Data Structures and Algorithms
Bloom Filters

CS 225
Brad Solomon

April 25, 2022
Center for Exascale-Enabled Scramjet Design (CEESD)

Flexible research opportunities in modeling and simulation code development

Summer Research Info: go.illinois.edu/ceesd
Project Homepage: ceesd.illinois.edu
Info Contact: gcevans@illinois.edu
Summer Dates: June 13–August 4
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
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 $N$)

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>

Table: http://doi.org/10.5334/dsj-2015-011

Estimated total volume of one array: 4.6 EB
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 resource limitations

- cache
- RAM
- disk
- network

(< 1 second, Hours - Days, Months, Years)

(Estimates are Time x 1 billion courtesy of https://gist.github.com/hellerbarde/2843375)
Memory-Constrained Data Structures

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

1)

2)
Reducing a hash table

What can we remove from a hash table?
Reducing a hash table

What can we remove from a hash table?

Take away values

\[ H(k_1) = i_1 \]

\[ m \]

\[ k_1 \rightarrow k_2 \rightarrow k_3 \]
\[ k_4 \rightarrow k_5 \]
\[ k_6 \]
\[ k_7 \rightarrow k_8 \]
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**

\[ H(k_1) = i_1 \]
Learning Objectives

Build a conceptual understanding of a bloom filter

Discuss probabilistic data structures and one-sided error

Formalize the math behind the bloom filter

Introduce extensions to the bloom filter
Bloom Filter: Insertion

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

\[ h(k) = k \mod 7 \]
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 \textit{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++

- The Pirate Bay

  - “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**
Probabilistic Accuracy: Bloom Filters

Are all outcomes possible for the bloom filter?

- True Positive
- False Negative
- False Positive
- True Negative
Probabilistic Accuracy: Bloom Filters

Item Inserted

<table>
<thead>
<tr>
<th>Bit Value = 1</th>
<th>Bit Value = 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H(z)$</td>
<td>$H(z)$</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>‘Yes’</td>
<td>‘No’</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>True Positive</td>
<td>False Negative</td>
</tr>
</tbody>
</table>

Item NOT inserted

<table>
<thead>
<tr>
<th>Bit Value = 1</th>
<th>Bit Value = 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H(z)$</td>
<td>$H(z)$</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>‘Yes’</td>
<td>‘No’</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>False Positive</td>
<td>True Negative</td>
</tr>
</tbody>
</table>
Probabilistic Accuracy: Bloom Filter

The bloom filter error is ‘one-sided’ — only false positives are possible!
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

$h_1$
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

\[
\begin{align*}
&h_1 & h_2 & h_3 & \ldots & h_k \\
0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 & 1 & 1
\end{align*}
\]
Bloom Filter: Repeated Trials

\[
h_{\{1,2,3,\ldots,k\}}(y)
\]

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></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>1</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>1</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>1</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>1</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>1</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>1</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>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Bloom Filter: Repeated Trials

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

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

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

$h_{\{1,2,3,...,k\}}(z)$
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?

$h_1$  $h_2$  $h_3$  ...  $h_k$
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

$$h_1(x) = x \% 10 \quad h_2(x) = 2x \% 10 \quad h_3(x) = (5+3x) \% 10$$

_\text{find}(1) \quad \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: ________________
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

What's the probability a specific bucket is 1 after one object is 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 one object is inserted?
Bloom Filter: Error Rate

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

What's the probability a specific bucket is 0 after one object is 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 0 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

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

The chance my bit is 1 by chance 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 $k$  
(B) small $k$

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

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

Taylor's expansion of $\ln(1 + x)$:

$$x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \ldots$$

"Mercator Series"
Bloom Filter: Error Rate

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

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

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

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

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

\[
\left(1 - e^{-\frac{nk}{m}}\right)^k = \ln 2 \cdot 10 = 6.93
\]

Figure by Ben Langmead
Bloom Filter: Optimal Parameters

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

Given any two values, we can optimize the third

\[ n = 100 \text{ items} \quad k = 3 \text{ hashes} \quad m = \frac{300}{\ln(2)} \approx 433 \text{ bits} \]

\[ m = 100 \text{ bits} \quad n = 20 \text{ items} \quad k = \ln(2) \frac{100}{20} \approx 3.47 \text{ hashes} \]

\[ m = 100 \text{ bits} \quad k = 2 \text{ items} \quad n = \ln(2) \frac{100}{2} \approx 34.6 \text{ items} \]
Bloom Filters

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

Ideal number of hash functions: \[ k^* = \ln 2 \cdot \frac{m}{n} \]

Optimal Bloom Filter does not change asymptotic scaling!

\[ m = \frac{nk}{\ln 2} \approx 1.44 \cdot nk \]
Bloom Filter: Website Caching

Sequence Bloom Trees

Imagine we have a large collection of text…

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

ATGGTTAGAATTAAACCCGG
TGCTAATAAACCUAGTGATG
CGATAGCACAGGTAGATCC
TACGTAGAGGTAGTCATTAGCC
TACGTAGAGGTCATTAGCCG
TGCTAATAAACCUAGTGATG
...

ATGGTTAGAATTAAACCCGG
TGCTAATAAACCUAGTGATG
CGATAGCACAGGTAGATCC
TACGTAGAGGTAGTCATTAGCC
TACGTAGAGGTCATTAGCCG
TGCTAATAAACCUAGTGATG
...
Bloom Filters: Unioning

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

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

\[ U = \]

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

<table>
<thead>
<tr>
<th>SRA 00001</th>
<th>SRA 00002</th>
<th>SRA 00003</th>
<th>SRA 00004</th>
<th>SRA 00005</th>
<th>SRA 00006</th>
<th>SRA 00007</th>
<th>SRA 00008</th>
</tr>
</thead>
</table>
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…