CS-GY 6763: Lecture 3
Exponential Concentration Inequalities, Fingerprinting

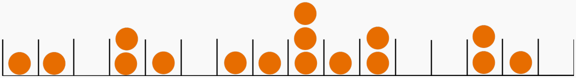
NYU Tandon School of Engineering, Prof. Christopher Musco

Lemma (Chebyshev's Inequality)

Let X be a random variable with expectation $\mathbb{E}[X]$ and variance $\sigma^2 = \mathrm{Var}[X]$. Then for any $k > 0$,

$$\Pr[|X - \mathbb{E}[X]| \geq k \cdot \sigma] \leq \frac{1}{k^2}$$

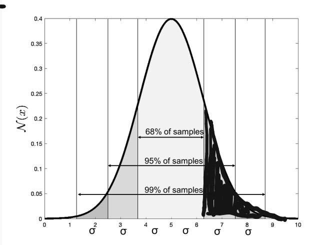
One application: Proved that if you throw n balls into n bins, the maximum loaded bin has $O(\sqrt{n})$ balls. We used Chebyshevs + Union Bound.



This lecture, we'll prove a bound of $O(\log n)$ using stronger tools.

Motivating question: Is Chebyshev's Inequality tight?

It is the worst case, but often not in reality.



68-95-99 rule for Gaussian bell-curve. $X \sim N(0, \sigma^2)$

Chebyshev's Inequality:

$$\Pr(|X - \mathbb{E}[X]| \geq 1\sigma) \leq 100\%$$
$$\Pr(|X - \mathbb{E}[X]| \geq 2\sigma) \leq 25\%$$
$$\Pr(|X - \mathbb{E}[X]| \geq 3\sigma) \leq 11\%$$
$$\Pr(|X - \mathbb{E}[X]| \geq 4\sigma) \leq 6\%.$$

Truth:

$$\Pr(|X - \mathbb{E}[X]| \geq 1\sigma) \approx 32\%$$
$$\Pr(|X - \mathbb{E}[X]| \geq 2\sigma) \approx 5\%$$
$$\Pr(|X - \mathbb{E}[X]| \geq 3\sigma) \approx 1\%$$
$$\Pr(|X - \mathbb{E}[X]| \geq 4\sigma) \approx .01\%$$

For $X \sim \mathcal{N}(\mu, \sigma^2)$:

$$\Pr[X = \mu \pm x] \sim \left(\frac{1}{\sigma\sqrt{2\pi}} e^{-x^2/2\sigma^2}\right)$$

Lemma (Gaussian Tail Bound)

For $X \sim \mathcal{N}(\mu, \sigma^2)$:

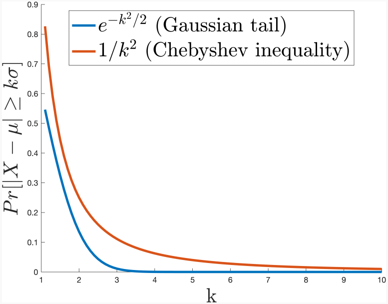
$$\Pr[|X - \mathbb{E}X| \geq k \cdot \sigma] \leq 2e^{-k^2/2}.$$

Compare this to:

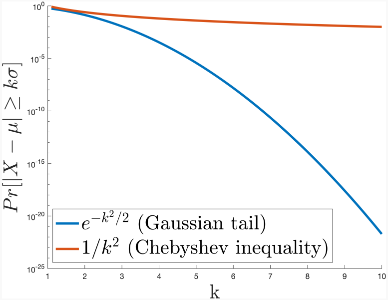
Lemma (Chebyshev's Inequality)

For $X \sim \mathcal{N}(\mu, \sigma^2)$:

$$\Pr[|X - \mathbb{E}X| \geq k \cdot \sigma] \leq \frac{1}{k^2}$$



Standard *y*-scale.



Logarithmic *y*-scale.

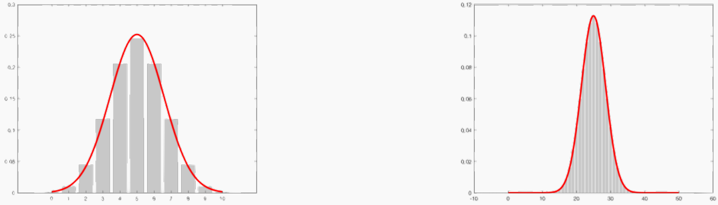
Takeaway: Gaussian random variables concentrate much tighter around their expectation than variance alone (i.e. Chebyshevs's inequality) predicts.

