CS70 – Spring 2024

Lecture 27 – April 25

Today: Intro. to Randomized Algorithms

- Finding large prime numbers
- Fingerprinting
- Pattern matching

1. Primality testing

Cryptographic applications (e.g. RSA) require very large prime numbers (~ 100s of digits)

$\boxed{Q:}$ How do we get hold of them ?

$\boxed{A:}$ Generate a random number with 100s of digits & test if it is prime!
If not, try another one ...

$\boxed{Q:}$ How many numbers do we expect to try ?

$\boxed{A:}$ Depends on the density of primes

Prime Number Theorem: Let $\pi(n)$ denote the number of primes $\leq n$. Then

$$\boxed{\frac{n}{\ln n} \leq \pi(n) \leq 1.26 \frac{n}{\ln n} \quad \forall n \geq 17}$$

Corollary: Roughly 1 in every $\ln n$ numbers $\leq n$ is prime. [Eg. $n = 10^{500} \Rightarrow \ln n \approx 1150$]

So no. of trials until we find a prime has Geometric$(1/\ln n)$ distribution

$$\Rightarrow \boxed{E[\#\text{trials}] \approx \ln n}$$

$$\Pr[\text{more than } k \ln n \text{ trials}] = \left(1 - \frac{1}{\ln n}\right)^{k \ln n} \leq \boxed{e^{-k}}$$

$$\left(1 - \frac{1}{m}\right)^m \sim e^{-1}$$

Q: How do we get hold of them?

A: Generate a <u>random</u> number with 100s of digits & test if it is prime!
If not, try another one...

<u>Bigger Question</u>: How do we test if a very large number n (with 100s of digits) is prime?

<u>Simple algorithm</u>: try all divisors!

for $a = 2, 3, \ldots, \sqrt{n}$

  if a divides n **then** **halt** & **output** "not prime"

output "prime"

Simple algorithm: try all divisors!

for $a = 2, 3, \ldots, \sqrt{n}$

if a divides n then halt & output "not prime"

output "prime"

Is this a good algorithm?

How many divisors will we have to try?

Think: $n \approx 10^{500}$ (500 digits, $\approx$ 1700 bits)

A simple randomized algorithm

repeat many times

  pick $a \in [2, \sqrt{n}]$ uniformly at random

  if a divides n **then halt** & **output** "not prime"

output "prime?"

witness

Is this a good algorithm?

A better randomized algorithm?

Need a better <u>witness</u> for n being not prime (so that the number of witnesses is large)

<u>Recall</u>: <u>Fermat's Little Theorem</u>

For any prime p and any $a \in [1, \dots, p-1]$:

$$a^{p-1} \equiv 1 \pmod{p}$$

<u>Corollary</u>: If we find an $a \in [1, \dots, n-1]$ s.t.

$$a^{n-1} \not\equiv 1 \pmod{n}$$

then we know for sure that n is <u>not</u> prime!

A better algorithm : Fermat Test

pick $a \in [1, \dots, n-1]$ u.a.r.

if $\gcd(a,n) \neq 1$ then halt & output "not prime"

if $a^{n-1} \not\equiv 1 \pmod{n}$ then —— " ——

else output "prime ?"

Properties

- Outputs "not prime" $\Rightarrow$ n is definitely not prime
- Outputs "prime ?" $\Rightarrow$ either n is prime, or it's not prime but the algorithm picked an a that's not a witness
- Running time : $O(\log n) = O(\#\text{ of digits in } n)$

- Running time: $O(\log n) = O(\#\text{ of } \underline{\text{digits}} \text{ in } n)$

Why?

