CS 473: Algorithms

Chandra Chekuri   Ruta Mehta

University of Illinois, Urbana-Champaign

Fall 2016

Fingerprinting

Lecture 11

September 28, 2016

Fingerprinting

Source: Wikipedia

Process of mapping a large data item to a much shorter bit string, called its fingerprint.

Fingerprints uniquely identifies data *for all practical purposes*.

Fingerprinting

Source: Wikipedia

Process of mapping a large data item to a much shorter bit string, called its fingerprint.

Fingerprints uniquely identifies data *for all practical purposes*.

Typically used to avoid comparison and transmission of bulky data.

Eg: Web browser can store/fetch file fingerprints to check if it is changed.

Fingerprinting
Source: Wikipedia

Process of mapping a large data item to a much shorter bit string, called its fingerprint.

Fingerprints uniquely identifies data *for all practical purposes*.

Typically used to avoid comparison and transmission of bulky data.

Eg: Web browser can store/fetch file fingerprints to check if it is changed.

As you may have guessed, fingerprint functions are hash functions.

Bloom Filters

Hashing:

1. To insert $\mathbf{x}$ in dictionary store $\mathbf{x}$ in table in location $\mathbf{h(x)}$
2. To lookup $\mathbf{y}$ in dictionary check contents of location $\mathbf{h(y)}$

Bloom Filters

Hashing:

1. To insert $\mathbf{x}$ in dictionary store $\mathbf{x}$ in table in location $\mathbf{h(x)}$
2. To lookup $\mathbf{y}$ in dictionary check contents of location $\mathbf{h(y)}$

Bloom Filter: tradeoff space for false positives

1. Storing items in dictionary expensive in terms of memory, especially if items are unwieldy objects such a long strings, images, etc with *non-uniform* sizes.
2. To insert $\mathbf{x}$ in dictionary set *bit* to $\mathbf{1}$ in location $\mathbf{h(x)}$ (initially all bits are set to $\mathbf{0}$)
3. To lookup $\mathbf{y}$ if bit in location $\mathbf{h(y)}$ is $\mathbf{1}$ say yes, else no.

Bloom Filters

Bloom Filters

Bloom Filter: tradeoff space for false positives

1. To insert $\mathbf{x}$ in dictionary set *bit* to $\mathbf{1}$ in location $\mathbf{h(x)}$ (initially all bits are set to $\mathbf{0}$)
2. To lookup $\mathbf{y}$ if bit in location $\mathbf{h(y)}$ is $\mathbf{1}$ say yes, else no
3. No false negatives but false positives possible due to collisions

Bloom Filters

Bloom Filter: tradeoff space for false positives

1. To insert $\mathbf{x}$ in dictionary set *bit* to $\mathbf{1}$ in location $\mathbf{h(x)}$ (initially all bits are set to $\mathbf{0}$)
2. To lookup $\mathbf{y}$ if bit in location $\mathbf{h(y)}$ is $\mathbf{1}$ say yes, else no
3. No false negatives but false positives possible due to collisions

Reducing false positives:

1. Pick $\mathbf{k}$ hash functions $\mathbf{h_1, h_2, \ldots, h_k}$ *independently*
2. To insert $\mathbf{x}$ for $\mathbf{1 \leq i \leq k}$ set bit in location $\mathbf{h_i(x)}$ in table $\mathbf{i}$ to $\mathbf{1}$
3. To lookup $\mathbf{y}$ compute $\mathbf{h_i(y)}$ for $\mathbf{1 \leq i \leq k}$ and say yes only if each bit in the corresponding location is $\mathbf{1}$, otherwise say no. If probability of false positive for one hash function is $\alpha < 1$ then with $\mathbf{k}$ independent hash function it is

Bloom Filters

Bloom Filter: tradeoff space for false positives

1. To insert $\mathbf{x}$ in dictionary set *bit* to $\mathbf{1}$ in location $\mathbf{h(x)}$ (initially all bits are set to $\mathbf{0}$)
2. To lookup $\mathbf{y}$ if bit in location $\mathbf{h(y)}$ is $\mathbf{1}$ say yes, else no
3. No false negatives but false positives possible due to collisions

Reducing false positives:

1. Pick $\mathbf{k}$ hash functions $\mathbf{h_1, h_2, \ldots, h_k}$ *independently*
2. To insert $\mathbf{x}$ for $\mathbf{1 \leq i \leq k}$ set bit in location $\mathbf{h_i(x)}$ in table $\mathbf{i}$ to $\mathbf{1}$
3. To lookup $\mathbf{y}$ compute $\mathbf{h_i(y)}$ for $\mathbf{1 \leq i \leq k}$ and say yes only if each bit in the corresponding location is $\mathbf{1}$, otherwise say no. If probability of false positive for one hash function is $\alpha < 1$ then with $\mathbf{k}$ independent hash function it is $\alpha^{\mathbf{k}}$.

Outline

Use of hash functions for designing fast algorithms

Problem

Given a text $\mathbf{T}$ of length $\mathbf{m}$ and pattern $\mathbf{P}$ of length $\mathbf{n}$, $\mathbf{m} \gg \mathbf{n}$, find all occurrences of $\mathbf{P}$ in $\mathbf{T}$.

Outline

Use of hash functions for designing fast algorithms

Problem

Given a text $\mathbf{T}$ of length $\mathbf{m}$ and pattern $\mathbf{P}$ of length $\mathbf{n}$, $\mathbf{m} \gg \mathbf{n}$, find all occurrences of $\mathbf{P}$ in $\mathbf{T}$.

Karp-Rabin Randomized Algorithm

Outline

Use of hash functions for designing fast algorithms

Problem

Given a text $\mathbf{T}$ of length $\mathbf{m}$ and pattern $\mathbf{P}$ of length $\mathbf{n}$, $\mathbf{m} \gg \mathbf{n}$, find all occurrences of $\mathbf{P}$ in $\mathbf{T}$.

Karp-Rabin Randomized Algorithm

- Sampling a prime
- String equality via $\mathbf{mod\ p}$ arithmetic
- Rabin's fingerprinting scheme – rolling hash
- Karp-Rabin pattern matching algorithm: $\mathbf{O(m+n)}$ time.

Sampling a prime

Problem

Given an integer $x > 0$, sample a prime uniformly at random from all the primes between 1 and x.

Sampling a prime

Problem

Given an integer $x > 0$, sample a prime uniformly at random from all the primes between 1 and x.

Procedure

1. Sample a number p uniformly at random from $\{1, \ldots, x\}$.
2. If p is a prime, then output p. Else go to Step (1).

Sampling a prime

Problem

Given an integer $x > 0$, sample a prime uniformly at random from all the primes between 1 and x.

Procedure

1. Sample a number p uniformly at random from $\{1, \ldots, x\}$.
2. If p is a prime, then output p. Else go to Step (1).