Why does this matter for algorithm design?

Theorem (CLT – Informal)

Any sum of mutually independent, (identically distributed) r.v.'s $X_1, \ldots, X_k$ with mean μ and finite variance σ^2 converges to a Gaussian r.v. with mean $k \cdot \mu$ and variance $k \cdot \sigma^2$, as $k \to \infty$.

$$S = \sum_{i=1}^{k} X_i \Longrightarrow \mathcal{N}(k \cdot \mu, k \cdot \sigma^2).$$



(a) Distribution of # of heads after 10 coin flips, compared to a Gaussian.

(b) Distribution of # of heads after 50 coin flips, compared to a Gaussian.

Recall:

Definition (Mutual Independence)

Random variables $X_1, \ldots, X_k$ are mutually independent if, for all possible values $v_1, \ldots, v_k$,

$$\Pr[X_1 = v_1, \ldots, X_k = v_k] = \Pr[X_1 = v_1] \cdot \ldots \cdot \Pr[X_k = v_k]$$

Strictly stronger than pairwise independence.

$n = 100$ $\quad$ $\mathrm{Var}(X_i] = \mathbb{E}[X_i^2] - \mathbb{E}[X_i]^2$

If I flip a fair coin 100 times, lower bound the chance I get between 30 and 70 heads?

50 $\quad$ $\Pr[|S - \mathbb{E}S| \geq 20] \leq 2e^{-k^2/2}$

For this problem, we will assume the limit of the CLT holds exactly – i.e., that this sum looks exactly like a Gaussian random variable. $\quad$ $20 = k \cdot \sigma$

Lemma (Gaussian Tail Bound)

For $X \sim \mathcal{N}(\mu, \sigma^2)$:

4 $\quad$ $2e^{-4^2/2}$

$$\Pr[|X - \mathbb{E}X| \geq k \cdot \sigma] \leq 2e^{-k^2/2}.$$

$S = \sum_{i=1}^{n} X_i \rightarrow X_i = \mathbb{1}[i\text{th coin is heads}]$ $\quad$ $\mathbb{E}(X_i) = 1/2$

$\mathbb{E}[S] = n/2 = 50 \quad \mathrm{Var}[S] = n/4 = 25 \quad \sigma = 5.$ $\quad$ $\mathrm{Var}(X_i) = 1/4$

$2e^{-8} = .06\%$. Chebyshev's inequality gave a bound of 6.25%.

These back-of-the-envelop calculations can be made rigorous! Lots of different "versions" of bound which do so.

- Chernoff bound
- Bernstein bound
- Hoeffding bound
- ...

Different assumptions on random varibles (e.g. binary vs. bounded), different forms (additive vs. multiplicative error), etc. **Wikipedia is your friend.**

Theorem (Chernoff Bound)

Let $X_1, X_2, \ldots, X_k$ be independent $\{0, 1\}$-valued random variables and let $p_i = \mathbb{E}[X_i]$, where $0 < p_i < 1$. Then the sum $S = \sum_{i=1}^{k} X_i$, which has mean $\mu = \sum_{i=1}^{k} p_i$, satisfies

$$\Pr[S \geq (1+\epsilon)\mu] \leq e^{\frac{-\epsilon^2 \mu}{2+\epsilon}}.$$

and for $0 < \epsilon < 1$

$$\Pr[S \leq (1-\epsilon)\mu] \leq e^{\frac{-\epsilon^2 \mu}{2}}.$$

Theorem (Chernoff Bound Corollary)

Let $X_1, X_2, \ldots, X_k$ be independent $\{0,1\}$-valued random variables and let $p_i = \mathbb{E}[X_i]$, where $0 < p_i < 1$. Let $S = \sum_{i=1}^{k} X_i$ and $\mathbb{E}[S] = \mu$. For $\epsilon \in (0,1)$,

$$\Pr[|S - \mu| \geq \epsilon\mu] \leq 2e^{-\epsilon^2\mu/3}$$

$\sigma = O(\sqrt{\mu})$

$\Pr[|S-\mu| \geq \underbrace{\epsilon\sqrt{\mu}}_{k} \cdot \underbrace{\sqrt{\mu}}_{\sigma}] \leq 2e^{-\epsilon^2\mu/3} = 2e^{-k^2/3}$

$k = \epsilon\sqrt{\mu}$

$k^2 = \epsilon^2\mu$

Why does this look like the Gaussian tail bound of $\Pr[|S - \mu| \geq k \cdot \sigma] \lesssim 2e^{-k^2/2}$? What is $\sigma(S)$?

