Tell me now if you have a conflict.
Final is Monday the 11th at noon.

Next assignment will be up by Monday.
Problem 5 due next Thursday.

Announcement

CS 180 - Fresh Tabbis (lot.)

11/19/2009
We want to be able to retrieve a
name quickly given a letter.

<table>
<thead>
<tr>
<th>Name</th>
<th>Last Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Austin</td>
<td>210</td>
</tr>
<tr>
<td>John</td>
<td>54</td>
</tr>
<tr>
<td>David</td>
<td>350</td>
</tr>
<tr>
<td>Karen</td>
<td>65</td>
</tr>
<tr>
<td>Erin</td>
<td>109</td>
</tr>
</tbody>
</table>

Ex: 1

Data Share
loops: O(m)
Space: O(m)

BST: balanced
Space: O(n)

List: looks like O(n) → m is # of students
      (not students)
Space: O(n) where m is # of lectures
      Looks like O(1)

Answer: locker # is index of array
        How much space/time would it take?
        How could we store this?
It's always easy to figure out now.

- Color + BPM
- URL + HTML page
- Flight # + Arrival info
- Course # + Scheduling info

Other examples:
Notice: an array is a dictionary

void remove (KeyType key);

void insert (KeyType key, data_type data);

A structure which supports the following:

Dictionaries:
Then we store \((k, v)\) in \([\text{A}][\text{H}(k)]\)

\(N\) should be much smaller than the number of keys.

We should consider the range \([0, N-1]\) as an integer in the range 0 to \(N\).

A hash function \(h\) maps each key in our array is smaller.

We would like to make the key a hash function.
- Don't have collisions.

- Are fast (O(1) time)

- Good hash functions:
Consider the test case $32 - 613 + 2^{13} - 32 - 613$

Cast to int - lots of things could go wrong.

Now what about joining or floating?

What can we do for int char short these?

Say we want 32 bits to fit in an int. 32-bit

First: map key to a number 32-bit
Assuming hash code is defined on ints,

\[
\begin{align*}
32 \times & \text{int} \\
& + \text{code} \\
\text{int} \ \text{hashCode} \ (\text{long} \ x) \\
& \Rightarrow \text{long} \ x \\
& \Rightarrow \text{unsigned} \ (x >> 32) + \text{int} (x) \\
\end{align*}
\]
This can be fine. Remember ASCII?

128-bits (full newest version)

<table>
<thead>
<tr>
<th>temp 01 and temp 10</th>
<th>pmote 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>357214</td>
<td>0</td>
</tr>
</tbody>
</table>
Hensel's rule: \( x-1 + a(x^2 + a(x^3 + \cdots)) \)

\[ 0 \leq a_1 a_2 a_3 \leq 1 \]

Let \( h(x) = x_0 a_1 + x_0 a_2 + x_0 a_3 + \cdots + x_0 a_k \)

Pick \( a \neq 1 \) and split off the last \( k \geq 3 \)-bit parts

A better idea: Polynomial Hash Codes
Chop it at 3 a.m.

What about overflow? (Remember we went only 34 bits in Ruby)

This simple merge sort will collapse.
\[ \text{10100010} \ldots \text{0 1001} \]

Shift bits in representation somehow

Share Shift first codes
- modular arithmetic

Want to spread things out evenly

Ideas? Need something to o

\[ O \neq N \neq 1 \] so is in our answer.

Need to make sure it is symmetric.

Once we have an integer key representation:

Compress into Map.