**Definition.** The variance of a (discrete) RV $X$ is

$$\text{Var}(X) = \mathbb{E}[(X - \mathbb{E}[X])^2] = \sum_x p_x(x) \cdot (x - \mathbb{E}[X])^2$$

**Theorem.** For any $a, b \in \mathbb{R}$, $\text{Var}(a \cdot X + b) = a^2 \cdot \text{Var}(X)$

**Theorem.** $\text{Var}(X) = \mathbb{E}[X^2] - \mathbb{E}[X]^2$
Review Important Facts about Independent Random Variables

**Theorem.** If $X, Y$ independent, $\mathbb{E}[X \cdot Y] = \mathbb{E}[X] \cdot \mathbb{E}[Y]$.

**Theorem.** If $X, Y$ independent, $\text{Var}(X + Y) = \text{Var}(X) + \text{Var}(Y)$.

**Corollary.** If $X_1, X_2, \ldots, X_n$ mutually independent,

$$\text{Var} \left( \sum_{i=1}^{n} X_i \right) = \sum_{i} \text{Var}(X_i)$$
Agenda

• Bloom Filters Example & Analysis
• Zoo of Discrete RVs
  – Uniform Random Variables
  – Bernoulli Random Variables
  – Binomial Random Variables
  – Applications
Basic Problem

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

Example. $U = \text{set of 128 bit strings}$
\[ |U| \approx 2^{128} \]
\[ S = \text{subset of strings of interest} \]
\[ |S| \approx 1000 \]

Two goals:

1. Very fast (ideally constant time) answers to queries “Is $x \in S$?”
   for any $x \in U$.
2. Minimal storage requirements.
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 $h_1, \ldots, h_k : U \rightarrow [m]$
Bloom Filters – Three operations

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

  function \textsc{initialize}(k, m)
  \hspace{1em} \text{for } i = 1, \ldots, k: \text{ do}
  \hspace{2em} t_i = \text{new bit vector of } m \text{ 0s}

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

  function \textsc{add}(x)
  \hspace{1em} \text{for } i = 1, \ldots, k: \text{ do}
  \hspace{2em} t_i[h_i(x)] = 1

• Check if $x \in S$

  function \textsc{contains}(x)
  \hspace{1em} \text{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** \( \text{INITIALIZE}(k, m) \)

\[
\text{for } i = 1, \ldots, k: \text{ do }
\]

\[
t_i = \text{new bit vector of } m \text{ 0s}
\]

- **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

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

<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: 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

add("thisisavirus.com")

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

**function** ADD($x$)

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

```plaintext
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

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

for \( i = 1, \ldots, k \): do

\( t_i[h_i(x)] = 1 \)

**add(“thisisavirus.com”)**

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

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

\( h_3(“thisisavirus.com”) \rightarrow 4 \)

<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>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")

$\begin{align*}
h_1(\text{"thisisavirus.com"}) &\rightarrow 2 \\
h_2(\text{"thisisavirus.com"}) &\rightarrow 1 \\
h_3(\text{"thisisavirus.com"}) &\rightarrow 4
\end{align*}$

<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>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

**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

```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("thisisavirus.com")
```

<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

```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(“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

\[
\text{function } \text{CONTAINS}(x) \\
\quad \text{return } t_1[h_1(x)] = 1 \land t_2[h_2(x)] = 1 \land \cdots \land t_k[h_k(x)] = 1
\]

contains("thisisavirus.com")

\[
\begin{align*}
\text{Index} & \quad 0 & \quad 1 & \quad 2 & \quad 3 & \quad 4 \\
\rightarrow & \\
t_1 & 0 & 0 & 1 & 0 & 0 \\
t_2 & 0 & 1 & 0 & 0 & 0 \\
t_3 & 0 & 0 & 0 & 0 & 1
\end{align*}
\]

\( h_1(\text{"thisisavirus.com"}) \rightarrow 2 \)
\( h_2(\text{"thisisavirus.com"}) \rightarrow 1 \)
Bloom Filters: Example

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

```python
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("thisisavirus.com")

$h_1(\text{"thisisavirus.com"}) \rightarrow 2$
$h_2(\text{"thisisavirus.com"}) \rightarrow 1$
$h_3(\text{"thisisavirus.com"}) \rightarrow 4$

<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>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**

