Balls and Bins (Advanced)

K. Subramani[1]

[1] Lane Department of Computer Science and Electrical Engineering
West Virginia University

6 March, 2012

Outline

Outline

Outline

Recap

Recap

Main issues

Recap

Main issues

The experiment of throwing m balls into n bins,

Recap

Main issues

The experiment of throwing m balls into n bins, each bin being chosen independently and uniformly at random.

Recap

Main issues

The experiment of throwing m balls into n bins, each bin being chosen independently and uniformly at random.Several questions regarding the above random process were examined,

Recap

Main issues

The experiment of throwing m balls into n bins, each bin being chosen independently and uniformly at random.Several questions regarding the above random process were examined, such as expected maximum load,

Recap

Main issues

The experiment of throwing m balls into n bins, each bin being chosen independently and uniformly at random.Several questions regarding the above random process were examined, such as expected maximum load, expected number of balls in a bin,

Recap

Main issues

The experiment of throwing m balls into n bins, each bin being chosen independently and uniformly at random.Several questions regarding the above random process were examined, such as expected maximum load, expected number of balls in a bin, expected number of empty bins, and

Recap

Main issues

The experiment of throwing m balls into n bins, each bin being chosen independently and uniformly at random.Several questions regarding the above random process were examined, such as expected maximum load, expected number of balls in a bin, expected number of empty bins, and expected number of bins with r balls.

Recap

Main issues

The experiment of throwing m balls into n bins, each bin being chosen independently and uniformly at random.Several questions regarding the above random process were examined, such as expected maximum load, expected number of balls in a bin, expected number of empty bins, and expected number of bins with r balls. We also examined the Poisson random variable and its applications to Balls and Bins questions.

The Poisson Approximation

Main Issues

The Poisson Approximation

Main Issues

- Is bin emptiness events independent?

The Poisson Approximation

Main Issues

- Is bin emptiness events independent?
- We know that if m balls are thrown uniformly and independently into n bins, the distribution is approximately Poisson with mean $\frac{m}{n}$.

The Poisson Approximation

Main Issues

- Is bin emptiness events independent?
- We know that if m balls are thrown uniformly and independently into n bins, the distribution is approximately Poisson with mean $\frac{m}{n}$.
- We wish to approximate the load at each bin with independent Poisson random variables.

The Poisson Approximation

Main Issues

- Is bin emptiness events independent?
- We know that if m balls are thrown uniformly and independently into n bins, the distribution is approximately Poisson with mean $\frac{m}{n}$.
- We wish to approximate the load at each bin with independent Poisson random variables.
- We will show that this can be achieved by provable bounds.

The Poisson Approximation

Main Issues

- Is bin emptiness events independent?
- We know that if m balls are thrown uniformly and independently into n bins, the distribution is approximately Poisson with mean $\frac{m}{n}$.
- We wish to approximate the load at each bin with independent Poisson random variables.
- We will show that this can be achieved by provable bounds.

Note

There is a difference between throwing m balls randomly and assigning each bin a number of balls that is Poisson distributed with mean $\frac{m}{n}$.

The Poisson Approximation

Main Issues

- Is bin emptiness events independent?
- We know that if m balls are thrown uniformly and independently into n bins, the distribution is approximately Poisson with mean $\frac{m}{n}$.
- We wish to approximate the load at each bin with independent Poisson random variables.
- We will show that this can be achieved by provable bounds.

Note

There is a difference between throwing m balls randomly and assigning each bin a number of balls that is Poisson distributed with mean $\frac{m}{n}$. However, if you use Poisson distribution and end with m balls, the distributions are identical!

Outline

Theorem I

Theorem I

Theorem

Let $X_i^{(m)}$, $1 \leq i \leq n$ be the number of balls in the i^{th} bin.

Theorem I

Theorem

Let $X_i^{(m)}$, $1 \leq i \leq n$ be the number of balls in the i^{th} bin. Let $Y_i^{(m)}$, $1 \leq i \leq n$ denote independent Poisson random variables with mean $\frac{m}{n}$.

Theorem I

Theorem

Let $X_i^{(m)}$, $1 \leq i \leq n$ be the number of balls in the i^{th} bin. Let $Y_i^{(m)}$, $1 \leq i \leq n$ denote independent Poisson random variables with mean $\frac{m}{n}$.
The distribution of $(Y_1^{(m)}, \ldots, Y_n^{(m)})$ conditioned on $\sum_i Y_i^{(m)} = k$ is the same as $(X_1^{(k)}, \ldots, X_n^{(k)})$.

Theorem II

Theorem II

Theorem

Let $f(x_1, x_2, \ldots, x_n)$ denote a nonnegative function.