$\mathrm{Var}[S] = \sum_{i=1}^{k} \mathrm{Var}[X_i] = \sum_{i=1}^{k} \mathbb{E}[X_i^2] - \mathbb{E}[X_i]^2$

$\leq \sum_{i=1}^{k} \mathbb{E}[X_i^2] = \sum_{i=1}^{k} \mathbb{E}[X_i] = \mu.$

$\mathrm{Var}(S) = O(\mu) = \mu$

Theorem (Bernstein Inequality)

Let $X_1, X_2, \ldots, X_k$ be independent random variables with each $X_i \in [-1, 1]$. Let $\mu_i = \mathbb{E}[X_i]$ and $\sigma_i^2 = \mathrm{Var}[X_i]$. Let $\mu = \sum_i \mu_i$ and $\sigma^2 = \sum_i \sigma_i^2$. Then, for $k \leq \frac{1}{2}\sigma$, $S = \sum_i X_i$ satisfies

$$\Pr[|S - \mu| > k \cdot \sigma] \leq 2e^{-k^2/4}.$$

QUANTITATIVE VERSIONS OF THE CLT

Theorem (Hoeffding Inequality)

Let $X_1, X_2, \ldots, X_k$ be independent random variables with each $X_i \in [a_i, b_i]$. Let $\mu_i = \mathbb{E}[X_i]$ and $\mu = \sum_i \mu_i$. Then, for any $k > 0$, $S = \sum_i X_i$ satisfies:

$$\Pr[|S - \mu| > k] \leq 2e^{-\frac{k^2}{\underbrace{\sum_{i=1}^{k}(b_i - a_i)^2}_{O(\sigma^2)}}}.$$

$[a_i \quad b_i]$

Variance is a natural measure of central tendency, but there are others.

$b = 4$

$$q^{\text{th}} \text{ central moment: } \mathbb{E}[(X - \mathbb{E}X)^q]$$

$q = 2$ gives the variance. Proof of Chebyshev's applies Markov's inequality to the random variable $(X - \mathbb{E}X)^2$).

Idea in brief: Apply Markov's inequality to $\mathbb{E}[(X - \mathbb{E}X)^q]$ for larger q, or more generally to $f(X - \mathbb{E}X)$ for some other non-negative function f. E.g., to $\exp(X - \mathbb{E}X)$.

If I flip a fair coin 100 times, lower bound the chance I get between 30 and 70 heads?

heads

$= 50$

Corollary of Chernoff bound: Let $S = \sum_{i=1}^{k} X_i$ and $\mu = \mathbb{E}[S]$. For $0 < \epsilon < 1$,

$$\Pr[|S - \mu| \geq \epsilon\mu] \leq 2e^{-\epsilon^2\mu/3}$$

50

$2e^{-.4^2 50/3}$

Here $X_i = \mathbb{1}[i^{\text{th}}$ flip is heads].

$\epsilon = \frac{2}{5} = .4$

$\epsilon\mu = \frac{2}{5} \cdot 50 = 20.$

6.25%

1.4%.

.06%

General Statement: Flip biased coin k times: i.e. the coin is heads with probability b. As long as $k \geq O\left(\frac{\log(1/\delta)}{\epsilon^2}\right)$,

$$\Pr[|\#\text{ heads} - b \cdot k| \geq \epsilon k] \leq \delta$$

$\mu = bk$

From chernoff, for any $\varepsilon' \in (0,1)$ $\longrightarrow \mu \cdot \varepsilon'$ $\varepsilon' = \varepsilon/b$

$$\Pr[|\#\text{heads} - b \cdot k| \geq \varepsilon' bk] \leq 2e^{-\varepsilon'^2 bk/3}$$

Set $\varepsilon' = \varepsilon/b$. * need $\varepsilon \leq 1/b$.

$$\Pr[|\#\text{heads} - bk| \geq \varepsilon k] \leq 2e^{-\varepsilon^2 k/3b} \leq 2e^{-\varepsilon^2 k/3}.$$

