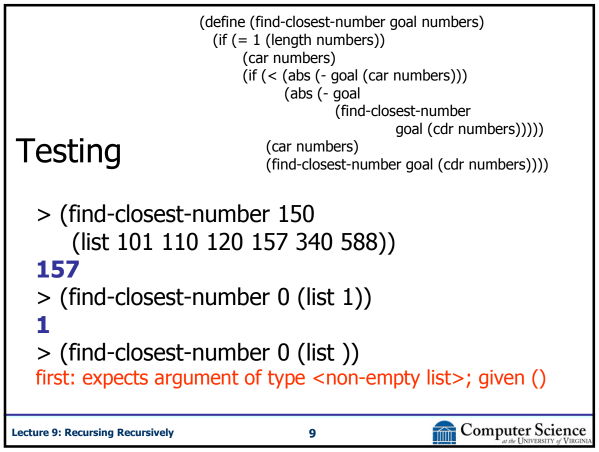
If there is only one number, that is the best match.



The Base Case

```
(define (find-closest-number goal numbers)
  (if (= 1 (length numbers))
      (car numbers)
      (if (< (abs (- goal (car numbers)))
             (abs (- goal
                     (find-closest-number
                      goal (cdr numbers)))))
          (car numbers)
          (find-closest-number goal (cdr numbers))))
```

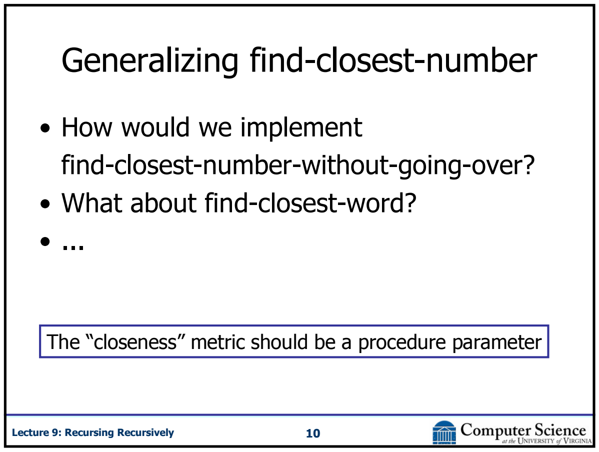
Same as before



Testing

```
(define (find-closest-number goal numbers)
  (if (= 1 (length numbers))
      (car numbers)
      (if (< (abs (- goal (car numbers)))
             (abs (- goal
                     (find-closest-number
                              goal (cdr numbers)))))
          (car numbers)
          (find-closest-number goal (cdr numbers))))
```

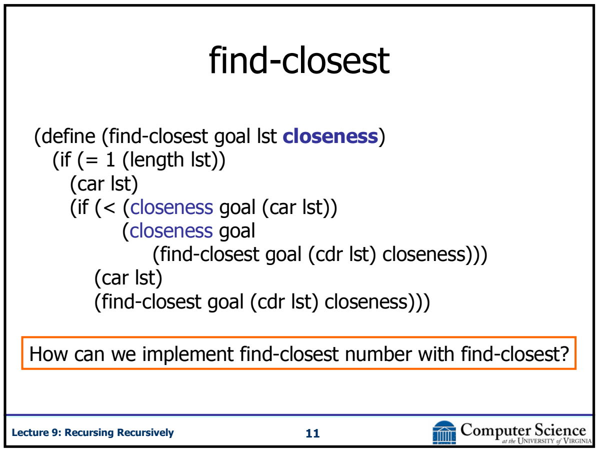
```
> (find-closest-number 150
     (list 101 110 120 157 340 588))
157
> (find-closest-number 0 (list 1))
1
> (find-closest-number 0 (list ))
first: expects argument of type <non-empty list>; given ()
```



Generalizing find-closest-number

- How would we implement find-closest-number-without-going-over?
- What about find-closest-word?
- ...