Theorem II

Theorem

Let $f(x_1, x_2, \ldots, x_n)$ *denote a nonnegative function. Then,*

Theorem II

Theorem

Let $f(x_1, x_2, \ldots, x_n)$ denote a nonnegative function. Then,

$$\mathbf{E}[f(X_1^{(m)}, \ldots, X_n^{(m)})] \leq e \cdot \sqrt{m} \cdot \mathbf{E}[f(Y_1^{(m)}, \ldots, Y_n^{(m)})]$$

Theorem II

Theorem

Let $f(x_1, x_2, \ldots, x_n)$ denote a nonnegative function. Then,

$$\mathbf{E}[f(X_1^{(m)}, \ldots, X_n^{(m)})] \leq e \cdot \sqrt{m} \cdot \mathbf{E}[f(Y_1^{(m)}, \ldots, Y_n^{(m)})]$$

Corollary

Any event that takes place with probability p in the Poisson case, takes place with probability at most

Theorem II

Theorem

Let $f(x_1, x_2, \ldots, x_n)$ denote a nonnegative function. Then,

$$\mathbf{E}[f(X_1^{(m)}, \ldots, X_n^{(m)})] \leq e \cdot \sqrt{m} \cdot \mathbf{E}[f(Y_1^{(m)}, \ldots, Y_n^{(m)})]$$

Corollary

Any event that takes place with probability p in the Poisson case, takes place with probability at most $p \cdot e \cdot \sqrt{m}$ in the exact case.

Theorem III

Theorem III

Theorem

Let $f(x_1, x_2, \ldots, x_n)$ denote a nonnegative function,

Theorem III

Theorem

Let $f(x_1, x_2, \ldots, x_n)$ *denote a nonnegative function, such that* $\mathbf{E}[f(X_1^{(m)}, \ldots, X_n^{(m)})]$ *is either monotonically increasing or monotonically decreasing in* m*.*

Theorem III

Theorem

Let $f(x_1, x_2, \ldots, x_n)$ *denote a nonnegative function, such that* $\mathbf{E}[f(X_1^{(m)}, \ldots, X_n^{(m)})]$ *is either monotonically increasing or monotonically decreasing in m. Then,*

Theorem III

Theorem

Let $f(x_1, x_2, \ldots, x_n)$ denote a nonnegative function, such that $\mathbf{E}[f(X_1^{(m)}, \ldots, X_n^{(m)})]$ is either monotonically increasing or monotonically decreasing in m. Then,

$$\mathbf{E}[f(X_1^{(m)}, \ldots, X_n^{(m)})] \leq 2 \cdot \mathbf{E}[f(Y_1^{(m)}, \ldots, Y_n^{(m)})]$$

Theorem III

Theorem

Let $f(x_1, x_2, \ldots, x_n)$ denote a nonnegative function, such that $\mathbf{E}[f(X_1^{(m)}, \ldots, X_n^{(m)})]$ is either monotonically increasing or monotonically decreasing in m. Then,

$$\mathbf{E}[f(X_1^{(m)}, \ldots, X_n^{(m)})] \leq 2 \cdot \mathbf{E}[f(Y_1^{(m)}, \ldots, Y_n^{(m)})]$$

Corollary

Let Δ be an event whose probability is either monotonically increasing or decreasing in the number of balls.

Theorem III

Theorem

Let $f(x_1, x_2, \ldots, x_n)$ *denote a nonnegative function, such that* $\mathbf{E}[f(X_1^{(m)}, \ldots, X_n^{(m)})]$ *is either monotonically increasing or monotonically decreasing in m. Then,*

$$\mathbf{E}[f(X_1^{(m)}, \ldots, X_n^{(m)})] \leq 2 \cdot \mathbf{E}[f(Y_1^{(m)}, \ldots, Y_n^{(m)})]$$

Corollary

Let Δ *be an event whose probability is either monotonically increasing or decreasing in the number of balls. If* Δ *has probability p in the Poisson case, then it has probability at most*

Theorem III

Theorem

Let $f(x_1, x_2, \ldots, x_n)$ denote a nonnegative function, such that $\mathbf{E}[f(X_1^{(m)}, \ldots, X_n^{(m)})]$ is either monotonically increasing or monotonically decreasing in m. Then,

$$\mathbf{E}[f(X_1^{(m)}, \ldots, X_n^{(m)})] \leq 2 \cdot \mathbf{E}[f(Y_1^{(m)}, \ldots, Y_n^{(m)})]$$

Corollary

Let Δ be an event whose probability is either monotonically increasing or decreasing in the number of balls. If Δ has probability p in the Poisson case, then it has probability at most $2 \cdot p$ in the exact case.

Theorem III

Theorem

Let $f(x_1, x_2, \ldots, x_n)$ denote a nonnegative function, such that $\mathbf{E}[f(X_1^{(m)}, \ldots, X_n^{(m)})]$ is either monotonically increasing or monotonically decreasing in m. Then,

$$\mathbf{E}[f(X_1^{(m)}, \ldots, X_n^{(m)})] \leq 2 \cdot \mathbf{E}[f(Y_1^{(m)}, \ldots, Y_n^{(m)})]$$

Corollary

Let Δ be an event whose probability is either monotonically increasing or decreasing in the number of balls. If Δ has probability p in the Poisson case, then it has probability at most $2 \cdot p$ in the exact case.