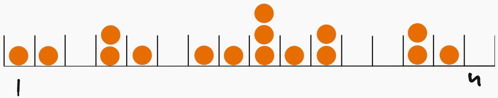
setting $k = O\left(\frac{\log(1/\delta)}{\varepsilon^2}\right)$ gives $2e^{-\varepsilon^2 k/3} \leq \delta$

Pay very little for higher probability – if you increase the number of coin flips by 4x, δ goes from $1/10 \rightarrow 1/100 \rightarrow 1/10000$

LOAD BALANCING

Recall: n jobs are distributed randomly to n servers using a hash function. Let S_i be the number of jobs sent to server i. What's the smallest B for which we can prove:

$$\Pr[\max_i S_i \geq \mathrm{B}] \leq 1/10$$



Recall: Suffices to prove that, for any i, $\Pr[S_i \geq \mathrm{B}] \leq 1/10n$:

$$\Pr[\max_i S_i \geq \mathrm{B}] = \Pr[S_1 \geq \mathrm{B} \text{ or } \ldots \text{ or } S_1 \geq \mathrm{B}]$$

$$\leq \Pr[S_1 \geq \mathrm{B}] + \ldots + \Pr[S_n \geq \mathrm{B}] \quad \text{(union bound).}$$

Theorem (Chernoff Bound) $2 + c\log(n) \le 2c\log(n)$

Let $X_1, X_2, \ldots, X_n$ be independent $\{0,1\}$-valued random variables and let $p_i = \mathbb{E}[X_i]$, where $0 < p_i < 1$. Then the sum $S = \sum_{j=1}^{n} X_i$, which has mean $\mu = \sum_{j=1}^{n} p_i$, satisfies

$$\Pr[S \ge (1+\epsilon)\mu] \le e^{\frac{-\epsilon^2 \cancel{\mu}}{2+\epsilon}}.$$

$S_i \le O(\log(n))$

$\mu = 1$

Consider a single bin. Let $X_j = \mathbb{1}[\text{ball } j \text{ lands in } \cancel{\text{that}} \text{ bin } i]$.

$S_i = \sum_{j=1}^{n} X_j \quad \mathbb{E}[X_j] = 1/n \quad \mathbb{E}[S_i] = \sum_{j=1}^{n} \mathbb{E}[X_i] = \sum_{j=1}^{n} 1/n = 1$

$$\Pr[S_i \ge (1 + c\log n)\overset{1}{\mu}] \le e^{\frac{-c^2\log^2 n}{2+c\log n}} \le e^{\frac{-c^2\log^2 n}{2c\log n}} \le e^{-.5c\log n} \le \frac{1}{10n},$$

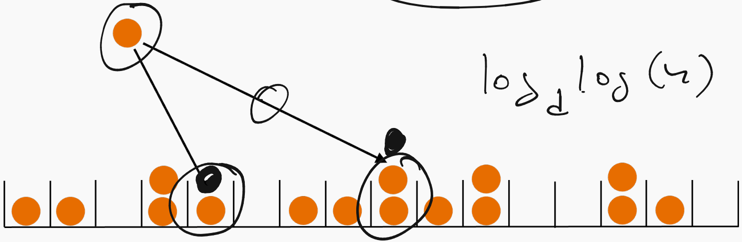
for sufficiently large c

$e^{-2\log(n)} = \frac{1}{n^2} \le \frac{1}{10n}$

So max load for randomized load balancing is $O(\log n)$! Best we could prove with Chebyshev's was $O(\sqrt{n})$.

Power of 2 Choices: Instead of assigning job to random server, choose 2 random servers and assign to the least loaded. With probability 1/10 the maximum load is bounded by:

(a) $O(\log n)$ (b) $O(\sqrt{\log n})$ (c) $O(\log \log n)$ (d) $O(1)$



$\log_d \log(n)$

BREAK

Recall from last class: $h(x) \sim \text{Unif}(1, \ldots, m)$

Definition (Universal hash function)

A random hash function $h : \mathcal{U} \to \{1, \ldots, m\}$ is universal if, for any fixed $x, y \in \mathcal{U}$,

$$\Pr[h(x) = h(y)] \leq \frac{1}{m}. \quad = \frac{1}{m}$$

Efficient construction: Let p be a prime number between $|\mathcal{U}|$ and $2|\mathcal{U}|$. Let a, b be random numbers in $0, \ldots, p$, $a \neq 0$.

$$h(x) = [a \cdot x + b \pmod{p}] \pmod{m}$$

is universal.

