Lecture 10: Bloom Filter
Agenda

- Variance
- Properties of Variance
- Independent Random Variables
- Properties of Independent Random Variables
- An Application: Bloom Filters!
Basic Problem

Problem: Store a subset $S$ of a large set $U$.

Example. $U =$ set of 128 bit strings
$S =$ subset of strings of interest

$|U| \approx 2^{128}$
$|S| \approx 1000$

Two goals:
1. Very fast (ideally constant time) answers to queries “Is $x \in S$?” for any $x \in U$.
2. Minimal storage requirements.
Naïve Solution I – Constant Time

**Idea:** Represent $S$ as an array $A$ with $2^{128}$ entries.

$$S = \{0, 2, \ldots, K\}$$

**Membership test:** To check $x \in S$ just check whether $A[x] = 1$.

$\rightarrow$ constant time! 👍😍

**Storage:** Require storing $2^{128}$ bits, even for small $S$. 😞😢

$$A[x] = \begin{cases} 1 & \text{if } x \in S \\ 0 & \text{if } x \notin S \end{cases}$$

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>...</th>
<th>K</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Naïve Solution II – Small Storage

**Idea:** Represent $S$ as a list with $|S|$ entries.

$$S = \{0,2, \ldots, K\}$$

**Storage:** Grows with $|S|$ only 😊 😊

**Membership test:** Check $x \in S$ requires time linear in $|S|$ 😞 😞

(Can be made logarithmic by using a tree)
Hash Table

Idea: Map elements in $S$ into an array $A$ of size $m$ using a hash function $h$

Membership test: To check $x \in S$ just check whether $A[h(x)] = x$

Storage: $m$ elements (size of array)

Hash function $h: U \rightarrow [m]$
Hash Table

Idea: Map elements in $S$ into an array $A$ of size $m$ using a hash function $h$

Membership test: To check $x \in S$ just check whether $A[h(x)] = x$

Storage: $m$ elements (size of array)

Challenge 1: Ensure $h(x) \neq h(y)$ for most $x, y \in S$

Challenge 2: Ensure $m = O(|S|)$
Collisions occur when $h(x) = h(y)$ for some distinct $x, y \in S$, i.e., two elements of set map to the same location.

- Common solution: chaining – at each location (bucket) in the table, keep linked list of all elements that hash there.
Good hash functions to keep collisions low

- The hash function $h$ is good iff it
  - distributes elements uniformly across the $m$ array locations so that
  - pairs of elements are mapped independently

“Universal Hash Functions” – see CSE 332
Hashing: summary

Hash Tables

• They store the data itself
• With a good hash function, the data is well distributed in the table and lookup times are small.
• However, they need at least as much space as all the data being stored, i.e., $m = \Omega(|S|)$

In some cases, $|S|$ is huge, or not known a-priori ...

Can we do better!? 
Bloom Filters to the rescue
(Named after Burton Howard Bloom)
Bloom Filters

- Stores information about a set of elements $S \subseteq U$.
- Supports two operations:
  1. $\text{add}(x)$ - adds $x \in U$ to the set $S$
  2. $\text{contains}(x)$ – ideally: true if $x \in S$, false otherwise

Possible false positives

Combine with fallback mechanism – can distinguish false positives from true positives with extra cost
Bloom Filters – Ingredients

Basic data structure is a $k \times m$ binary array “the Bloom filter”
- $k$ rows $t_1, \ldots, t_k$, each of size $m$
- Think of each row as an $m$-bit vector

$k$ different hash functions $h_1, \ldots, h_k : U \rightarrow [m]$

<table>
<thead>
<tr>
<th></th>
<th>$t_1$</th>
<th>$t_2$</th>
<th>$t_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Bloom Filters – Three operations

• Set up Bloom filter for $S = \emptyset$

function **INITIALIZE**($k, m$)
  
  for $i = 1, \ldots, k$: do
  
  $t_i = \text{new bit vector of } m \text{ 0s}$

• Update Bloom filter for $S \leftarrow S \cup \{x\}$

function **ADD**($x$)
  
  for $i = 1, \ldots, k$: do
  
  $t_i[h_i(x)] = 1$

• Check if $x \in S$

function **CONTAINS**($x$)
  
  return $t_1[h_1(x)] = 1 \land t_2[h_2(x)] = 1 \land \cdots \land t_k[h_k(x)] = 1$
Bloom Filters - Initialization