Lemma

When n balls are thrown independently into n bins, the maximum load is at least $\frac{\ln n}{\ln \ln n}$ with probability at least $(1 - \frac{1}{n})$, for sufficiently large n.

Outline

Chain Hashing

Main Issues

Chain Hashing

Main Issues

- The password checking problem.

Chain Hashing

Main Issues

- The password checking problem.
- The Hashing Approach and Hash functions.

Chain Hashing

Main Issues

- The password checking problem.
- The Hashing Approach and Hash functions. $f : U \to [0, n-1]$.

Chain Hashing

Main Issues

- The password checking problem.
- The Hashing Approach and Hash functions. $f : U \rightarrow [0, n-1]$.
- Chain Hashing.

Chain Hashing

Main Issues

- The password checking problem.
- The Hashing Approach and Hash functions. $f : U \rightarrow [0, n-1]$.
- Chain Hashing.
- Assumption: Hash function maps words into bins in random fashion.

Chain Hashing

Main Issues

- The password checking problem.
- The Hashing Approach and Hash functions. $f : U \rightarrow [0, n-1]$.
- Chain Hashing.
- Assumption: Hash function maps words into bins in random fashion.
- For each $x \in U$, the probability that $f(x) = j$ is $\frac{1}{n}$,

Chain Hashing

Main Issues

- The password checking problem.
- The Hashing Approach and Hash functions. $f : U \rightarrow [0, n-1]$.
- Chain Hashing.
- Assumption: Hash function maps words into bins in random fashion.
- For each $x \in U$, the probability that $f(x) = j$ is $\frac{1}{n}$, and

Chain Hashing

Main Issues

- The password checking problem.
- The Hashing Approach and Hash functions. $f : U \rightarrow [0, n-1]$.
- Chain Hashing.
- Assumption: Hash function maps words into bins in random fashion.
- For each $x \in U$, the probability that $f(x) = j$ is $\frac{1}{n}$, and the values of $f(x)$ for each x are independent of each other.

Chain Hashing

Main Issues

- The password checking problem.
- The Hashing Approach and Hash functions. $f : U \rightarrow [0, n-1]$.
- Chain Hashing.
- Assumption: Hash function maps words into bins in random fashion.
- For each $x \in U$, the probability that $f(x) = j$ is $\frac{1}{n}$, and the values of $f(x)$ for each x are independent of each other.
- Search approach.

Chain Hashing

Main Issues

- The password checking problem.
- The Hashing Approach and Hash functions. $f : U \rightarrow [0, n-1]$.
- Chain Hashing.
- Assumption: Hash function maps words into bins in random fashion.
- For each $x \in U$, the probability that $f(x) = j$ is $\frac{1}{n}$, and the values of $f(x)$ for each x are independent of each other.
- Search approach.
- If the word is not in dictionary, expected time is $\frac{m}{n}$,

Chain Hashing

Main Issues

- The password checking problem.
- The Hashing Approach and Hash functions. $f : U \rightarrow [0, n-1]$.
- Chain Hashing.
- Assumption: Hash function maps words into bins in random fashion.
- For each $x \in U$, the probability that $f(x) = j$ is $\frac{1}{n}$, and the values of $f(x)$ for each x are independent of each other.
- Search approach.
- If the word is not in dictionary, expected time is $\frac{m}{n}$, otherwise, expected time is $1 + \frac{m-1}{n}$.

Chain Hashing

Main Issues

- The password checking problem.
- The Hashing Approach and Hash functions. $f : U \rightarrow [0, n-1]$.
- Chain Hashing.
- Assumption: Hash function maps words into bins in random fashion.
- For each $x \in U$, the probability that $f(x) = j$ is $\frac{1}{n}$, and the values of $f(x)$ for each x are independent of each other.
- Search approach.
- If the word is not in dictionary, expected time is $\frac{m}{n}$, otherwise, expected time is $1 + \frac{m-1}{n}$.
- Choosing $n = m$, gives constant expected search time.

Chain Hashing

Main Issues

- The password checking problem.
- The Hashing Approach and Hash functions. $f : U \to [0, n-1]$.
- Chain Hashing.
- Assumption: Hash function maps words into bins in random fashion.
- For each $x \in U$, the probability that $f(x) = j$ is $\frac{1}{n}$, and the values of $f(x)$ for each x are independent of each other.
- Search approach.
- If the word is not in dictionary, expected time is $\frac{m}{n}$, otherwise, expected time is $1 + \frac{m-1}{n}$.
- Choosing $n = m$, gives constant expected search time.
- When $n = m$, the maximum load is $\Theta(\frac{\ln n}{\ln \ln n})$, w.h.p.;

Chain Hashing

Main Issues

- The password checking problem.
- The Hashing Approach and Hash functions. $f : U \rightarrow [0, n-1]$.
- Chain Hashing.
- Assumption: Hash function maps words into bins in random fashion.
- For each $x \in U$, the probability that $f(x) = j$ is $\frac{1}{n}$, and the values of $f(x)$ for each x are independent of each other.
- Search approach.
- If the word is not in dictionary, expected time is $\frac{m}{n}$, otherwise, expected time is $1 + \frac{m-1}{n}$.
- Choosing $n = m$, gives constant expected search time.
- When $n = m$, the maximum load is $\Theta(\frac{\ln n}{\ln \ln n})$, w.h.p.; faster than binary search.