We're not going to prove this, but this year I want to give a flavor for what tools are used.

One of the most famous applications of randomness in algorithm design.

Computational Problem: Given a number x, is x prime?

Recall:

- A number is prime if it can only be divided evenly by 1 and itself.
- The first few primes are $2, 3, 5, 7, 11, 13, 17, 19, \ldots$.
- Is 2023 prime? ✗
- What about 49301977064557719291?

n: input length

2^n $\sqrt{2^n} = 2^{O(n)}$

How would you design a generic algorithm to check?

Suppose we have an integer represented as a length n binary string.

$$x = 0110100010110101\ldots1010001110$$

The naive prime checking algorithm runs in $O(2^n)$ time.

NYU Greene Super Computer has 2 petaFLOPS of throughput. When $n = 128$, would need 1 million Greene Super computers running for 1 million years to check is x is prime.

RANDOMIZED PRIMALITY TEST

$2^{O(n)}$ $\quad$ $2^{O(n^{1/4})}$

(**Miller-Rabin 1976, 1980:**) There is a randomized algorithm running in $O(n^3 \log_2(1/\delta))$ time that, with probability $1 - \delta$ determines if an n-bit integer x is prime.

- $n = 128$
- $\delta = 10^{-9}$ (chance of winning the Powerball Jackpot)
- $n^3 \log_2(1/\delta) \approx 60$ million operations.

Could check in $< .1$ second on my laptop.

This was a really big break through!

RANDOMIZED PRIMALITY TEST

Took over 20 more years to find a deterministic polynomial time primality test.

PRIMES is in P

Manindra Agrawal Neeraj Kayal
Nitin Saxena*

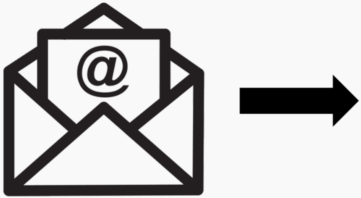
Department of Computer Science & Engineering
Indian Institute of Technology Kanpur
Kanpur-208016, INDIA
Email: {manindra,kayaln,nitinsa}@iitk.ac.in

Abstract

We present an unconditional deterministic polynomial-time algorithm that determines whether an input number is prime or composite.

Basis for modern public-key cryptography.

Goal: Bob wants to send Alice an email. Wants to encrypt it in some way so that even if it is intercepted, no one can read it besides Alice.



```
fsxtwdrweodautbeiqxnhmwtnhzutzgf
orzjfbryvokrtggbyeofchfyafekoyia
gcdfndzqyunfwgwfivfzvzlrehcgxctj
pcerkwdeysaruspsutecqvrbcetoiocg
oqnajpsslcjdlcwbdcaxcglbtevfzhiu
eutxubpmqnmxylonkaplmmqcvblbfltl
dbjasqnsavwlgbazbmkbphezgeavtmet
desrgtomtjrsxsrlshikycbpmhvncuhm
ipzzwehmjnmriareccuxfqhjjezdsuqt
zhnwmyummpzimvydkvscsbtjrciquyfn
```

WHY DO PRIMES MATTER?

Basis for modern public-key cryptography.

Goal: Bob wants to send Alice an email. Wants to **encrypt** it in some way so that even if it is intercepted, no one can read it besides Alice.

Option 1: Share some sort of secret key/codebook in advance.

Impractical if you have a large number of uncoordinated senders and receivers.

Option 2: Create a 1-way lock box.

Anyone can deliver, only Alice can open/read the messages.

RSA crytosystem (Rivest, Shamir, Adleman 1977):

- **Private key:** Two large (e.g. 128 bit) prime numbers p, q.
- **Public key:** Based on $z = p \times q$.

Even though checking if a number of prime can be done quickly, we do not have efficient algorithms for factoring numbers. E.g. for finding p, q based on z.[1]

[1] At least on classical computers we don't... different story on quantum computers.

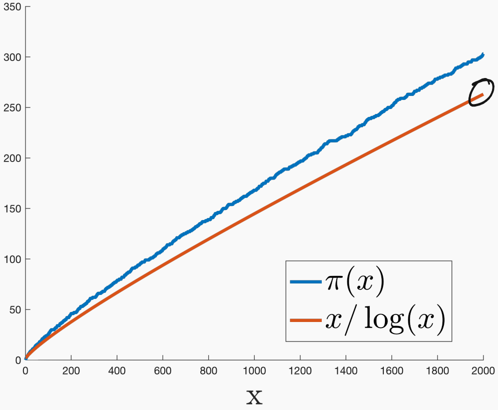
How to find a 128 bit prime number p? Use randomness, twice.