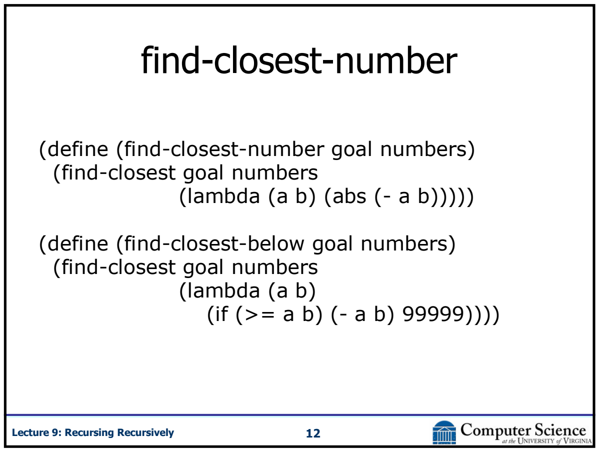
The "closeness" metric should be a procedure parameter



find-closest

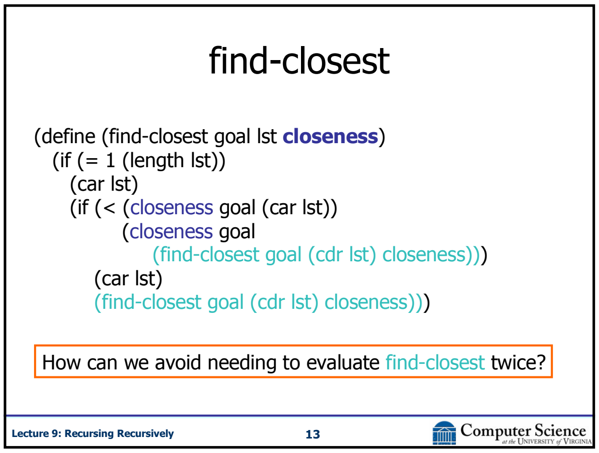
```
(define (find-closest goal lst closeness)
  (if (= 1 (length lst))
    (car lst)
    (if (< (closeness goal (car lst))
           (closeness goal
               (find-closest goal (cdr lst) closeness)))
        (car lst)
        (find-closest goal (cdr lst) closeness)))
```

How can we implement find-closest number with find-closest?



find-closest-number

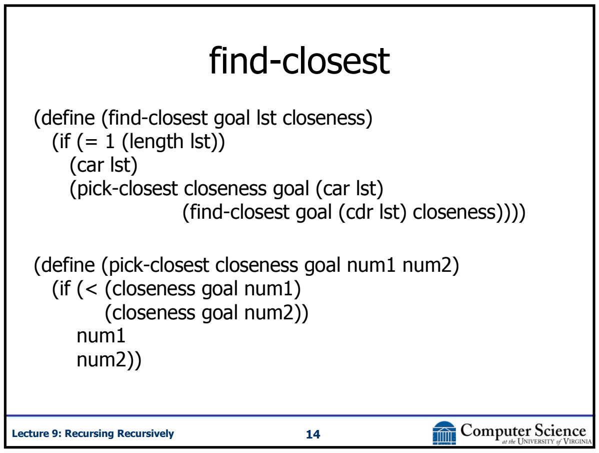
```
(define (find-closest-number goal numbers)
  (find-closest goal numbers
                (lambda (a b) (abs (- a b)))))

(define (find-closest-below goal numbers)
  (find-closest goal numbers
                (lambda (a b)
                   (if (>= a b) (- a b) 99999))))
```



