http://www.cs.unh.edu/~ruml/cs758

2 handouts: slides, asst 3
Searching
Dictionaries

Dictionaries


$n$ items, key length $k$

<table>
<thead>
<tr>
<th>Structure</th>
<th>Find</th>
<th>Insert</th>
<th>Delete</th>
</tr>
</thead>
<tbody>
<tr>
<td>List (unsorted)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>List (sorted)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Array (unsorted)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Array (sorted)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Heap</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hash table</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Binary tree (unbalanced)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Binary tree (balanced)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Break

- asst 2
- asst 3
Hash Tables
applications:
1. object method tables
2. string matching
3. nearest points
4. set operations: $\cup, \cap, -$ 

first methods:
1. direct-address tables: small key range. eg, bit vectors.
2. chaining: deletion?
Time Complexity

$n$ items in $m$ buckets

time complexity of search =
$n$ items in $m$ buckets

time complexity of search = number of items per bucket

assume nice hash: $P(h(i) = x) = 1/m$
$n$ items in $m$ buckets

time complexity of search = number of items per bucket

assume nice hash: $P(h(i) = x) = 1/m$

let $X_i$ be 1 iff $h(i) = x$, 0 otherwise

\[
E\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} E[X_i]
\]

\[
= \sum_{i=1}^{n} 1/m
\]

\[
= n/m
\]

let $\alpha = \frac{n}{m}$ ‘load factor’

expected number of items per bucket is $\alpha$

expected time is $\Theta(1 + \alpha)$
probability that $k$ of $n$ elements land in same of $m$ bins:

let $\alpha = \frac{n}{m}$ ‘load factor’

\[
\binom{n}{k} \left(\frac{1}{m}\right)^k \left(1 - \frac{1}{m}\right)^{n-k} \approx \frac{\alpha^k}{e^{\alpha} k!}
\]

<table>
<thead>
<tr>
<th>$k$</th>
<th>probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.37</td>
</tr>
<tr>
<td>1</td>
<td>0.37</td>
</tr>
<tr>
<td>2</td>
<td>0.18</td>
</tr>
<tr>
<td>3</td>
<td>0.06</td>
</tr>
<tr>
<td>4</td>
<td>0.015</td>
</tr>
<tr>
<td>5</td>
<td>0.003</td>
</tr>
<tr>
<td>$&gt;5$</td>
<td>0.002 total</td>
</tr>
</tbody>
</table>

if $n = m$, $\approx \frac{1}{e^k}$.
Open Addressing

1. **linear probing:** \( h(k, i) = (h_1(k) + i) \mod m \) for increasing \( i \)
   - the runs

2. **double hashing:** \( h(k, i) = (h_1(k) + ih_2(k)) \mod m \) for increasing \( i \)
   - requires: \( h_2 \neq 0, h_2(k) \) and \( m \) relatively prime
   - eg, \( m \) prime and \( h_2(k) < m \)
   - or, \( m = 2^x \) and \( h_2(k) \) odd

moral: low load factor

deletion?
Implementing Sets

operations: $\cup, \cap, -$
EOLQs

- What’s still confusing?
- What question didn’t you get to ask today?
- What would you like to hear more about?

Please write down your most pressing question about algorithms and put it in the box on your way out.

Thanks!