- Pick a random 128 bit number.
- Check if it's prime (using randomized primality test).
- If not, repeat.

Roughly how many tries do you expect this to take?

Let $\pi(x)$ denote the number of primes less than some integer x. **Informally:**

$$\pi(x) \sim \frac{x}{\log(x)}$$



$$\frac{x/\log(x)}{x} = \frac{1}{\log(x)}$$

Formally: For $x > 17$,

$$\frac{x}{\log(x)} \leq \pi(x) \leq \frac{x}{\log(x) - 4}$$

So if we select a random 128 bit number p, the chance that it is prime is great than:

$$\frac{1}{\log(2^{128})} \geq \frac{1}{90} = O\left(\frac{1}{128}\right)$$

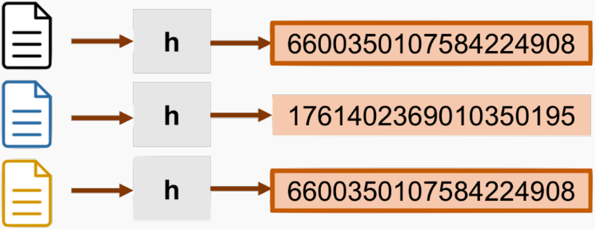
After a few hundred tries, we will almost definitely find a prime number. **In general, need $O(n)$ tries to find a prime with n bits.**

Finding large prime numbers is also a critical step in constructing efficiently computable universal hash.

Remainder of lecture: Discuss a simple but really important application of prime numbers to hashing.

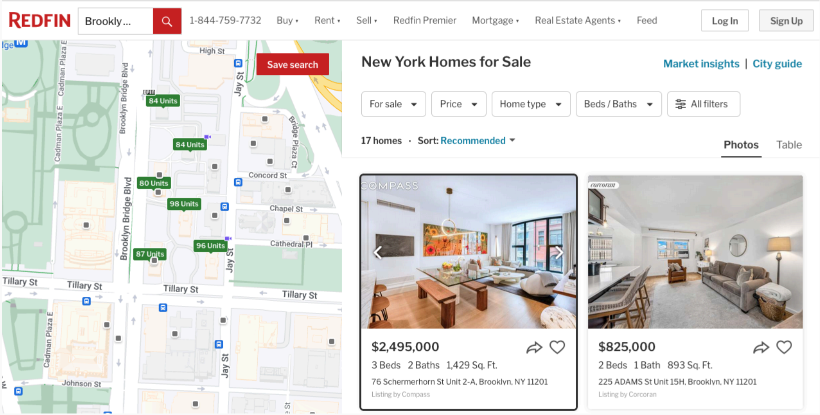
Goal: Construct a compact "fingerprint" $h(f)$ for any given file f with two properties:

- The fingerprints $h(f_1)$ and $h(f_2)$ should be different with high probability if the contents of f_1 and f_2 differ at all.
- If the contents of f_1 and f_2 are identical, we should have $h(f_1) = h(f_2)$.

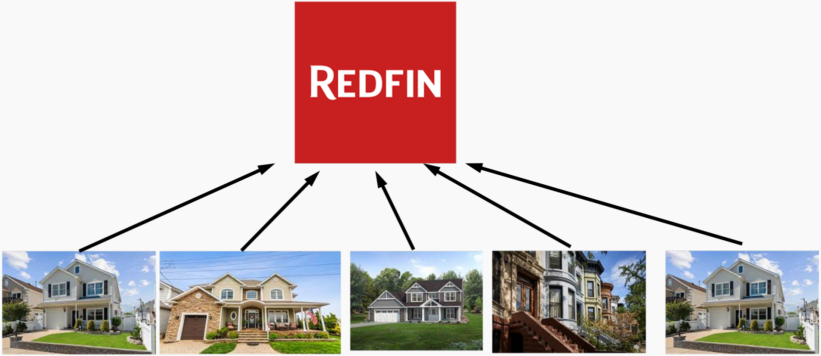


- Quickly check if two versions of the same file are identical (e.g. in version control systems like Git). Do not need to communicate the entire file between servers. Also used in webcaching and content delivery networks.
- Check that a file pieced together from multiple parts is not missing anything.