find-closest

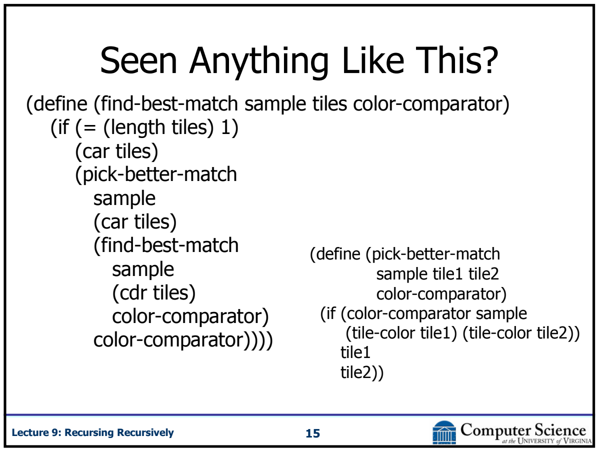
```
(define (find-closest goal lst closeness)
  (if (= 1 (length lst))
    (car lst)
    (if (< (closeness goal (car lst))
           (closeness goal
              (find-closest goal (cdr lst) closeness)))
       (car lst)
       (find-closest goal (cdr lst) closeness)))
```

How can we avoid needing to evaluate find-closest twice?



find-closest

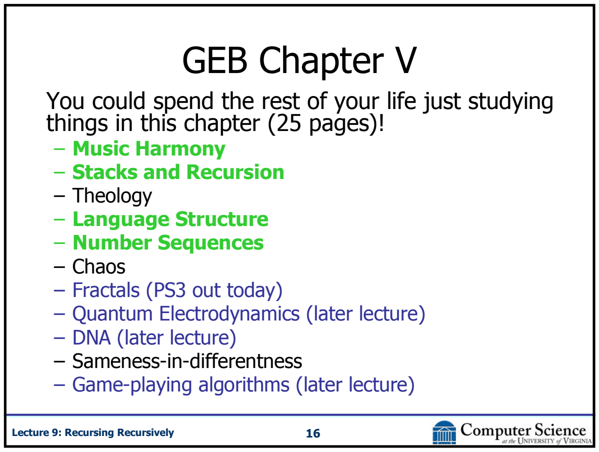
```
(define (find-closest goal lst closeness)
  (if (= 1 (length lst))
    (car lst)
    (pick-closest closeness goal (car lst)
                 (find-closest goal (cdr lst) closeness))))

(define (pick-closest closeness goal num1 num2)
  (if (< (closeness goal num1)
         (closeness goal num2))
     num1
     num2))
```



Seen Anything Like This?

```
(define (find-best-match sample tiles color-comparator)
   (if (= (length tiles) 1)
      (car tiles)
      (pick-better-match
        sample
        (car tiles)
        (find-best-match
          sample
          (cdr tiles)
          color-comparator)
        color-comparator))))
```

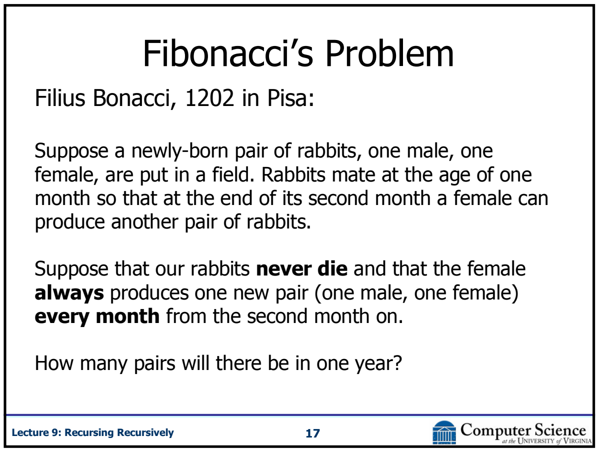
```
(define (pick-better-match
           sample tile1 tile2
           color-comparator)
  (if (color-comparator sample
        (tile-color tile1) (tile-color tile2))
      tile1
      tile2))
```



GEB Chapter V

You could spend the rest of your life just studying things in this chapter (25 pages)!