```python
def 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``)

- $h_1(``thisisavirus.com``) → 2$
- $h_2(``thisisavirus.com``) → 1$
- $h_3(``thisisavirus.com``) → 4$

Since all conditions satisfied, returns True (correctly)

<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

add("totallynotsuspicious.com")

function $\text{ADD}(x)$

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

$t_i[h_i(x)] = 1$

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

$t_i[h_i(x)] = 1$

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

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

<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>

Function $\text{ADD}(x)$

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

$t_i[h_i(x)] = 1$

add(“totallynotsuspicious.com”)

- $h_1(“totallynotsuspicious.com”) \rightarrow 1$
- $h_2(“totallynotsuspicious.com”) \rightarrow 0$
- $h_3(“totallynotsuspicious.com”) \rightarrow 4$
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

```python
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 $\rightarrow$</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 \( \text{CONTAINS}(x) \)
    return \( t_1[h_1(x)] = 1 \land t_2[h_2(x)] = 1 \land \cdots \land t_k[h_k(x)] = 1 \)

contains(“verynormalsite.com”)

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

<table>
<thead>
<tr>
<th>Index ( t )</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

\[
\text{function} \ \text{CONTAINS}(x) \\
\quad \text{return} \ t_1[h_1(x)] == 1 \land t_2[h_2(x)] == 1 \land \cdots \land t_k[h_k(x)] == 1
\]

contains(“verynormalsite.com”) 

\[
\begin{align*}
h_1(“verynormalsite.com”) & \rightarrow 2 \\
h_2(“verynormalsite.com”) & \rightarrow 0
\end{align*}
\]

<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>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>

contains(“verynormalsite.com”)

$h_1(“verynormalsite.com”) → 2$
$h_2(“verynormalsite.com”) → 0$
$h_3(“verynormalsite.com”) → 4$

function CONTAINS(x)  
return $t_1[h_1(x)] == 1 ∧ t_2[h_2(x)] == 1 ∧ \cdots ∧ t_k[h_k(x)] == 1$
Bloom Filters: False Positives

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

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

<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)

contains(“verynormalsite.com”)

\( h_1(“verynormalsite.com”) \rightarrow 2 \)
\( h_2(“verynormalsite.com”) \rightarrow 0 \)
\( h_3(“verynormalsite.com”) \rightarrow 4 \)
**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, so \( E_i \)'s are 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 $h_i(x) \neq h_i(x_n)$

$$P(E_i^c) = \sum_{z=1}^{m} P(h_i(x) = z) \cdot P(E_i^c | h_i(x) = z)$$

LTP
False positive probability – Events

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

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

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

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

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

\( P(E_i^c) = \sum_{z=1}^{m} P(h_i(x) = z) \cdot P(E_i^c | h_i(x) = z) = \left(1 - \frac{1}{m}\right)^n \)
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 ... and $h_i(x) \neq h_i(x_n)$

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

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

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

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

\[ FPR = 1.28\% \]
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</th>