APPLICATIONS OF FINGER PRINTING



APPLICATIONS OF FINGER PRINTING



Images from databases of local real estate agencies.

Fingerprints used as file names for the images to make sure we did not reupload new images that we already had, and to detect duplicate images and listings.

Goal: Construct a compact "fingerprint" function $h(f)$ such that:

- $h(f_1) \neq h(f_2)$ if $f_1 \neq f_2$ with high probability.

Ideally, length of $h(f_1)$ (i.e. the size of the integers hashed to) is much less than the file size.

(**Rabin Fingerprint**) **(1981):** Let file $f = 010\ldots1101$ of length n be interpreted as an n bit integer. So something between 0 and 2^n.

Construct h **randomly:** Choose random prime number p between 2 and $tn\log(tn)$ for a constant t.

$$h(f) = f \pmod{p}.$$

$O(\log t + \log n)$

How many bits does $h(f)$ take to store?

$\log(p)$ bits to store

$\leq \log(tn\log(tn)) \leq \log((tn)^2) = O(\log(tn))$

$$h(f) = f \pmod{p} \quad \text{for prime } p \in \{2, \ldots, tn\log(tn)\}$$

Claim: If $f_1 \neq f_2$ then $h(f_1) = h(f_2)$ with probability $\leq \frac{2}{t}$.

Since our fingerprint only takes $O(\log n + \log t)$ space, we can set t to be super large, so effectively the probability of $h(f_1)$ and $h(f_2)$ colliding is negligible for all real-world applications.

E.g. set fingerprint length to $\log n + 28$ bits and you are more likely to win the Powerball.

$$h(f) = f \pmod{p} \quad \text{for prime } p \in \{2, \ldots, tn \log(tn)\}$$

Claim: If $f_1 \neq f_2$ then $h(f_1) = h(f_2)$ with probability just $\frac{2}{t}$.

$f_1 \pmod{p} = f_2 \pmod{p}$

First observation: If $h(f_1) = h(f_2)$, then:

$$(f_1 - f_2) \pmod{p} = 0.$$

In other words, we only fail if the $f_1 - f_2$ is divisible by p.

Question: What is the chance that $f_1 - f_2$, which is an integer less than 2^n, is divisible by a random prime $p \in \{2, \ldots, tn\log(tn)\}$?

$\rightarrow I$

Number of distinct prime factors of $f_1 - f_2$: At most n.

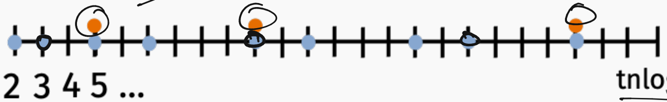
$I = 2 \cdot 3^2 \cdot 5 \cdot 7^7 \cdot \ldots \quad I \geq 2^{\#\text{ of prime factors}}$

Number of primes between $\{2, \ldots, tn\log(tn)\}$: At least $\frac{tn\log(tn)}{\log(tn\log(tn))}$ via prime number theorem.

= prime number

= prime factors of f_1-f_2

at most n.



$tn\log(tn) = X$

$\leq \log((tn)^2) = 2\log(tn)$

Chance we pick a prime factor of $f_1 - f_2$ is less than:

$$\frac{\cancel{n}}{\frac{t\cancel{n}\log(tn)}{\log(tn\log(tn))}} = \frac{\log(tn\log(tn))}{t\log(tn)} \leq \frac{2\cancel{\log(tn)}}{t\cancel{\log(tn)}} = \frac{2}{t}$$

Conclusion: The chance that a random prime $p \in \{2, \ldots, tn\log(tn)\}$ is a factor of $f_1 - f_2$ is $\leq \frac{2}{t}$.

So, for two files $f_1 \neq f_2$, the chance that $h(f_1) = h(f_2) \leq \frac{2}{t}$.

Set $t = 10^{18}$ (the chance you win the Powerball twice in a row).

Fingerprint size: At most $2\log_2(nt) = 2\log(n) + 2\log_2(10^{18})$ bits.

Suppose we are fingerprinting 1mb image files. $n \approx 8 \cdot 10^6$, so our fingerprint has size:

166 bits

This amounts to a 50,000x reduction over sending and comparing the original files.