- **Music Harmony**
- **Stacks and Recursion**
- Theology
- **Language Structure**
- **Number Sequences**
- Chaos
- Fractals (PS3 out today)
- Quantum Electrodynamics (later lecture)
- DNA (later lecture)
- Sameness-in-differentness
- Game-playing algorithms (later lecture)



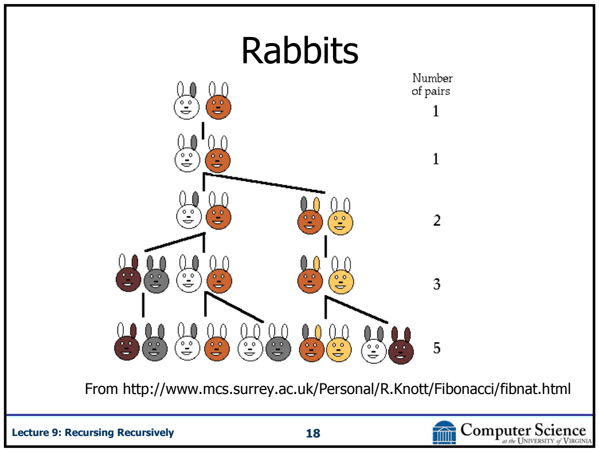
Fibonacci's Problem

Filius Bonacci, 1202 in Pisa:

Suppose a newly-born pair of rabbits, one male, one female, are put in a field. Rabbits mate at the age of one month so that at the end of its second month a female can produce another pair of rabbits.

Suppose that our rabbits **never die** and that the female **always** produces one new pair (one male, one female) **every month** from the second month on.

How many pairs will there be in one year?



Rabbits

From http://www.mcs.surrey.ac.uk/Personal/R.Knott/Fibonacci/fibnat.html

Fibonacci Numbers

GEB p. 136:

> These numbers are best defined recursively by the pair of formulas
>
> FIBO (*n*) = FIBO (n – 1) + FIBO (n – 2) for n > 2
>
> FIBO (1) = FIBO (2) = 1

Can we turn this into a Scheme procedure?

Defining FIBO

1. Be optimistic - assume you can solve it, if you could, how would you solve a bigger problem.
2. Think of the simplest version of the problem, something you can already solve.
3. Combine them to solve the problem.

These numbers are best defined recursively by the pair of formulas
FIBO (*n*) =
FIBO (n – 1)
+ FIBO (n – 2)
for n > 2
FIBO (1) = FIBO (2) = 1

Defining fibo

```
;;; (fibo n) evaluates to the nth Fibonacci
;;; number
(define (fibo n)
  (if (or (= n 1) (= n 2))
     1 ;;; base case
     (+ (fibo (- n 1))
        (fibo (- n 2)))))
```

FIBO (1) = FIBO (2) = 1

FIBO (*n*) =
FIBO (n – 1)
+ FIBO (n – 2)
for n > 2

Fibo Results

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

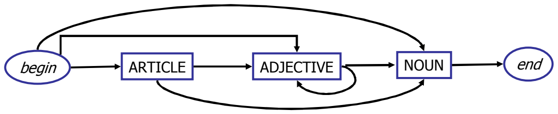
Still working after 4 hours…

Why can't our 4Mx Apollo Guidance Computer figure out how many rabbits there will be in 5 years?

To be continued...

Recursive Transition Networks

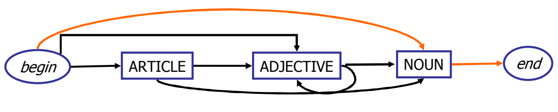
ORNATE NOUN



Can we describe this using Backus Naur Form?

