6.090
Building Programming Experience

Lecture 7

1/22/2007

Outline

- Association Lists
- Trees

Association lists

- Want to represent data in a table
- Add new entries
- Look up entries

Name	Value
a	5
b	6
c	(5 6)
d	7

Association Lists

- Build it out of a list
  - `((a 5)`
    `(b 6)`
    `(c (5 6))`
    `(d 7))`

Name	Value
a	5
b	6
c	(5 6)
d	7

Association Lists

```
(define (assoc-add n v lst)
  (cons (list n v) lst))

(define table
  (assoc-add 'a 5
   (assoc-add 'b 6
    null)))
```

Name	Value
a	5
b	6

Association Lists

- Lookup by name:

```
(define (a-lookup n lst)
  (cond ((null? lst) null)
        ((eq? n (caar lst))
         (cadar lst))
        (else
         (a-lookup n (cdr lst)))))
```

Association List Builtins

- assq (builtin) is a little different:

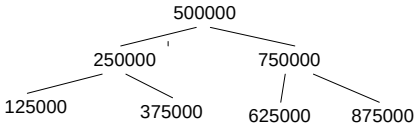
```
(assq 'b '((a 1) (b 2) (c 3)))
  →(b 2)
```

Trees

- Suppose you have a list of sorted numbers:
  - (1,2,4,6,7,9, …, 999996, 999998, 999999)
- Is 789431 in this list?
- How many comparisons will this take?
  - Much too slow

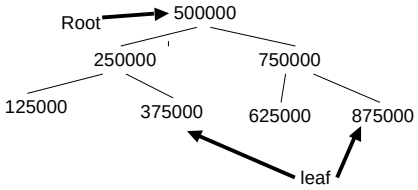
Trees

- Rather than a list, let's build a new data structure: a binary tree



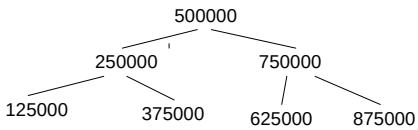
Trees

- Rather than a list, let's build a new data structure: a binary tree



Tree Lookups

- Is 765239 in the tree?
  - Start at the root
  - Is root value smaller or larger?
  - Pick the appropriate branch (subtree)
  - Repeat until found, or until no subtree



How long do lookups take?

- For a balanced binary tree:
  - Each lookup halves the size of the tree
  - 1000000 leaves, only ~20 comparisons
  - 2^{20} ~= 1000000
  - Much faster than a linear search

Trees in Scheme

- Many possible implementations
- Download lec7.scm from website
- Complete abstractions in groups

One caveat

- For lookups to be fast, tree must be balanced
- How long will lookups take in this tree?