COMPSCI 690RA: Randomized Algorithms and Probabilistic Data Analysis

Prof. Cameron Musco

University of Massachusetts Amherst. Spring 2022.

Lecture 3

Logistics

- Problem Set 1 had its due date postponed until Tuesday 2/15 at 8pm.
- We will still have a weekly quiz this week, also due Tuesday 2/15 at 8pm.
- Most people think the lectures are 'just right' or 'a bit too fast'. I'll try to slow down a bit. If you feel that you are really falling behind, let me know.

Summary

Last Time:

- Concentration bounds – Markov's and Chebyshev's inequalities.
- The union bound.
- Quicksort analysis
- Coupon collecting, statistical estimation
- Randomized load balancing and ball-into-bins

Summary

Last Time:

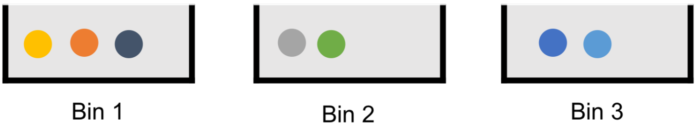
- Concentration bounds – Markov's and Chebyshev's inequalities.
- The union bound.
- Quicksort analysis
- Coupon collecting, statistical estimation
- Randomized load balancing and ball-into-bins

Today:

- Stronger concentration bounds for sums of independent random variables. I.e., exponential concentration bounds.
- Randomized hash function and fingerprints.
- Applications to fast pattern mining and efficient communication protocols.

Balls Into Bins

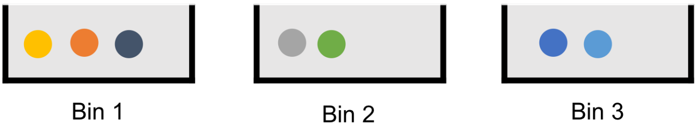
I throw m balls independently and uniformly at random into n bins. What is the maximum number of balls any bin?



- Applications to randomized load balancing
- Analysis of hash tables using chaining.

Balls Into Bins

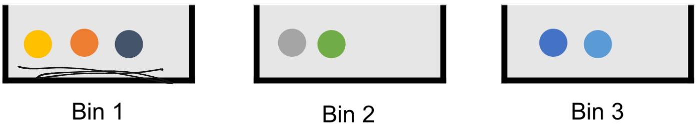
I throw m balls independently and uniformly at random into n bins. What is the maximum number of balls any bin?



- Applications to randomized load balancing
- Analysis of hash tables using chaining.
- **Direct Proof:** For any bin i, $\Pr[\mathbf{b}_i \geq \frac{c \ln n}{\ln \ln n}] \leq \frac{1}{n^{c-o(1)}}$. Thus, via union bound, the maximum load is exceeds $\frac{c \ln n}{\ln \ln n}$ with probability at most $\frac{1}{n^{c-1-o(1)}}$.

Balls Into Bins

I throw m balls independently and uniformly at random into n bins. What is the maximum number of balls any bin?



- Applications to randomized load balancing
- Analysis of hash tables using chaining.
- **Direct Proof:** For any bin i, $\Pr[\mathbf{b}_i \geq \frac{c \ln n}{\ln \ln n}] \leq \frac{1}{n^{c-o(1)}}$. Thus, via union bound, the maximum load is exceeds $\frac{c \ln n}{\ln \ln n}$ with probability at most $\frac{1}{n^{c-1-o(1)}}$.
- Proof using Chebyshev's inequality gives a weak bound of $O(\sqrt{n})$ for the maximum load.

Exponential Concentration Bounds

Higher Moments

Markov's Inequality: $\Pr[X \geq t] \leq \frac{\mathbb{E}[X]}{t}$. First moment.

Chebyshev's Inequality: $\Pr[X \geq t] \leq \frac{\mathbb{E}[X^2]}{t^2}$. Second moment.

$Pr(X^2 \geq t^2)$

Higher Moments

Markov's Inequality: $\Pr[X \geq t] \leq \frac{\mathbb{E}[X]}{t}$. First moment.

Chebyshev's Inequality: $\Pr[X \geq t] \leq \frac{\mathbb{E}[X^2]}{t^2}$. Second moment.

Often (not always!) we can obtain tighter bounds by looking to higher moments of the random variable.

$$\Pr(X \geq t) = \Pr(X^4 \geq t^4) \leq \frac{\mathbb{E} X^4}{t^4}$$

Higher Moments

Markov's Inequality: $\Pr[X \geq t] \leq \frac{\mathbb{E}[X]}{t}$. First moment.

Chebyshev's Inequality: $\Pr[X \geq t] \leq \frac{\mathbb{E}[X^2]}{t^2}$. Second moment.

Often (not always!) we can obtain tighter bounds by looking to higher moments of the random variable.

Moment Generating Function: Consider for any $z > 0$:

$$M_z(X) = e^{z \cdot X} = \sum_{k=0}^{\infty} \frac{z^k X^k}{k!}$$

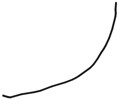
Higher Moments

Markov's Inequality: $\Pr[X \geq t] \leq \frac{\mathbb{E}[X]}{t}$. First moment.

Chebyshev's Inequality: $\Pr[X \geq t] \leq \frac{\mathbb{E}[X^2]}{t^2}$. Second moment.

Often (not always!) we can obtain tighter bounds by looking to higher moments of the random variable.

Moment Generating Function: Consider for any $z > 0$:



$$M_z(X) = e^{z \cdot X} = \sum_{k=0}^{\infty} \frac{z^k X^k}{k!}$$

$e^{z \cdot t}$ is non-negative, and monotonic for any $z > 0$. So can bound via Markov's inequality, $\Pr[X \geq t] = \Pr[M_z(X) \geq e^{zt}] \leq \frac{\mathbb{E}[M_z(X)]}{e^{zt}}$.

Higher Moments

Markov's Inequality: $\Pr[X \geq t] \leq \frac{\mathbb{E}[X]}{t}$. First moment.

Chebyshev's Inequality: $\Pr[X \geq t] \leq \frac{\mathbb{E}[X^2]}{t^2}$. Second moment.

Often (not always!) we can obtain tighter bounds by looking to higher moments of the random variable.

Moment Generating Function: Consider for any $z > 0$:

$$M_z(X) = e^{z \cdot X} = \sum_{k=0}^{\infty} \frac{z^k X^k}{k!}$$

$e^{z \cdot t}$ is non-negative, and monotonic for any $z > 0$. So can bound via Markov's inequality, $\Pr[X \geq t] = \Pr[M_z(X) \geq e^{zt}] \leq \frac{\mathbb{E}[M_z(X)]}{e^{zt}}$.

By appropriately picking z and bounding $\mathbb{E}[M_z(X)]$, we can obtain a variety of exponential tail bounds. Typically require that X is a sum of bounded and independent random variables

$\mathbb{E} XY = \mathbb{E} X \cdot \mathbb{E} Y$

$\Pr(X \geq t) \leq \frac{\mathbb{E}(X^k)}{t^k}$

$\Pr(X \geq t) = \Pr(X^3 \geq t^3) \leq ?$

$X = X_1 + X_2 + \dots X_n$

$\mathbb{E}(e^{zX}) = \mathbb{E} e^{z(X_1 + X_2 + \dots X_n)} = \mathbb{E}(e^{zX_1} \cdot e^{zX_2} \cdots e^{zX_n})$

$\mathbb{E}(e^{zX}) = \prod_{i=1}^{n} \mathbb{E} e^{zX_i}$

let f be monotonic function above zero $\Pr(X \geq t) = \Pr(f(X) \geq f(t)) \leq \frac{\mathbb{E} f(X)}{f(t)}$

The Chernoff Bound

Chernoff Bound ~~(simplified version)~~: Consider independent random variables $X_1, \ldots, X_n$ taking values in $\{0, 1\}$ and let $X = \sum_{i=1}^n X_i$. Let $\mu = \mathbb{E}[X] = \mathbb{E}[\sum_{i=1}^n X_i]$. For any $\delta \geq 0$

$\delta = 5$

$$\Pr\left(X \geq (1+\delta)\mu\right) \leq \frac{e^{\delta\mu}}{(1+\delta)^{(1+\delta)\mu}}$$

$$\frac{e^{5\mu}}{6^{6\mu}}$$

Is X binomially distributed?

