Randomness in Computing

Lecture 16

Last time
• Hashing
• Universal hash families

Today
• Using universal hash families
• Perfect hashing
• Bloom filters
Motivating example
Password checker to prevent people from using common passwords.

- $S$ is the set of common passwords

- **Universe**: set $U$
- $S \subseteq U$ and $m = |S|$
- $m \ll |U|$

**Goal**: A data structure for storing $S$ that supports the search query

"Does $w \in S$?" for all words $w \in U$. 
Solutions

Deterministic solutions

• Store $S$ as a sorted array (or as a binary search tree)

Search time: $O(\log m)$, Space: $O(m)$

• Store an array that for each $w \in U$ has 1 if $w \in S$ and 0 otherwise.

Search time: $O(1)$, Space: $O(|U|)$

A randomized solution

• Hashing
Chain Hashing

- **Hash table:** \( n \) bins, words that fall in the same bin are chained into a linked list.

- **Hash function:** \( h : U \rightarrow [n] \)

To construct the table

- hash all elements of \( S \)

To search for word \( w \)

- check if \( w \) is in bin \( h(w) \)

**Desiderata for \( h \):**

- \( O(1) \) evaluation time.
- \( O(1) \) space to store \( h \).
A set $\mathcal{H}$ of hash functions is **universal** if for every pair $w_1, w_2 \in U$ and for $h$ chosen uniformly from $\mathcal{H}$

$$\Pr[h(w_1) = h(w_2)] \leq \frac{1}{n}$$

**Constructing a universal hash family**

- Fix a prime $p \geq |U|$ and think of the range as $\{0, 1, \ldots, n - 1\}$.
- Define $h_{a,b}(x) = ((ax + b) \mod p) \mod n$
  $$\mathcal{H} = \{h_{a,b} \mid a \in [p - 1], 0 \leq b \leq p - 1\}$$

**Theorem**

$\mathcal{H}$ is universal.
Using a universal family

As before:

• If \( w \notin S \), expected number of words in bin \( h(w) \) is \( \leq \frac{m}{n} \)

• If \( w \in S \), expected number of words in bin \( h(w) \) is \( \leq 1 + \frac{m - 1}{n} \)

The previous guarantee on max load no longer holds!

Goal: Given \( S \), find a hash function with no collisions for words in \( S \).
Perfect hashing: no collisions

**Theorem**

If \( h: U \rightarrow \{0,1,\ldots,n - 1\} \) is chosen uniformly at random from a universal hash family, then \( \forall S \) of size \( m \), such that \( n \geq m^2 \),
\[
\Pr[h \text{ is perfect}] \geq 1/2.
\]

**Proof:** Let \( s_1, \ldots, s_m \) be elements of \( S \).
Perfect hashing

**Theorem**

If $h: U \rightarrow \{0,1, \ldots, n-1\}$ is chosen uniformly at random from a universal hash family, then $\forall S$ of size $m$, such that $n \geq m^2$, $\Pr[h \text{ is perfect}] \geq 1/2$.

- Select $h \in \mathcal{H}$ until a perfect $h$ is found.
- Expected number of tries is at most 2.
- Each try takes $O(m)$ time.
- **Drawback:** $\Omega(m^2)$ space.
2-level scheme for perfect hashing

- Set $m = n$.
- Select $h \in \mathcal{H}$ until $h$ with at most $m$ collisions is found.
- For each bin $i$ with collisions, that is, with $k > 1$ items:
  - select a new hash function $h_i$ with $k^2$ bins from a universal family until $h_i$ has no collisions.
2-level scheme for perfect hashing

- Set $m = n$.
- Select $h \in \mathcal{H}$ until $h$ with at most $m$ collisions is found.
- For each bin $i$ with collisions, that is, with $k > 1$ items:
  - select a new hash function $h_i$ with $k^2$ bins from a universal family until $h_i$ has no collisions.

**Theorem**

2-level scheme achieves perfect hashing with $O(m)$ space.

**A solution for static dictionary problem with:**

- $O(1)$ worst case guarantee on search time.
- $O(m)$ space.
- Expected $O(m)$ preprocessing time.
**Theorem**

2-level scheme achieves perfect hashing with $O(m)$ space.

**Proof:**

- Let $X = \# \text{ of collisions in Stage 1}$. 

- We showed before: $\Pr \left[ X > \frac{m^2}{n} \right] \leq \frac{1}{2}$. 

- Now $m = n$: $\Pr[X > m] \leq \frac{1}{2}$. 

- So at least half of $h \in \mathcal{H}$ have $\leq m$ collisions. 

- Assume we found such $h$. 

Analysis of 2-level scheme

**Theorem**

2-level scheme achieves perfect hashing with $O(m)$ space.

**Proof (continued):**
Conclusion: 2-level hashing

A solution for static dictionary problem with:

• $O(1)$ worst case guarantee on search time.
• $O(m)$ space.
• Expected $O(m)$ preprocessing time.
Approximate solutions
for static dictionary problem

- **False positives:** If \( w \in S \), our data structure must answer correctly. If \( w \notin S \), we may err with small probability.
- E.g., we prevent all unsuitable passwords and some suitable ones, too.

**Fingerprints**

- Use hash function \( h \)
- Store sorted list \( L \) of fingerprints \( h(x), x \in S \).
- To see if \( w \in S \), perform binary search for \( h(w) \).
Bloom filters

- Trade off between space and false positive probability
- Parameters $k, n$
- Bloom filter: array of $n$ bits $A[1], \ldots, A[n]$
  - Initially: all bits are 0
  $k$ independent random hash functions $h_1, \ldots, h_k$ with range $[n]$
- To represent set $S$
  - For each $x \in S$ and $i \in [k]$, set bits $A[h_i(x)]$ to 1.
- To search for $w$:
  - If for all $i \in [k]$, bits $A[h_i(w)] = 1$, accept, o.w. reject.