Data Structures and Algorithms
Bloom and Counting Sketches

CS 225
G Carl Evans

April 26, 2023
Bloom Filter: Search

The bloom filter is a *probabilistic* data structure!

If the value in the BF is 0:

If the value in the BF is 1:
Probabilistic Accuracy: One-sided error

Dataset:

search with one-sided error

Query:

search with one-sided error

...
Bloom Filter: Repeated Trials

\[ h_{1,2,3,...,k}(y) \]
Bloom Filter: Repeated Trials

If any query yields 0, item is not in the set

$h_{1,2,3,...,k}(y)$
If all queries yield 1, item *may* be in the set; or we might have collided *k* times.

$h_{1,2,3,...,k}(y)$
Bloom Filter: Repeated Trials

But doesn’t this hurt our storage costs by storing $k$ separate filters?

$h_1 \quad h_2 \quad h_3 \quad \cdots \quad h_k$
Bloom Filter: Repeated Trials

Rather than use a new filter for each hash, one filter can use $k$ hashes

$S = \{ 6, 8, 4 \}$

$h_1(x) = x \% 10 \quad h_2(x) = 2x \% 10 \quad h_3(x) = (5+3x) \% 10$
Bloom Filter: Repeated Trials

Rather than use a new filter for each hash, one filter can use $k$ hashes

$h_1(x) = x \mod 10$ \hspace{1cm} h_2(x) = 2x \mod 10 \hspace{1cm} h_3(x) = (5+3x) \mod 10$

_find(1)

find(16)
Bloom Filter

A probabilistic data structure storing a set of values

Built from a bit vector of length $m$ and $k$ hash functions

Insert / Find runs in: _________________

Delete is not possible (yet)!

$H = \{h_1, h_2, \ldots, h_k\}$
Bloom Filter: Error Rate

Given bit vector of size $m$ and $k$ SUHA hash function $h_{\{1,2,3,\ldots,k\}}$

What is our expected FPR after $n$ objects are inserted?
Bloom Filter: Error Rate

Given bit vector of size $m$ and 1 SUHA hash function

What's the probability a specific bucket is 1 after one object is inserted?

Same probability given $k$ SUHA hash function?
Bloom Filter: Error Rate

Given bit vector of size $m$ and $k$ SUHA hash function $h_{\{1,2,3,...,k\}}$

Probability a specific bucket is 0 after one object is inserted?

After $n$ objects are inserted?
Bloom Filter: Error Rate

Given bit vector of size $m$ and $k$ SUHA hash function

What's the probability a specific bucket is 1 after $n$ objects are inserted?
Bloom Filter: Error Rate

Given bit vector of size $m$ and $k$ SUHA hash function $h_{1,2,3,...,k}$

What is our expected FPR after $n$ objects are inserted?

The probability my bit is 1 after $n$ objects inserted

$$\left(1 - \left(1 - \frac{1}{m}\right)^n\right)^k$$

The number of [assumed independent] trials
Bloom Filter: Error Rate

Vector of size $m$, $k$ SUHA hash function, and $n$ objects

To minimize the FPR, do we prefer…

(A) large $k$  

(B) small $k$

\[
\left(1 - \left(1 - \frac{1}{m}\right)^{nk}\right)^k
\]
Bloom Filter: Optimal Error Rate

So how can we find the minimum error rate?
Bloom Filter: Optimal Error Rate

**Claim:** The optimal hash function is when $k^* = \ln 2 \cdot \frac{m}{n}$

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

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

Derivative is zero when $k^* = \ln 2 \cdot \frac{m}{n}$
Bloom Filter: Error Rate

\( (1 - e^{-nk/m})^k \)

\( m/n = 10 \)

\( k^* = \ln 2 \cdot 10 = 6.93 \)

Figure by Ben Langmead
Bloom Filter: Optimal Parameters

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

Given any two values, we can optimize the third

- \( n = 100 \) items  \( k = 3 \) hashes  \( m = \)
- \( m = 100 \) bits  \( n = 20 \) items  \( k = \)
- \( m = 100 \) bits  \( k = 2 \) items  \( n = \)
Bloom Filter: Optimal Parameters

\[
m = \frac{nk}{\ln 2} \approx 1.44 \cdot nk
\]

Optimal hash function is still \(O(n)\)!

\(n = 60\) billion — 130 trillion

\(n = 250,000\) files vs \(~10^{15}\) nucleotides vs 260 TB
Bloom Filter: Website Caching

0
1
0
1
0
1

Loaded this before?

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...
Bloom Filters: Unioning

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

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

\[U\]

= 

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

- **STAR (CPU time):** >2.5 years
- **STAR (15-thread):** >2 days
- **SRA-BLAST:** 19 mins
- **SBT:** Single CPU

<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

A **sketch** is a compact (reduced) representation of a dataset that acts as a replacement for calculations.

Sometimes we need more information than ‘presence/absence’…
### Count Min Sketch

<table>
<thead>
<tr>
<th></th>
<th>000</th>
<th>110</th>
<th>010</th>
<th>001</th>
<th>100</th>
<th>110</th>
<th>000</th>
<th>000</th>
<th>000</th>
<th>100</th>
<th>111</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Count Min Sketch
Count Min Sketch

A **count-min sketch** has $k$ rows, each with its own hash function.