The Chernoff Bound

Chernoff Bound (simplified version): Consider independent random variables $X_1, \ldots, X_n$ taking values in $\{0,1\}$ and let $X = \sum_{i=1}^n X_i$. Let $\mu = \mathbb{E}[X] = \mathbb{E}[\sum_{i=1}^n X_i]$. For any $\delta \geq 0$

$$\Pr\left(X \geq (1+\delta)\mu\right) \leq \frac{e^{\delta\mu}}{(1+\delta)^{(1+\delta)\mu}}$$

Chernoff Bound (alternate version): Consider independent random variables $X_1, \ldots, X_n$ taking values in $\{0,1\}$ and let $X = \sum_{i=1}^n X_i$. Let $\mu = \mathbb{E}[X] = \mathbb{E}[\sum_{i=1}^n X_i]$. For any $\delta \geq 0$

$\delta \approx 0$ ≈ 0

$$\Pr\left(\left|\sum_{i=1}^n X_i - \mu\right| \geq \delta\mu\right) \leq 2\exp\left(-\frac{\delta^2\mu}{2+\delta}\right).$$

As δ gets larger and larger, the bound falls off exponentially fast.

Balls Into Bins Via Chernoff Bound

Recall that $\mathbf{b}_i$ is the number of balls landing in bin i, when we randomly throw n balls into n bins.

- $\mathbf{b}_i = \sum_{i=1}^{n} \mathbf{I}_{i,j}$ where $\mathbf{I}_{i,j} = 1$ with probability $1/n$ and 0 otherwise. $\mathbf{I}_{i,1}, \ldots \mathbf{I}_{i,n}$ are independent.

Balls Into Bins Via Chernoff Bound

Recall that b_i is the number of balls landing in bin i, when we randomly throw n balls into n bins.

- $\mathsf{b}_i = \sum_{i=1}^{n} \mathsf{I}_{i,j}$ where $\mathsf{I}_{i,j} = 1$ with probability $1/n$ and 0 otherwise. $\mathsf{I}_{i,1}, \ldots \mathsf{I}_{i,n}$ are independent.
- Apply Chernoff bound with $\mu = \mathbb{E}[\mathsf{b}_i] = 1$:

$$\Pr[\mathsf{b}_i \geq k+1] \leq \frac{e^k}{(1+k)^{(1+k)}}.$$

$$Pr(b_i \geq (1+\delta)\mu) \leq \frac{e^{\delta\mu}}{(1+\delta)^{(1+\delta)}}$$

$\mu = 1 \quad \delta = k$

Balls Into Bins Via Chernoff Bound

Recall that $\mathbf{b}_i$ is the number of balls landing in bin i, when we randomly throw n balls into n bins.

- $\mathbf{b}_i = \sum_{i=1}^{n} \mathbf{I}_{i,j}$ where $\mathbf{I}_{i,j} = 1$ with probability $1/n$ and 0 otherwise. $\mathbf{I}_{i,1}, \ldots \mathbf{I}_{i,n}$ are independent.
- Apply Chernoff bound with $\mu = \mathbb{E}[\mathbf{b}_i] = 1$:

$$\Pr[\mathbf{b}_i \geq k] \leq \frac{e^k}{(1+k)^{(1+k)}}.$$

- For $k \geq \frac{c \log n}{\log \log n}$ we have:

$$\Pr[\mathbf{b}_i \geq k] \leq \frac{e^{\frac{c \log n}{\log \log n}}}{\left(\frac{c \log n}{\log \log n}\right)^{\frac{c \log n}{\log \log n}}} =$$

$$e^{-[c \log\log n - c \log\log\log n] \cdot \frac{c \log n}{\log \log n}}$$

$$e^{-c^2 \log n} \cdot e^{\frac{c \log n}{\log\log n}}$$

$$\approx e^{-O(\log n)} = 1/n^{O(1)}$$

Balls Into Bins Via Chernoff Bound

Recall that $\mathbf{b}_i$ is the number of balls landing in bin i, when we randomly throw n balls into n bins.

- $\mathbf{b}_i = \sum_{i=1}^{n} \mathbf{I}_{i,j}$ where $\mathbf{I}_{i,j} = 1$ with probability $1/n$ and 0 otherwise. $\mathbf{I}_{i,1}, \ldots \mathbf{I}_{i,n}$ are independent.
- Apply Chernoff bound with $\mu = \mathbb{E}[\mathbf{b}_i] = 1$:

$$\Pr[\mathbf{b}_i \geq k] \leq \frac{e^k}{(1+k)^{(1+k)}}.$$

- For $k \geq \frac{c \log n}{\log \log n}$ we have:

$$\Pr[\mathbf{b}_i \geq k] \leq \frac{e^{\frac{c \log n}{\log \log n}}}{\left(\frac{c \log n}{\log \log n}\right)^{\frac{c \log n}{\log \log n}}} = \frac{1}{n^{c-o(1)}}$$

Balls Into Bins Via Chernoff Bound

Recall that $\mathbf{b}_i$ is the number of balls landing in bin i, when we randomly throw n balls into n bins.

- $\mathbf{b}_i = \sum_{i=1}^{n} \mathbf{I}_{i,j}$ where $\mathbf{I}_{i,j} = 1$ with probability $1/n$ and 0 otherwise. $\mathbf{I}_{i,1}, \ldots \mathbf{I}_{i,n}$ are independent.
- Apply Chernoff bound with $\mu = \mathbb{E}[\mathbf{b}_i] = 1$:

$$\Pr[\mathbf{b}_i \geq k] \leq \frac{e^k}{(1+k)^{(1+k)}}.$$

- For $k \geq \frac{c \log n}{\log \log n}$ we have:

$$\Pr[\mathbf{b}_i \geq k] \leq \frac{e^{\frac{c \log n}{\log \log n}}}{\left(\frac{c \log n}{\log \log n}\right)^{\frac{c \log n}{\log \log n}}} = \frac{1}{n^{c - o(1)}}$$

Upshot: We recover the right bound for balls into bins.

Bernstein Inequality

Bernstein Inequality: Consider independent random variables $X_1, \ldots, X_n$ all falling in $[-M, M]$ and let $X = \sum_{i=1}^{n} X_i$. Let $\mu = \mathbb{E}[X]$ and $\sigma^2 = \text{Var}[X] = \sum_{i=1}^{n} \text{Var}[X_i]$. For any $t \geq 0$:

$$\Pr\left(\left|\sum_{i=1}^{n} X_i - \mu\right| \geq t\right) \leq 2\exp\left(-\frac{t^2}{2\sigma^2 + \frac{4}{3}Mt}\right).$$

$$\sum_{i=1}^{n} Var(X_i) \leq nM^2$$

Bernstein Inequality

Bernstein Inequality: Consider independent random variables $X_1, \ldots, X_n$ all falling in $[-M, M]$ and let $X = \sum_{i=1}^n X_i$. Let $\mu = \mathbb{E}[X]$ and $\sigma^2 = \text{Var}[X] = \sum_{i=1}^n \text{Var}[X_i]$. For any $t \geq 0$:

$$\Pr\left(\left|\sum_{i=1}^n X_i - \mu\right| \geq t\right) \leq 2\exp\left(-\frac{t^2}{2\sigma^2 + \frac{4}{3}Mt}\right).$$

Assume that $M = 1$ and plug in $t = s \cdot \sigma$ for $s \leq \sigma$.