Function `initialize(k, m)`

For $i = 1, \ldots, k$: do

$t_i =$ new bit vector of $m$ 0s

- **Number of hash functions**
- **Size of array associated to each hash function.**

for each hash function, initialize an empty bit vector of size $m$
Bloom Filters: Example

Bloom filter $t$ of length $m = 5$ that uses $k = 3$ hash functions

```plaintext
function INITIALIZE($k, m$)
    for $i = 1, \ldots, k$: do
        $t_i =$ new bit vector of $m$ 0s
```

<table>
<thead>
<tr>
<th>Index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$t_2$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$t_3$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Bloom Filters: Add

**function** `ADD(x)`

for $i = 1, \ldots, k$: do

$t_i[h_i(x)] = 1$

for each hash function $h_i$

Index into $i$-th bit-vector, at index produced by hash function and set to 1

$h_i(x) \rightarrow$ result of hash function $h_i$ on $x$
Bloom Filters: Example

Bloom filter $t$ of length $m = 5$ that uses $k = 3$ hash functions

**function** $ADD(x)$

for $i = 1, \ldots, k$:

$\begin{align*}
  t_i[h_i(x)] &= 1 \\
\end{align*}$

```add("thisisavirus.com")
$$
\begin{align*}
  h_1("thisisavirus.com") \rightarrow 2
\end{align*}
```

<table>
<thead>
<tr>
<th>Index $\rightarrow$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$t_2$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$t_3$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Bloom Filters: Example

Bloom filter t of length $m = 5$ that uses $k = 3$ hash functions

**function** $ADD(x)$

for $i = 1, \ldots, k$: do

t$_i[h_i(x)] = 1$

add("thisisavirus.com")

$h_1("thisisavirus.com") \rightarrow 2$

$h_2("thisisavirus.com") \rightarrow 1$

<table>
<thead>
<tr>
<th>Index $\rightarrow$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>t$_1$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>t$_2$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>t$_3$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Bloom Filters: Example

Bloom filter $t$ of length $m = 5$ that uses $k = 3$ hash functions

\[
\text{function } \text{ADD}(x) \\
\text{for } i = 1, \ldots, k: \text{ do} \\
\quad t_i[h_i(x)] = 1
\]

\[
\begin{array}{c|c|c|c|c|c}
\text{Index} & 0 & 1 & 2 & 3 & 4 \\
\hline
\hline
\text{t}_1 & 0 & 0 & \textbf{1} & 0 & 0 \\
\hline
\text{t}_2 & 0 & \textbf{1} & 0 & 0 & 0 \\
\hline
\text{t}_3 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

add("thisisavirus.com")

$h_1("thisisavirus.com") \rightarrow 2$

$h_2("thisisavirus.com") \rightarrow 1$

$h_3("thisisavirus.com") \rightarrow 4$
Bloom Filter: Example

Bloom filter \( t \) of length \( m = 5 \) that uses \( k = 3 \) hash functions

**function** \( \text{ADD}(x) \)

\[
\text{for } i = 1, \ldots, k: \text{ do } \\
t_i[h_i(x)] = 1
\]

**add(“thisisavirus.com”)**

\[
\begin{align*}
h_1(“thisisavirus.com”) & \rightarrow 2 \\
h_2(“thisisavirus.com”) & \rightarrow 1 \\
h_3(“thisisavirus.com”) & \rightarrow 4
\end{align*}
\]

<table>
<thead>
<tr>
<th>Index ( i )</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_1 )</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( t_2 )</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( t_3 )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Bloom Filters: Contains

```plaintext
function CONTAINS(x)
    return \( t_1[h_1(x)] == 1 \land t_2[h_2(x)] == 1 \land \cdots \land t_k[h_k(x)] == 1 \)
```