Checking if p is prime

- Agrawal-Kayal-Saxena primality test: deterministic but slow
- Miller-Rabin randomized primality test: fast but randomized

  outputs 'prime' when it is not *with very low probability*.

Sampling a Prime: Analysis

Is the returned prime *sampled uniformly at random*?

Sampling a Prime: Analysis

Is the returned prime *sampled uniformly at random*?

$\pi(x)$: number of primes in $\{1, \ldots, x\}$,

Lemma

For a fixed prime $p^* \leq x$, $\Pr[\text{algorithm outputs } p^*] = 1/\pi(x)$.

Sampling a Prime: Analysis

Is the returned prime *sampled uniformly at random*?

$\pi(x)$: number of primes in $\{1, \ldots, x\}$,

Lemma

For a fixed prime $p^* \leq x$, $\Pr[$*algorithm outputs* $p^*] = 1/\pi(x)$.

Proof.

A : Event that a prime is picked in a round. $\Pr[A] = \pi(x)/x$.

Sampling a Prime: Analysis

Is the returned prime *sampled uniformly at random*?

$\pi(x)$: number of primes in $\{1, \ldots, x\}$,

Lemma

For a fixed prime $p^* \leq x$, $\Pr[\textit{algorithm outputs } p^*] = 1/\pi(x)$.

Proof.

A : Event that a prime is picked in a round. $\Pr[A] = \pi(x)/x$.

B : Number (prime) p^* is picked. $\Pr[B] = 1/x$. $B \subset A$.

Sampling a Prime: Analysis

Is the returned prime *sampled uniformly at random*?

$\pi(\mathbf{x})$ **:** number of primes in $\{1, \ldots, x\}$,

Lemma

For a fixed prime $\mathbf{p}^* \leq \mathbf{x}$, $\mathbf{Pr}[$*algorithm outputs* $\mathbf{p}^*] = 1/\pi(\mathbf{x})$.

Proof.

A : Event that a prime is picked in a round. $\mathbf{Pr}[A] = \pi(x)/x$.

B : Number (prime) $\mathbf{p}^*$ is picked. $\mathbf{Pr}[B] = 1/x$. $B \subset A$.

$$\mathbf{Pr}[B|A] = \frac{\mathbf{Pr}[A \cap B]}{\mathbf{Pr}[A]} = \frac{\mathbf{Pr}[B]}{\mathbf{Pr}[A]} = \frac{1/x}{\pi(x)/x} = \frac{1}{\pi(x)}$$

□

Sampling a Prime: Analysis

Is the returned prime *sampled uniformly at random*?

$\pi(x)$: number of primes in $\{1, \ldots, x\}$,

Lemma

For a fixed prime $p^* \leq x$, $\Pr[\textit{algorithm outputs } p^*] = 1/\pi(x)$.

Proof.

A : Event that a prime is picked in a round. $\Pr[A] = \pi(x)/x$.

B : Number (prime) p^* is picked. $\Pr[B] = 1/x$. $B \subset A$.

$$\Pr[B|A] = \frac{\Pr[A \cap B]}{\Pr[A]} = \frac{\Pr[B]}{\Pr[A]} = \frac{1/x}{\pi(x)/x} = \frac{1}{\pi(x)} \qquad \square$$

Running time in expectation

Q: How many samples in expectation before termination?

A: $x/\pi(x)$. Exercise.

How many primes between 0 and x

$\pi(x)$ **:** Number of primes between 0 and x.

Prime Number Theorem

$$\lim_{x \to \infty} \frac{\pi(x)}{x / \ln x} = 1$$

By Jacques Hadamard and Charles Jean de la Vallée-Poussin in 1896

How many primes between 0 and x

$\pi(x)$ **:** Number of primes between $\mathbf{0}$ and $\mathbf{x}$.

Prime Number Theorem

$$\lim_{x \to \infty} \frac{\pi(x)}{x/\ln x} = 1$$

By Jacques Hadamard and Charles Jean de la Vallée-Poussin in 1896

Chebyshev (from 1848)

$$\pi(x) \geq \frac{7}{8} \frac{x}{\ln x} = (1.262..) \frac{x}{\lg x} > \frac{x}{\lg x}$$

How many primes between 0 and x

$\pi(x)$ **:** Number of primes between 0 and x.

Prime Number Theorem

$$\lim_{x\to\infty} \frac{\pi(x)}{x/\ln x} = 1$$

By Jacques Hadamard and Charles Jean de la Vallée-Poussin in 1896

Chebyshev (from 1848)

$$\pi(x) \geq \frac{7}{8}\frac{x}{\ln x} = (1.262..)\frac{x}{\lg x} > \frac{x}{\lg x}$$

- $y \sim \{1, \ldots, x\}$ u.a.r., then y is a prime w.p. $\frac{\pi(x)}{x} > \frac{1}{\lg x}$.

How many primes between 0 and x

$\pi(x)$ **:** Number of primes between 0 and x.

Prime Number Theorem

$$\lim_{x\to\infty} \frac{\pi(x)}{x/\ln x} = 1$$

By Jacques Hadamard and Charles Jean de la Vallée-Poussin in 1896

Chebyshev (from 1848)

$$\pi(x) \geq \frac{7}{8}\frac{x}{\ln x} = (1.262..)\frac{x}{\lg x} > \frac{x}{\lg x}$$

- $y \sim \{1, \ldots, x\}$ u.a.r., then y is a prime w.p. $\frac{\pi(x)}{x} > \frac{1}{\lg x}$.
- If we want $k \geq 4$ primes then $x \geq 2k \lg k$ suffices.

$$\pi(x) \geq \pi(2k \lg k) = \frac{k(2 \lg k)}{\lg 2 + \lg k + \lg \lg k} \geq k$$

String Equality

Problem

Alice, the captain of a Mars lander, receives an N-bit string $\mathbf{x}$, and Bob, back at mission control, receives a string $\mathbf{y}$. They know nothing about each others strings, but want to check if $\mathbf{x} = \mathbf{y}$.

String Equality

Problem

Alice, the captain of a Mars lander, receives an N-bit string $\mathbf{x}$, and Bob, back at mission control, receives a string $\mathbf{y}$. They know nothing about each others strings, but want to check if $\mathbf{x} = \mathbf{y}$.

Alice sends Bob $\mathbf{x}$, and Bob confirms if $\mathbf{x} = \mathbf{y}$. But sending $\mathbf{N}$ bits is costly! *Can they share less communication and check equality?*

String Equality

Problem

Alice, the captain of a Mars lander, receives an N-bit string $\mathbf{x}$, and Bob, back at mission control, receives a string $\mathbf{y}$. They know nothing about each others strings, but want to check if $\mathbf{x} = \mathbf{y}$.