Bernstein Inequality

Bernstein Inequality: Consider independent random variables $X_1, \ldots, X_n$ all falling in [-1,1] and let $X = \sum_{i=1}^{n} X_i$. Let $\mu = \mathbb{E}[X]$ and $\sigma^2 = \text{Var}[X] = \sum_{i=1}^{n} \text{Var}[X_i]$. For any $s \geq 0$:

$$\Pr\left(\left|\sum_{i=1}^{n} X_i - \mu\right| \geq s\sigma\right) \leq 2\exp\left(-\frac{s^2}{4}\right).$$

Assume that $M = 1$ and plug in $t = s \cdot \sigma$ for $s \leq \sigma$.

Bernstein Inequality

Bernstein Inequality: Consider independent random variables $X_1, \ldots, X_n$ all falling in [-1,1] and let $X = \sum_{i=1}^n X_i$. Let $\mu = \mathbb{E}[X]$ and $\sigma^2 = \text{Var}[X] = \sum_{i=1}^n \text{Var}[X_i]$. For any $s \geq 0$:

$$\Pr\left(\left|\sum_{i=1}^n X_i - \mu\right| \geq s\sigma\right) \leq 2\exp\left(-\frac{s^2}{4}\right).$$

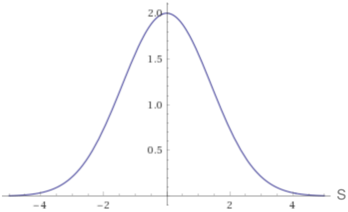
Assume that $M = 1$ and plug in $t = s \cdot \sigma$ for $s \leq \sigma$.

Compare to Chebyshev's: $\Pr\left(\left|\sum_{i=1}^n X_i - \mu\right| \geq s\sigma\right) \leq \frac{1}{s^2}$.

- An exponentially stronger dependence on s!

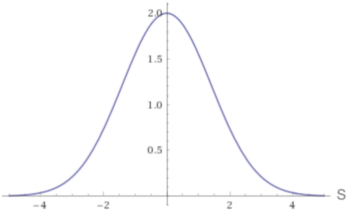
Interpretation as a Central Limit Theorem

Simplified Bernstein: Probability of a sum of independent, bounded random variables lying $\geq s$ standard deviations from its mean is $\approx \exp\left(-\frac{s^2}{4}\right)$. Can plot this bound for different s:



Interpretation as a Central Limit Theorem

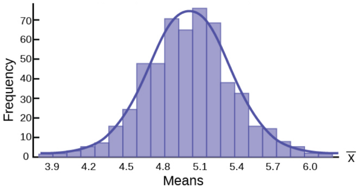
Simplified Bernstein: Probability of a sum of independent, bounded random variables lying $\geq s$ standard deviations from its mean is $\approx \exp\left(-\frac{s^2}{4}\right)$. Can plot this bound for different s:



- Looks like a Gaussian (normal) distribution – can think of Bernstein's inequality as giving a quantitative version of the central limit theorem.
- The distribution of the sum of bounded independent random variables can be upper bounded with a Gaussian distribution.

Central Limit Theorem

Stronger Central Limit Theorem: The distribution of the sum of *n bounded* independent random variables converges to a Gaussian (normal) distribution as *n* goes to infinity.



- The Gaussian distribution is so important since many random variables can be approximated as the sum of a large number of small and roughly independent random effects. Thus, their distribution looks Gaussian by CLT.

Sampling for Approximation

I have an $n \times n$ matrix with entries in $[0, 1]$. I want to estimate the sum of entries. I sample s entries uniformly at random with replacement, take their sum, and multiply it by n^2/s. How large must s be so that this method returns the correct answer, up to error $\pm\epsilon \cdot n^2$ with probability at least $1 - 1/n$?

(a) $O(n^2)$ (b) $O(n/\epsilon)$ (c) $O(\log n/\epsilon)$ (d) $O(\log n/\epsilon^2)$

I have an $n \times n$ matrix with entries in $[0, 1]$. I want to estimate the sum of entries. I sample s entries uniformly at random with replacement, take their sum, and multiply it by n^2/s. How large must s be so that this method returns the correct answer, up to error $\pm\epsilon \cdot n^2$ with probability at least $1 - 1/n$?

(a) $O(n^2)$ (b) $O(n/\epsilon)$ (c) $O(\log n/\epsilon)$ (d) $O(\log n/\epsilon^2)$

Bernstein Inequality: Consider independent random variables $X_1, \ldots, X_n$ all falling in $[-M, M]$ and let $X = \sum_{i=1}^n X_i$. Let $\mu = \mathbb{E}[X]$ and $\sigma^2 = \text{Var}[X] = \sum_{i=1}^n \text{Var}[X_i]$. For any $t \geq 0$:

$$\Pr\left(\left|\sum_{i=1}^n X_i - \mu\right| \geq t\right) \leq 2\exp\left(-\frac{t^2}{2\sigma^2 + \frac{4}{3}Mt}\right).$$

Sampling for Approximation

I have an $n \times n$ matrix with entries in $[0, 1]$. I want to estimate the sum of entries. I sample s entries uniformly at random with replacement, take their sum, and multiply it by n^2/s. How large must s be so that this method returns the correct answer, up to error $\pm\epsilon \cdot n^2$ with probability at least $1 - 1/n$?

(a) $O(n^2)$ (b) $O(n/\epsilon)$ (c) $O(\log n/\epsilon)$ (d) $O(\log n/\epsilon^2)$

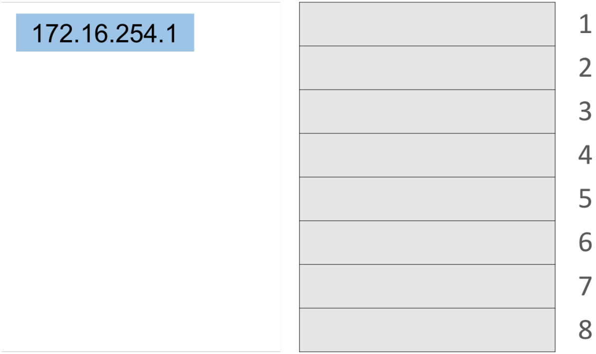
Application: Linear Probing

Linear Probing

Linear probing is the simplest form of open addressing for hash tables. If an item is hashed into a full bucket, keep trying buckets until you find an empty one.

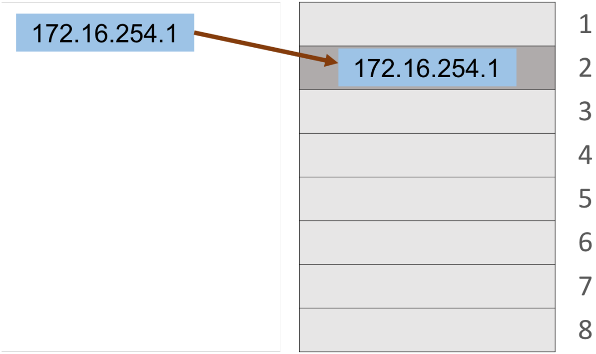
Linear Probing

Linear probing is the simplest form of open addressing for hash tables. If an item is hashed into a full bucket, keep trying buckets until you find an empty one.



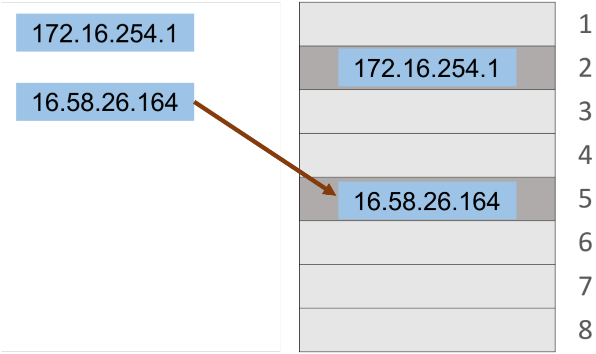
Linear Probing

Linear probing is the simplest form of open addressing for hash tables. If an item is hashed into a full bucket, keep trying buckets until you find an empty one.