Chain Hashing

Main Issues

- The password checking problem.
- The Hashing Approach and Hash functions. $f : U \rightarrow [0, n-1]$.
- Chain Hashing.
- Assumption: Hash function maps words into bins in random fashion.
- For each $x \in U$, the probability that $f(x) = j$ is $\frac{1}{n}$, and the values of $f(x)$ for each x are independent of each other.
- Search approach.
- If the word is not in dictionary, expected time is $\frac{m}{n}$, otherwise, expected time is $1 + \frac{m-1}{n}$.
- Choosing $n = m$, gives constant expected search time.
- When $n = m$, the maximum load is $\Theta(\frac{\ln n}{\ln \ln n})$, w.h.p.; faster than binary search.
- Wasted space.

Outline

Bit String Hashing

Bit String Hashing

Main Issues

Bit String Hashing

Main Issues

- Goal is to save space instead of time.

Bit String Hashing

Main Issues

- Goal is to save space instead of time.
- Assume that passwords are restricted to 64 bits and that a hash function maps words into 32 bits.

Bit String Hashing

Main Issues

- Goal is to save space instead of time.
- Assume that passwords are restricted to 64 bits and that a hash function maps words into 32 bits.
- Fingerprinting and false positives.

Bit String Hashing

Main Issues

- Goal is to save space instead of time.
- Assume that passwords are restricted to 64 bits and that a hash function maps words into 32 bits.
- Fingerprinting and false positives.
- The approximate set membership problem:

Bit String Hashing

Main Issues

- Goal is to save space instead of time.
- Assume that passwords are restricted to 64 bits and that a hash function maps words into 32 bits.
- Fingerprinting and false positives.
- The approximate set membership problem:
  Given a set $S = \{s_1, s_2, \ldots, s_m\}$ of m elements from a large universe U,

Bit String Hashing

Main Issues

- Goal is to save space instead of time.
- Assume that passwords are restricted to 64 bits and that a hash function maps words into 32 bits.
- Fingerprinting and false positives.
- The approximate set membership problem:
  Given a set $S = \{s_1, s_2, \ldots, s_m\}$ of m elements from a large universe U, we would like to represent elements in such a way,

Bit String Hashing

Main Issues

- Goal is to save space instead of time.
- Assume that passwords are restricted to 64 bits and that a hash function maps words into 32 bits.
- Fingerprinting and false positives.
- The approximate set membership problem:
  Given a set $S = \{s_1, s_2, \ldots, s_m\}$ of m elements from a large universe U, we would like to represent elements in such a way, as to efficiently answer queries of the form "Is $x \in S$"?

Bit String Hashing

Main Issues

- Goal is to save space instead of time.
- Assume that passwords are restricted to 64 bits and that a hash function maps words into 32 bits.
- Fingerprinting and false positives.
- The approximate set membership problem:
  Given a set $S = \{s_1, s_2, \ldots, s_m\}$ of m elements from a large universe U, we would like to represent elements in such a way, as to efficiently answer queries of the form “Is $x \in S$”?
  The disallowed passwords correspond to S.

Bit String Hashing

Main Issues

- Goal is to save space instead of time.
- Assume that passwords are restricted to 64 bits and that a hash function maps words into 32 bits.
- Fingerprinting and false positives.
- The approximate set membership problem:
  Given a set $S = \{s_1, s_2, \ldots, s_m\}$ of m elements from a large universe U, we would like to represent elements in such a way, as to efficiently answer queries of the form "Is $x \in S$"?
  The disallowed passwords correspond to S. We also want to be space efficient and are willing to tolerate some error.

Bit String Hashing

Main Issues

- Goal is to save space instead of time.
- Assume that passwords are restricted to 64 bits and that a hash function maps words into 32 bits.
- Fingerprinting and false positives.
- The approximate set membership problem:
  Given a set $S = \{s_1, s_2, \ldots, s_m\}$ of m elements from a large universe U, we would like to represent elements in such a way, as to efficiently answer queries of the form "Is $x \in S$"? The disallowed passwords correspond to S. We also want to be space efficient and are willing to tolerate some error.
- Assume that we use b bits to create a fingerprint.

