Fingerprinting for String Matching

Lecture 11

Feb 20, 2019

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.

Hash functions are an example of fingerprinting.

Outline

Use of fingerprinting for designing fast algorithms

String equality

Given two strings x and y determine if $x = y$ with very little communication.

Outline

Use of fingerprinting for designing fast algorithms

String equality

Given two strings x and y determine if $x = y$ with very little communication.

Problem

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

Outline

Use of fingerprinting for designing fast algorithms

String equality

Given two strings x and y determine if $x = y$ with very little communication.

Problem

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

Karp-Rabin Randomized Algorithm

Outline

Use of fingerprinting for designing fast algorithms

String equality

Given two strings x and y determine if $x = y$ with very little communication.

Problem

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

Karp-Rabin Randomized Algorithm

It involves:

- Sampling a prime
- String equality via $mod\ p$ arithmetic
- Rabin's fingerprinting scheme – rolling hash

Part I

Sampling a Prime

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

Proof.

Event A : a prime is picked in a round. $\Pr[A] = \pi(x)/x$.
Event B : number (prime) p^* is picked. $\Pr[B] = 1/x$.
$\Pr[A \cap B] = \Pr[B] = 1/x$. **Why?** Because $B \subset A$.

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

$\square$

Sampling a prime: Expected number of samples

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).

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 $\mathbf{0}$ and x.

J. Hadamard and C. J. de la Vallée-Poussin (1896)

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

How many primes between 0 and x

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

J. Hadamard and C. J. de la Vallée-Poussin (1896)

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

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.

J. Hadamard and C. J. de la Vallée-Poussin (1896)

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

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.

J. Hadamard and C. J. de la Vallée-Poussin (1896)

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

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.

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

Part II

String Equality

String Equality

Problem

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

String Equality

Problem

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

Alice sends Bob x, and Bob confirms if $x = y$. But sending 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 x, and Bob, back at mission control, receives a string y. They know nothing about each others strings, but want to check if $x = y$.

Alice sends Bob x, and Bob confirms if $x = y$. But sending 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 $O(\lg N)$ may suffice!!!

String Equality

Problem

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

Alice sends Bob x, and Bob confirms if $x = y$. But sending 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 $O(\lg N)$ may suffice!!!
  - If $x = y$, then $\mathbf{Pr}[\text{Bob says } \textit{equal}] = 1$.
  - If $x \neq y$, then $\mathbf{Pr}[\text{Bob says } \textit{un-equal}] = 0.9999$.

N versus log N

Question: Given x, y what is basic information that Alice can send to Bob about x?

N versus log N

Question: Given x, y what is basic information that Alice can send to Bob about x?

Alice can send $|x|$. How many bits does this take?

N versus log N

Question: Given x, y what is basic information that Alice can send to Bob about x?

Alice can send $|x|$. How many bits does this take? $\lceil \log N \rceil$

Thus one can assume that Alice and Bob have equal length strings for simplicity.

N versus log N

Question: Given x, y what is basic information that Alice can send to Bob about x?

Alice can send $|x|$. How many bits does this take? $\lceil \log N \rceil$

Thus one can assume that Alice and Bob have equal length strings for simplicity.

If $x \neq y$ they differ in at least one bit. How many bits does it take to specify the location of a bit where they differ?

N versus log N

Question: Given x, y what is basic information that Alice can send to Bob about x?

Alice can send $|x|$. How many bits does this take? $\lceil \log N \rceil$

Thus one can assume that Alice and Bob have equal length strings for simplicity.

If $x \neq y$ they differ in at least one bit. How many bits does it take to specify the location of a bit where they differ? $\lceil \log N \rceil$

How many binary strings of length N are there?

N versus log N

Question: Given x, y what is basic information that Alice can send to Bob about x?

Alice can send $|x|$. How many bits does this take? $\lceil \log N \rceil$

Thus one can assume that Alice and Bob have equal length strings for simplicity.

If $x \neq y$ they differ in at least one bit. How many bits does it take to specify the location of a bit where they differ? $\lceil \log N \rceil$

How many binary strings of length N are there? 2^N Information theoretically no deterministic protocol can send less than N bits but randomization with smaller error allows one to get $O(\log N)$ bits.