Returns True if the bit vector \( t_i \) for each hash function has bit 1 at index determined by \( h_i(x) \),

Returns False otherwise
Bloom Filters: Example

Bloom filter $t$ of length $m = 5$ that uses $k = 3$ hash functions

**Function** $\text{CONTAINS}(x)$

```
return $t_1[h_1(x)] == 1 \land t_2[h_2(x)] == 1 \land \cdots \land t_k[h_k(x)] == 1$
```

contains(“thisisavirus.com”)

<table>
<thead>
<tr>
<th>Index →</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$t_2$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$t_3$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Bloom Filters: Example

Bloom filter \( t \) of length \( m = 5 \) that uses \( k = 3 \) hash functions

**function** `contains(x)`

```
return \( t_1[h_1(x)] = 1 \land t_2[h_2(x)] = 1 \land \cdots \land t_k[h_k(x)] = 1 \)
```

contains(“thisisavirus.com”)

\( h_1(“thisisavirus.com”) \rightarrow 2 \)

True

<table>
<thead>
<tr>
<th>Index ( \rightarrow )</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_1 )</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( t_2 )</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( t_3 )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Bloom Filters: Example

Bloom filter $t$ of length $m = 5$ that uses $k = 3$ hash functions

```
function CONTAINS(x)
    return $t_1[h_1(x)] == 1 \land t_2[h_2(x)] == 1 \land \cdots \land t_k[h_k(x)] == 1$
```

contains("thisisavirus.com")

$h_1(“thisisavirus.com”) \rightarrow 2$

$h_2(“thisisavirus.com”) \rightarrow 1$

<table>
<thead>
<tr>
<th>Index $\rightarrow$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$t_2$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$t_3$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Bloom Filters: Example

Bloom filter $t$ of length $m = 5$ that uses $k = 3$ hash functions

function `CONTAINS(x)`

```
return $t_1[h_1(x)] = 1 \land t_2[h_2(x)] = 1 \land \cdots \land t_k[h_k(x)] = 1$
```

contains("thisisavirus.com")

$h_1("thisisavirus.com") \rightarrow 2$

$h_2("thisisavirus.com") \rightarrow 1$

$h_3("thisisavirus.com") \rightarrow 4$

<table>
<thead>
<tr>
<th>Index →</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$t_2$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$t_3$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Bloom Filters: Example

Bloom filter \( t \) of length \( m = 5 \) that uses \( k = 3 \) hash functions

**Function** `CONTAINS(x)`

```python
function CONTAINS(x)
    return \( t_1[h_1(x)] == 1 \wedge t_2[h_2(x)] == 1 \wedge \cdots \wedge t_k[h_k(x)] == 1 \)
```

<table>
<thead>
<tr>
<th>Index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_1 )</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( t_2 )</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( t_3 )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Since all conditions satisfied, returns **True** (correctly)

\[
\begin{align*}
\text{contains(“thisisavirus.com”)} & \quad h_1(“thisisavirus.com”) \rightarrow 2 \\
                           & \quad h_2(“thisisavirus.com”) \rightarrow 1 \\
                           & \quad h_3(“thisisavirus.com”) \rightarrow 4
\end{align*}
\]
Bloom Filters: False Positives

Bloom filter \( t \) of length \( m = 5 \) that uses \( k = 3 \) hash functions

add(“totallynotsuspicious.com”)

\[
\text{function} \ \text{ADD}(x) \\
\text{for } i = 1, \ldots, k: \text{ do} \\
t_i[h_i(x)] = 1
\]

<table>
<thead>
<tr>
<th>Index →</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_1 )</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( t_2 )</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( t_3 )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
**Bloom Filters: False Positives**

Bloom filter \( t \) of length \( m = 5 \) that uses \( k = 3 \) hash functions

```plaintext
function \( \text{ADD}(x) \)
    for \( i = 1, \ldots, k \): do
        \( t_i[h_i(x)] = 1 \)
```

add(“totallynotsuspicious.com”)

\( h_1(“totallynotsuspicious.com”) \rightarrow 1 \)

<table>
<thead>
<tr>
<th>Index ( \rightarrow )</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_1 )</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( t_2 )</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( t_3 )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Bloom Filters: False Positives

Bloom filter $t$ of length $m = 5$ that uses $k = 3$ hash functions.