Bit String Hashing

Main Issues

- Goal is to save space instead of time.
- Assume that passwords are restricted to 64 bits and that a hash function maps words into 32 bits.
- Fingerprinting and false positives.
- The approximate set membership problem:
  Given a set $S = \{s_1, s_2, \ldots, s_m\}$ of m elements from a large universe U, we would like to represent elements in such a way, as to efficiently answer queries of the form "Is $x \in S$"? The disallowed passwords correspond to S. We also want to be space efficient and are willing to tolerate some error.
- Assume that we use b bits to create a fingerprint. The probability that an acceptable password has a fingerprint that is different from any specific password is

Bit String Hashing

Main Issues

- Goal is to save space instead of time.
- Assume that passwords are restricted to 64 bits and that a hash function maps words into 32 bits.
- Fingerprinting and false positives.
- The approximate set membership problem:
  Given a set $S = \{s_1, s_2, \ldots, s_m\}$ of m elements from a large universe U, we would like to represent elements in such a way, as to efficiently answer queries of the form "Is $x \in S$"? The disallowed passwords correspond to S. We also want to be space efficient and are willing to tolerate some error.
- Assume that we use b bits to create a fingerprint. The probability that an acceptable password has a fingerprint that is different from any specific password is $(1 - \frac{1}{2^b})$.

Bit String Hashing (contd.)

Bit String Hashing (contd.)

Main Issues (contd.)

Bit String Hashing (contd.)

Main Issues (contd.)

- Since S has size m, what is the probability that an acceptable password fingerprint will match the fingerprints of one of the elements of S?

Bit String Hashing (contd.)

Main Issues (contd.)

- Since S has size m, what is the probability that an acceptable password fingerprint will match the fingerprints of one of the elements of S? $1-(1-\frac{1}{2^b})^m$

Bit String Hashing (contd.)

Main Issues (contd.)

- Since S has size m, what is the probability that an acceptable password fingerprint will match the fingerprints of one of the elements of S? $1-(1-\frac{1}{2^b})^m \geq 1-e^{\frac{-m}{2 \cdot b}}$.

Bit String Hashing (contd.)

Main Issues (contd.)

- Since S has size m, what is the probability that an acceptable password fingerprint will match the fingerprints of one of the elements of S? $1-(1-\frac{1}{2^b})^m \geq 1-e^{\frac{-m}{2 \cdot b}}$.
- Since we want the probability of a false positive to be less than c, we need,

Bit String Hashing (contd.)

Main Issues (contd.)

- Since S has size m, what is the probability that an acceptable password fingerprint will match the fingerprints of one of the elements of S? $1-(1-\frac{1}{2^b})^m \geq 1-e^{\frac{-m}{2\cdot b}}$.
- Since we want the probability of a false positive to be less than c, we need,

$$1-e^{\frac{-m}{2\cdot b}} \leq c$$

Bit String Hashing (contd.)

Main Issues (contd.)

- Since S has size m, what is the probability that an acceptable password fingerprint will match the fingerprints of one of the elements of S? $1-(1-\frac{1}{2^b})^m \geq 1-e^{\frac{-m}{2\cdot b}}$.
- Since we want the probability of a false positive to be less than c, we need,

$$\begin{aligned} 1-e^{\frac{-m}{2\cdot b}} &\leq c \\ \Rightarrow e^{\frac{-m}{2\cdot b}} &\geq 1-c \end{aligned}$$

Bit String Hashing (contd.)

Main Issues (contd.)

- Since S has size m, what is the probability that an acceptable password fingerprint will match the fingerprints of one of the elements of S? $1-(1-\frac{1}{2^b})^m \geq 1-e^{\frac{-m}{2\cdot b}}$.
- Since we want the probability of a false positive to be less than c, we need,

$$\begin{aligned} 1-e^{\frac{-m}{2\cdot b}} &\leq c \\ \Rightarrow e^{\frac{-m}{2\cdot b}} &\geq 1-c \\ \Rightarrow b &\geq \log_2 \frac{m}{\ln \frac{1}{1-c}} \end{aligned}$$

Bit String Hashing (contd.)

Main Issues (contd.)

- Since S has size m, what is the probability that an acceptable password fingerprint will match the fingerprints of one of the elements of S? $1-(1-\frac{1}{2^b})^m \geq 1-e^{\frac{-m}{2\cdot b}}$.
- Since we want the probability of a false positive to be less than c, we need,

$$\begin{aligned} 1-e^{\frac{-m}{2\cdot b}} &\leq c \\ \Rightarrow e^{\frac{-m}{2\cdot b}} &\geq 1-c \\ \Rightarrow b &\geq \log_2 \frac{m}{\ln \frac{1}{1-c}} \end{aligned}$$

- If we choose $b = 2 \cdot \log_2 m$, the probability of a false positive is:

Bit String Hashing (contd.)

Main Issues (contd.)

- Since S has size m, what is the probability that an acceptable password fingerprint will match the fingerprints of one of the elements of S? $1-(1-\frac{1}{2^b})^m \geq 1-e^{\frac{-m}{2\cdot b}}$.
- Since we want the probability of a false positive to be less than c, we need,

$$\begin{aligned} 1-e^{\frac{-m}{2\cdot b}} &\leq c \\ \Rightarrow e^{\frac{-m}{2\cdot b}} &\geq 1-c \\ \Rightarrow b &\geq \log_2 \frac{m}{\ln \frac{1}{1-c}} \end{aligned}$$

- If we choose $b = 2 \cdot \log_2 m$, the probability of a false positive is: $1-(1-\frac{1}{m^2})^m$

Bit String Hashing (contd.)