→ $\gcd(a,n)$ runs in $O(\log n)$ steps via Euclid's Algorithm

→ a^{n-1} can be computed in $O(\log n)$ steps by <u>repeated squaring</u>:

e.g. for a^{53}

$53 = 32 + 16 + 4 + 1 = 110101$ in binary

$a^{53} = a^{32} \times a^{16} \times a^{4} \times a^{1}$

→ enough to compute $a, a^2, a^4, a^8, a^{16}, a^{32}$

Density of witnesses

Let $\mathbb{Z}_n^* = \{a \in [1, \ldots, n-1] : \gcd(a,n) = 1\}$

Defn: A witness for n is a number $a \in \mathbb{Z}_n^*$ s.t. $a^{n-1} \not\equiv 1 \pmod{n}$

Claim: For any n, if $\exists$ a witness a for n then at least half of all $a \in \mathbb{Z}_n^*$ are witnesses!

Corollary: The Fermat Test is correct with probability $\geq 1/2$ on all inputs n except for non-primes n that have no witnesses at all

"Carmichael Numbers"

A better algorithm: Fermat Test

pick $a \in [1, \dots, n-1]$ u.a.r.

if $\gcd(a,n) \neq 1$ then halt & output "not prime"

if $a^{n-1} \neq 1 \pmod{n}$ then —— " ——

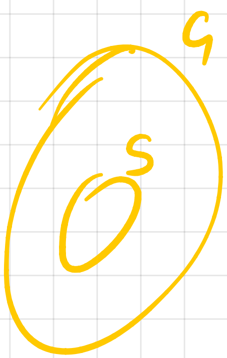
else output "prime?"

Properties

- Outputs "not prime" $\Longrightarrow$ n is definitely not prime
- Outputs "prime?" $\Longrightarrow$ either n is a Carmichael Number, or $\boxed{\Pr[n \text{ is prime}] \geq 1/2}$

<u>Claim</u>: For any n, if $\exists$ a witness a for n then <u>at least half</u> of all $a \in \mathbb{Z}_n^*$ are witnesses!

<u>Proof</u>: $\mathbb{Z}_n^*$ is a <u>group</u> under multiplication $\pmod n$. I.e.:



$|S| \mid |G|$

- [Identity]: $1 \in \mathbb{Z}_n^*$
- [Inverses]: $a \in \mathbb{Z}_n^* \Rightarrow a^{-1} \in \mathbb{Z}_n^*$
- [Closure]: $a, b \in \mathbb{Z}_n^* \Rightarrow ab \in \mathbb{Z}_n^*$

The set $S = \{a \in \mathbb{Z}_n^* : a^{n-1} = 1 \pmod n\}$ is a <u>subgroup</u> (because $a, b \in S \Rightarrow ab \in S$)

<u>Lagrange's Theorem</u>: $|S|$ divides $|\mathbb{Z}_n^*|$

n is not a CN $\Rightarrow |S| < |\mathbb{Z}_n^*| \Rightarrow |S| \leq \frac{1}{2} |\mathbb{Z}_n^*|$ ✓

Two remaining questions:

1. Why is $\Pr[n \text{ is prime}] \geq 1/2$ good enough?

2. What about Carmichael Numbers?

1. Why is $\Pr[n \text{ is prime}] \geq 1/2$ good enough?

A: Just repeat the Fermat Test k times, choosing a independently each time

Suppose n is not prime and not a CN

Then $\boxed{\Pr[\text{all } k \text{ tests output "prime?"}] \leq 2^{-k}}$

If we take $k = 1000$ (say), this is negligible!

2. What about Carmichael Numbers?

Defn: n is a Carmichael Number if it is not prime and $a^{n-1} \equiv 1 \pmod{n}$ $\forall a \in \mathbb{Z}_n^*$

CNs are rare: 255 CNs $\leq 10^8$

~ 20 million CNs $\leq 10^{21}$

First few CNs: 561, 1105, 1729, ...

