Next week test on conflict, tel
- Final is 8am on Wed
- The next HW is over hydrogen, next
- HW due next Tuesday

Announcements

CS 2100 - PWSHINS
Let \( m = \text{ # of people} \) and \( n = \text{ # of people} \). Quickly given a locker number, we want to be able to retrieve a name.

<table>
<thead>
<tr>
<th>Name</th>
<th>Locker #</th>
</tr>
</thead>
<tbody>
<tr>
<td>David</td>
<td>201</td>
</tr>
<tr>
<td>Nick</td>
<td>63</td>
</tr>
<tr>
<td>Tracy</td>
<td>101</td>
</tr>
<tr>
<td>Kevin</td>
<td>355</td>
</tr>
<tr>
<td>Don</td>
<td>26</td>
</tr>
</tbody>
</table>

Ex: Locker # Name

New Problem: Data Store
1. Create a vector of strings or names
2. List
3. Insert/Remove: O(n)
4. Lookup: O(1)
5. Insert/Remove: O(n)
Balanced BST (AVL)
to share and look up.
Not always easy to figure out how

- Directors & Moviers
- Color and BMP
- NFL and NHL page
- Flight # and arrival info
- Course # and schedule info

Other examples
Note: Everything is based on graphs.

Vol. 1: Insert, Find (Insert a tree, Find a node)

Defining:
A data structure which supports the

Dictionary class
(go back 3 slices)

Other alternatives:

An array is a dictionary.

First thing to note:

Data Structures
\* \* O(1) Lookup

But then the key nodes to get smaller. We would like to use O(n) space, not O(n^2). Assuming we have an array, it is not very space efficient. Hashing.
Then given \( f(x) \), we solve \( A_{\text{null}} \) \( \neq \emptyset \) if \( N > n \) or \( n \neq 0 \) of course smaller than 1.

Then in our dictionary keys in hash function \( h \) maps each integer in the range \([0, N-1]\).
Good hash functions:

1. Don't have collisions. 
2. Few collisions. 
   a. Test if $h(k_1) = h(k_2)$
5. Key Space (size m)

$N - 2$
$N - 1$
0
m

Don't have collisions. 2 Few collisions.
Since not perfect handle collisions somehow.

Compress that number to $2^{n-1}$

Make key a number with few collisions

So we have a few steps.
Just cast to a #

Ex: char, int, or short (all 32-bit) (Remember, keys can be any type!)

I take key and map it to a number
XOR

5x: Long or Float - 64 bits

It needs to be 32 bits
def hash_code(x):
    return int(unsigned_long(x) >> 32)

int hash_code(long x)
Goal: a single unit

(OR XOR)

011 + 111 + 010

Erin

Think ASCII

What about strings?
We won't to avoid collisions between similar strings (or other types).

But, in some cases, a shorter life this...
\[ p(x) = 69 \cdot 37^3 + 105 \cdot 37^2 + 114 \cdot 37 + 110 \]

**Ex:** Let \( p(x) = x \cdot a_{k-1} + x^{k-2} \cdot a_{k-2} + \ldots + x^2 \cdot a_2 + x \cdot a_1 + a_0 \).

Pick a \( a \neq 1 \) and split data into \( 32 \)-bit parts. Then use Poly nomial Hash Codes.
In order to find the $k$th derivative of a polynomial function $f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x^1 + a_0$, we can use the following recursive formula:

$$f^{(k)}(x) = a_n (n-k) x^{n-k} + a_{n-1} (n-k-1) x^{n-k-1} + \cdots + a_1 (2) x + a_0$$

This formula allows us to find the $k$th derivative of $f(x)$ by substituting the coefficients and the order of the derivative into the formula. Once we have the $k$th derivative, we can use it to answer the question: How long does this take?
Chop down a XOR

What about overflows?

(Works for fists, shrimps, etc.)

This roughly makes it less likely that phenomenal

Shoehorning
Also works well in practice.

Instead of multiplying by $p$, shift each 32-bit piece by some $\#$ of bits.

Alternative to polynomial hashing.