Last POTD today! No labs this week!
Learning Objectives

Review and finalize fundamentals of bloom filters

Discuss strategies for counting the occurrences of objects

Introduce the concept of cardinality and cardinality estimation
Sketch

A “sketch” is a compact (reduced) representation of a dataset that acts as a replacement for calculations.
Bloom Filters

A probabilistic data structure storing a set of values $h_{\{1,2,3,\ldots,k\}}$

Has three key properties:

- $k$, number of hash functions
- $n$, expected number of insertions
- $m$, filter size in bits

Expected false positive rate:

$$\left(1 - \left(1 - \frac{1}{m}\right)^{nk}\right)^k \approx \left(1 - e^{-\frac{nk}{m}}\right)^k$$

Optimal accuracy when:

$$k^* = \ln 2 \cdot \frac{m}{n}$$
Bloom Filter: Website Caching

Loaded this before?

0 1 0 1 0 1

Cache this page!

Add to filter (but don’t cache!)

Sequence Bloom Trees

Imagine we have a large collection of text…

And our goal is to search these files for a query of interest…
Sequence Bloom Trees

SRA 00001  SRA 00002  SRA 00003  SRA 00004  SRA 00005  SRA 00006  SRA 00007  SRA 00008
Bloom Filters: Unioning

Bloom filters can be trivially merged using bit-wise union.

\[
\begin{array}{cccc}
0 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 \\
2 & 1 & 2 & 1 \\
3 & 1 & 3 & 0 \\
4 & 0 & 4 & 0 \\
5 & 0 & 5 & 0 \\
6 & 1 & 6 & 1 \\
7 & 0 & 7 & 1 \\
8 & 0 & 8 & 1 \\
9 & 1 & 9 & 1 \\
\end{array}
\]

\[ U = \]

\[
\begin{array}{cccc}
0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 \\
2 & 2 & 2 & 2 \\
3 & 3 & 3 & 3 \\
4 & 4 & 4 & 4 \\
5 & 5 & 5 & 5 \\
6 & 6 & 6 & 6 \\
7 & 7 & 7 & 7 \\
8 & 8 & 8 & 8 \\
9 & 9 & 9 & 9 \\
\end{array}
\]
Sequence Bloom Trees

Are $\geq \theta$ fraction of query kmers $\in$ this Bloom filter?

If YES, move to children

If NO, stop looking at this subtree (Global mismatch)
Sequence Bloom Trees

<table>
<thead>
<tr>
<th></th>
<th>SRA</th>
<th>FASTA.gz</th>
<th>SBT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leaves</td>
<td>4966 GB</td>
<td>2692 GB</td>
<td>63 GB</td>
</tr>
<tr>
<td>Full Tree</td>
<td>-</td>
<td>-</td>
<td>200 GB</td>
</tr>
</tbody>
</table>

Bloom Filters: Tip of the Iceberg


There are many more than shown here…
Counting Sketches

Sometimes we need more information than ‘presence/absence’…
Counting Bloom Filters

Instead of using one bit per register, let's use multiple!
Counting Bloom Filters

$S = \{ 16, 8, 4, 13, 29, 11, 22 \}$

$h_1(k) = k \% 7 \quad h_2(k) = 2k+1 \% 7$
Counting Bloom Filters

\[ S = \{ 16, 8, 4, 13, 29, 11, 22 \} \]

\[ h_1(k) = k \mod 7 \quad h_2(k) = 2k + 1 \mod 7 \]

<table>
<thead>
<tr>
<th>_find(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>_find(5)</td>
</tr>
</tbody>
</table>
Counting Bloom Filters

\[ S = \{ 16, 8, 4, 13, 29, 11, 22 \} \]

\[ h_1(k) = k \mod 7 \quad h_2(k) = 2k + 1 \mod 7 \]

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
</tr>
</tbody>
</table>

\_delete(8) \_delete(5)
Counting Bloom Filters

A probabilistic data structure storing a set of values

Has **four** key properties:

- $k$, number of hash functions
- $n$, expected number of insertions
- $m$, filter size in **registers**
- $b$, number of bits per register

Can handle deletions at the cost of allowing false negatives!
Counting Bloom Filters

Pro:

Con:
Counting Bloom Filters

*At time of insertion, what information do we have?*

| 3 | 4 | 2 | 1 | 4 | 6 | 8 | 7 | 2 | 0 |
Minimal Increase

\[ S = \{1, 3, 5, 8\} \]

\[ h_1(k) = k \mod 5 \quad h_2(k) = (3k + 1) \mod 5 \quad h_3(k) = |k - 4| \mod 5 \]
## Minimal Increase

\[ S = \{1, 3, 5, 8\} \]

\[ h_1(k) = k \mod 5 \quad h_2(k) = (3k+1) \mod 5 \quad h_3(k) = |k - 4| \mod 5 \]

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>4</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>3</td>
<td>4</td>
<td></td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Naive**

- \_find(3)
- \_find(5)
- \_find(8)

**Minimal Increase**

- \_find(3)
- \_find(5)
- \_find(8)
Counting Bloom Filters

Do we know anything about our collision frequency at insertion?

| 3 | 4 | 2 | 1 | 4 | 6 | 8 | 7 | 2 | 0 |
Spectral Bloom Filter

A counting bloom filter with two key optimizations:

1) **Minimal Increase**: On insert, only increment counts that have the minimum value.