N versus log N

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

N versus log N

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

$\boldsymbol{\lg N = 38}$

Universal Hashing?

Question: Can we use universal hashing? Alice sends $h(x)$ to Bob and Bob checks if $h(x) = h(y)$. If range of h is $[m]$ and h is universal then $\Pr[h(x) = h(y)] \leq 1/m$ if $x \neq q$. Can choose m sufficiently large to make this small. Only need to send $O(\log m)$ bits?

Universal Hashing?

Question: Can we use universal hashing? Alice sends $h(x)$ to Bob and Bob checks if $h(x) = h(y)$. If range of h is $[m]$ and h is universal then $\Pr[h(x) = h(y)] \leq 1/m$ if $x \neq q$. Can choose m sufficiently large to make this small. Only need to send $O(\log m)$ bits?

- **Scenario 1:** Both Alice and Bob know h apriori
  - This means Alice cannot pick randomness specifically for each new x. Will violate randomized guarantee if used repeatedly.
- **Scenario 2;** Alice has to send h also to Bob
  - Consider scheme using primes. Universe $\mathcal{U}$ is set of all 2^N strings implies $p > 2^N$ and $a, b \in \mathbb{Z}_p$. Alice needs to send p, a, b which is $\Omega(N)$ bits!

String Equality: Randomized Algorithm

x, y : N-bit strings. Interpret them as integers in binary

String Equality: Randomized Algorithm

x, y : N-bit strings. Interpret them as integers in binary

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

String Equality: Randomized Algorithm

x, y : N-bit strings. Interpret them as integers in binary

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

Procedure

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

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

String Equality: Randomized Algorithm

x, y : N-bit strings. Interpret them as integers in binary

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

Procedure

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

x, y : N-bit strings. Interpret them as integers in binary

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

Procedure

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

x, y : N-bit strings.

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

Procedure

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 $x \neq y$ then, $\mathbf{Pr}[\textit{Bob says } \text{equal}] \leq 1/5$ *(error probability).*

String Equality: Randomized Algorithm

x, y : N-bit strings.

(Recall) If $M = \lceil 2(sN) \lg sN \rceil$, then sN primes in $\{1, \ldots, M\}$.

Procedure

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 $x \neq y$ then, $\Pr[\textit{Bob says}\ \text{equal}] \leq 1/s$ *(error probability).*

Question.

Let $x = 6 = 2 * 3$. If we draw a p u.a.r. from $\{2, 3, 5, 7\}$, then what is the probability that $x \; mod \; p = 0$?

- **(A)** 0.
- **(B)** 1.
- **(C)** $1/4$.
- **(D)** $1/2$.
- **(E)** none of the above.

Question.

Let $x = 6 = 2 * 3$. If we draw a p u.a.r. from $\{2, 3, 5, 7\}$, then what is the probability that $x \; mod \; p = 0$?

- **(A)** 0.
- **(B)** 1.
- **(C)** 1/4.
- **(D)** 1/2.
- **(E)** none of the above.

Now, let $y = 21$. What is the probability that $(y - x) \; mod \; p = 15 \; mod \; p = 0$?

- **(A)** 0.
- **(B)** 1.
- **(C)** 1/4.
- **(D)** 1/2.

String Equality: Randomized Algorithm

Error probability

x, y N-bit string, $M = \lceil 2(sN) \lg sN \rceil$, and $h_p(x) = x \mod p$

Lemma

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

Proof.

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

String Equality: Randomized Algorithm

Error probability

x, y N-bit string, $M = \lceil 2(sN) \lg sN \rceil$, and $h_p(x) = x \mod p$

Lemma

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

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 $D \leq 2^N$.

String Equality: Randomized Algorithm

Error probability

x, y N-bit string, $M = \lceil 2(sN) \lg sN \rceil$, and $h_p(x) = x \mod p$

Lemma

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

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 with repetitions.

String Equality: Randomized Algorithm

Error probability

x, y N-bit string, $M = \lceil 2(sN) \lg sN \rceil$, and $h_p(x) = x \mod p$

Lemma

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

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 with repetitions. All $p_i \geq 2 \Rightarrow D \geq 2^k$.

