Tracking Rumors

Suppose that we want to track gossip in a rumor mill.

Seiichi —> Mike —> Amir —> Joseph

Seiichi —> Lindsey —> Amir —> Derrick
Simplifying assumption: each person tells at most two others
Is a rumor mill simply a list of people?

No, because there are relationships among people
How about this?:

; A person is
; (make-person image person person person)

No, because some people don't gossip to anyone else—or they gossip to an empty rumor mill...
Representing Rumor Mills

How about this?

; A rumor-mill is either
;   - empty
;   - (make-gossip image rumor-mill rumor-mill)

(define-struct gossip (who next1 next2))

This looks promising...
Example Rumor Mills

; A rumor-mill is either
;   - empty
;   - (make-gossip image rumor-mill rumor-mill)

empty
Example Rumor Mills

; A rumor-mill is either
;   - empty
;   - (make-gossip image rumor-mill rumor-mill)

(make-gossip empty empty)

Joseph
Example Rumor Mills

; A rumor-mill is either
;   - empty
;   - (make-gossip image rumor-mill rumor-mill)

(make-gossip empty)

(make-gossip empty empty)
Example Rumor Mills

; A rumor-mill is either
;   - empty
;   - (make-gossip image rumor-mill rumor-mill)

(make-gossip)
  (make-gossip empty empty)
  (make-gossip)
    (make-gossip empty)
    (make-gossip empty empty))
  (make-gossip empty empty)))
Example Using Abbreviations

(define joseph-mill
  (make-gossip
   empty empty))

(define amir-mill
  (make-gossip
   empty joseph-mill))

(define derrick-mill
  (make-gossip
   empty empty))

(define lindsey-mill
  (make-gossip
   amir-mill derrick-mill))

(define mike-mill
  (make-gossip
   empty empty))

(define seiichi-mill
  (make-gossip
   mike-mill lindsey-mill))
; A rumor-mill is either
;   - empty
;   - (make-gossip image rumor-mill rumor-mill)

(define (func-for-rumor-mill rm)
  (cond
[(empty? rm) ...]
[(gossip? rm)
  ... (gossip-who rm)
  ... (func-for-rumor-mill (gossip-next1 rm))
  ... (func-for-rumor-mill (gossip-next2 rm)) ...])))
Rumor Program Examples

- Implement the function `informed?` which takes a person image and a rumor mill and determines whether the person is part of the rumor mill.

- Implement `rumor-delay` which takes a rumor mill and determines the maximum number of days required for a rumor to reach everyone, assuming that each person waits a day before passing on a rumor.

- Implement `add-gossip` which takes a rumor mill and two person images—one new and one old—and adds the new person to the rumor mill, receiving rumors from the old person; the old person must not already have two next persons.

- Implement `rumor-chain` which takes a person image and a rumor mill and returns a list of person images representing everyone who must pass on the rumor for it to reach the given person; return `false` if the given person is never informed.
More Pipes

- In the Mid-Term I example, we had all straight pipes in a pipeline.
- Real pipes end in faucets (open or closed) and sometimes branch.
More Pipes

• In the Mid-Term I example, we had all straight pipes in a pipeline
• Real pipes end in faucets (open or closed) and sometimes branch
More Pipes

• In the Mid-Term I example, we had all straight pipes in a pipeline
• Real pipes end in faucets (open or closed) and sometimes branch
More Pipes

- In the Mid-Term I example, we had all straight pipes in a pipeline
- Real pipes end in faucets (open or closed) and sometimes branch
More Pipes

- In the Mid-Term I example, we had all straight pipes in a pipeline.
- Real pipes end in faucets (open or closed) and sometimes branch.

```scheme
; A pipeline is either
;   - bool
;   - (make-straight sym pipeline)
;   - (make-branch pipeline pipeline)
(define-struct straight (kind next))
(define-struct branch (next1 next2))
```
Example Pipelines

; A pipeline is either
; - bool
; - (make-straight sym pipeline)
; - (make-branch pipeline pipeline)

false
Example Pipelines

; A pipeline is either
;   - bool
;   - (make-straight sym pipeline)
;   - (make-branch pipeline pipeline)

true
Example Pipelines

; A pipeline is either
;  - bool
;  - (make-straight sym pipeline)
;  - (make-branch pipeline pipeline)

(make-straight 'copper false)
Example Pipelines

; A pipeline is either
;   - bool
;   - (make-straight sym pipeline)
;   - (make-branch pipeline pipeline)

(make-straight 'copper
  (make-straight 'lead false))

![Pipe Diagram]
Example Pipelines

; A pipeline is either
;  - bool
;  - (make-straight sym pipeline)
;  - (make-branch pipeline pipeline)

(make-branch
  (make-branch (make-straight 'copper true) false)
  (make-branch false false))
Programming with Pipelines

; A pipeline is either
;   - bool
;   - (make-straight sym pipeline)
;   - (make-branch pipeline pipeline)

(define (func-for-pipeline pl)
  (cond
    [(boolean? pl) ...]
    [(straight? pl)
     ... (straight-kind pl)
     ... (func-for-pipeline (straight-next pl)) ...]
    [(branch? pl)
     ... (func-for-pipeline (branch-next1 pl))
     ... (func-for-pipeline (branch-next2 pl)) ...]]))
Pipeline Examples

• Implement the function `water-running?` which takes a pipeline and determines whether any faucets are open

• Implement the function `modernize` which takes a pipeline and converts all 'lead straight pipes to 'copper

• Implement the function `off` which takes a pipeline and turns off all the faucets

• Implement the function `lead-off` which takes a pipeline and turns off all the faucets that receive water through a lead pipe

• Implement the function `twice-as-long` which takes a pipeline and inserts a 'copper straight pipe before every existing piece of the pipeline