(a) Linear Probing

\[ h(k, i) = k \mod 13 + i \]

<table>
<thead>
<tr>
<th>insert these</th>
<th>how many slots do you need to probe to find…</th>
</tr>
</thead>
<tbody>
<tr>
<td>inorder</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>25</td>
</tr>
<tr>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>24</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
</tr>
<tr>
<td>6</td>
<td>25</td>
</tr>
</tbody>
</table>

\[ \alpha = \frac{3}{4} \]

expected # of slots = 4

(b) Double Hashing

\[ h(k, i) = (h_1(k) + ih_2(k)) \mod m \]

= \((k \mod 13) + i(k \mod 12)\)

<table>
<thead>
<tr>
<th>insert the same elements</th>
<th># probes to find…</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, 8, 14, 2, 6, 24, 12, 25</td>
<td>2</td>
</tr>
<tr>
<td>load factor (\alpha = \frac{3}{4})</td>
<td>5</td>
</tr>
<tr>
<td>expected # of probes = 4</td>
<td>26</td>
</tr>
<tr>
<td># probes to find…</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>24</td>
</tr>
</tbody>
</table>
Recitation 7

Agenda

(11.3) Universal hashing
(11.5) Perfect hashing
- MP5 #6 (2)

Reminders

Feedback today
Submit early: avoid the rush - soln soon - ret. W
Exam conflicts: email (reason, Times R 8am-8pm avail) W 10/15 7:30-9:30
Warmup: linear probing & double hashing problems.

universal hashing

idea: choose hash function from a family of hash functions.
- can be done differently on each run.

\[ H: \mathbb{U} \to \mathbb{Z}_m \]

universal:
\[ \forall k_1, k_2 \in \mathbb{U}, k_1 \neq k_2 \]

\[ \text{num } (h \in H) \text{ for which } h(k_1) = h(k_2) \quad \text{(they collide)} \quad \left( \frac{1}{m} \right) \]

So chance of collision is just \( \frac{1}{m} \).

now an adversary cannot force worst-case run time.

our universal hash: \[ h_{a,b}(k) = (ak+b) \mod p \mod m \]

\[ H_{pm} = \left\{ h_{a,b} : a \in \mathbb{Z}_p^*, b \in \mathbb{Z}_p \right\} \]

- \( \mathbb{Z}_p^* = \{ 1, 2, \ldots, p-1 \} \) integers mod \( p \)
- \( \mathbb{Z}_p = \{ 0, 1, 2, \ldots, p-1 \} \)
- \( p \) is prime
- and \( p > |U| \)

more desirable hash properties:

- one way: infeasible given \( x \) to find \( x \) s.t. \( h(x) = y \)
- collision resistance: can't find \( x, x' \) s.t. \( h(x) = h(x') \)
- weak: given \( x \) can't find \( x' \neq x \) s.t. \( h(x') = h(x) \)
- psudorandom: looks random (can't be bias is also repeatable)
- non-malleability: given \( h(x) \) can't produce \( h(x') \) for some \( x' \) related to \( x \)

= collisions can be brute forced in \( O(2^{d/2}) \) (b-day problem)
= Inversions \( O(2^d) \)

ex: passwords, file modifications
Perfect Hashing

- Static keyset
- 2 level hashing scheme

\[ \rightarrow \begin{array}{c} X \ X \ X \\ \downarrow \downarrow \\ X \end{array} \]

- Universal hash @ each level.
- Goal: guarantee \( \Theta(1) \) for operations. (w/o excess space allocation)

**Level 1**: Same as for hashing w/ chaining. (chosen from \( H_{p,m} \))

**Level 2**: A secondary hash table for the keys that collide in slot \( (i) \) with its own hash function \( (h_i) \) chosen from \( H_{p,m} \).

**No Collisions**

Size \( m_i = (\text{# colliding elements})^2 \) \( \leftarrow \) Why? b/c the math works out this way.

\[ \Rightarrow \text{this gives probability of any collisions existing} = \frac{1}{2} \]

\[ \binom{n}{2} \text{ pairs that could collide} \quad \Pr(\text{any collision}) = \frac{1}{m_i} = \frac{n^2}{2} \]

So \( \frac{n!}{2!(n-2)!} = \frac{n(n-1)}{2n^2} < \frac{1}{2} \)

- if collide re-choose \( h_i \) and try again. We expect to have to try 2 \( h_i \)'s to find a non-colliding one.

Overall mem = \( O(n) \) \( \Rightarrow \)

See book for proof

Idea: it is unlikely that the first hash will use only a few of its slots (since it's universal). In order to get bad space performance this would have to happen.

**Example Exercise**

**Cuckoo Hashing**

- \( X \) can be in \( h_1(x) \) or \( h_2(x) \) \( \Rightarrow \) lookup takes constant time (2 slots)
- One elt per slot
- If slots full, evict element there.
  - \( \infty \) loops: set time limit & rehash (can resize too)

Constant amortized cost.
(c) Perfect Hashing

**Level 1**

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>-12</td>
</tr>
<tr>
<td>3</td>
<td>17</td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>

**Level 2**

Choose $a, b$ arbitrarily and see if any collisions occur.

$1 \leq a \leq 53$

$0 \leq b < 53$

Place the following els in their level 1 hash bucket:

17 8 39 41 -1 10

23 27 -12 5 6

Solutions depend on choice of $a, b$.
MD5: message digest 1991

\[ m = \cdots m_3 m_2 m_1 \]

Compression function

Broken 4

MD6
- larger input size
- parallel
- 4:1 compression @ each node
- sequential mode
- in between modes
- every compression function is "keyed"
Please comment on

PSETS (difficulty, length, fun?...)
A:
B:

LECTURES (pace, material, technique...)

Please comment on

PSETS (difficulty, length, fun?)
A:
B:

LECTURES (pace, material, technique...)

RECIPIATIONS (pace, material, technique...)

OTHER

OTHER