Alice sends Bob $\mathbf{x}$, and Bob confirms if $\mathbf{x} = \mathbf{y}$. But sending $\mathbf{N}$ bits is costly! *Can they share less communication and check equality?*

Possibilities:

- If want 100% surety then NO.
- If OK with 99.99% surety then $\mathbf{O(\lg N)}$ may suffice!!!

String Equality

Problem

Alice, the captain of a Mars lander, receives an N-bit string $\mathbf{x}$, and Bob, back at mission control, receives a string $\mathbf{y}$. They know nothing about each others strings, but want to check if $\mathbf{x} = \mathbf{y}$.

Alice sends Bob $\mathbf{x}$, and Bob confirms if $\mathbf{x} = \mathbf{y}$. But sending $\mathbf{N}$ bits is costly! *Can they share less communication and check equality?*

Possibilities:

- If want 100% surety then NO.
- If OK with 99.99% surety then $\mathbf{O(\lg N)}$ may suffice!!!
  - If $\mathbf{x} = \mathbf{y}$, then $\mathbf{Pr}[\text{Bob says } \textit{equal}] = \mathbf{1}$.
  - If $\mathbf{x} \neq \mathbf{y}$, then $\mathbf{Pr}[\text{Bob says } \textit{un-equal}] = \mathbf{0.9999}$.

HOW?

String Equality: Randomized Algorithm

(Recall) $\mathbf{5N}$ primes in $\{1, \ldots, M\}$ if $M = \lceil 2(5N) \lg 5N \rceil$.

String Equality: Randomized Algorithm

(Recall) $\mathbf{5N}$ primes in $\{1, \ldots, M\}$ if $M = \lceil 2(5N) \lg 5N \rceil$.

Define $h_p(x) = x \mod p$

1. Alice picks a random prime p from $\{1, \ldots M\}$.

String Equality: Randomized Algorithm

(Recall) $5N$ primes in $\{1, \ldots, M\}$ if $M = \lceil 2(5N) \lg 5N \rceil$.

$$\text{Define } h_p(x) = x \mod p$$

1. Alice picks a random prime p from $\{1, \ldots M\}$.
2. She sends Bob prime p, and also $h_p(x) = x \mod p$.
3. Bob checks if $h_p(y) = h_p(x)$. If so, he says *equal* else *un-equal*.

String Equality: Randomized Algorithm

(Recall) $\mathbf{5N}$ primes in $\{1, \ldots, M\}$ if $M = \lceil 2(5N) \lg 5N \rceil$.

Define $h_p(x) = x \mod p$

1. Alice picks a random prime p from $\{1, \ldots M\}$.
2. She sends Bob prime p, and also $h_p(x) = x \mod p$.
3. Bob checks if $h_p(y) = h_p(x)$. If so, he says *equal* else *un-equal*.

Lemma

If $x = y$ *then Bob always says* equal.

String Equality: Randomized Algorithm

(Recall) $\mathbf{5N}$ primes in $\{1, \ldots, M\}$ if $M = \lceil 2(5N) \lg 5N \rceil$.

Define $h_p(x) = x \mod p$

1. Alice picks a random prime p from $\{1, \ldots M\}$.
2. She sends Bob prime p, and also $h_p(x) = x \mod p$.
3. Bob checks if $h_p(y) = h_p(x)$. If so, he says *equal* else *un-equal*.

Lemma

If $x = y$ *then Bob always says* equal.

Lemma

If $x \neq y$ *then,* $\Pr[$*Bob says* equal$] \leq 1/5$ *(error probability).*

String Equality: Randomized Algorithm

Error probability

Let $\mathbf{M} = \lceil 2(\mathbf{sN}) \lg \mathbf{sN} \rceil$ and $\mathbf{h_p(x) = x \mod p}$

Lemma

If $\mathbf{x} \neq \mathbf{y}$ *then,* $\mathbf{Pr}[$*Bob says* equal$] = \mathbf{Pr}[\mathbf{h_p(x) = h_p(y)}] \leq \mathbf{1/s}$

Proof.

Given $\mathbf{x} \neq \mathbf{y}$, $\mathbf{h_p(x) = h_p(y)} \Rightarrow \mathbf{x \mod p = y \mod p}$.

String Equality: Randomized Algorithm

Error probability

Let $\mathbf{M} = \lceil 2(\mathbf{sN}) \lg \mathbf{sN} \rceil$ and $\mathbf{h_p(x) = x \mod p}$

Lemma

If $\mathbf{x} \neq \mathbf{y}$ *then,* $\mathbf{Pr}[$*Bob says* equal$] = \mathbf{Pr}[\mathbf{h_p(x) = h_p(y)}] \leq \mathbf{1/s}$

Proof.

Given $\mathbf{x} \neq \mathbf{y}$, $\mathbf{h_p(x) = h_p(y)} \Rightarrow \mathbf{x \mod p = y \mod p}$.

- $\mathbf{D} = |\mathbf{x} - \mathbf{y}|$, then $\mathbf{D \mod p = 0}$, and $\mathbf{D} \leq \mathbf{2^N}$.

String Equality: Randomized Algorithm

Error probability

Let $\mathbf{M} = \lceil 2(\mathbf{sN}) \lg \mathbf{sN} \rceil$ and $\mathbf{h_p(x) = x \mod p}$

Lemma

If $\mathbf{x} \neq \mathbf{y}$ *then,* $\mathbf{Pr}[$*Bob says* equal$] = \mathbf{Pr}[\mathbf{h_p(x) = h_p(y)}] \leq \mathbf{1/s}$

Proof.

Given $\mathbf{x} \neq \mathbf{y}$, $\mathbf{h_p(x) = h_p(y) \Rightarrow x \mod p = y \mod p}$.

- $\mathbf{D} = |\mathbf{x} - \mathbf{y}|$, then $\mathbf{D \mod p = 0}$, and $\mathbf{D} \leq \mathbf{2^N}$.
- $\mathbf{D = p_1 \ldots p_k}$ prime factorization. All $\mathbf{p_i} \geq \mathbf{2} \Rightarrow \mathbf{D} \geq \mathbf{2^k}$.

String Equality: Randomized Algorithm

Error probability

Let $M = \lceil 2(sN) \lg sN \rceil$ and $h_p(x) = x \mod p$

Lemma

If $x \neq y$ *then,* $\Pr[$*Bob says* equal$] = \Pr[h_p(x) = h_p(y)] \leq 1/s$

Proof.

Given $x \neq y$, $h_p(x) = h_p(y) \Rightarrow x \mod p = y \mod p$.

- $D = |x - y|$, then $D \mod p = 0$, and $D \leq 2^N$.
- $D = p_1 \ldots p_k$ prime factorization. All $p_i \geq 2 \Rightarrow D \geq 2^k$.
- $2^k \leq D \leq 2^N \Rightarrow k \leq N$. D has at most N divisors.