<th>Bloom Filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>(optimistic)</td>
<td></td>
</tr>
<tr>
<td>$5,000,000 \times 40B = 200$MB</td>
<td>$2,500,000 \times 30 = 75,000,000$ bits</td>
</tr>
<tr>
<td></td>
<td>$&lt; 10$ MB</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%**

\[
\text{false positives} = 100000 \times 0.03 \times 500\text{ms} + 2000 \times 500\text{ms}\]

\[
\frac{\text{1ms} + \frac{100000 \times 0.03 \times 500\text{ms} + 2000 \times 500\text{ms}}{102000}}{\text{malicious URLs}} \approx 25.51\text{ms}
\]

- vs. **500ms** all the time.
Bloom Filters typical of....

... randomized algorithms and randomized data structures.

- Simple
- Fast
- Efficient
- Elegant
- Useful!
Brain Break
Motivation for “Named” Random Variables

Random Variables that show up all over the place.
   – Easily solve a problem by recognizing it’s a special case of one of these random variables.

Each RV introduced today will show:
   – A general situation it models
   – Its name and parameters
   – Its PMF, Expectation, and Variance
   – Example scenarios you can use it
### Welcome to the Zoo! (Preview)

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Probability Mass Function</th>
<th>Expected Value</th>
<th>Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X \sim \text{Unif}(a,b)$</td>
<td>$P(X = k) = \frac{1}{b - a + 1}$</td>
<td>$\mathbb{E}[X] = \frac{a + b}{2}$</td>
<td>$\text{Var}(X) = \frac{(b - a)(b - a + 2)}{12}$</td>
</tr>
<tr>
<td>$X \sim \text{Ber}(p)$</td>
<td>$P(X = 1) = p, P(X = 0) = 1 - p$</td>
<td>$\mathbb{E}[X] = p$</td>
<td>$\text{Var}(X) = p(1 - p)$</td>
</tr>
<tr>
<td>$X \sim \text{Bin}(n,p)$</td>
<td>$P(X = k) = \binom{n}{k} p^k (1 - p)^{n-k}$</td>
<td>$\mathbb{E}[X] = np$</td>
<td>$\text{Var}(X) = np(1 - p)$</td>
</tr>
<tr>
<td>$X \sim \text{Geo}(p)$</td>
<td>$P(X = k) = (1 - p)^{k-1}p$</td>
<td>$\mathbb{E}[X] = \frac{1}{p}$</td>
<td>$\text{Var}(X) = \frac{1 - p}{p^2}$</td>
</tr>
<tr>
<td>$X \sim \text{NegBin}(r,p)$</td>
<td>$P(X = k) = \binom{k-1}{r-1} p^r (1 - p)^{k-r}$</td>
<td>$\mathbb{E}[X] = \frac{r}{p}$</td>
<td>$\text{Var}(X) = \frac{r(1 - p)}{p^2}$</td>
</tr>
<tr>
<td>$X \sim \text{HypGeo}(N,K,n)$</td>
<td></td>
<td>$\mathbb{E}[X] = \frac{K}{N}$</td>
<td>$\text{Var}(X) = n \frac{K(N-K)(N-n)}{N^2(N-1)}$</td>
</tr>
</tbody>
</table>
Agenda

• Bloom Filters Example & Analysis
• Zoo of Discrete RVs, Part I
  – Uniform Random Variables
  – Bernoulli Random Variables
  – Binomial Random Variables
  – Applications
Discrete Uniform Random Variables

A discrete random variable \( X \) **equally likely** to take any (integer) value between integers \( a \) and \( b \) (inclusive), is **uniform**.

**Notation:**

**PMF:**

**Expectation:**

**Variance:**

**Example:** value shown on one roll of a fair die
Discrete Uniform Random Variables

A discrete random variable $X$ equally likely to take any (integer) value between integers $a$ and $b$ (inclusive), is uniform.

Notation: $X \sim \text{Unif}(a, b)$

PMF: $P(X = i) = \frac{1}{b - a + 1}$

Expectation: $\mathbb{E}[X] = \frac{a + b}{2}$

Variance: $\text{Var}(X) = \frac{(b-a)(b-a+2)}{12}$

Example: value shown on one roll of a fair die is $\text{Unif}(1, 6)$:
- $P(X = i) = 1/6$
- $\mathbb{E}[X] = 7/2$
- $\text{Var}(X) = 35/12$
Agenda

- Bloom Filters Example & Analysis
- Zoo of Discrete RVs, Part I
  - Uniform Random Variables
  - Bernoulli Random Variables
  - Binomial Random Variables
  - Applications
Bernoulli Random Variables