String Equality: Randomized Algorithm

Error probability

x, y N-bit string, $M = \lceil 2(sN) \lg sN \rceil$, and $h_p(x) = x \mod p$

Lemma

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

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 with repetitions. 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 prime divisors.

String Equality: Randomized Algorithm

Error probability

x, y N-bit string, $M = \lceil 2(sN) \lg sN \rceil$, and $h_p(x) = x \mod p$

Lemma

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

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 $D \leq 2^N$.
- $D = p_1 \dots p_k$ prime factorization with repetitions. 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 prime divisors.
- Probability that a random prime p from $\{1, \dots, M\}$ is a divisor $= \frac{k}{\pi(M)} \leq \frac{N}{\pi(M)}$

String Equality: Randomized Algorithm

Error probability

x, y N-bit string, $M = \lceil 2(sN) \lg sN \rceil$, and $h_p(x) = x \mod p$

Lemma

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

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 with repetitions. 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 prime divisors.
- Probability that a random prime p from $\{1, \ldots, M\}$ is a divisor $= \frac{k}{\pi(M)} \leq \frac{N}{\pi(M)} \leq \frac{N}{M/\lg M} = \frac{N}{2(sN)\lg sN} \lg M \leq \frac{1}{s}$

□

Error Probability and Communication

Low Error Probability

1. Choose large enough s. Error prob: $1/s$.

Error Probability and Communication

Low Error Probability

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

Error Probability and Communication

Low Error Probability

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

Error Probability and Communication

Low Error Probability

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

Error Probability and Communication

Low Error Probability

1. Choose large enough s. Error prob: $1/s$.
2. Alice repeats the process R times, and Bob says *equal* only if he gets equal all 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 $\leq M$. # bits: $2 \lg M \leq 4(\lg s + \lg N)$.

Error Probability and Communication

Low Error Probability

1. Choose large enough s. Error prob: $1/s$.
2. Alice repeats the process R times, and Bob says *equal* only if he gets equal all 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 $\leq M$. # bits: $2 \lg M \leq 4(\lg s + \lg N)$.

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

Error Probability and Communication

Low Error Probability

1. Choose large enough s. Error prob: $1/s$.
2. Alice repeats the process R times, and Bob says *equal* only if he gets equal all 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 $\leq M$. # bits: $2 \lg M \leq 4(\lg s + \lg N)$.

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

Second approach will send $10(2 \lg(10N \lg 5N)) \leq 1280$ bits.

Verifying inequality

Question: Algorithm is Monte Carlo. Suppose $x \neq y$. Can Alice and Bob find with high probability an index i such that $x_i \neq y_i$ and verify it? Assuming here that Alice and Bob can communicate over multiple rounds adaptively.

Verifying inequality

Question: Algorithm is Monte Carlo. Suppose $x \neq y$. Can Alice and Bob find with high probability an index i such that $x_i \neq y_i$ and verify it? Assuming here that Alice and Bob can communicate over multiple rounds adaptively.

Exercise: Show how Alice and Bob can do this with $O(\log^2 N)$ bits of communication. *Hint:* Use binary search.

Verifying inequality

Question: Algorithm is Monte Carlo. Suppose $x \neq y$. Can Alice and Bob find with high probability an index i such that $x_i \neq y_i$ and verify it? Assuming here that Alice and Bob can communicate over multiple rounds adaptively.

Exercise: Show how Alice and Bob can do this with $O(\log^2 N)$ bits of communication. *Hint:* Use binary search.

Using above find a Las Vegas algorithm that communicates $O(\log N)$ bits in expectation and $O(N)$ bits in the worst case but is always correct.

Multiple strings

We want to check equality between several pairs of strings $(x_1, y_1), \ldots, (x_k, y_k)$ where all strings are N-bits long.

Suppose we pick random prime p and use hash function h_p to check equality of all pairs. Will it work? What range should p be chosen from to ensure that **all** of the answers are correct with probability at least $(1 - \delta)$ for some given parameter δ?