String Equality: Randomized Algorithm

Error probability

Let $\mathbf{M} = \lceil 2(\mathbf{sN}) \lg \mathbf{sN} \rceil$ and $\mathbf{h_p(x) = x \mod p}$

Lemma

If $\mathbf{x \neq y}$ *then,* $\mathbf{Pr}[$*Bob says* equal$] = \mathbf{Pr[h_p(x) = h_p(y)] \leq 1/s}$

Proof.

Given $\mathbf{x \neq y}$, $\mathbf{h_p(x) = h_p(y) \Rightarrow x \mod p = y \mod p}$.

- $\mathbf{D = |x - y|}$, then $\mathbf{D \mod p = 0}$, and $\mathbf{D \leq 2^N}$.
- $\mathbf{D = p_1 \ldots p_k}$ prime factorization. All $\mathbf{p_i \geq 2 \Rightarrow D \geq 2^k}$.
- $\mathbf{2^k \leq D \leq 2^N \Rightarrow k \leq N}$. $\mathbf{D}$ has at most $\mathbf{N}$ divisors.
- Probability that a random prime $\mathbf{p}$ from $\mathbf{\{1, \ldots, M\}}$ is a divisor,

$$\leq \frac{\mathbf{N}}{\pi(\mathbf{M})}$$

String Equality: Randomized Algorithm

Error probability

Let $\mathbf{M} = \lceil 2(\mathbf{sN}) \lg \mathbf{sN} \rceil$ and $\mathbf{h_p(x) = x \mod p}$

Lemma

If $\mathbf{x \neq y}$ *then,* $\mathbf{Pr}[$*Bob says* equal$] = \mathbf{Pr[h_p(x) = h_p(y)] \leq 1/s}$

Proof.

Given $\mathbf{x \neq y}$, $\mathbf{h_p(x) = h_p(y) \Rightarrow x \mod p = y \mod p}$.

- $\mathbf{D = |x - y|}$, then $\mathbf{D \mod p = 0}$, and $\mathbf{D \leq 2^N}$.
- $\mathbf{D = p_1 \ldots p_k}$ prime factorization. All $\mathbf{p_i \geq 2 \Rightarrow D \geq 2^k}$.
- $\mathbf{2^k \leq D \leq 2^N \Rightarrow k \leq N}$. $\mathbf{D}$ has at most $\mathbf{N}$ divisors.
- Probability that a random prime $\mathbf{p}$ from $\mathbf{\{1, \ldots, M\}}$ is a divisor,

$$\leq \frac{\mathbf{N}}{\pi(\mathbf{M})} \leq \frac{\mathbf{N}}{\mathbf{M}/\lg \mathbf{M}} = \frac{\mathbf{N}}{2(\mathbf{sN})\lg \mathbf{sN}} \lg \mathbf{M} \leq \frac{1}{\mathbf{s}}$$

Low Error Probability and Communication.

Low Error Probability

1. Choose large enough $\mathbf{s}$. Error prob: $\mathbf{1/s}$.
2. Alice repeats the process $\mathbf{R}$ times, and Bob says *equal* only if he gets equal all $\mathbf{R}$ times.

Low Error Probability and Communication.

Low Error Probability

1. Choose large enough $\mathbf{s}$. Error prob: $\mathbf{1/s}$.
2. Alice repeats the process $\mathbf{R}$ times, and Bob says *equal* only if he gets equal all $\mathbf{R}$ times.
   Error probability: $\frac{\mathbf{1}}{\mathbf{s^R}}$.

Low Error Probability and Communication.

Low Error Probability

1. Choose large enough $\mathbf{s}$. Error prob: $\mathbf{1/s}$.
2. Alice repeats the process $\mathbf{R}$ times, and Bob says *equal* only if he gets equal all $\mathbf{R}$ times.

   Error probability: $\frac{1}{s^R}$. For $\mathbf{s = 5, R = 10}$, $\frac{1}{5^{10}} \le \mathbf{0.000001}$.

Low Error Probability and Communication.

Low Error Probability

1. Choose large enough $\mathbf{s}$. Error prob: $\mathbf{1/s}$.
2. Alice repeats the process $\mathbf{R}$ times, and Bob says *equal* only if he gets equal all $\mathbf{R}$ times.
   Error probability: $\frac{1}{s^R}$. For $\mathbf{s = 5, R = 10}$, $\frac{1}{5^{10}} \leq \mathbf{0.000001}$.

$$M = \lceil 2(sN) \lg sN \rceil$$

Amount of Communication

Each round sends 2 integers $\leq \mathbf{M}$. # bits $\mathbf{2 \lg M \leq 4(\lg s + \lg N)}$.

Low Error Probability and Communication.

Low Error Probability

1. Choose large enough $\mathbf{s}$. Error prob: $\mathbf{1/s}$.
2. Alice repeats the process $\mathbf{R}$ times, and Bob says *equal* only if he gets equal all $\mathbf{R}$ times.
   Error probability: $\frac{1}{s^R}$. For $s = 5, R = 10$, $\frac{1}{5^{10}} \leq 0.000001$.

$$M = \lceil 2(sN) \lg sN \rceil$$

Amount of Communication

Each round sends 2 integers $\leq \mathbf{M}$. # bits $2 \lg M \leq 4(\lg s + \lg N)$.

If $\mathbf{x}$ and $\mathbf{y}$ are copies of Wikipedia, about 25 billion characters. If 8 bits per character, then $N \approx 2^{38}$ bits.

Low Error Probability and Communication.

Low Error Probability

1. Choose large enough $\mathbf{s}$. Error prob: $\mathbf{1/s}$.
2. Alice repeats the process $\mathbf{R}$ times, and Bob says *equal* only if he gets equal all $\mathbf{R}$ times.
   Error probability: $\frac{1}{s^R}$. For $\mathbf{s = 5, R = 10}$, $\frac{1}{5^{10}} \le \mathbf{0.000001}$.

$$\mathbf{M = \lceil 2(sN) \lg sN \rceil}$$

Amount of Communication

Each round sends 2 integers $\le \mathbf{M}$. # bits $\mathbf{2 \lg M \le 4(\lg s + \lg N)}$.

If $\mathbf{x}$ and $\mathbf{y}$ are copies of Wikipedia, about 25 billion characters. If 8 bits per character, then $\mathbf{N \approx 2^{38}}$ bits.

Second approach will send $\mathbf{10(2 \lg 10N \lg 5N) \le 1280}$ bits.

Part I

Karp-Rabin Pattern Matching Algorithm

Pattern Matching

Given a string $\mathbf{T}$ of length $\mathbf{m}$ and pattern $\mathbf{P}$ of length $\mathbf{n}$, s.t. $\mathbf{m} \gg \mathbf{n}$, find all occurrences of $\mathbf{P}$ in $\mathbf{T}$.