- There are similar randomized algorithms (using more complicated witnesses) that handle CNs
- There are also deterministic algorithms (that are always correct) but they are too inefficient ($\sim O((\log n)^6)$)

Proof that 561 is a Carmichael Number:

$561 = 3 \times 11 \times 17$

Claim: $a^{560} \equiv 1 \pmod{561} \quad \forall a \in \mathbb{Z}^*_{561}$

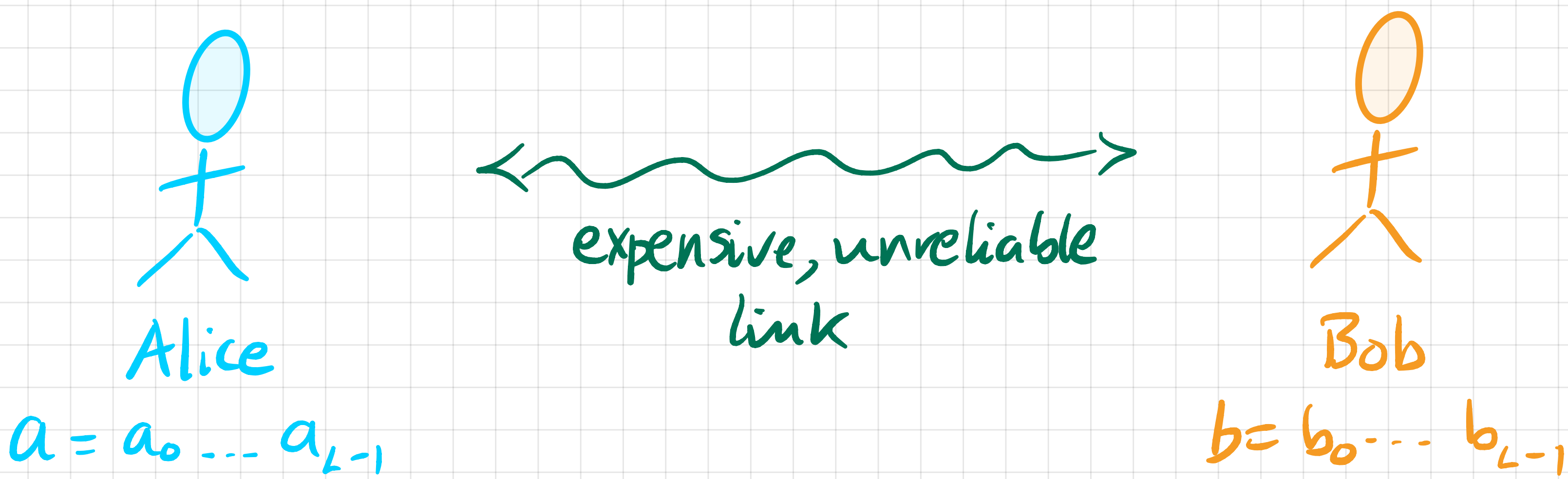
Sufficient to show $a^{560} \equiv 1$ $\left.\begin{array}{l} \bmod 3 \\ \bmod 11 \\ \bmod 17 \end{array}\right\}$ $\Rightarrow$ by C.R.T. $a^{560} \equiv 1 \bmod 561$