Each bucket is $b$-bits (allowing $2^b$ count values)

| $h_1(x)$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $h_2(x)$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $h_3(x)$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $h_4(x)$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $h_5(x)$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
Count Min Sketch Insertion

\[ S = \{ 1, 3, 8, 16 \} \]

\[ h_1(k) = k \mod 7 \quad h_2(k) = k + 3(k \mod 2) \mod 7 \quad h_3(k) = |k - 4| \mod 7 \]
Count Min Sketch Find

\( h_1(k) = k \mod 7 \quad h_2(k) = k + 3(k \mod 2) \mod 7 \quad h_3(k) = |k - 4| \mod 7 \)

\_find(16)

\_find(1)

\[ S = \{1, 3, 8, 16\} \]
Count Min Sketch Find

What is our estimated count of $x$?

How many **known** collisions?

<table>
<thead>
<tr>
<th>$h_1(x)$</th>
<th>10</th>
<th>5</th>
<th>13</th>
<th>17</th>
<th>8</th>
<th>1</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_2(x)$</td>
<td>3</td>
<td>7</td>
<td>20</td>
<td>12</td>
<td>2</td>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td>$h_3(x)$</td>
<td>12</td>
<td>6</td>
<td>7</td>
<td>5</td>
<td>9</td>
<td>18</td>
<td>3</td>
</tr>
<tr>
<td>$h_4(x)$</td>
<td>4</td>
<td>19</td>
<td>26</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>$h_5(x)$</td>
<td>6</td>
<td>6</td>
<td>8</td>
<td>11</td>
<td>6</td>
<td>7</td>
<td>16</td>
</tr>
</tbody>
</table>
### Improving Count Min Sketch

Given what we know about collisions here, can we improve our insert strategy?

<table>
<thead>
<tr>
<th></th>
<th>$h_1(x)$</th>
<th>$h_2(x)$</th>
<th>$h_3(x)$</th>
<th>$h_4(x)$</th>
<th>$h_5(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10</td>
<td>3</td>
<td>12</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>7</td>
<td>6</td>
<td>19</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>20</td>
<td>7</td>
<td>26</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>17</td>
<td>12</td>
<td>5</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>2</td>
<td>9</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>9</td>
<td>18</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>7</td>
<td>3</td>
<td>1</td>
<td>16</td>
</tr>
</tbody>
</table>
Count Min Sketch Improved Insertion

\( S = \{ \ldots, 1, 3, 8, 16, \ldots \} \)

\( h_1(k) = k \mod 7 \quad h_2(k) = (k+3(k\mod 2)) \mod 7 \quad h_3(k) = |k - 4| \mod 7 \)
## Count Min Sketch Insertion

**Minimal Increase:** When inserting, only update the minimum count.

### Default

<table>
<thead>
<tr>
<th>$h_1$</th>
<th>7</th>
<th>11</th>
<th>11</th>
<th>2</th>
<th>5</th>
<th>7</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_2$</td>
<td>4</td>
<td>9</td>
<td>8</td>
<td>5</td>
<td>5</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>$h_3$</td>
<td>15</td>
<td>13</td>
<td>2</td>
<td>6</td>
<td>9</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

### Minimal Increase

<table>
<thead>
<tr>
<th>$h_1$</th>
<th>7</th>
<th>9</th>
<th>10</th>
<th>2</th>
<th>5</th>
<th>7</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_2$</td>
<td>4</td>
<td>9</td>
<td>7</td>
<td>5</td>
<td>5</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td>$h_3$</td>
<td>15</td>
<td>12</td>
<td>2</td>
<td>5</td>
<td>9</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>
Count-Min Sketch

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

**Minimal increase** reduces overcounting by identifying collisions.

Count value returned here [underestimates / overestimates / matches] the true count of the query.
Trivia Point 1: Deletion is Possible!

<table>
<thead>
<tr>
<th>$h_1(x)$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>2</th>
<th>3</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_2(x)$</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>$h_3(x)$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>$h_4(x)$</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$h_5(x)$</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Trivia Point 2: Counting Bloom Filters

<table>
<thead>
<tr>
<th>$h_1(x)$</th>
<th>10</th>
<th>5</th>
<th>13</th>
<th>17</th>
<th>8</th>
<th>1</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_2(x)$</td>
<td>3</td>
<td>7</td>
<td>20</td>
<td>12</td>
<td>2</td>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td>$h_3(x)$</td>
<td>12</td>
<td>6</td>
<td>7</td>
<td>5</td>
<td>9</td>
<td>18</td>
<td>3</td>
</tr>
<tr>
<td>$h_4(x)$</td>
<td>4</td>
<td>19</td>
<td>26</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>$h_5(x)$</td>
<td>6</td>
<td>6</td>
<td>8</td>
<td>11</td>
<td>6</td>
<td>7</td>
<td>16</td>
</tr>
</tbody>
</table>
Trivia Point 2: Counting Bloom Filters

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

\[ h_1(k) = k \mod 7 \quad h_2(k) = (2k+1) \mod 7 \]
Trivia Point 2: Counting Bloom Filters

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

\[ h_1(k) = k \mod 7 \quad h_2(k) = 2k+1 \mod 7 \]
Next Time: Cardinality

**Cardinality:** how many *distinct* values in a data stream?