Honors Algorithms
G22.3520-001 Fall 2006

Lecture 3
Perfect Hashing

We have $n$ fixed items $a_1, \ldots, a_n$

We want to be able to build a table with these items, so that lookups take constant time — in the worst case

Basic strategy: universal hashing

$m = \# \text{ slots}$

We don’t want any collisions
\[
\text{Pr[collision]} \leq \sum_{i=1}^{n} \sum_{j=1}^{i-1} \text{Pr}[h_K(a_i) = h_K(a_j)] \\
\leq \frac{n(n-1)}{2m}
\]

Assume \( m \geq n(n-1) \), so that we get a collision with probability \( \leq 1/2 \)

Strategy:

repeat

choose a random hash key
hash \( a_1, \ldots, a_n \) using this key

until no collisions
Good news: each iteration succeeds with probability $\geq \frac{1}{2}$

$\therefore$ expected # of iterations $\leq 2$

Bad news: *HUGE* table

A better approach: two levels of universal hashing

- Level 1 segregates items so that not too many go into any one slot
- Level 2 applies the basic strategy to each Level-1 slot
Suppose there are $m \geq 2n$ Level-1 slots

Step 1:

repeat

  choose a random hash key $K$
  hash $a_1, \ldots, a_n$ using $K$

let $L_s := \# \text{ items in slot } s$

let $V' := \sum_s L_s (L_s - 1) = \sum_s L_s^2 - n$

until $V' \leq n$

Step 2:

For each Level-1 slot $s$, use Basic Strategy to hash all items in slot $s$ into a hash table with (at least) $L_s (L_s - 1)$ slots
Analysis

Tool: Markov’s inequality

let $X$ be a random variable taking non-negative values

let $\mu := E[X]

For all $t > 0$: $Pr[X \geq t] \leq \mu/t$

Set $t = 2\mu$: $Pr[X \geq 2\mu] \leq 1/2$

Step 1:

Lecture 1: $E[V'] \leq n^2/m \leq n/2$

Markov: $Pr[V' \geq n] \leq 1/2$

$\therefore$ expected # of iterations $\leq 2$
Analysis (cont’d)

Step 2:

For each slot $s$, we build a sub-table with (at least) $L_s(L_s - 1)$ slots

$\therefore$ we can quickly find a good key for this sub-table

Summary:

- Total expected running time $= O(n)$
- Total size of data structure $= O(n)$
Beyond Pairwise Independence: Uniform Hashing Assumption

let $\mathcal{H} = \{h_k\}_{k \in \mathcal{K}}$ be a family of hash functions, $h_k : \mathcal{U} \to \{0, \ldots, m - 1\}$

we want to hash data sets of size (up to) $n$

let $K$ be uniformly distributed over $\mathcal{K}$

**Uniform Hashing Assumption:**

- each $h_K(a)$ is uniformly distributed over $\{0, \ldots, m - 1\}$
- the family $\{h_K(a)\}_{a \in \mathcal{U}}$ is $n$-wise independent
A very strong assumption
Hard to achieve in practice
Often the assumption is just heuristically applied
“off the shelf” cryptographic hash functions
The Max Load — Revisited

Suppose we hash \( n \) items into \( n \) slots

Let \( M = \max \# \) of data items that hash to any one slot

**Theorem.** Under the Uniform Hashing Assumption,

\[
E[M] = O\left(\frac{\log n}{\log \log n}\right).
\]

**Note:** compare to \( O(\sqrt{n}) \) for pairwise independent hashing
**General Fact:** let $X$ be a random variable that takes only non-negative integer values

Then $E[X] = \sum_{j \geq 1} \Pr[X \geq j]$

Proof by picture ($p_i = \Pr[X = i]$):

\[ p_1 \]
\[ p_2 \quad p_2 \]
\[ p_3 \quad p_3 \quad p_3 \]
\[ \vdots \quad \cdots \quad \cdots \]

$E[X] = \sum_{i \geq 1} i \Pr[X = i]$

Row $i$ sums to $i \Pr[X = i]$

Column $j$ sums to $\Pr[X \geq j]$
Proof of Theorem.