Multiple strings

We want to check equality between several pairs of strings $(x_1, y_1), \ldots, (x_k, y_k)$ where all strings are N-bits long.

Suppose we pick random prime p and use hash function h_p to check equality of all pairs. Will it work? What range should p be chosen from to ensure that **all** of the answers are correct with probability at least $(1 - \delta)$ for some given parameter δ?

Use union bound to figure out how large s should be.

Part III

Karp-Rabin Pattern Matching Algorithm

Pattern Matching

Given a string T of length m and pattern P of length n, s.t. $m \gg n$,

- find whether P is a substring of T
- more generally, find all positions where P matches with T.

Example

T=abracadabra, P=ab.

Pattern Matching

Given a string T of length m and pattern P of length n, s.t. $m \gg n$,

- find whether P is a substring of T
- more generally, find all positions where P matches with T.

Example

T=abracadabra, P=ab.

Index set of all matches: $S = \{1, 8\}$.

Pattern Matching

Given a string T of length m and pattern P of length n, s.t. $m \gg n$,

- find whether P is a substring of T
- more generally, find all positions where P matches with T.

Example

T=abracadabra, P=ab.

Index set of all matches: $S = \{1, 8\}$.

For $j \geq i$, let $T[i,j] = T[i]T[i+1] \ldots T[j]$.

Pattern Matching

Given a string T of length m and pattern P of length n, s.t. $m \gg n$,

- find whether P is a substring of T
- more generally, find all positions where P matches with T.

Example

T=abracadabra, P=ab.

Index set of all matches: $S = \{1, 8\}$.

For $j \geq i$, let $T[i,j] = T[i]T[i+1]\ldots T[j]$.

Brute force algorithm

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

- If $\text{match}(T[i, i+n-1], P)$ then $S = S \cup \{i\}$.

Pattern Matching

Given a string T of length m and pattern P of length n, s.t. $m \gg n$,

- find whether P is a substring of T
- more generally, find all positions where P matches with T.

Example

T=abracadabra, P=ab.

Index set of all matches: $S = \{1, 8\}$.

For $j \geq i$, let $T[i, j] = T[i]T[i+1] \ldots T[j]$.

Brute force algorithm

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

- If $\text{match}(T[i, i+n-1], P)$ then $S = S \cup \{i\}$.

$O(mn)$ run-time.

Using Fingerprinting

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

Brute force algorithm using fingerprinting

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

- If $h_p(T[i, i + n - 1]) = h_p(P)$ then $S = S \cup \{i\}$.

Using Fingerprinting

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

Brute force algorithm using fingerprinting

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

- If $h_p(T[i, i + n - 1]) = h_p(P)$ 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 Fingerprinting

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

Brute force algorithm using fingerprinting

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

- If $h_p(T[i, i + n - 1]) = h_p(P)$ 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.

Do we need to recompute fingerprints from scratch for each i?

mod p math

Let a and b be (non-negative) integers.

$$(a + b) \; mod \; p = ((a \; mod \; p) + (b \; mod \; p)) \; mod \; p$$

mod p math

Let a and b be (non-negative) integers.

$$(a + b) \ mod \ p = ((a \ mod \ p) + (b \ mod \ p)) \ mod \ p$$

$$(a \cdot b) \ mod \ p = ((a \ mod \ p) \cdot (b \ mod \ p)) \ mod \ p$$

Rolling Hash

$x = T[i \dots i + n - 1]$ and $x' = T[i + 1, i + n]$.
Let $x = x_1 x_2 \dots x_n$ and $x' = x'_1 x'_2 \dots x'_n$

Example

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

Rolling Hash

$x = T[i \ldots i+n-1]$ and $x' = T[i+1, i+n]$.
Let $x = x_1 x_2 \ldots x_n$ and $x' = x'_1 x'_2 \ldots x'_n$

Example

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

$$x' = 2(x - x_1 2^{n-1}) + x'_n$$

Rolling Hash

$x = T[i \ldots i + n - 1]$ and $x' = T[i + 1, i + n]$.
Let $x = x_1 x_2 \ldots x_n$ and $x' = x'_1 x'_2 \ldots x'_n$