Example

$\mathbf{T}$=abracadabra, $\mathbf{P}$=ab.

Pattern Matching

Given a string $\mathbf{T}$ of length $\mathbf{m}$ and pattern $\mathbf{P}$ of length $\mathbf{n}$, s.t. $\mathbf{m} \gg \mathbf{n}$, find all occurrences of $\mathbf{P}$ in $\mathbf{T}$.

Example

$\mathbf{T}$=abracadabra, $\mathbf{P}$=ab.

Solution $\mathbf{S} = \{1, 8\}$.

Pattern Matching

Given a string $\mathbf{T}$ of length $\mathbf{m}$ and pattern $\mathbf{P}$ of length $\mathbf{n}$, s.t. $\mathbf{m} \gg \mathbf{n}$, find all occurrences of $\mathbf{P}$ in $\mathbf{T}$.

Example

$\mathbf{T}$=abracadabra, $\mathbf{P}$=ab.

Solution $\mathbf{S} = \{1, 8\}$.

For $\mathbf{j} > \mathbf{i}$, let $\mathbf{T_{i \dots j} = T[i]T[i+1] \dots T[j]}$.

Pattern Matching

Given a string $\mathbf{T}$ of length $\mathbf{m}$ and pattern $\mathbf{P}$ of length $\mathbf{n}$, s.t. $\mathbf{m} \gg \mathbf{n}$, find all occurrences of $\mathbf{P}$ in $\mathbf{T}$.

Example

$\mathbf{T}$=abracadabra, $\mathbf{P}$=ab.

Solution $\mathbf{S} = \{1, 8\}$.

For $\mathbf{j} > \mathbf{i}$, let $\mathbf{T_{i \ldots j}} = \mathbf{T[i]T[i+1] \ldots T[j]}$.

Brute force algorithm

$\mathbf{S} = \emptyset$. For each $\mathbf{i} = \mathbf{1 \ldots m - n + 1}$

- If $\mathbf{T_{i \ldots i+n-1}} = \mathbf{P}$ then $\mathbf{S} = \mathbf{S} \cup \{\mathbf{i}\}$.

Pattern Matching

Given a string $\mathbf{T}$ of length $\mathbf{m}$ and pattern $\mathbf{P}$ of length $\mathbf{n}$, s.t. $\mathbf{m} \gg \mathbf{n}$, find all occurrences of $\mathbf{P}$ in $\mathbf{T}$.

Example

$\mathbf{T}$=abracadabra, $\mathbf{P}$=ab.

Solution $\mathbf{S} = \{1, 8\}$.

For $\mathbf{j} > \mathbf{i}$, let $\mathbf{T_{i\ldots j} = T[i]T[i+1]\ldots T[j]}$.

Brute force algorithm

$\mathbf{S} = \emptyset$. For each $\mathbf{i = 1 \ldots m - n + 1}$

- If $\mathbf{T_{i\ldots i+n-1} = P}$ then $\mathbf{S = S \cup \{i\}}$.

$\mathbf{O(mn)}$ run-time.

Using Hash Function

Pick a prime $\mathbf{p}$ u.a.r. from $\{1, \ldots, M\}$. $\mathbf{h_p(x) = x \bmod p}$.

Brute force algorithm using hash function

$\mathbf{S} = \emptyset$. For each $\mathbf{i = 1 \ldots m - n + 1}$

- If $\mathbf{h_p(T_{i \ldots i+n-1}) = h_p(P)}$ then $\mathbf{S} = \mathbf{S} \cup \{\mathbf{i}\}$.

Using Hash Function

Pick a prime $\mathbf{p}$ u.a.r. from $\{1, \ldots, M\}$. $h_p(x) = x \bmod p$.

Brute force algorithm using hash function

$S = \emptyset$. For each $i = 1 \ldots m - n + 1$

- If $h_p(T_{i \ldots i+n-1}) = h_p(P)$ then $S = S \cup \{i\}$.

If x is of length n, then computing $h_p(x)$ takes $O(n)$ running time.

Overall $O(mn)$ running time.

Using Hash Function

Pick a prime $\mathbf{p}$ u.a.r. from $\{1, \ldots, M\}$. $\mathbf{h_p(x) = x \bmod p}$.

Brute force algorithm using hash function

$\mathbf{S} = \emptyset$. For each $\mathbf{i = 1 \ldots m - n + 1}$

- If $\mathbf{h_p(T_{i \ldots i+n-1}) = h_p(P)}$ then $\mathbf{S = S \cup \{i\}}$.

If $\mathbf{x}$ is of length $\mathbf{n}$, then computing $\mathbf{h_p(x)}$ takes $\mathbf{O(n)}$ running time.

Overall $\mathbf{O(mn)}$ running time.

Can we compute $\mathbf{h_p(T_{i+1 \ldots i+n})}$ using $\mathbf{h_p(T_{i \ldots i+n-1})}$ fast?

Rolling Hash

$\mathbf{x} = \mathbf{T}_{\mathbf{i} \ldots \mathbf{i}+\mathbf{n}-1}$ and $\mathbf{x}' = \mathbf{T}_{\mathbf{i}+1 \ldots \mathbf{i}+\mathbf{n}}$.

Example

$\mathbf{x} = \mathbf{1011001}$, and $\mathbf{x}' = \mathbf{0110010}$ (or $\mathbf{x}' = \mathbf{0110011}$).

Rolling Hash

$\mathbf{x} = \mathbf{T}_{\mathbf{i}\ldots\mathbf{i}+\mathbf{n}-1}$ and $\mathbf{x}' = \mathbf{T}_{\mathbf{i}+1\ldots\mathbf{i}+\mathbf{n}}$.

Example

$\mathbf{x} = \mathbf{1011001}$, and $\mathbf{x}' = \mathbf{0110010}$ (or $\mathbf{x}' = \mathbf{0110011}$).

$$\mathbf{x}' = 2(\mathbf{x} - \mathbf{x}_{\mathbf{hb}} 2^{\mathbf{n}-1}) + \mathbf{x}'_{\mathbf{lb}}$$

Rolling Hash

$x = T_{i \ldots i+n-1}$ and $x' = T_{i+1 \ldots i+n}$.

Example

$x = 1011001$, and $x' = 0110010$ (or $x' = 0110011$).

$$x' = 2(x - x_{hb} 2^{n-1}) + x'_{lb}$$

$$\begin{aligned} h_p(x') &= x' \bmod p \\ &= (2(x \bmod p) - x_{hb}(2^n \bmod p) + x'_{lb}) \bmod p \\ &= (2h_p(x) - x_{hb} h_p(2^n) + x'_{lb}) \bmod p \end{aligned}$$

Karp-Rabin Algorithm

$\mathbf{p}$ **:** a random prime from $\{1, \ldots, M\}$.