Recursive Transition Networks

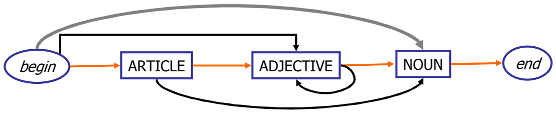
ORNATE NOUN



ORNATE NOUN ::= *NOUN*

Recursive Transition Networks

ORNATE NOUN

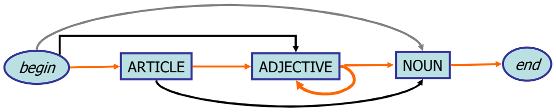


ORNATE NOUN ::= *NOUN*

ORNATE NOUN ::= *ARTICLE ADJECTIVE NOUN*

Recursive Transition Networks

ORNATE NOUN



ORNATE NOUN ::= *ARTICLE ADJECTIVE NOUN*

ORNATE NOUN ::= *ARTICLE ADJECTIVE ADJECTIVE NOUN*

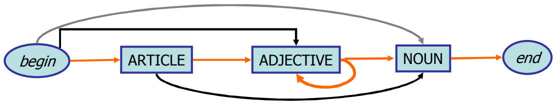
ORNATE NOUN ::= *ARTICLE ADJECTIVE ADJECTIVE ADJECTIVE NOUN*

ORNATE NOUN ::= *ARTICLE ADJECTIVE ADJECTIVE ADJECTIVE ADJECTIVE NOUN*

ORNATE NOUN ::= *ARTICLE ADJECTIVE ADJECTIVE ADJECTIVE ADJECTIVE ADJECTIVE NOUN*

Recursive Transition Networks

ORNATE NOUN



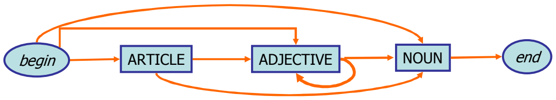
ORNATE NOUN ::= *ARTICLE ADJECTIVES NOUN*

ADJECTIVES ::= *ADJECTIVE ADJECTIVES*

ADJECTIVES ::=

Recursive Transition Networks

ORNATE NOUN



ORNATE NOUN ::= *OPTARTICLE ADJECTIVES NOUN*

ADJECTIVES ::= *ADJECTIVE ADJECTIVES*

ADJECTIVES ::= ε

OPTARTICLE ::= *ARTICLE*

OPTARTICLE ::= ε

Which notation is *better*?

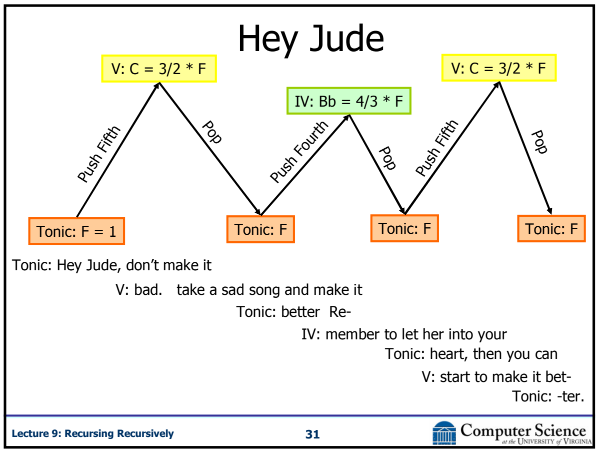
Music Harmony

Kleines Harmonisches Labyrinth

(Little Harmonic Labyrinth)