A random variable $X$ that takes value 1 ("Success") with probability $p$, and 0 ("Failure") otherwise. $X$ is called a Bernoulli random variable.

Notation: $X \sim \text{Ber}(p)$

PMF: $P(X = 1) = p$, $P(X = 0) = 1 - p$

Expectation: $E(X) = p$

Variance: $\text{Var}(X) = p(1 - p)$

Poll:

[Link: pollev.com/paulbeame028]

Mean | Variance
---|---
A. $p$, $p$
B. $p$, $1 - p$
C. $p$, $p(1 - p)$
D. $p$, $p^2$

Clue: balanced
Bernoulli Random Variables

A random variable $X$ that takes value 1 ("Success") with probability $p$, and 0 ("Failure") otherwise. $X$ is called a Bernoulli random variable.

**Notation:** $X \sim \text{Ber}(p)$

**PMF:** $P(X = 1) = p$, $P(X = 0) = 1 - p$

**Expectation:** $\mathbb{E}[X] = p$  
**Note:** $\mathbb{E}[X^2] = p$

**Variance:** $\text{Var}(X) = \mathbb{E}[X^2] - \mathbb{E}[X]^2 = p - p^2 = p(1 - p)$

**Examples:**
- Coin flip
- Randomly guessing on a MC test question
- A server in a cluster fails
- Any indicator RV
Agenda

• Bloom Filters Example & Analysis
• Zoo of Discrete RVs, Part I
  – Uniform Random Variables
  – Bernoulli Random Variables
  – Binomial Random Variables
  – Applications
Binomial Random Variables

A discrete random variable $X$ that is the number of successes in $n$ independent random variables $Y_i \sim \text{Ber}(p)$. $X$ is a Binomial random variable where $X = \sum_{i=1}^{n} Y_i$

Examples:
- # of heads in $n$ coin flips
- # of 1s in a randomly generated $n$ bit string
- # of servers that fail in a cluster of $n$ computers
- # of bit errors in file written to disk
- # of elements in a bucket of a large hash table

Poll:
pollev.com/paulbeame028

$P(X = k)$
A. $p^k(1-p)^{n-k}$
B. $np$
C. $\binom{n}{k} p^k (1-p)^{n-k}$
D. $\binom{n}{n-k} p^k (1-p)^{n-k}$
Binomial Random Variables

A discrete random variable $X$ that is the number of successes in $n$ independent random variables $Y_i \sim Ber(p)$. $X$ is a Binomial random variable where $X = \sum_{i=1}^{n} Y_i$

Notation: $X \sim Bin(n, p)$

PMF: $P(X = k) = \binom{n}{k}p^k(1 - p)^{n-k}$

Expectation:

Variance:
Binomial Random Variables

A discrete random variable $X$ that is the number of successes in $n$ independent random variables $Y_i \sim \text{Ber}(p)$. $X$ is a \textbf{Binomial random variable} where $X = \sum_{i=1}^{n} Y_i$

\textbf{Notation:} $X \sim \text{Bin}(n, p)$

\textbf{PMF:} $P(X = k) = \binom{n}{k} p^k (1 - p)^{n-k}$

\textbf{Expectation:} $\mathbb{E}[X] = np$

\textbf{Variance:} $\text{Var}(X) = np(1 - p)$
Mean, Variance of the Binomial

If $Y_1, Y_2, ..., Y_n \sim \text{Ber}(p)$ and independent (i.i.d.), then $X = \sum_{i=1}^{n} Y_i$, $X \sim \text{Bin}(n, p)$

Claim $\mathbb{E}[X] = np$

$$\mathbb{E}[X] = \mathbb{E}\left[ \sum_{i=1}^{n} Y_i \right] = \sum_{i=1}^{n} \mathbb{E}[Y_i] = n\mathbb{E}[Y_1] = np$$

Claim $\text{Var}(X) = np(1 - p)$

$$\text{Var}(X) = \text{Var}\left( \sum_{i=1}^{n} Y_i \right) = \sum_{i=1}^{n} \text{Var}(Y_i) = n\text{Var}(Y_1) = np(1 - p)$$
Binomial PMFs

PMF for $X \sim \text{Bin}(10,0.5)$

PMF for $X \sim \text{Bin}(10,0.25)$
Binomial PMFs

PMF for $X \sim \text{Bin}(30, 0.5)$

PMF for $X \sim \text{Bin}(30, 0.1)$
Example

Sending a binary message of length 1024 bits over a network with probability 0.999 of correctly sending each bit in the message without corruption (independent of other bits).

Let $X$ be the number of corrupted bits.

What is $\mathbb{E}[X]$?

Poll:

pollev.com/paulbeame028

- a. 1022.99
- b. 1.024
- c. 1.02298
- d. 1
- e. Not enough information to compute