Main Issues (contd.)

- Since S has size m, what is the probability that an acceptable password fingerprint will match the fingerprints of one of the elements of S? $1-(1-\frac{1}{2^b})^m \geq 1-e^{\frac{-m}{2\cdot b}}$.
- Since we want the probability of a false positive to be less than c, we need,

$$\begin{aligned} 1-e^{\frac{-m}{2\cdot b}} &\leq c \\ \Rightarrow e^{\frac{-m}{2\cdot b}} &\geq 1-c \\ \Rightarrow b &\geq \log_2 \frac{m}{\ln \frac{1}{1-c}} \end{aligned}$$

- If we choose $b = 2\cdot\log_2 m$, the probability of a false positive is: $1-(1-\frac{1}{m^2})^m < \frac{1}{m}$.

If our dictionary has 2^{16} words, using 32 bits when hashing, leads to an error probability of at most $\frac{1}{65,536}$.

Outline

Bloom Filters

Bloom Filters

Main Issues

Bloom Filters

Main Issues

- Chain hashing optimizes time,

Bloom Filters

Main Issues

- Chain hashing optimizes time, while bit string hashing optimizes space.

Bloom Filters

Main Issues

- Chain hashing optimizes time, while bit string hashing optimizes space. Can we get a better trade-off?

Bloom Filters

Main Issues

- Chain hashing optimizes time, while bit string hashing optimizes space. Can we get a better trade-off? Bloom filters!

Bloom Filters

Main Issues

- Chain hashing optimizes time, while bit string hashing optimizes space. Can we get a better trade-off? Bloom filters!
- A Bloom filter consists of an array of n bits, $A[0]$ to $A[n-1]$,

Bloom Filters

Main Issues

- Chain hashing optimizes time, while bit string hashing optimizes space. Can we get a better trade-off? Bloom filters!
- A Bloom filter consists of an array of n bits, $A[0]$ to $A[n-1]$, initially set to 0.

Bloom Filters

Main Issues

- Chain hashing optimizes time, while bit string hashing optimizes space. Can we get a better trade-off? Bloom filters!
- A Bloom filter consists of an array of n bits, $A[0]$ to $A[n-1]$, initially set to 0.
- k hash function h_1, h_2, $\ldots$, h_k with range $\{0, \ldots, n-1\}$ are used.

Bloom Filters

Main Issues

- Chain hashing optimizes time, while bit string hashing optimizes space. Can we get a better trade-off? Bloom filters!
- A Bloom filter consists of an array of n bits, $A[0]$ to $A[n-1]$, initially set to 0.
- k hash function h_1, h_2, ..., h_k with range $\{0,\ldots,n-1\}$ are used.
- We wish to represent a set $S = \{s_1, s_2, \ldots, s_m\}$ of m elements.

Bloom Filters

Main Issues

- Chain hashing optimizes time, while bit string hashing optimizes space. Can we get a better trade-off? Bloom filters!
- A Bloom filter consists of an array of n bits, $A[0]$ to $A[n-1]$, initially set to 0.
- k hash function h_1, h_2, ..., h_k with range $\{0,\ldots,n-1\}$ are used.
- We wish to represent a set $S = \{s_1, s_2, \ldots, s_m\}$ of m elements.
- Set $A[h_i(s)]$ to 1, for each $1 \le i \le k$ and each $s \in S$.

Bloom Filters

Main Issues

- Chain hashing optimizes time, while bit string hashing optimizes space. Can we get a better trade-off? Bloom filters!
- A Bloom filter consists of an array of n bits, $A[0]$ to $A[n-1]$, initially set to 0.
- k hash function h_1, h_2, $\ldots$, h_k with range $\{0, \ldots, n-1\}$ are used.
- We wish to represent a set $S = \{s_1, s_2, \ldots, s_m\}$ of m elements.
- Set $A[h_i(s)]$ to 1, for each $1 \leq i \leq k$ and each $s \in S$.
- How to check if $x \in S$?

Bloom Filters

Main Issues

- Chain hashing optimizes time, while bit string hashing optimizes space. Can we get a better trade-off? Bloom filters!
- A Bloom filter consists of an array of n bits, $A[0]$ to $A[n-1]$, initially set to 0.
- k hash function $h_1, h_2, \ldots, h_k$ with range $\{0, \ldots, n-1\}$ are used.
- We wish to represent a set $S = \{s_1, s_2, \ldots, s_m\}$ of m elements.
- Set $A[h_i(s)]$ to 1, for each $1 \leq i \leq k$ and each $s \in S$.
- How to check if $x \in S$? Check all locations $A[h_i(x)]$, $1 \leq i \leq k$.