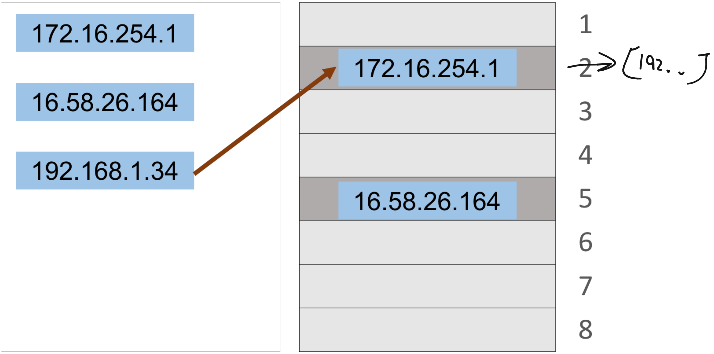
Linear Probing

Linear probing is the simplest form of open addressing for hash tables. If an item is hashed into a full bucket, keep trying buckets until you find an empty one.



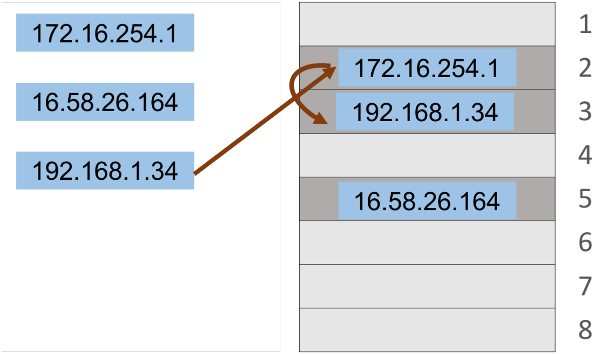
Linear Probing

Linear probing is the simplest form of open addressing for hash tables. If an item is hashed into a full bucket, keep trying buckets until you find an empty one.



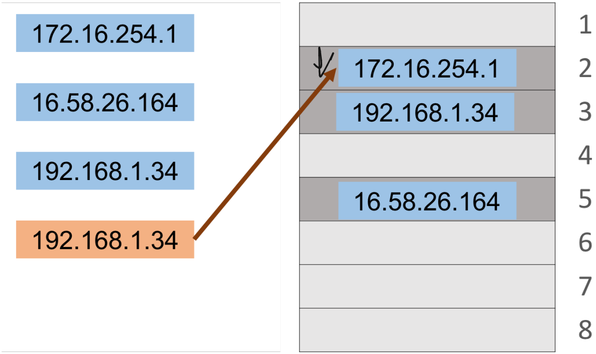
Linear Probing

Linear probing is the simplest form of open addressing for hash tables. If an item is hashed into a full bucket, keep trying buckets until you find an empty one.



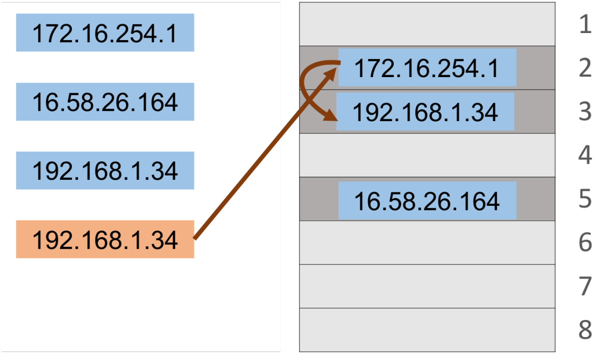
Linear Probing

Linear probing is the simplest form of open addressing for hash tables. If an item is hashed into a full bucket, keep trying buckets until you find an empty one.



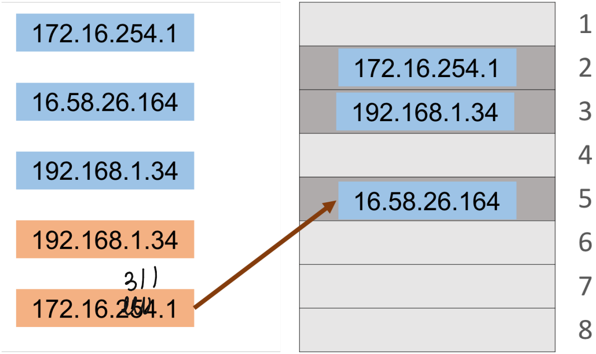
Linear Probing

Linear probing is the simplest form of open addressing for hash tables. If an item is hashed into a full bucket, keep trying buckets until you find an empty one.



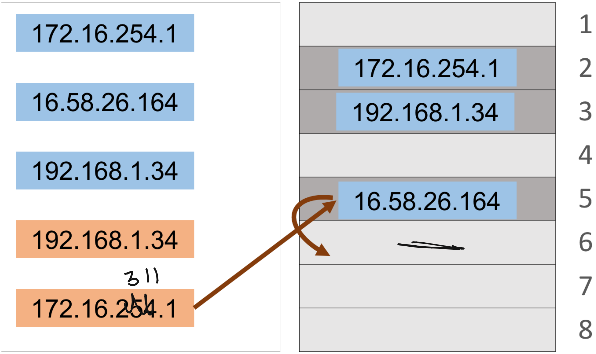
Linear Probing

Linear probing is the simplest form of open addressing for hash tables. If an item is hashed into a full bucket, keep trying buckets until you find an empty one.



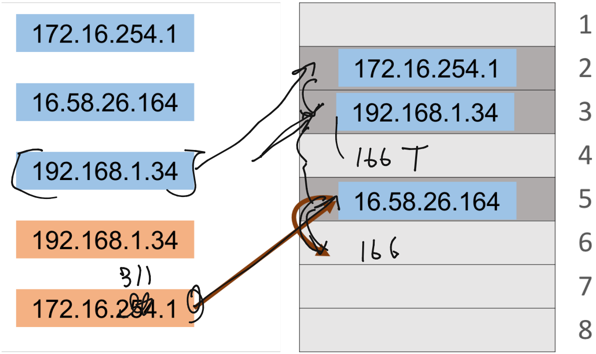
Linear Probing

Linear probing is the simplest form of open addressing for hash tables. If an item is hashed into a full bucket, keep trying buckets until you find an empty one.



Linear Probing

Linear probing is the simplest form of open addressing for hash tables. If an item is hashed into a full bucket, keep trying buckets until you find an empty one.



Simple and potentially very efficient – but performance can degrade as the hash table fills up.

Linear Probing Expected Runtime

Theorem: If the hash table has n inserted items and $m \geq 2n$ buckets, then linear probing requires $O(1)$ expected time per insertion/query.

Linear Probing Expected Runtime

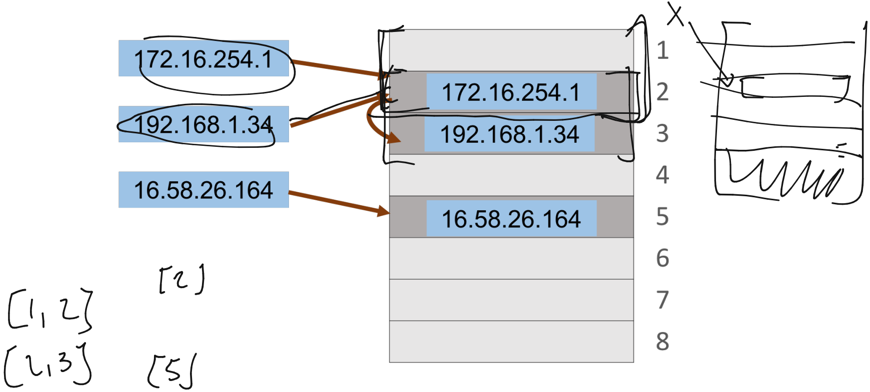
Theorem: If the hash table has n inserted items and $m \geq 2n$ buckets, then linear probing requires $O(1)$ expected time per insertion/query.

Definition: For any interval $I \subset [n]$, let $\mathbf{L}(I) = |\{x : \mathbf{h}(x) \in I\}|$ be the number of items hashed to the interval. We say I is full if $\mathbf{L}(I) \geq |I|$.

