Lecture 10: Puzzling Pegboards

CS150: Computer Science
University of Virginia
Computer Science

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

Menu

- Problem Sets 2 and 3
- Pegboard Puzzler

Problem Sets

- Not just meant to review stuff you should already know
  - Get you to explore new ideas
  - Motivate what is coming up in the class
- The main point of the PSs is ***learning***, not ***evaluation***
  - Don't give up if you can't find the answer in the book (you won't solve many problems this way)
  - Do discuss with other students

PS2: Question 3

Why is

```
(define (higher-card? card1 card2)
  (> (card-rank card1) (card-rank card2)
```

better than

```
(define (higher-card? card1 card2)
  (> (car card1) (car card2))
```

?

PS2: Question 8, 9

- Predict how long it will take
- Identify ways to make it faster

Most of next week and much of many later classes will be focused on how computer scientists **predict** how long programs will take, and on how to **make them faster**.

Can we do better?

```
(define (find-best-hand hole-cards community-cards)
  (car (sort (possible-hands hole-cards
                             community-cards))
             higher-hand?))
```

Hmmm....

```
(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))
```


find-bestest

```
(define (find-bestest lst bestiness)
  (if (= 1 (length lst))
    (car lst)
    (pick-bestier bestiness
                  (car lst)
                  (find-bestest goal (cdr lst) bestiness))))

(define (pick-bestier bestiness num1 num2)
  (if (< (bestiness num1)
         (bestiness num2))
      num1
      num2))
```


find-best-hand

```
(define (find-bestest lst bestiness)
  (if (= 1 (length lst)) (car lst)
    (pick-bestier bestiness
                  (car lst)
                  (find-bestest goal (cdr lst) bestiness))))
(define (pick-bestier bestiness num1 num2)
  (if (< (bestiness num1) (bestiness num2))
    num1 num2))
```

```
(define (find-best-hand lst)
  (find-bestest lst higher-hand?))
```

Next week: how much better is this?

PS3: Lindenmayer System Fractals

L-Systems

CommandSequence ::= **(** *CommandList* **)**
CommandList ::= *Command CommandList*
CommandList ::=
Command ::= **F**
Command ::= **R***Angle*
Command ::= **O***CommandSequence*

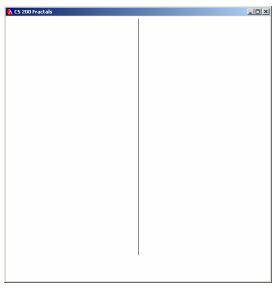
L-System Rewriting

CommandSequence ::= **(** *CommandList* **)**
CommandList ::= *Command CommandList*
CommandList ::=
Command ::= **F**
Command ::= **R***Angle*
Command ::= **O***CommandSequence*

Start: (F)
Rewrite Rule:
F → (F O(R30 F) F O(R-60 F) F)

Work like BNF replacement rules, except replace all instances at once!

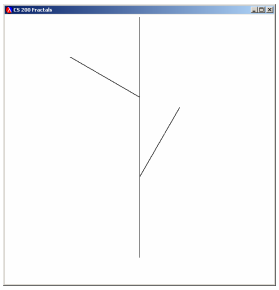
Why is this a better model for biological systems?



Level 0

Start: (F)

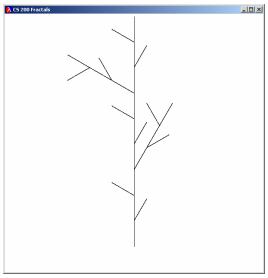
(F)

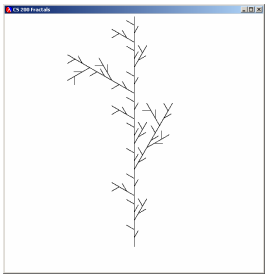


Level 1

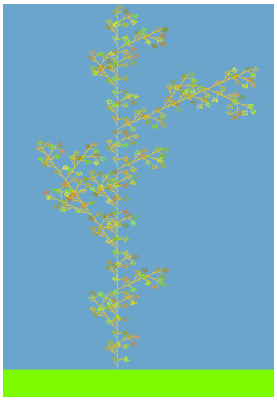
F → (F O(R30 F) F O(R-60 F) F)

(F O(R30 F) F O(R-60 F) F)



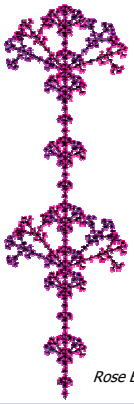


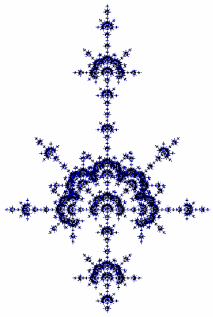
Level 2 Level 3



The Great Lambda Tree of Ultimate Knowledge and Infinite Power

(Level 5 with color)





Tie Dye by Bill Ingram

Rose Bush by Jacintha Henry and Rachel Kay

Pegboard Puzzle

1,1

2,1 2,2

3,1 3,2 3,3

4,1 4,2 4,3 4,4

5,1 5,2 5,3 5,4 5,5

Solving the Pegboard Puzzle

- How to represent the state of the board?
  - Which holes have pegs in them
- How can we simulate a jump?
  - board state, jump positions → board state
- How can we generate a list of all possible jumps on a given board?
- How can we find a winning sequence of jumps?

Removing a Peg

```
;;; remove-peg evaluates to the board you get by removing a
;;; peg at posn from the passed board (removing a peg adds a
;;; hole)

(define (remove-peg board posn)
  (make-board (board-rows board)
              (cons posn (board-holes board))))
```


Adding a Peg

```
;;; add-peg evaluates to the board you get by
;;; adding a peg at posn to board (adding a
;;; peg removes a hole)

(define (add-peg board posn)
  (make-board (board-rows board)
              (remove-hole (board-holes board) posn)))
```


Remove Hole

```
(define (remove-hole lst posn)
  (if (same-position (car lst) posn)
      (cdr lst)
      (cons (car lst) (remove-hole (cdr lst) posn))))
```

Could we define remove-hole using map?

No. (length (map f lst)) is always the same as (length lst), but remove-hole needs to remove elements from the list.

What if we had a procedure (filter proc lst) that removes from lst all elements for which proc (applied to that element) is false?

Filter

```
(define (filter proc lst)
  (if (null? lst)
      null
      (if (proc (car lst)) ; proc is true, keep it
          (cons (car lst) (filter proc (cdr lst)))
          (filter proc (cdr lst))))) ; proc is false, drop it
```

```
> (filter (lambda (x) (> x 0)) (list 1 4 -3 2))
(1 4 2)
```


Filter Remove

```
(define (filter proc lst)
  (if (null? lst)
      null
      (if (proc (car lst)) ; proc is true, keep it
          (cons (car lst) (filter proc (cdr lst)))
          (filter proc (cdr lst))))) ; proc is false, drop it

(define (remove-hole lst posn)
  (filter (lambda (pos)
            (not (same-position pos posn)))
          lst))
```


Solving the Peg Board Game

- Try all possible moves on the board
- Try all possible moves from the positions you get after each possible first move
- Try all possible moves from the positions you get after trying each possible move from the positions you get after each possible first move
- …

Charge

- Next class: we'll finish a pegboard puzzle solver and find out if how hard it is to be "genius"
- I have office hours now
- Make progress on PS3