1. Set $S = \emptyset$. Compute $h_p(T_{1\ldots n})$, $h_p(2^n)$, and $h_p(P)$.
2. For each $i = 1, \ldots, m - n + 1$
    1. If $h_p(T_{i\ldots i+n-1}) = h_p(P)$, then $S = S \cup \{i\}$.
    2. Compute $h_p(T_{i+1\ldots i+n})$ using $h_p(T_{i\ldots i+n-1})$ and $h_p(2^n)$ by applying rolling hash.

Karp-Rabin Algorithm

$\mathbf{p}$ **:** a random prime from $\{1, \ldots, M\}$.

1. Set $S = \emptyset$. Compute $h_p(T_{1\ldots n})$, $h_p(2^n)$, and $h_p(P)$.
2. For each $i = 1, \ldots, m - n + 1$
    1. If $h_p(T_{i\ldots i+n-1}) = h_p(P)$, then $S = S \cup \{i\}$.
    2. Compute $h_p(T_{i+1\ldots i+n})$ using $h_p(T_{i\ldots i+n-1})$ and $h_p(2^n)$ by applying rolling hash.

Running Time

- In Step 1, computing $h_p(x)$ for an n bit x is in $O(n)$ time.

Karp-Rabin Algorithm

$\mathbf{p}$: a random prime from $\{1, \ldots, M\}$.

1. Set $S = \emptyset$. Compute $h_p(T_{1 \ldots n})$, $h_p(2^n)$, and $h_p(P)$.
2. For each $i = 1, \ldots, m - n + 1$
    1. If $h_p(T_{i \ldots i+n-1}) = h_p(P)$, then $S = S \cup \{i\}$.
    2. Compute $h_p(T_{i+1 \ldots i+n})$ using $h_p(T_{i \ldots i+n-1})$ and $h_p(2^n)$ by applying rolling hash.

Running Time

- In Step 1, computing $h_p(x)$ for an n bit x is in $O(n)$ time.

Assuming $O(\lg M)$ bit arithmetic can be done in $O(1)$ time,

- Since $h_p(.)$ produces $\lg M$ bit numbers, both steps inside **for loop** can be done in $O(1)$ time.

Karp-Rabin Algorithm

$\mathbf{p}$: a random prime from $\{1, \ldots, M\}$.

1. Set $S = \emptyset$. Compute $h_p(T_{1 \ldots n})$, $h_p(2^n)$, and $h_p(P)$.
2. For each $i = 1, \ldots, m - n + 1$
    1. If $h_p(T_{i \ldots i+n-1}) = h_p(P)$, then $S = S \cup \{i\}$.
    2. Compute $h_p(T_{i+1 \ldots i+n})$ using $h_p(T_{i \ldots i+n-1})$ and $h_p(2^n)$ by applying rolling hash.

Running Time

- In Step 1, computing $h_p(x)$ for an n bit x is in $O(n)$ time.

Assuming $O(\lg M)$ bit arithmetic can be done in $O(1)$ time,

- Since $h_p(.)$ produces $\lg M$ bit numbers, both steps inside **for loop** can be done in $O(1)$ time.
- Overall $O(m + n)$ time.

Karp-Rabin Algorithm

$\mathbf{p}$: a random prime from $\{1, \ldots, \mathbf{M}\}$.

1. Set $\mathbf{S} = \emptyset$. Compute $\mathbf{h_p(T_{1\ldots n})}$, $\mathbf{h_p(2^n)}$, and $\mathbf{h_p(P)}$.
2. For each $\mathbf{i = 1, \ldots, m - n + 1}$
    1. If $\mathbf{h_p(T_{i\ldots i+n-1}) = h_p(P)}$, then $\mathbf{S = S \cup \{i\}}$.
    2. Compute $\mathbf{h_p(T_{i+1\ldots i+n})}$ using $\mathbf{h_p(T_{i\ldots i+n-1})}$ and $\mathbf{h_p(2^n)}$ by applying rolling hash.

Running Time

- In Step 1, computing $\mathbf{h_p(x)}$ for an $\mathbf{n}$ bit $\mathbf{x}$ is in $\mathbf{O(n)}$ time.

Assuming $\mathbf{O(\lg M)}$ bit arithmetic can be done in $\mathbf{O(1)}$ time,

- Since $\mathbf{h_p(.)}$ produces $\mathbf{\lg M}$ bit numbers, both steps inside **for loop** can be done in $\mathbf{O(1)}$ time.
- Overall $\mathbf{O(m + n)}$ time. Can't do better.

Karp-Rabin Algorithm: Error Probability

1. Set $\mathbf{S} = \emptyset$. Compute $\mathbf{h_p(T_{1\ldots n})}$, $\mathbf{h_p(2^n)}$, and $\mathbf{h_p(P)}$.
2. For each $\mathbf{i = 1, \ldots, m - n + 1}$
   1. If $\mathbf{h_p(T_{i\ldots i+n-1}) = h_p(P)}$, then $\mathbf{S = S \cup \{i\}}$.
   2. Compute $\mathbf{h_p(T_{i+1\ldots i+n})}$ using $\mathbf{h_p(T_{i\ldots i+n-1})}$ and $\mathbf{h_p(2^n)}$.

Lemma

If match at any position $\mathbf{i}$ *then* $\mathbf{i} \in \mathbf{S}$*. In otherwords if* $\mathbf{T_{i\ldots i+n-1} = P}$*, then* $\mathbf{i} \in \mathbf{S}$*.*

All matched positions are in $\mathbf{S}$.

Karp-Rabin Algorithm: Error Probability

1. Set $\mathbf{S} = \emptyset$. Compute $\mathbf{h_p(T_{1\ldots n})}$, $\mathbf{h_p(2^n)}$, and $\mathbf{h_p(P)}$.
2. For each $\mathbf{i = 1, \ldots, m - n + 1}$
   1. If $\mathbf{h_p(T_{i\ldots i+n-1}) = h_p(P)}$, then $\mathbf{S = S} \cup \{\mathbf{i}\}$.
   2. Compute $\mathbf{h_p(T_{i+1\ldots i+n})}$ using $\mathbf{h_p(T_{i\ldots i+n-1})}$ and $\mathbf{h_p(2^n)}$.

Lemma

If match at any position $\mathbf{i}$ *then* $\mathbf{i} \in \mathbf{S}$. *In otherwords if* $\mathbf{T_{i\ldots i+n-1} = P}$, *then* $\mathbf{i} \in \mathbf{S}$.

All matched positions are in $\mathbf{S}$.

Can it contain unmatched positions?