Bloom Filters

Main Issues

- Chain hashing optimizes time, while bit string hashing optimizes space. Can we get a better trade-off? Bloom filters!
- A Bloom filter consists of an array of n bits, $A[0]$ to $A[n-1]$, initially set to 0.
- k hash function h_1, h_2, $\ldots$, h_k with range $\{0, \ldots, n-1\}$ are used.
- We wish to represent a set $S = \{s_1, s_2, \ldots, s_m\}$ of m elements.
- Set $A[h_i(s)]$ to 1, for each $1 \leq i \leq k$ and each $s \in S$.
- How to check if $x \in S$? Check all locations $A[h_i(x)]$, $1 \leq i \leq k$.
- How could we go wrong?

Bloom Filters

Main Issues

- Chain hashing optimizes time, while bit string hashing optimizes space. Can we get a better trade-off? Bloom filters!
- A Bloom filter consists of an array of n bits, $A[0]$ to $A[n-1]$, initially set to 0.
- k hash function $h_1, h_2, \ldots, h_k$ with range $\{0, \ldots, n-1\}$ are used.
- We wish to represent a set $S = \{s_1, s_2, \ldots, s_m\}$ of m elements.
- Set $A[h_i(s)]$ to 1, for each $1 \leq i \leq k$ and each $s \in S$.
- How to check if $x \in S$? Check all locations $A[h_i(x)]$, $1 \leq i \leq k$.
- How could we go wrong? False positives.

Analyzing error probability

Analyzing error probability

Analysis

Analyzing error probability

Analysis

Analyzing error probability

Analysis

- What is the probability that a specific bit is 0, after the preprocessing?

Analyzing error probability

Analysis

- What is the probability that a specific bit is 0, after the preprocessing? $(1 - \frac{1}{n})^{k \cdot m}$

Analyzing error probability

Analysis

- What is the probability that a specific bit is 0, after the preprocessing? $(1 - \frac{1}{n})^{k \cdot m} = e^{-\frac{k \cdot m}{n}}$.

Analyzing error probability

Analysis

- What is the probability that a specific bit is 0, after the preprocessing? $(1-\frac{1}{n})^{k\cdot m} = e^{-\frac{k\cdot m}{n}}$.
- Assume that a fraction $p = e^{-\frac{k\cdot m}{n}}$ of the entries are still 0, after S has been hashed into the Bloom filter.

Analyzing error probability

Analysis

- What is the probability that a specific bit is 0, after the preprocessing? $(1-\frac{1}{n})^{k\cdot m} = e^{-\frac{k\cdot m}{n}}$.
- Assume that a fraction $p = e^{-\frac{k\cdot m}{n}}$ of the entries are still 0, after S has been hashed into the Bloom filter.
- The probability of a false positive is then precisely the probability that all the hash functions map the input string $x \notin S$, to 1,

Analyzing error probability

Analysis

- What is the probability that a specific bit is 0, after the preprocessing? $(1-\frac{1}{n})^{k\cdot m} = e^{-\frac{k\cdot m}{n}}$.
- Assume that a fraction $p = e^{-\frac{k\cdot m}{n}}$ of the entries are still 0, after S has been hashed into the Bloom filter.
- The probability of a false positive is then precisely the probability that all the hash functions map the input string $x \notin S$, to 1, which is,

Analyzing error probability

Analysis

- What is the probability that a specific bit is 0, after the preprocessing? $(1 - \frac{1}{n})^{k \cdot m} = e^{-\frac{k \cdot m}{n}}$.
- Assume that a fraction $p = e^{-\frac{k \cdot m}{n}}$ of the entries are still 0, after S has been hashed into the Bloom filter.
- The probability of a false positive is then precisely the probability that all the hash functions map the input string $x \notin S$, to 1, which is,

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

Analyzing error probability

Analysis

- What is the probability that a specific bit is 0, after the preprocessing? $(1-\frac{1}{n})^{k \cdot m} = e^{-\frac{k \cdot m}{n}}$.
- Assume that a fraction $p = e^{-\frac{k \cdot m}{n}}$ of the entries are still 0, after S has been hashed into the Bloom filter.
- The probability of a false positive is then precisely the probability that all the hash functions map the input string $x \notin S$, to 1, which is,

$$(1-(1-\frac{1}{n})^{k \cdot m})^k = (1-e^{-\frac{k \cdot m}{n}})^k$$

Analyzing error probability

Analysis

- What is the probability that a specific bit is 0, after the preprocessing? $(1-\frac{1}{n})^{k\cdot m} = e^{-\frac{k\cdot m}{n}}$.
- Assume that a fraction $p = e^{-\frac{k\cdot m}{n}}$ of the entries are still 0, after S has been hashed into the Bloom filter.
- The probability of a false positive is then precisely the probability that all the hash functions map the input string $x \notin S$, to 1, which is,

$$\begin{aligned}(1-(1-\frac{1}{n})^{k\cdot m})^k &= (1-e^{-\frac{k\cdot m}{n}})^k \\ &= f()\end{aligned}$$

Analyzing error probability

Analysis

- What is the probability that a specific bit is 0, after the preprocessing? $(1-\frac{1}{n})^{k\cdot m} = e^{-\frac{k\cdot m}{n}}$.
- Assume that a fraction $p = e^{-\frac{k\cdot m}{n}}$ of the entries are still 0, after S has been hashed into the Bloom filter.
- The probability of a false positive is then precisely the probability that all the hash functions map the input string $x \notin S$, to 1, which is,

$$
\begin{aligned}
(1-(1-\frac{1}{n})^{k\cdot m})^k &= (1-e^{-\frac{k\cdot m}{n}})^k \\
&= f()
\end{aligned}
$$

- Optimizing for k, we get $k = \ln 2$ and $f \approx (0.6185)^{\frac{m}{n}}$.

