Learning Objectives

Build a conceptual understanding of a bloom filter

Review probabilistic data structures and one-sided error

Formalize the math behind the bloom filter

Introduce extensions to the bloom filter
Data Structures Review

What method would you use to build a search index on a collection of objects?
Memory-Constrained Data Structures

What method would you use to build a search index on a collection of objects *in a memory-constrained environment*?

Constrained by Big Data (Large $N$)

Google Index Estimate: >60 billion webpages
Google Universe Estimate (2013): >130 trillion webpages
What method would you use to build a search index on a collection of objects in a memory-constrained environment?

Constrained by Big Data (Large $N$)

Sequence Read Archive Size: $>60$ petabases ($10^{15}$)
Memory-Constrained Data Structures

What method would you use to build a search index on a collection of objects in a memory-constrained environment?

Constrained by Big Data (Large $N$)

<table>
<thead>
<tr>
<th>Sky Survey Projects</th>
<th>Data Volume</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPOSS (The Palomar Digital Sky Survey)</td>
<td>3 TB</td>
</tr>
<tr>
<td>2MASS (The Two Micron All-Sky Survey)</td>
<td>10 TB</td>
</tr>
<tr>
<td>GBT (Green Bank Telescope)</td>
<td>20 PB</td>
</tr>
<tr>
<td>GALEX (The Galaxy Evolution Explorer)</td>
<td>30 TB</td>
</tr>
<tr>
<td>SDSS (The Sloan Digital Sky Survey)</td>
<td>40 TB</td>
</tr>
<tr>
<td>SkyMapper Southern Sky Survey</td>
<td>500 TB</td>
</tr>
<tr>
<td>PanSTARRS (The Panoramic Survey Telescope and Rapid Response System)</td>
<td>~ 40 PB expected</td>
</tr>
<tr>
<td>LSST (The Large Synoptic Survey Telescope)</td>
<td>~ 200 PB expected</td>
</tr>
<tr>
<td>SKA (The Square Kilometer Array)</td>
<td>~ 4.6 EB expected</td>
</tr>
</tbody>
</table>


Estimated total volume of one array: 4.6 EB
What method would you use to build a search index on a collection of objects in a memory-constrained environment?

Constrained by resource limitations

- cache: < 1 second
- RAM: Hours - Days
- disk: Months
- network: Years

(Estimates are Time x 1 billion courtesy of https://gist.github.com/hellerbarde/2843375)
What method would you use to build a search index on a collection of objects \textit{in a memory-constrained environment}?
Reducing storage costs

1) Throw out information that isn’t needed

2) Compress the dataset
Reducing a hash table

What can we remove from a hash table?

$H(k_1) = i_1$

$m$

$k_1 \rightarrow v_1$

$k_2 \rightarrow v_2$

$k_3 \rightarrow v_3$

$k_4 \rightarrow v_4$

$k_5 \rightarrow v_5$

$k_6 \rightarrow v_6$

$k_7 \rightarrow v_7$

$k_8 \rightarrow v_8$
Reducing a hash table

What can we remove from a hash table?

Take away values
Reducing a hash table

What can we remove from a hash table?

Take away values and keys

\[ H(k_1) = i_1 \]
Reducing a hash table

What can we remove from a hash table?

Take away values and keys

This is a *bloom filter*
Bloom Filter: Insertion

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

\[ h(k) = k \% 7 \]

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Bloom Filter: Insertion

An item is inserted into a bloom filter by hashing and then setting the hash-valued bit to 1

If the bit was already one, it stays 1
Bloom Filter: Deletion

\[ S = \{ 16, 8, 4, 13, 29, 11, 22 \} \]
\[ h(k) = k \% 7 \]

\_delete(13)

\_delete(29)
Bloom Filter: Deletion

Due to hash collisions and lack of information, items cannot be deleted!
Bloom Filter: Search

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

$h(k) = k \% 7$

_find(16)

_find(20)

_find(3)
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: Malicious Websites

Imagine we have a detection oracle that identifies if a site is malicious.

- CS 225
- Introduction to Data Structures and Algorithms with C++

- "Not malicious"

- "Malicious"
Probabilistic Accuracy: Malicious Websites

Imagine we have a detection oracle that identifies if a site is malicious.

True Positive:

False Positive:

False Negative:

True Negative:
Imagine we have a **bloom filter** that **stores malicious sites**...

