

CS-257L

Nonimperative Programming: Scheme!

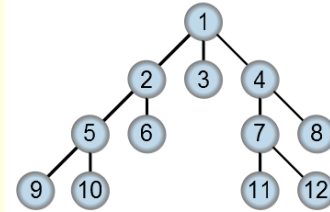
Instructor:

Joel Castellanos

e-mail: joel@unm.edu

Web: <http://cs.unm.edu/~joel/>

Office: Farris Engineering
Center (FEC) room 321



5/7/2008

Data Structure for a Graph

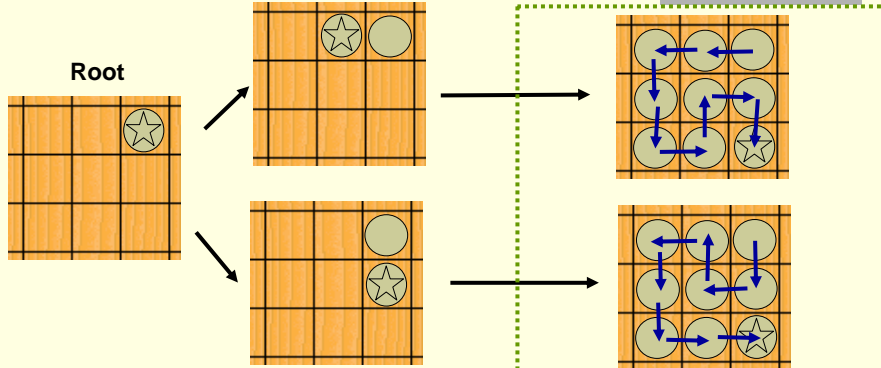
The way a graph is represented and the information stored depends on the needs of the application.

- Are the edges directed?
- Is there a logical root?
- In what ways will the data in the graph need to be accessed?
- Can there be more than one path to a node? (graph or tree).
- How will the graph / tree be build?
 - All at once?
 - Will insertions need to be made between existing edges?

5/7/2008

2

Tron Look-Ahead: Graph or Tree?



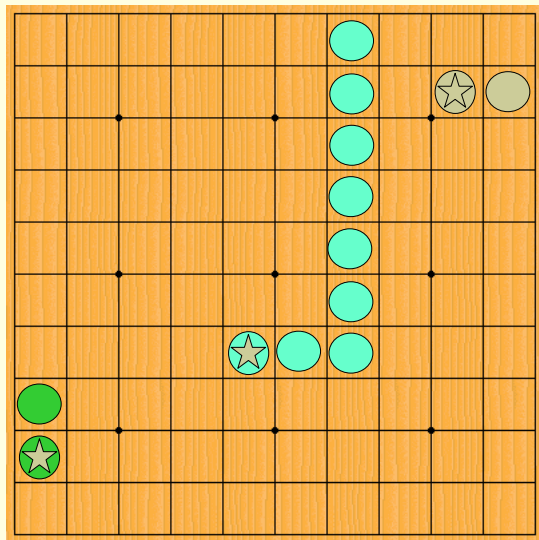
- **Advantage:** Trees are easier to build and use than general graphs.
- **Disadvantage:** One occurrence causes not just one extra node, but one extra subtree.

Is it OK to have one state represented in more than one node?

5/7/2008

3

Tron Look-Ahead: Ignore the Irrelevant



For a look-ahead of 3 moves,
How many children does this state have?

5/7/2008

4

Node Object - Data

```
(load "writeln.scm")
(define make-instance-node (lambda (myName)
  (define name myName)
  (define parent void)
  (define childList ())

  (define node (lambda argList
    (define cmd (car argList))
    (define args (cdr argList))
    (writeln "cmd=" cmd)
    (writeln "args=" args)
  ))
  node
))

(define myNode1 (make-instance-node 'snake))
(myNode1 'setName 'vertebrate)
```

```
Output:
(cmd= setName)
(arg= vertebrate)
```

5/7/2008

5

Node Object - Methods

```
(define make-instance-node (lambda
(myName)
  (define ...
    (define node (lambda argList
      (define cmd (car argList))
      (define args (cdr argList))
      (cond
        ((eq? cmd 'setName) (set!
name (car args)))
        ((eq? cmd 'getName) name)
      ))
      node
    ))
  ))
```

```
Output:
vertebrate
mammal
```

5/7/2008

6

Node Object - Methods

```
(define make-instance-node (lambda
  (myName)
  (define name void)
  (define parentIdx void)
  (define childList ())

  (define node (lambda argList
    (define cmd (car argList))
    (define args (cdr argList))
    (cond
      ((eq? cmd 'setName) (set! name
        (car args)))
      ((eq? cmd 'getName) name)
```

5/7/2008

7