Linear Probing Expected Runtime

Theorem: If the hash table has n inserted items and $m \geq 2n$ buckets, then linear probing requires $O(1)$ expected time per insertion/query.

Definition: For any interval $I \subset [m]$, let $\mathbf{L}(I) = |\{x : \mathbf{h}(x) \in I\}|$ be the number of items hashed to the interval. We say I is full if $\mathbf{L}(I) \geq |I|$.

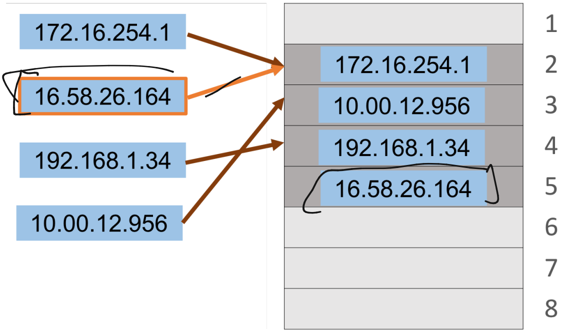


Which intervals in this table are full?

Analysis via Full Intervals

Claim Let $\mathbf{T}(x)$ denote the number of steps required for an insertion/query operation for item x. If $\mathbf{T}(x) > k$, there are at least k full intervals of different lengths containing $\mathbf{h}(x)$.

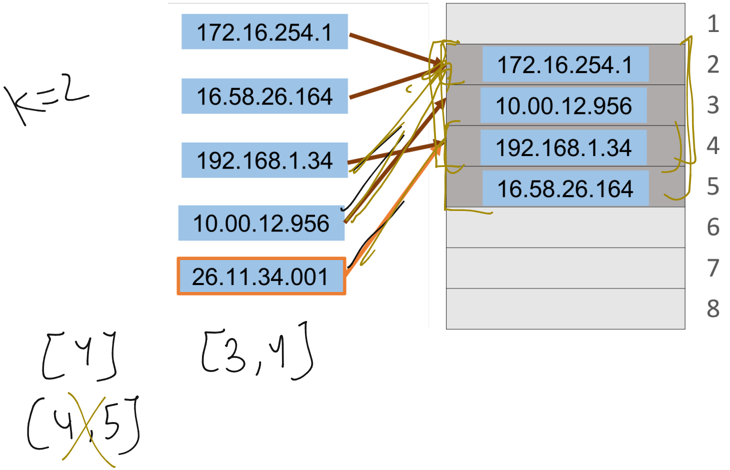
k=3



[2]
[2,3]
[2,4] [2,5]

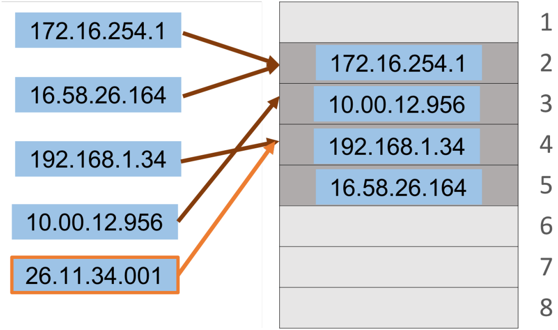
Analysis via Full Intervals

Claim Let $\mathbf{T}(x)$ denote the number of steps required for an insertion/query operation for item x. If $\mathbf{T}(x) > k$, there are at least k full intervals of different lengths containing $\mathbf{h}(x)$.



Analysis via Full Intervals

Claim Let $\mathbf{T}(x)$ denote the number of steps required for an insertion/query operation for item x. If $\mathbf{T}(x) > k$, there are at least k full intervals of different lengths containing $\mathbf{h}(x)$.



Let $\mathbf{I}_j = 1$ if $\mathbf{h}(x)$ lies in some length-j full interval, $\mathbf{I}_j = 0$ otherwise.
Operation time for x is can be bounded as $\mathbf{T}(x) \leq \sum_{j=1}^{n} \mathbf{I}_j$.

Expectation Analysis

$\mathsf{I}_j = 1$ if $\mathsf{h}(x)$ lies in some length-j full interval, $\mathsf{I}_j = 0$ otherwise.
Expected operation time for any x is:

$$\mathbb{E}[\mathsf{T}(x)] \le \sum_{j=1}^{m} \mathbb{E}[\mathsf{I}_j].$$

Expectation Analysis

$\mathsf{I}_j = 1$ if $\mathsf{h}(x)$ lies in some length-j full interval, $\mathsf{I}_j = 0$ otherwise. Expected operation time for any x is:

$$\mathbb{E}[\mathsf{T}(x)] \leq \sum_{j=1}^{n} \mathbb{E}[\mathsf{I}_j].$$

Observe that $\mathsf{h}(x)$ lies in at most 1 length-1 interval, 2 length-2 intervals, etc. So we can upper bound this expectation by:

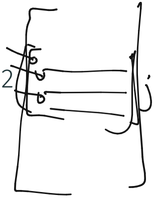
union bound

$$\mathbb{E}[\mathsf{T}(x)] \leq \sum_{j=1}^{n} j \cdot \Pr[\text{any length-}j\text{ interval is full}].$$

Expectation Analysis

$I_j = 1$ if $h(x)$ lies in some length-j full interval, $I_j = 0$ otherwise. Expected operation time for any x is:

$$\mathbb{E}[T(x)] \leq \sum_{j=1}^{n} \mathbb{E}[I_j].$$



Observe that $h(x)$ lies in at most 1 length-1 interval, 2 length-2 intervals, etc. So we can upper bound this expectation by:

$$\mathbb{E}[T(x)] \leq \sum_{j=1}^{n} j \cdot \Pr[\text{any length-}j\text{ interval is full}].$$

A length-j interval is full if the number of items hashed into it, $L(I)$ is at least j. Note that when $m \geq 2n$, $\mathbb{E}[L(I)] = j/2$.

Expectation Analysis

$\mathbf{I}_j = 1$ if $\mathbf{h}(x)$ lies in some length-j full interval, $\mathbf{I}_j = 0$ otherwise. Expected operation time for any x is:

$$\mathbb{E}[\mathbf{T}(x)] \leq \sum_{j=1}^{n} \mathbb{E}[\mathbf{I}_j].$$

Observe that $\mathbf{h}(x)$ lies in at most 1 length-1 interval, 2 length-2 intervals, etc. So we can upper bound this expectation by:

$$\mathbb{E}[\mathbf{T}(x)] \leq \sum_{j=1}^{n} j \cdot \Pr[\text{any length-}j\text{ interval is full}].$$

A length-j interval is full if the number of items hashed into it, $\mathbf{L}(I)$ is at least j. Note that when $m \geq 2n$, $\mathbb{E}[\mathbf{L}(I)] = j/2$. Applying a Chernoff bound with $\delta = 1/2$, $\mu = \mathbb{E}[\mathbf{L}(I)] = j/2$:

if $L(I) \geq j$

$|L(I) - \mu| \geq \frac{j}{2}$

$$\Pr[\mathbf{L}(I) \geq j] \leq \Pr[|\mathbf{L}(I) - \mu| \geq \delta \cdot \mu]$$

$$\leq 2e^{-\frac{(1/2)^2 \cdot j/2}{2+1/2}} \leq 2e^{-\frac{1^2 \cdot j/2}{2+1}} = 2\exp(-j)$$

Expectation Analysis

$\mathbf{I}_j = 1$ if $\mathbf{h}(x)$ lies in some length-j full interval, $\mathbf{I}_j = 0$ otherwise. Expected operation time for any x is:

$$\mathbb{E}[\mathbf{T}(x)] \leq \sum_{j=1}^{n} \mathbb{E}[\mathbf{I}_j].$$

Observe that $\mathbf{h}(x)$ lies in at most 1 length-1 interval, 2 length-2 intervals, etc. So we can upper bound this expectation by:

$$\mathbb{E}[\mathbf{T}(x)] \leq \sum_{j=1}^{n} j \cdot \Pr[\text{any length-}j \text{ interval is full}].$$