Outline

Breaking Symmetry

Main Issues

Breaking Symmetry

Main Issues

- Resource contention.

Breaking Symmetry

Main Issues

- Resource contention.
- Sequential ordering.

Breaking Symmetry

Main Issues

- Resource contention.
- Sequential ordering.
- The Hashing approach.

Breaking Symmetry

Main Issues

- Resource contention.
- Sequential ordering.
- The Hashing approach. Hash each user identifier into b bits.

Breaking Symmetry

Main Issues

- Resource contention.
- Sequential ordering.
- The Hashing approach. Hash each user identifier into b bits.
- How likely that two users will have the same hash value?

Breaking Symmetry

Main Issues

- Resource contention.
- Sequential ordering.
- The Hashing approach. Hash each user identifier into b bits.
- How likely that two users will have the same hash value? Fix one user.

Breaking Symmetry

Main Issues

- Resource contention.
- Sequential ordering.
- The Hashing approach. Hash each user identifier into b bits.
- How likely that two users will have the same hash value? Fix one user. What is the probability that some other user obtains the same hash value?

Breaking Symmetry

Main Issues

- Resource contention.
- Sequential ordering.
- The Hashing approach. Hash each user identifier into b bits.
- How likely that two users will have the same hash value? Fix one user. What is the probability that some other user obtains the same hash value?

$$1 - (1 - \frac{1}{2^b})^{n-1}$$

Breaking Symmetry

Main Issues

- Resource contention.
- Sequential ordering.
- The Hashing approach. Hash each user identifier into b bits.
- How likely that two users will have the same hash value? Fix one user. What is the probability that some other user obtains the same hash value?

$$1-(1-\frac{1}{2^b})^{n-1} \leq \frac{n-1}{2^b}$$

Breaking Symmetry

Main Issues

- Resource contention.
- Sequential ordering.
- The Hashing approach. Hash each user identifier into b bits.
- How likely that two users will have the same hash value? Fix one user. What is the probability that some other user obtains the same hash value?

$$1 - (1 - \frac{1}{2^b})^{n-1} \leq \frac{n-1}{2^b}$$

By the union bound, the probability that any user has the same hash values as the fixed user is

Breaking Symmetry

Main Issues

- Resource contention.
- Sequential ordering.
- The Hashing approach. Hash each user identifier into b bits.
- How likely that two users will have the same hash value? Fix one user. What is the probability that some other user obtains the same hash value?

$$1 - (1 - \frac{1}{2^b})^{n-1} \leq \frac{n-1}{2^b}$$

By the union bound, the probability that any user has the same hash values as the fixed user is $\frac{n \cdot (n-1)}{2^b}$.

Breaking Symmetry

Main Issues

- Resource contention.
- Sequential ordering.
- The Hashing approach. Hash each user identifier into b bits.
- How likely that two users will have the same hash value? Fix one user. What is the probability that some other user obtains the same hash value?

$$1-(1-\frac{1}{2^b})^{n-1} \leq \frac{n-1}{2^b}$$

By the union bound, the probability that any user has the same hash values as the fixed user is $\frac{n \cdot (n-1)}{2^b}$. In order to guarantee success with probability $(1-\frac{1}{n})$, choose $b =$

Breaking Symmetry

Main Issues

- Resource contention.
- Sequential ordering.
- The Hashing approach. Hash each user identifier into b bits.
- How likely that two users will have the same hash value? Fix one user. What is the probability that some other user obtains the same hash value?

$$1-(1-\frac{1}{2^b})^{n-1} \leq \frac{n-1}{2^b}$$

By the union bound, the probability that any user has the same hash values as the fixed user is $\frac{n \cdot (n-1)}{2^b}$. In order to guarantee success with probability $(1-\frac{1}{n})$, choose $b = 3 \cdot \log n_2$.

Breaking Symmetry

Main Issues

- Resource contention.
- Sequential ordering.
- The Hashing approach. Hash each user identifier into b bits.
- How likely that two users will have the same hash value? Fix one user. What is the probability that some other user obtains the same hash value?

$$1-(1-\frac{1}{2^b})^{n-1} \leq \frac{n-1}{2^b}$$

By the union bound, the probability that any user has the same hash values as the fixed user is $\frac{n\cdot(n-1)}{2^b}$. In order to guarantee success with probability $(1-\frac{1}{n})$, choose $b = 3 \cdot \log n_2$.

Can also be used for the leader election problem.