Association lists

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>5</td>
</tr>
<tr>
<td>b</td>
<td>6</td>
</tr>
<tr>
<td>c</td>
<td>(5 6)</td>
</tr>
<tr>
<td>d</td>
<td>7</td>
</tr>
</tbody>
</table>

Association Lists

- Build it out of a list
- \((\text{assoc-add } \text{'a} \text{ 5})\)
- \((\text{assoc-add } \text{'b} \text{ 6})\)
- \((\text{assoc-add } \text{'c} \text{ (5 6)})\)
- \((\text{assoc-add } \text{'d} \text{ 7})\)

Association Lists

- Lookup by name:
- \((\text{define } (\text{a-lookup } \text{n} \text{ lst)})\)
- \((\text{define table})\)
- \((\text{assoc-add 'a} \text{ 5})\)
- \((\text{assoc-add 'b} \text{ 6})\)
- \((\text{null})\)
Association List Builtins

• assq (builtin) is a little different:

\[(\text{assq } \ 'b' \ ((a\ 1)\ (b\ 2)\ (c\ 3)))\]

\[\rightarrow (b\ 2)\]

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

\[
\text{Root} \rightarrow \begin{array}{c}
250000 \\
125000 \quad 375000 \quad 625000 \\
500000 \\
\end{array}
\]

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
  - \[2^{20} \approx 1000000\]
  - 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?

```
1 2 3 4 5 6 ...
```