Example

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

$$\begin{aligned} x' &= 2(x - x_1 2^{n-1}) + x'_n \\ &= 2x - x_1 2^n + x'_n \end{aligned}$$

Rolling Hash

$x = T[i \dots i+n-1]$ and $x' = T[i+1, i+n]$.
Let $x = x_1 x_2 \dots x_n$ and $x' = x'_1 x'_2 \dots x'_n$

Example

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

$$\begin{aligned} x' &= 2(x - x_1 2^{n-1}) + x'_n \\ &= 2x - x_1 2^n + x'_n \end{aligned}$$

$$\begin{aligned} h_p(x') &= x' \bmod p \\ &= (2(x \bmod p) - x_1(2^n \bmod p) + x'_n) \bmod p \\ &= (2h_p(x) - x_1 h_p(2^n) + x'_n) \bmod p \end{aligned}$$

Karp-Rabin Algorithm

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

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

Karp-Rabin Algorithm

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

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

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

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

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

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

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

1. Set $S = \emptyset$. Compute $h_p(T[1, n])$, $h_p(2^n)$, and $h_p(P)$.
2. For each $i = 1, \ldots, m - n + 1$
   1. If $h_p(T[i, i + n - 1]) = h_p(P)$, then $S = S \cup \{i\}$.
   2. Compute $h_p(T[i + 1, i + n])$ using $h_p(T[i, i + n - 1])$ 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. Can't do better.

Karp-Rabin Algorithm: Error Analysis

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

Lemma

If match at any position i then $i \in S$. In otherwords if $T[i, i + n - 1] = P$, then $i \in S$.

All matched positions are in S.

Karp-Rabin Algorithm: Error Analysis

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

Lemma

If match at any position i then $i \in S$. In otherwords if $T[i, i + n - 1] = P$, then $i \in S$.

All matched positions are in S.

Can it contain unmatched positions?

Karp-Rabin Algorithm: Error Analysis

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

Lemma

If match at any position i then $i \in S$. In otherwords if $T[i, i + n - 1] = P$, then $i \in S$.

All matched positions are in S.

Can it contain unmatched positions? YES!

Karp-Rabin Algorithm: Error Analysis

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

Lemma

If match at any position i then $i \in S$. In otherwords if $T[i, i + n - 1] = P$, then $i \in S$.

All matched positions are in S.

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

Karp-Rabin Algorithm: Error Analysis

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

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

Karp-Rabin Algorithm: Error Analysis

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

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

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

Karp-Rabin Algorithm: Error Analysis

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

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

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

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

Karp-Rabin Algorithm: Error Analysis

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

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

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

- If $T[i, i+n-1] \neq P$, $\mathbf{Pr}[i \in S] \leq 1/s$.
- $\mathbf{Pr}[S$ contains an incorrect index$]$

Karp-Rabin Algorithm: Error Analysis

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

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

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

- If $T[i, i+n-1] \neq P$, $\mathbf{Pr}[i \in S] \leq 1/s$.
- **Pr[**S contains an incorrect index**]** $\leq m/s$ (Union bound).

Karp-Rabin Algorithm: Error Analysis

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

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

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

- If $T[i, i+n-1] \neq P$, $\Pr[i \in S] \leq 1/s$.
- **Pr[*S* contains an incorrect index]** $\leq m/s$ (Union bound).
- To ensure S is correct with at least **0.99** probability, we need

$$1 - \frac{m}{s} \geq 0.99 \Rightarrow \frac{m}{s} \leq \frac{1}{100} \Rightarrow s \geq 100m$$

.

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.

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

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 $m \approx 2^{38}$,

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 $m \approx 2^{38}$,

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

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 $m \approx 2^{38}$,

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

64-bit arithmetic is doable on laptops!

Deterministic Pattern Matching

$O(n + m)$ (linear time) deterministic algorithms are known

- Boyer-Moore algorithm
- Knuth-Morris-Pratt (KMP) algorithm

Why randomization?

- generalizes to settings (two-dimensional settings) where standard algorithms do not
- generalizes to multiple string pattern matchings easily