<table>
<thead>
<tr>
<th></th>
<th>Bit Value = 1</th>
<th>Bit Value = 0</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Item Inserted</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>True Positive</td>
<td>![Yes](H(z) = 0, 1)</td>
<td>![No](H(z) = 0, 0)</td>
</tr>
<tr>
<td>False Negative</td>
<td>![Yes](H(z) = 0, 1)</td>
<td>![No](H(z) = 0, 0)</td>
</tr>
<tr>
<td><strong>Item NOT inserted</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>False Positive</td>
<td>![Yes](H(z) = 0, 1)</td>
<td>![No](H(z) = 0, 0)</td>
</tr>
<tr>
<td>True Negative</td>
<td>![Yes](H(z) = 0, 1)</td>
<td>![No](H(z) = 0, 0)</td>
</tr>
</tbody>
</table>
Probabilistic Accuracy: One-sided error

We will NEVER have a False Negative:

We will get some False Positives:

We will NEVER have a False Negative:
Probabilistic Accuracy: One-sided error

Dataset:

search with one-sided error

Query:

search with one-sided error

...
Bloom Filter: Repeated Trials

Use many hashes/filters; add each item to each filter
Bloom Filter: Repeated Trials

Use many hashes/filters; add each item to each filter
Bloom Filter: Repeated Trials

Use many hashes/filters; add each item to each filter

\[ h_1 \quad h_2 \quad h_3 \]
Bloom Filter: Repeated Trials

Use many hashes/filters; add each item to each filter

\[ h_1 \quad h_2 \quad h_3 \quad \ldots \quad h_k \]
Bloom Filter: Repeated Trials

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

If any query yields 0, item is not in the set
Bloom Filter: Repeated Trials

If all queries yield 1, item may be in the set; or we might have collided $k$ times.
Bloom Filter: Repeated Trials

Using repeated trials, even a very bad filter can still have a very low FPR!

If we have $k$ bloom filter, each with a FPR $p$, what is the likelihood that all filters return the value ‘1’ for an item we didn’t insert?
Bloom Filter: Repeated Trials

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

<table>
<thead>
<tr>
<th>$h_1$</th>
<th>$h_2$</th>
<th>$h_3$</th>
<th>...</th>
<th>$h_k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
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

- \( h_1(x) = x \mod 10 \)
- \( h_2(x) = 2x \mod 10 \)
- \( h_3(x) = (5+3x) \mod 10 \)

_\text{find}(1)_

_\text{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)!
Bloom Filter: Error Rate

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

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 $h_{\{1,2,3,\ldots,k\}}$

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

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 $h_{\{1,2,3,...,k\}}$

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

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

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 $k$  

\[
\left( 1 - \left( 1 - \frac{1}{m} \right)^{nk} \right)^k
\]

(B) small $k$
Bloom Filter: Optimal Error Rate

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

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

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

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

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

\[
\frac{d}{dk} \left( 1 - e^{-nk/m} \right)^k \approx \frac{d}{dk} \left( k \cdot \ln(1 - e^{-nk/m}) \right)
\]

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

\[ \left( 1 - e^{-\frac{n k}{m}} \right)^k \]

\[ k^* = \ln 2 \cdot 10 = 6.93 \]
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 \( m = \) \( 100 \) bits
- \( m = 100 \) bits \( n = 20 \) items \( k = \) \( 2 \) items
- \( m = 100 \) bits \( k = 2 \) items \( n = \)
Bloom Filter: Optimal Parameters

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

Optimal hash function is still $O(m)$!

$n = 250,000$ files vs $\sim 10^{15}$ nucleotides vs 260 TB

$n = 60$ billion — 130 trillion
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…
Bloom Filters: Unioning

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

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th></th>
<th>0</th>
<th>0</th>
<th></th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td>1</td>
<td>0</td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td></td>
<td>2</td>
<td>1</td>
<td></td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td></td>
<td>3</td>
<td>0</td>
<td></td>
<td>3</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td></td>
<td>4</td>
<td>0</td>
<td></td>
<td>4</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td></td>
<td>5</td>
<td>0</td>
<td></td>
<td>5</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td></td>
<td>6</td>
<td>1</td>
<td></td>
<td>6</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td></td>
<td>7</td>
<td>1</td>
<td></td>
<td>7</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td></td>
<td>8</td>
<td>1</td>
<td></td>
<td>8</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td></td>
<td>9</td>
<td>1</td>
<td></td>
<td>9</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

U = 4
# Sequence Bloom Trees

| SRA 00001 | SRA 00002 | SRA 00003 | SRA 00004 | SRA 00005 | SRA 00006 | SRA 00007 | SRA 00008 |
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…