**function** $\text{ADD}(x)$

for $i = 1, \ldots, k$: do

$t_i[h_i(x)] = 1$

add(“totallynotsuspicious.com”)

$h_1(“totallynotsuspicious.com”) \rightarrow 1$

$h_2(“totallynotsuspicious.com”) \rightarrow 0$

<table>
<thead>
<tr>
<th>Index $\rightarrow$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$t_2$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$t_3$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Bloom Filters: False Positives

Bloom filter $t$ of length $m = 5$ that uses $k = 3$ hash functions

function $\text{ADD}(x)$

for $i = 1, \ldots, k$: do

$t_i[h_i(x)] = 1$

add("totallynotsuspicious.com")

$h_1("totallynotsuspicious.com") \rightarrow 1$
$h_2("totallynotsuspicious.com") \rightarrow 0$
$h_3("totallynotsuspicious.com") \rightarrow 4$

<table>
<thead>
<tr>
<th>Index $\rightarrow$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$t_2$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$t_3$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Bloom Filters: False Positives

Bloom filter $t$ of length $m = 5$ that uses $k = 3$ hash functions

**function** ADD($x$)

for $i = 1, \ldots, k$: do

$t_i[h_i(x)] = 1$

---

add("totallynotsuspicious.com")

$h_1("totallynotsuspicious.com") \rightarrow 1$

$h_2("totallynotsuspicious.com") \rightarrow 0$

$h_3("totallynotsuspicious.com") \rightarrow 4$

---

<table>
<thead>
<tr>
<th>Index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$t_2$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$t_3$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Bloom Filters: False Positives

Bloom filter \( t \) of length \( m = 5 \) that uses \( k = 3 \) hash functions

```plaintext
function CONTAINS(x)
    return \( t_1[h_1(x)] = 1 \land t_2[h_2(x)] = 1 \land \ldots \land t_k[h_k(x)] = 1 \)
```

contains(“verynormalsite.com”)

<table>
<thead>
<tr>
<th>Index →</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_1 )</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( t_2 )</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( t_3 )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Bloom Filters: False Positives

Bloom filter $t$ of length $m = 5$ that uses $k = 3$ hash functions

```
function CONTAINS(x)
    return t_1[h_1(x)] == 1 ∧ t_2[h_2(x)] == 1 ∧ ... ∧ t_k[h_k(x)] == 1
```

`contains("verynormalsite.com")`

```
h_1("verynormalsite.com") → 2
```

<table>
<thead>
<tr>
<th>Index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$t_2$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$t_3$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
**Bloom Filters: False Positives**

Bloom filter $t$ of length $m = 5$ that uses $k = 3$ hash functions

```plaintext
function CONTAINS(x)
    return $t_1[h_1(x)] == 1 \land t_2[h_2(x)] == 1 \land \cdots \land t_k[h_k(x)] == 1$
```

contains(“verynormalsite.com”)

<table>
<thead>
<tr>
<th>Index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$t_2$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$t_3$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

$h_1(“verynormalsite.com”) \rightarrow 2$

$h_2(“verynormalsite.com”) \rightarrow 0$
Bloom Filters: False Positives

Bloom filter $t$ of length $m = 5$ that uses $k = 3$ hash functions

```
function CONTAINS(x)
    return $t_1[h_1(x)] == 1 \land t_2[h_2(x)] == 1 \land \cdots \land t_k[h_k(x)] == 1$
```

Contains("verynormalsite.com")

- $h_1("verynormalsite.com") \rightarrow 2$
- $h_2("verynormalsite.com") \rightarrow 0$
- $h_3("verynormalsite.com") \rightarrow 4$

<table>
<thead>
<tr>
<th>Index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$t_2$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$t_3$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Bloom Filters: False Positives

Bloom filter $t$ of length $m = 5$ that uses $k = 3$ hash functions

<table>
<thead>
<tr>
<th>Index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$t_2$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$t_3$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Since all conditions satisfied, returns True (incorrectly)
Analysis: False positive probability

**Question:** For an element \( x \in U \), what is the probability that \( \text{contains}(x) \) returns true if \( \text{add}(x) \) was never executed before?