Karp-Rabin Algorithm: Error Probability

1. Set $\mathbf{S} = \emptyset$. Compute $\mathbf{h_p(T_{1\ldots n})}$, $\mathbf{h_p(2^n)}$, and $\mathbf{h_p(P)}$.
2. For each $\mathbf{i = 1, \ldots, m - n + 1}$
   1. If $\mathbf{h_p(T_{i\ldots i+n-1}) = h_p(P)}$, then $\mathbf{S = S \cup \{i\}}$.
   2. Compute $\mathbf{h_p(T_{i+1\ldots i+n})}$ using $\mathbf{h_p(T_{i\ldots i+n-1})}$ and $\mathbf{h_p(2^n)}$.

Lemma

If match at any position $\mathbf{i}$ *then* $\mathbf{i} \in \mathbf{S}$. *In otherwords if* $\mathbf{T_{i\ldots i+n-1} = P}$, *then* $\mathbf{i} \in \mathbf{S}$.

All matched positions are in $\mathbf{S}$.

Can it contain unmatched positions? YES!

Karp-Rabin Algorithm: Error Probability

1. Set $\mathbf{S} = \emptyset$. Compute $\mathbf{h_p(T_{1 \dots n})}$, $\mathbf{h_p(2^n)}$, and $\mathbf{h_p(P)}$.
2. For each $\mathbf{i = 1, \dots, m - n + 1}$
   1. If $\mathbf{h_p(T_{i \dots i+n-1}) = h_p(P)}$, then $\mathbf{S = S} \cup \{\mathbf{i}\}$.
   2. Compute $\mathbf{h_p(T_{i+1 \dots i+n})}$ using $\mathbf{h_p(T_{i \dots i+n-1})}$ and $\mathbf{h_p(2^n)}$.

Lemma

If match at any position $\mathbf{i}$ *then* $\mathbf{i} \in \mathbf{S}$. *In otherwords if* $\mathbf{T_{i \dots i+n-1} = P}$, *then* $\mathbf{i} \in \mathbf{S}$.

All matched positions are in $\mathbf{S}$.

Can it contain unmatched positions? YES! With what probability?

Karp-Rabin Algorithm: Error Probability

Pr[S contains an index i, while there is no match at i]

1. Set $\mathbf{S} = \emptyset$. Compute $\mathbf{h_p(T_{1\ldots n})}$, $\mathbf{h_p(2^n)}$, and $\mathbf{h_p(P)}$.
2. For each $\mathbf{i = 1, \ldots, m - n + 1}$
   1. If $\mathbf{h_p(T_{i\ldots i+n-1}) = h_p(P)}$, then $\mathbf{S = S \cup \{i\}}$.
   2. Compute $\mathbf{h_p(T_{i+1\ldots i+n})}$ using $\mathbf{h_p(T_{i\ldots i+n-1})}$ and $\mathbf{h_p(2^n)}$.

Karp-Rabin Algorithm: Error Probability

Pr[S contains an index i, while there is no match at i]

1. Set $\mathbf{S = \emptyset}$. Compute $\mathbf{h_p(T_{1\ldots n})}$, $\mathbf{h_p(2^n)}$, and $\mathbf{h_p(P)}$.
2. For each $\mathbf{i = 1, \ldots, m - n + 1}$
   1. If $\mathbf{h_p(T_{i\ldots i+n-1}) = h_p(P)}$, then $\mathbf{S = S \cup \{i\}}$.
   2. Compute $\mathbf{h_p(T_{i+1\ldots i+n})}$ using $\mathbf{h_p(T_{i\ldots i+n-1})}$ and $\mathbf{h_p(2^n)}$.

Set $\mathbf{M = \lceil 2(sn) \lg sn \rceil}$. Given $\mathbf{x \neq y}$, $\mathbf{Pr[h_p(x) = h_p(y)] \leq 1/s}$.

Karp-Rabin Algorithm: Error Probability

Pr[S contains an index i, while there is no match at i]

1. Set $\mathbf{S = \emptyset}$. Compute $\mathbf{h_p(T_{1\ldots n})}$, $\mathbf{h_p(2^n)}$, and $\mathbf{h_p(P)}$.
2. For each $\mathbf{i = 1, \ldots, m - n + 1}$
   1. If $\mathbf{h_p(T_{i\ldots i+n-1}) = h_p(P)}$, then $\mathbf{S = S \cup \{i\}}$.
   2. Compute $\mathbf{h_p(T_{i+1\ldots i+n})}$ using $\mathbf{h_p(T_{i\ldots i+n-1})}$ and $\mathbf{h_p(2^n)}$.

Set $\mathbf{M = \lceil 2(sn) \lg sn \rceil}$. Given $\mathbf{x \neq y}$, $\mathbf{Pr[h_p(x) = h_p(y)] \leq 1/s}$.

False positive: Pr[S contains an i, while no match at i]

Karp-Rabin Algorithm: Error Probability

Pr[S contains an index i, while there is no match at i]

1. Set $\mathbf{S} = \emptyset$. Compute $\mathbf{h_p(T_{1\ldots n})}$, $\mathbf{h_p(2^n)}$, and $\mathbf{h_p(P)}$.
2. For each $\mathbf{i = 1, \ldots, m - n + 1}$
    1. If $\mathbf{h_p(T_{i\ldots i+n-1}) = h_p(P)}$, then $\mathbf{S = S \cup \{i\}}$.
    2. Compute $\mathbf{h_p(T_{i+1\ldots i+n})}$ using $\mathbf{h_p(T_{i\ldots i+n-1})}$ and $\mathbf{h_p(2^n)}$.

Set $\mathbf{M = \lceil 2(sn) \lg sn \rceil}$. Given $\mathbf{x \neq y}$, $\mathbf{Pr[h_p(x) = h_p(y)] \leq 1/s}$.

False positive: Pr[S contains an i, while no match at i]

- Given $\mathbf{T_{i\ldots i+n-1} \neq P}$, $\mathbf{Pr[i \in S] \leq 1/s}$.

Karp-Rabin Algorithm: Error Probability

Pr[S contains an index i, while there is no match at i]

1. Set $\mathbf{S} = \emptyset$. Compute $\mathbf{h_p(T_{1\ldots n})}$, $\mathbf{h_p(2^n)}$, and $\mathbf{h_p(P)}$.
2. For each $\mathbf{i = 1, \ldots, m - n + 1}$
   1. If $\mathbf{h_p(T_{i\ldots i+n-1}) = h_p(P)}$, then $\mathbf{S = S \cup \{i\}}$.
   2. Compute $\mathbf{h_p(T_{i+1\ldots i+n})}$ using $\mathbf{h_p(T_{i\ldots i+n-1})}$ and $\mathbf{h_p(2^n)}$.

Set $\mathbf{M} = \lceil \mathbf{2(sn) \lg sn} \rceil$. Given $\mathbf{x \neq y}$, $\mathbf{Pr[h_p(x) = h_p(y)] \leq 1/s}$.