A length-j interval is full if the number of items hashed into it, $\mathbf{L}(I)$ is at least j. Note that when $m \geq 2n$, $\mathbb{E}[\mathbf{L}(I)] = j/2$. Applying a Chernoff bound with $\delta = 1/2$, $\mu = \mathbb{E}[\mathbf{L}(I)] = j/2$:

$$\Pr[\mathbf{L}(I) \geq j] \leq \Pr[|\mathbf{L}(I) - \mu| \geq \delta \cdot \mu]$$

$$\leq 2e^{-\frac{(1/2)^2 \cdot j/2}{2+1/2}} = 2e^{-c \cdot j}.$$

Finishing the Analysis

Expected operation time for any x is:

$$\mathbb{E}[\mathsf{T}(x)] \leq \sum_{j=1}^{n} j \cdot \Pr[\text{any length-}j\text{ interval is full}]$$

Finishing the Analysis

Expected operation time for any x is:

$$\begin{aligned}\mathbb{E}[\mathsf{T}(x)] &\leq \sum_{j=1}^{n} j \cdot \Pr[\text{any length-}j\text{ interval is full}] \\ &\leq \underbrace{\sum_{j=1}^{n} j \cdot 2e^{-c \cdot j}}_{O(1)}\end{aligned}$$

Finishing the Analysis

Expected operation time for any x is:

$$\begin{aligned}\mathbb{E}[\mathsf{T}(x)] &\leq \sum_{j=1}^{n} j \cdot \Pr[\text{any length-}j\text{ interval is full}] \\ &\leq \sum_{j=1}^{n} j \cdot 2e^{-c \cdot j} \qquad e^{-c} \\ &= O(1).\end{aligned}$$

Finishing the Analysis

Expected operation time for any x is:

$$\begin{aligned} \mathbb{E}[\mathsf{T}(x)] &\leq \sum_{j=1}^{n} j \cdot \Pr[\text{any length-}j\text{ interval is full}] \\ &\leq \sum_{j=1}^{n} j \cdot 2e^{-c \cdot j} \\ &= O(1). \end{aligned}$$

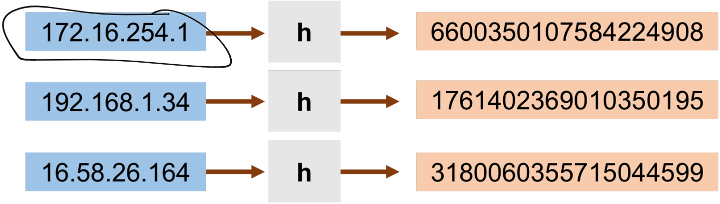
This matches the expected operation cost of chaining when $m \geq 2n$.
In practice, linear probing is typically much faster.

$\Pr(T(x) \geq ck) \leq \exp(-k)$

Random Hashing and Fingerprinting

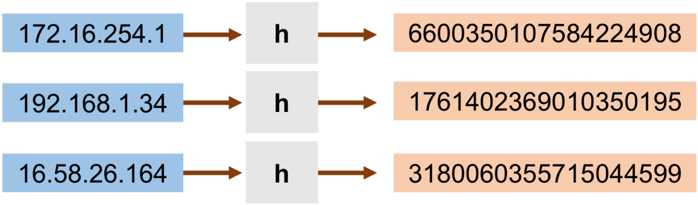
Random Hash Functions

A random hash function maps inputs to random outputs.



Random Hash Functions

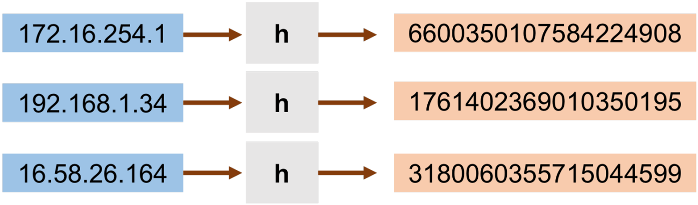
A random hash function maps inputs to random outputs.



h is picked randomly, but after it is picked it is fixed – so a single input is always mapped to the same output.

Random Hash Functions

A random hash function maps inputs to random outputs.



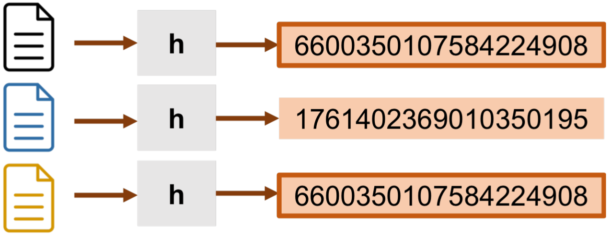
h is picked randomly, but after it is picked it is fixed – so a single input is always mapped to the same output.

```
import random
a = random.randint(1,100)
b = random.randint(1,100)
def myHash(x):
  return (a*x+b) % 100
```

```
import random
def myHash(x):
  a = random.randint(1,100)
  b = random.randint(1,100)
  return (a*x+b) % 100
```

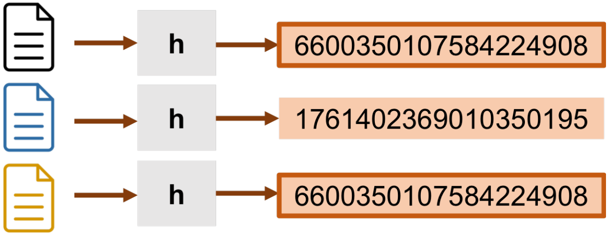
Fingerprinting

Random hash functions are often used to reduce large files down to hash 'fingerprints', which can be used to check equality of files (deduplication), detect updates/corruptions, etc.



Fingerprinting

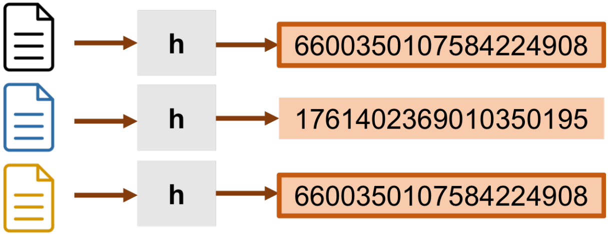
Random hash functions are often used to reduce large files down to hash ‘fingerprints’, which can be used to check equality of files (deduplication), detect updates/corruptions, etc.



- Key requirement is that two distinct files are unlikely to have the same hash – low collision probability.

Fingerprinting

Random hash functions are often used to reduce large files down to hash ‘fingerprints’, which can be used to check equality of files (deduplication), detect updates/corruptions, etc.



- Key requirement is that two distinct files are unlikely to have the same hash – low collision probability.
- In practice *h* is often a deterministic ‘cryptographic’ hash function like SHA or MD5 – hard to analyze formally.

Rabin Fingerprint

Rabin Fingerprint: Interpret a bit string $x_1, x_2, \ldots, x_n$ as the binary representation of the integer $x = \sum_{i=1}^{n} x_i \cdot 2^{i-1}$. Let

$$\mathrm{h}(x) = x \mod p,$$

where p is a randomly chosen prime in $[1, tn \log tn]$.

Rabin Fingerprint: Interpret a bit string $x_1, x_2, \ldots, x_n$ as the binary representation of the integer $x = \sum_{i=1}^{n} x_i \cdot 2^{i-1}$. Let

$$h(x) = x \mod p,$$

where p is a randomly chosen prime in $[1, tn \log tn]$.

Prime Number Theorem: There are $\approx \frac{tn \log tn}{\log(tn \log tn)} = \Theta(tn)$ primes in $[1, tn \log tn]$. So p is chosen randomly from $\Theta(tn)$ possible values.

Rabin Fingerprint

Rabin Fingerprint: Interpret a bit string $x_1, x_2, \ldots, x_n$ as the binary representation of the integer $x = \sum_{i=1}^{n} x_i \cdot 2^{i-1}$. Let

$$\mathbf{h}(x) = x \mod p,$$

where p is a randomly chosen prime in $[1, tn \log tn]$.