Probability over what?! Over the choice of the \( h_1, \ldots, h_k \)

Assumptions for the analysis (somewhat stronger than for ordinary hashing):

- Each \( h_i(x) \) is uniformly distributed in \([m]\) for all \( x \) and \( i \)
- Hash function outputs for each \( h_i \) are mutually independent (not just in pairs)
- Different hash functions are independent of each other
False positive probability – Events

Assume we perform \( \text{add}(x_1), \ldots, \text{add}(x_n) \)

+ \( \text{contains}(x) \) for \( x \notin \{x_1, \ldots, x_n\} \)

Event \( E_i \) holds iff \( h_i(x) \in \{h_i(x_1), \ldots, h_i(x_n)\} \)

\[
P(\text{false positive}) = P(E_1 \cap E_2 \cap \cdots \cap E_k) = \prod_{i=1}^{k} P(E_i)
\]

\( h_1, \ldots, h_k \) independent
False positive probability – Events

Event $E_i$ holds iff $h_i(x) \in \{h_i(x_1), \ldots, h_i(x_n)\}$

Event $E_i^c$ holds iff $h_i(x) \neq h_i(x_1)$ and ... and $h_i(x) \neq h_i(x_n)$

$$P(E_i^c) = \sum_{z=1}^{m} P(h_i(x) = z) \cdot P(E_i^c | h_i(x) = z)$$
False positive probability – Events

\[ P(E_i^c \mid h_i(x) = z) = P(h_i(x_1) \neq z, \ldots, h_i(x_n) \neq z \mid h_i(x) = z) \]

= \[ P(h_i(x_1) \neq z, \ldots, h_i(x_n) \neq z) \]

= \[ \prod_{j=1}^{n} P(h_i(x_j) \neq z) \]

= \[ \prod_{j=1}^{n} \left(1 - \frac{1}{m}\right) = \left(1 - \frac{1}{m}\right)^n \]

\[ P(E_i^c) = \sum_{z=1}^{m} P(h_i(x) = z) \cdot P(E_i^c \mid h_i(x) = z) = \left(1 - \frac{1}{m}\right)^n \]
False positive probability – Events

Event $E_i$ holds iff $h_i(x) \in \{h_i(x_1), \ldots, h_i(x_n)\}$

Event $E_i^c$ holds iff $h_i(x) \neq h_i(x_1)$ and $\ldots$ and $h_i(x) \neq h_i(x_n)$

$$P(E_i^c) = \left(1 - \frac{1}{m}\right)^n$$

$$FPR = \prod_{i=1}^{k} \left(1 - P(E_i^c)\right) = \left(1 - \left(1 - \frac{1}{m}\right)^n\right)^k$$
False Positivity Rate – Example

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

e.g., \( n = 5,000,000 \)
\( k = 30 \)
\( m = 2,500,000 \)

\[ \text{FPR} = 1.28\% \]
Comparison with Hash Tables - **Space**

- Google storing 5 million URLs, each URL 40 bytes.
- Bloom filter with \( k = 30 \) and \( m = 2,500,000 \)

<table>
<thead>
<tr>
<th>Hash Table (optimistic)</th>
<th>Bloom Filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 5,000,000 \times 40B = 200MB )</td>
<td>( 2,500,000 \times 30 = 75,000,000 \text{ bits} )</td>
</tr>
</tbody>
</table>
Time

- Say avg user visits 102,000 URLs in a year, of which 2,000 are malicious.
- 0.5 seconds to do lookup in the database, 1ms for lookup in Bloom filter.
- Suppose the false positive rate is 3%

\[
1\text{ms} + \frac{100000 \times 0.03 \times 500\text{ms}}{102000} + 2000 \times 500\text{ms} \approx 25.51\text{ms}
\]
Bloom Filters typical of….

... randomized algorithms and randomized data structures.

- Simple
- Fast
- Efficient
- Elegant
- Useful!