2) **Recurring Minimum**: Insertions that have only a single minimum value have unusually high collision likelihood!

For these values, create a second spectral bloom filter and store them in both.
Bloom Filters: Tip of the Iceberg II


There are many more than shown here…
The hidden problem with bloom filters...
Bloom Filter: Optimal Parameters

\[ k^* = \ln 2 \cdot \frac{m}{n} \]  

Given any two values, we can optimize the third

... but we often have to guess an approximate value for \( n \)!
Cardinality: how many *distinct* values in a data stream?

<table>
<thead>
<tr>
<th>3201</th>
<th>946</th>
<th>5581</th>
<th>8945</th>
<th>6145</th>
<th>8126</th>
<th>3887</th>
<th>8925</th>
</tr>
</thead>
<tbody>
<tr>
<td>1246</td>
<td>8324</td>
<td>4549</td>
<td>9100</td>
<td>5598</td>
<td>8499</td>
<td>8970</td>
<td>3921</td>
</tr>
<tr>
<td>8575</td>
<td>4859</td>
<td>4960</td>
<td>42</td>
<td>6901</td>
<td>4336</td>
<td>9228</td>
<td>3317</td>
</tr>
<tr>
<td>399</td>
<td>6925</td>
<td>2660</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Cardinality

I take cards labeled 1--1,000 and choose a random subset of size $N$ to hide in my hat.

We want to estimate $N$.

We can see one representative from the cards in the hat; which to pick?

Minimum, median, maximum? Something else?
Cardinality

What if *minimum* was 500? ...10? ... 4?
If minimum is 95, what's our estimate for $N$?
Cardinality

What if \textbf{minimum} was 500? ...10? ... 4?

If minimum is 95, what's our estimate for \( N \)?

\[
N + 1 \approx 10.5
\]

\[
N \approx 9.5
\]

Conceptually: If we scatter \( N \) points randomly across the interval, we end up with \( N + 1 \) parts, each about \( 1000/(N + 1) \) long.

Assuming our first ‘partition’ is about average:  

\[
95 \approx 1000/(N + 1)
\]

\[
N + 1 \approx 10.5
\]

\[
N \approx 9.5
\]
Now imagine we have a SUHA hash (let $h_{64}$ be a 64-bit hash)

The randomness in the hash function turns any dataset-cardinality problem into the “hat problem”
Cardinality

Let \( M = \min(X_1, X_2, \ldots, X_N) \), where each \( X_i \) is an independent uniform draw between \([0, 1]\)

Claim: \( \mathbb{E}[M] = \frac{1}{N + 1} \)
Cardinality

<table>
<thead>
<tr>
<th>Attempt 1</th>
<th>0.455</th>
<th>0.220</th>
<th>0.951</th>
<th>0.236</th>
<th>0.979</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attempt 2</td>
<td>0.968</td>
<td>0.234</td>
<td>0.835</td>
<td>0.642</td>
<td>0.349</td>
</tr>
<tr>
<td>Attempt 3</td>
<td>0.774</td>
<td>0.484</td>
<td>0.309</td>
<td>0.526</td>
<td>0.143</td>
</tr>
</tbody>
</table>
Cardinality

Can the $k^{th}$-smallest hash value estimate the cardinality better than the minimum?
Cardinality

Can the $k^{th}$-smallest hash value estimate the cardinality better than the *minimum*?
Can the $k^{th}$-smallest hash value estimate the cardinality better than the minimum?

\[
\begin{align*}
    E[M_1] &= \frac{1}{N + 1} \\
    E[M_k] &= \frac{k}{N + 1}
\end{align*}
\]
Cardinality

\[
\frac{1}{N + 1} = \frac{\mathbf{E}[M_k]}{k}
\]
Cardinality

\[
\frac{1}{N + 1} = \frac{\mathbb{E}[M_k]}{k} = \left[ \mathbb{E}[M_1] + (\mathbb{E}[M_2] - \mathbb{E}[M_1]) + \ldots + (\mathbb{E}[M_k] - \mathbb{E}[M_{k-1}]) \right] \cdot \frac{1}{k}
\]

\(k^{th}\) minimum value (KMV)  

Averages \(k\) estimates for \(\frac{1}{N + 1}\)
Cardinality

True cardinality = 1,000