Prime Number Theorem: There are $\approx \frac{tn \log tn}{\log(tn \log tn)} = \Theta(tn)$ primes in $[1, tn \log tn]$. So p is chosen randomly from $\Theta(tn)$ possible values.

Claim: For $x, y \in [0, 2^n]$ with $x \neq y$, $\Pr[\mathbf{h}(x) = \mathbf{h}(y))] = O(1/t)$.

Rabin Fingerprint

Rabin Fingerprint: Interpret a bit string $x_1, x_2, \ldots, x_n$ as the binary representation of the integer $x = \sum_{i=1}^{n} x_i \cdot 2^{i-1}$. Let

$$\mathbf{h}(x) = x \mod p,$$

where p is a randomly chosen prime in $[1, tn \log tn]$.

Prime Number Theorem: There are $\approx \frac{tn \log tn}{\log(tn \log tn)} = \Theta(tn)$ primes in $[1, tn \log tn]$. So p is chosen randomly from $\Theta(tn)$ possible values.

Claim: For $x, y \in [0, 2^n]$ with $x \neq y$, $\Pr[\mathbf{h}(x) = \mathbf{h}(y))] = O(1/t)$.

- If $\mathbf{h}(x) = \mathbf{h}(y)$, then it must be that $x - y \mod p = 0$. I.e., p divides $x - y$.

$\Pr(p \text{ divides } x-y) \leq 1/t$

Rabin Fingerprint

Rabin Fingerprint: Interpret a bit string $x_1, x_2, \ldots, x_n$ as the binary representation of the integer $x = \sum_{i=1}^{n} x_i \cdot 2^{i-1}$. Let

$$\mathbf{h}(x) = x \mod p,$$

where p is a randomly chosen prime in $[1, tn \log tn]$.

Prime Number Theorem: There are $\approx \frac{tn \log tn}{\log(tn \log tn)} = \Theta(tn)$ primes in $[1, tn \log tn]$. So p is chosen randomly from $\Theta(tn)$ possible values.

Claim: For $x, y \in [0, 2^n]$ with $x \neq y$, $\Pr[\mathbf{h}(x) = \mathbf{h}(y))] = O(1/t)$.

- If $\mathbf{h}(x) = \mathbf{h}(y)$, then it must be that $x - y \mod p = 0$. I.e., p divides $x - y$.
- $x - y$ is an integer in the range $[-2^n, 2^n]$. What is the probability that p divides $x - y$?

Rabin Fingerprint Analysis

Think-Pair-Share 1: How many unique prime factors can an integer in $[-2^n, 2^n]$ have?

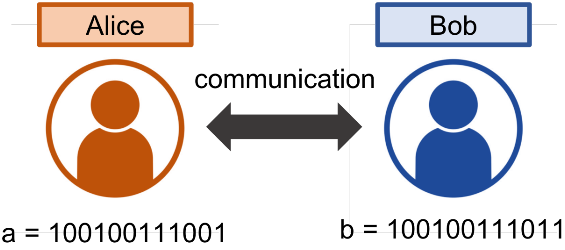
Think-Pair-Share 2: What is the probability that a random prime p chosen from $[1, tn \log tn]$ divides $x - y \in [-2^n, 2^n]$?

Recall: There are $\Theta(tn)$ primes in the range $[1, tn \log tn]$.

Application 1: Communication Complexity

Fingerprinting for Equality Testing

Equality Testing Communication Problem: Alice has some bit string $a \in \{0,1\}^n$. Bob has some string $b \in \{0,1\}$. How many bits do they need to communicate to determine if $a = b$ with probability at least $2/3$?



Fingerprinting for Equality Testing

Equality Testing Protocol:

- Alice picks a random prime $p \in [1, tn \log tn]$ for some large constant t.
- Alice sends p, along with the Rabin fingerprint $\mathsf{h}(a) := a \mod p$ to Bob.
- Bob uses p to compute $\mathsf{h}(b) := b \mod p$.
- If $\mathsf{h}(a) = \mathsf{h}(b)$, Bob sends 'YES' to Alice. Else, he sends 'No'.

Fingerprinting for Equality Testing

Equality Testing Protocol:

- Alice picks a random prime $p \in [1, tn \log tn]$ for some large constant t.
- Alice sends p, along with the Rabin fingerprint $\mathsf{h}(a) := a \mod p$ to Bob.
- Bob uses p to compute $\mathsf{h}(b) := b \mod p$.
- If $\mathsf{h}(a) = \mathsf{h}(b)$, Bob sends 'YES' to Alice. Else, he sends 'No'.

Correctness: If $a = b$ both Alice and Bob always output 'YES'. If $a \neq b$ they output 'NO' with probability $1 - O(1/t) \geq 2/3$ if t is set large enough.

Fingerprinting for Equality Testing

Equality Testing Protocol:

- Alice picks a random prime $p \in [1, tn \log tn]$ for some large constant t.
- Alice sends p, along with the Rabin fingerprint $\mathsf{h}(a) := a \mod p$ to Bob.
- Bob uses p to compute $\mathsf{h}(b) := b \mod p$.
- If $\mathsf{h}(a) = \mathsf{h}(b)$, Bob sends 'YES' to Alice. Else, he sends 'No'.

Correctness: If $a = b$ both Alice and Bob always output 'YES'. If $a \neq b$ they output 'NO' with probability $1 - O(1/t) \geq 2/3$ if t is set large enough.

Complexity: Uses just $O(\log n)$ bits of communication in total.

Fingerprinting for Equality Testing

Equality Testing Protocol:

- Alice picks a random prime $p \in [1, tn \log tn]$ for some large constant t.
- Alice sends p, along with the Rabin fingerprint $\mathbf{h}(a) := a \mod p$ to Bob. [$O(\log p) = O(\log n)$ bits]
- Bob uses p to compute $\mathbf{h}(b) := b \mod p$.
- If $\mathbf{h}(a) = \mathbf{h}(b)$, Bob sends 'YES' to Alice. Else, he sends 'No'. [1 bit]

Correctness: If $a = b$ both Alice and Bob always output 'YES'. If $a \neq b$ they output 'NO' with probability $1 - O(1/t) \geq 2/3$ if t is set large enough.

Complexity: Uses just $O(\log n)$ bits of communication in total.

Deterministic Equality Testing

How many bits must Alice and Bob send if they want to check equality of $a, b \in \{0, 1\}^n$ without using randomness?

Deterministic Equality Testing

How many bits must Alice and Bob send if they want to check equality of $a, b \in \{0, 1\}^n$ without using randomness?

Claim: Any deterministic protocol for equality testing requires sending $\Omega(n)$ bits.

Deterministic Equality Testing

How many bits must Alice and Bob send if they want to check equality of $a, b \in \{0, 1\}^n$ without using randomness?

Claim: Any deterministic protocol for equality testing requires sending $\Omega(n)$ bits.

- An exponential separation between randomized and deterministic protocols!
- Unlike for running times, for communication complexity problems there are often large provable separations between randomized and deterministic protocols.

Deterministic Equality Testing Lower Bound

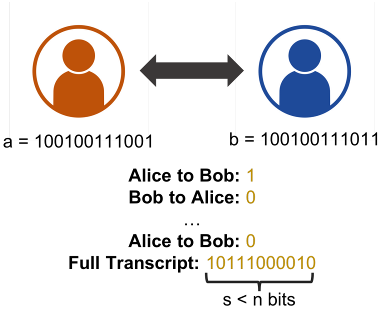
Claim: Any deterministic protocol for equality testing requires sending $\Omega(n)$ bits.

- Assume without loss of generality that Alice and Bob alternate sending 1 bit at a time – at most doubles the number of bits.

Deterministic Equality Testing Lower Bound

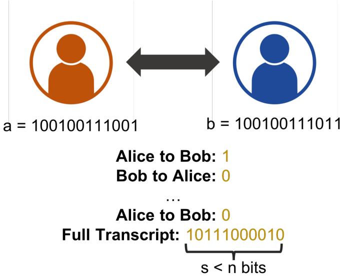
Claim: Any deterministic protocol for equality testing requires sending $\Omega(n)$ bits.