Claim 1: for $j = 1, \ldots, n$: $\Pr[M \geq j] \leq n/j!$

Proof: We are hashing $a_1, \ldots, a_n$

$M \geq j$ iff for some subset of indices $\{i_1, \ldots, i_j\}$, the items $a_{i_1}, \ldots, a_{i_j}$ hash to the same slot

For any fixed subset, this happens with probability $1/n^{j-1}$:

- $a_{i_1}$ can hash into any slot $s$
- the other $j-1$ must hash into slot $s$
Summing over all subsets of size $j$:

$$\Pr[M \geq j] \leq \binom{n}{j} \cdot \frac{1}{n^{j-1}}$$

$$= \frac{n(n-1) \cdots (n-j+1)}{j!} \cdot \frac{1}{n^{j-1}}$$

$$\leq \frac{n}{j!}$$

That proves the claim
Define \( f(n) := \) least \( j \) such that \( n/j! \leq 1 \)

**Claim 2:** \( f(n) = O(\log n / \log \log n) \)

Sketch: we want \( \log n \leq \log j! \approx j \log j \)

This happens when \( j \) is roughly \( \log n / \log \log n \)

We have

\[
E[M] = \sum_{j \geq 1} \Pr[M \geq j] \\
\leq \sum_{j \leq f(n)} \Pr[M \geq j] + \sum_{j > f(n)} \Pr[M \geq j] \\
\leq f(n) + \sum_{j > f(n)} \frac{n}{j!} \leq f(n) + \sum_{i \geq 1} 1/2^i \\
= f(n) + 1 = O(\log n / \log \log n) 
\]

QED
Bloom Filters

A fixed set $S = \{a_1, \ldots, a_n\} \subset U$

Data structure: an array of $m$ bits

Use $\ell$ hash functions $h_1, \ldots, h_\ell$

set bits $h_i(a_j)$ for $i = 1, \ldots, \ell, j = 1, \ldots, n$

to test if $a \in U$:

- test if bits $h_1(a), \ldots, h_\ell(a)$ are all set

Pros: very compact (just a bit vector – no pointer, no data)

Cons: “false positives”
Analysis: \( a \notin S \) is a false positive if
\[
\forall i' \ \exists j, \ i : h_{i'}(a) = h_i(a_j)
\]

For any fixed \( i', j, i \):
\[
\Pr[h_{i'}(a) = h_i(a_j)] = \frac{1}{m}
\]

For any fixed \( i' \):
\[
\Pr \left[ \forall j, \ i : h_{i'}(a) \neq h_i(a_j) \right] = \left(1 - \frac{1}{m}\right)^{n \ell}
\]

False positive rate:
\[
\Pr \left[ \forall i' \ \exists j, \ i : h_{i'}(a) = h_i(a_j) \right] = \left(1 - \left(1 - \frac{1}{m}\right)^{n \ell}\right)^\ell
\]
Use the approximation $1 + x \approx e^x$

False positive rate:

$$\left(1 - (1 - 1/m)^{n\ell}\right)\ell \approx (1 - e^{-\ell n/m})\ell$$

For fixed $m/n$, this is minimized at $\ell = (m/n)\log 2$

For this $\ell$, false positive rate $\approx (0.62)^{m/n}$

Example: $m/n = 10$

<table>
<thead>
<tr>
<th>$\ell$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>.0952</td>
<td>.0329</td>
<td>.0174</td>
<td>.0118</td>
<td>.00943</td>
<td>.00844</td>
<td>.00819</td>
<td>.00846</td>
</tr>
</tbody>
</table>

We get < 1% false positive rate with 10 bits per dictionary entry
Bloom Filters: properties and applications

Can add items to dictionary, but not delete

can compute union and set difference of Bloom filters (bit-wise OR)

Reduce workload on databases,

Minimize access to large/slow memory

Privacy: can distribute/publish a Bloom filter, without explicitly revealing the items in the dictionary