False positive: Pr[S contains an i, while no match at i]

- Given $\mathbf{T_{i\ldots i+n-1} \neq P}$, $\mathbf{Pr[i \in S] \leq 1/s}$.
- **Pr[**Any index in **S** is wrong**]**

Karp-Rabin Algorithm: Error Probability

Pr[S contains an index i, while there is no match at i]

1. Set $\mathbf{S} = \emptyset$. Compute $\mathbf{h_p(T_{1 \ldots n})}$, $\mathbf{h_p(2^n)}$, and $\mathbf{h_p(P)}$.
2. For each $\mathbf{i = 1, \ldots, m - n + 1}$
   1. If $\mathbf{h_p(T_{i \ldots i+n-1}) = h_p(P)}$, then $\mathbf{S = S \cup \{i\}}$.
   2. Compute $\mathbf{h_p(T_{i+1 \ldots i+n})}$ using $\mathbf{h_p(T_{i \ldots i+n-1})}$ and $\mathbf{h_p(2^n)}$.

Set $\mathbf{M = \lceil 2(sn) \lg sn \rceil}$. Given $\mathbf{x \neq y}$, $\mathbf{Pr[h_p(x) = h_p(y)] \leq 1/s}$.

False positive: Pr[S contains an i, while no match at i]

- Given $\mathbf{T_{i \ldots i+n-1} \neq P}$, $\mathbf{Pr[i \in S] \leq 1/s}$.
- $\mathbf{Pr[}$Any index in $\mathbf{S}$ is wrong$\mathbf{]} \leq \mathbf{m/s}$ (Union bound).

Karp-Rabin Algorithm: Error Probability

Pr[S contains an index i, while there is no match at i]

1. Set $\mathbf{S} = \emptyset$. Compute $\mathbf{h_p(T_{1\ldots n})}$, $\mathbf{h_p(2^n)}$, and $\mathbf{h_p(P)}$.
2. For each $\mathbf{i = 1, \ldots, m - n + 1}$
   1. If $\mathbf{h_p(T_{i\ldots i+n-1}) = h_p(P)}$, then $\mathbf{S = S \cup \{i\}}$.
   2. Compute $\mathbf{h_p(T_{i+1\ldots i+n})}$ using $\mathbf{h_p(T_{i\ldots i+n-1})}$ and $\mathbf{h_p(2^n)}$.

Set $\mathbf{M = \lceil 2(sn) \lg sn \rceil}$. Given $\mathbf{x \neq y}$, $\mathbf{Pr[h_p(x) = h_p(y)] \leq 1/s}$.

False positive: Pr[S contains an i, while no match at i]

- Given $\mathbf{T_{i\ldots i+n-1} \neq P}$, $\mathbf{Pr[i \in S] \leq 1/s}$.
- **Pr[**Any index in **S** is wrong**]** $\mathbf{\leq m/s}$ (Union bound).
- To ensure **S** is correct with at least **0.99** probability, we need

$$\mathbf{1 - \frac{m}{s} = 0.99 \Leftrightarrow \frac{m}{s} = \frac{1}{100} \Leftrightarrow s = 100m}$$

Karp-Rabin Algorithm

Back to running time

Running Time

- In Step 1, computing $\mathbf{h_p(x)}$ for an $\mathbf{n}$ bit $\mathbf{x}$ is in $\mathbf{O(n)}$ time.

Assuming $\mathbf{O(\lg M)}$ bit arithmetic can be done in $\mathbf{O(1)}$ time,

- Since $\mathbf{h_p(.)}$ produces $\mathbf{\lg M}$ bit numbers, both steps inside **for loop** can be done in $\mathbf{O(1)}$ time.
- Overall $\mathbf{O(m+n)}$ time. Can't do better.

$$\mathbf{M = \lceil 200mn \lg 100mn \rceil \Rightarrow \lg M = O(\lg m)}$$

Running Time

- In Step 1, computing $\mathbf{h_p(x)}$ for an $\mathbf{n}$ bit $\mathbf{x}$ is in $\mathbf{O(n)}$ time.

Assuming $\mathbf{O(\lg M)}$ bit arithmetic can be done in $\mathbf{O(1)}$ time,

- Since $\mathbf{h_p(.)}$ produces $\mathbf{\lg M}$ bit numbers, both steps inside **for loop** can be done in $\mathbf{O(1)}$ time.
- Overall $\mathbf{O(m+n)}$ time. Can't do better.

$$\mathbf{M = \lceil 200mn \lg 100mn \rceil \Rightarrow \lg M = O(\lg m)}$$

Even if $\mathbf{T}$ is entire Wikipedia, with bit length $\mathbf{m \approx 2^{38}}$,

Karp-Rabin Algorithm

Back to running time

Running Time

- In Step 1, computing $h_p(x)$ for an n bit x is in $O(n)$ time.

Assuming $O(\lg M)$ bit arithmetic can be done in $O(1)$ time,

- Since $h_p(.)$ produces $\lg M$ bit numbers, both steps inside **for loop** can be done in $O(1)$ time.
- Overall $O(m + n)$ time. Can't do better.

$$M = \lceil 200mn \lg 100mn \rceil \Rightarrow \lg M = O(\lg m)$$

Even if T is entire Wikipedia, with bit length $m \approx 2^{38}$,

$$\lg M \approx 64 \text{ (assuming bit-length of } n \leq 2^{16})$$

Running Time

- In Step 1, computing $\mathbf{h_p(x)}$ for an $\mathbf{n}$ bit $\mathbf{x}$ is in $\mathbf{O(n)}$ time.

Assuming $\mathbf{O(\lg M)}$ bit arithmetic can be done in $\mathbf{O(1)}$ time,

- Since $\mathbf{h_p(.)}$ produces $\mathbf{\lg M}$ bit numbers, both steps inside **for loop** can be done in $\mathbf{O(1)}$ time.
- Overall $\mathbf{O(m + n)}$ time. Can't do better.

$$\mathbf{M = \lceil 200mn \lg 100mn \rceil \Rightarrow \lg M = O(\lg m)}$$

Even if $\mathbf{T}$ is entire Wikipedia, with bit length $\mathbf{m \approx 2^{38}}$,

$$\mathbf{\lg M \approx 64} \text{ (assuming bit-length of } \mathbf{n \leq 2^{16}}\text{)}$$

$\mathbf{64}$-bit arithmetic is doable on laptops!

Take away points

1. Hashing is a powerful and important technique. Many practical applications.
2. Randomization fundamental to understanding hashing.
3. Good and efficient hashing possible in theory and practice with proper definitions (universal, perfect, etc).
4. Related ideas of creating a compact fingerprint/sketch for objects is very powerful in theory and practice.