Hey Jude

John Lennon and Paul McCartney, 1968



Hey Jude

V: C = 3/2 * F

IV: Bb = 4/3 * F

V: C = 3/2 * F

Push Fifth

Pop

Push Fourth

Pop

Push Fifth

Pop

Tonic: F = 1

Tonic: F

Tonic: F

Tonic: F

Tonic: Hey Jude, don't make it

V: bad. take a sad song and make it

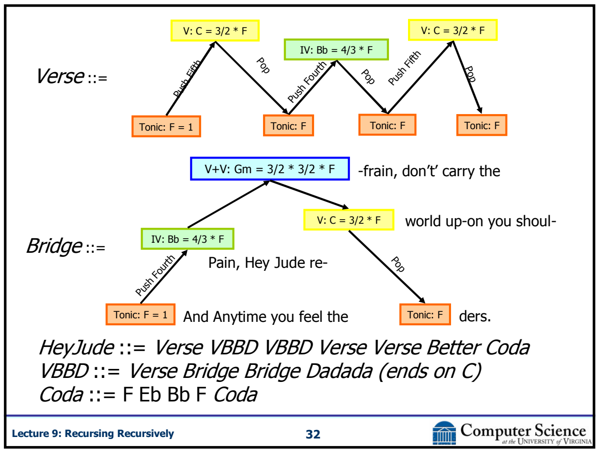
Tonic: better Re-

IV: member to let her into your

Tonic: heart, then you can

V: start to make it bet-

Tonic: -ter.



Verse ::=

V: C = 3/2 * F

IV: Bb = 4/3 * F

V: C = 3/2 * F

Push Fifth

Pop

Push Fourth

Pop

Push Fifth

Pop

Tonic: F = 1

Tonic: F

Tonic: F

Tonic: F

V+V: Gm = 3/2 * 3/2 * F -frain, don't' carry the

V: C = 3/2 * F world up-on you shoul-

Bridge ::=

IV: Bb = 4/3 * F

Pain, Hey Jude re-

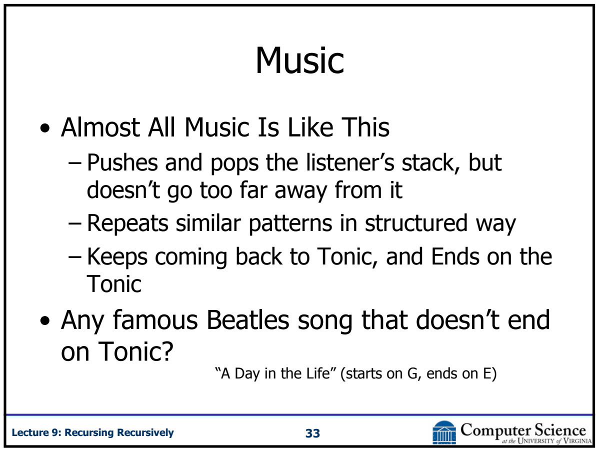
Push Fourth

Pop

Tonic: F = 1 And Anytime you feel the

Tonic: F ders.

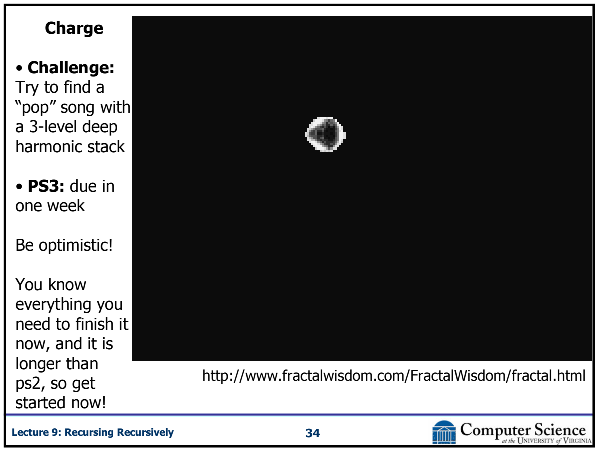
HeyJude ::= *Verse VBBD VBBD Verse Verse Better Coda*
VBBD ::= *Verse Bridge Bridge Dadada (ends on C)*
Coda ::= F Eb Bb F *Coda*



Music

- Almost All Music Is Like This
  - Pushes and pops the listener's stack, but doesn't go too far away from it
  - Repeats similar patterns in structured way
  - Keeps coming back to Tonic, and Ends on the Tonic
- Any famous Beatles song that doesn't end on Tonic?

"A Day in the Life" (starts on G, ends on E)



Charge

- **Challenge:** Try to find a "pop" song with a 3-level deep harmonic stack
- **PS3:** due in one week

Be optimistic!

You know everything you need to finish it now, and it is longer than ps2, so get started now!

http://www.fractalwisdom.com/FractalWisdom/fractal.html