- Assume without loss of generality that Alice and Bob alternate sending 1 bit at a time – at most doubles the number of bits.
- If Alice and Bob send $s < n$ bits, in total, there are 2^s possible conversations they may have.



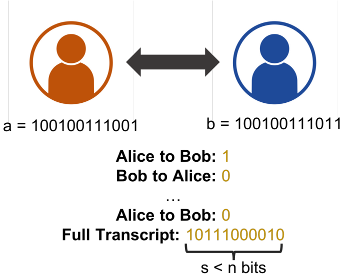
Deterministic Equality Testing Lower Bound

If Alice and Bob send $s < n$ bits, in total, there are 2^s possible conversations they may have.



Deterministic Equality Testing Lower Bound

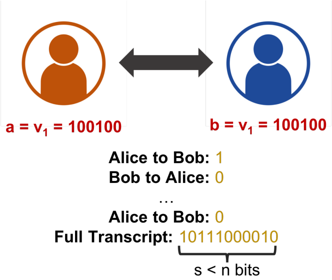
If Alice and Bob send $s < n$ bits, in total, there are 2^s possible conversations they may have.



- Since there are $2^n > 2^s$ possible inputs, there must be two different inputs $v_1 \neq v_2$, such that given $a = b = v_1$ or $a = b = v_2$, the protocol outputs 'YES' and has identical transcripts.

Deterministic Equality Testing Lower Bound

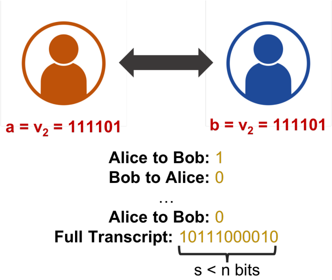
If Alice and Bob send $s < n$ bits, in total, there are 2^s possible conversations they may have.



- Since there are $2^n > 2^s$ possible inputs, there must be two different inputs $v_1 \neq v_2$, such that given $a = b = v_1$ or $a = b = v_2$, the protocol outputs 'YES' and has identical transcripts.

Deterministic Equality Testing Lower Bound

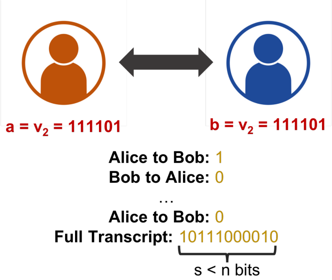
If Alice and Bob send $s < n$ bits, in total, there are 2^s possible conversations they may have.



- Since there are $2^n > 2^s$ possible inputs, there must be two different inputs $v_1 \neq v_2$, such that given $a = b = v_1$ or $a = b = v_2$, the protocol outputs 'YES' and has identical transcripts.

Deterministic Equality Testing Lower Bound

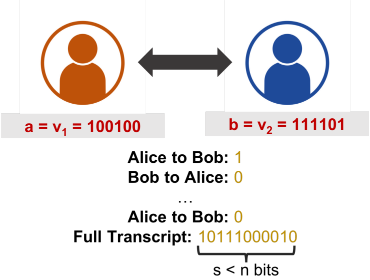
If Alice and Bob send $s < n$ bits, in total, there are 2^s possible conversations they may have.



- Since there are $2^n > 2^s$ possible inputs, there must be two different inputs $v_1 \neq v_2$, such that given $a = b = v_1$ or $a = b = v_2$, the protocol outputs 'YES' and has identical transcripts.
- But then the players will send the same messages and output 'YES' also when Alice is given $a = v_1$ and Bob is given $b = v_2$. This violates correctness!

Deterministic Equality Testing Lower Bound

If Alice and Bob send $s < n$ bits, in total, there are 2^s possible conversations they may have.



- Since there are $2^n > 2^s$ possible inputs, there must be two different inputs $v_1 \neq v_2$, such that given $a = b = v_1$ or $a = b = v_2$, the protocol outputs 'YES' and has identical transcripts.
- But then the players will send the same messages and output 'YES' also when Alice is given $a = v_1$ and Bob is given $b = v_2$. This violates correctness!

Application 2: Pattern Matching

Pattern Matching

Given some document $x = x_1 x_2 \ldots x_n$ and a pattern $y = y_1 y_2 \ldots y_m$, find some j such that

$$x_j x_{j+1}, \ldots, x_{j+m-1} = y_1 y_2 \ldots y_m.$$

x = The quick brown **fox** jumped across the pond…

y = fox

Can assume without loss of generality that the strings are binary strings.

Pattern Matching

Given some document $x = x_1 x_2 \ldots x_n$ and a pattern $y = y_1 y_2 \ldots y_m$, find some j such that

$$x_j x_{j+1}, \ldots, x_{j+m-1} = y_1 y_2 \ldots y_m.$$

x = The quick brown **fox** jumped across the pond…

y = fox

Can assume without loss of generality that the strings are binary strings.

What is the 'naive' running time required to solve this problem?

Rolling Hash

We will use the fact that the Rabin fingerprint is a rolling hash.

Rolling Hash

We will use the fact that the Rabin fingerprint is a rolling hash.

- Letting $X_j = \sum_{i=0}^{m-1} x_{j+i} \cdot 2^{m-1-i}$ be the integer value represented by the binary string $x_j x_{j+1}, \ldots, x_{j+m-1}$, we have

$$X_{j+1} = 2 \cdot X_j - 2^m x_j + x_{j+m}.$$

Rolling Hash

We will use the fact that the Rabin fingerprint is a rolling hash.

- Letting $X_j = \sum_{i=0}^{m-1} x_{j+i} \cdot 2^{m-1-i}$ be the integer value represented by the binary string $x_j x_{j+1}, \ldots, x_{j+m-1}$, we have

$$X_{j+1} = 2 \cdot X_j - 2^m x_j + x_{j+m}.$$

- Thus, since for any X, $\mathbf{h}(X) = X \mod p$,

$$\mathbf{h}(X_{j+1}) = 2 \cdot \mathbf{h}(X_j) - 2^m x_j + x_{j+m} \mod p.$$

- Given $\mathbf{h}(X_j)$, this hash value can be computed using just $O(1)$ arithmetic operations.

Rabin-Karp Algorithm

The Rabin-Karp pattern matching algorithm is then:

- Pick a random prime $p \in [1, ctm \log mt]$, for $t = n^2$.
- Let $Y = \mathsf{h}(y)$ be the Rabin fingerprint of the pattern.
- Let $H = \mathsf{h}(X_1)$ be the Rabin fingerprint of the first block of text.
- For $j = 1, \ldots, x_{n-m+1}$
    - If $Y == H$, return j.
    - Else, $H = \mathsf{h}(X_{j+1}) = 2 \cdot \mathsf{h}(X_j) - 2^m x_j + x_{j+m} \mod p$.

Rabin-Karp Algorithm

The Rabin-Karp pattern matching algorithm is then:

- Pick a random prime $p \in [1, ctm \log mt]$, for $t = n^2$.
- Let $Y = \mathbf{h}(y)$ be the Rabin fingerprint of the pattern.
- Let $H = \mathbf{h}(X_1)$ be the Rabin fingerprint of the first block of text.
- For $j = 1, \ldots, x_{n-m+1}$
  - If $Y == H$, return j.
  - Else, $H = \mathbf{h}(X_{j+1}) = 2 \cdot \mathbf{h}(X_j) - 2^m x_j + x_{j+m} \mod p$.

Runtime: We require $O(m+n)$ time – $O(m)$ for the initial hash computations, and $O(1)$ for each iteration of the for loop.

Correctness: The probability of a false positive at any step is upper bounded by $\frac{1}{t} = \frac{1}{t^2}$, so via a union bound, the probably of a false positive overall is at most $\frac{n}{t^2} = \frac{1}{n}$.

Questions?