Now note: $a^2 \equiv 1 \pmod 3 \Rightarrow a^{560} \equiv 1 \pmod 3$

$a^{10} \equiv 1 \pmod{11} \Rightarrow a^{560} \equiv 1 \pmod{11}$

$a^{16} \equiv 1 \pmod{17} \Rightarrow a^{560} \equiv 1 \pmod{17}$ ✓

2. Fingerprinting & Pattern Matching

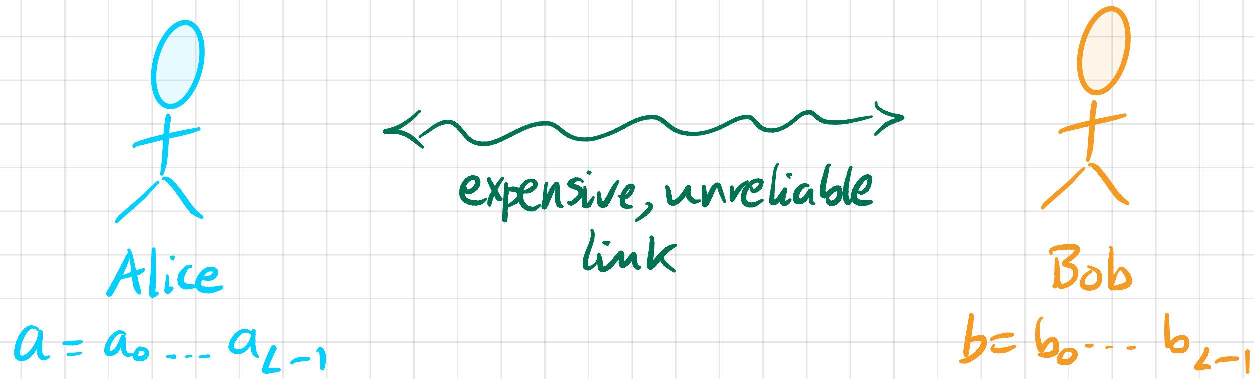


Alice & Bob each have a copy of a large database consisting of L bits (L very large)

They want to check if their copies are identical

But they <u>don't</u> want to send all L bits

<u>Idea</u>: Send a much smaller <u>fingerprint</u> of their data



Idea: Send a much smaller fingerprint of their data

View $a = a_0 \dots a_{L-1}$ & $b = b_0 \dots b_{L-1}$ as L-bit numbers

Alice: picks a random prime $p \in [2 \dots T]$

computes $F_p(a) := a \bmod p$

sends p and $F_p(a)$ to Bob

Bob: computes $F_p(b) := b \bmod p$

if $F_p(a) \neq F_p(b)$ outputs "not identical"

else outputs "identical?"

Alice: picks a random prime $p \in [2 \ldots T]$

computes $F_p(a) := a \bmod p$

sends p and $F_p(a)$ to Bob

Bob: computes $F_p(b) := b \bmod p$

if $F_p(a) \neq F_p(b)$ outputs "not identical"

else outputs "identical?"

Properties

Outputs "not identical" $\Rightarrow$ definitely $a \neq b$

Outputs "identical?" $\Rightarrow$ either $a = b$ or

$\boxed{a \neq b \text{ but } a = b \bmod p}$

error

Outputs "identical?" $\Rightarrow$ either $a = b$ or

$$\boxed{a \neq b \text{ but } a \equiv b \bmod p}$$

Claim: If $a \neq b$ and p is a random prime in $[2..T]$ then $\Pr[a \equiv b \pmod p] \leq \frac{L \ln T}{T}$

Proof: $a \equiv b \pmod p \iff a - b \equiv 0 \pmod p$

$\iff p \mid |a-b|$

But $|a-b|$ is an L-bit number, so it has at most L prime factors!

Hence $\Pr[a \equiv b \pmod p] \leq \frac{L}{\pi(T)} \leq \frac{L \ln T}{T}$

prime number thm.

Corollary: $\boxed{\Pr[\text{error}] \leq \frac{L \ln T}{T}}$

Corollary: $\boxed{\Pr[\text{error}] \leq \frac{L \ln T}{T}}$

How should we set T?

If we set $T = 4L\ln L$ then

$$\Pr[\text{error}] \leq L \cdot \frac{\ln L + \ln\ln L + \ln 4}{4L\ln L}$$

$$= \frac{1}{4}\left[1 + \frac{\ln\ln L}{\ln L} + \frac{\ln 4}{\ln L}\right]$$

$$\leq \boxed{\frac{1}{2}}$$

$\longrightarrow$ Can boost to $\leq 2^{-k}$ with k independent trials

Also: $T = 4L\ln L \Rightarrow p$ has $O(\log L)$ bits

So fingerprint $F_p(a)$ sent by Alice is exponentially smaller than the database itself!

Example:

Suppose $L = 2^{33}$ $(\approx 1\,GB)$

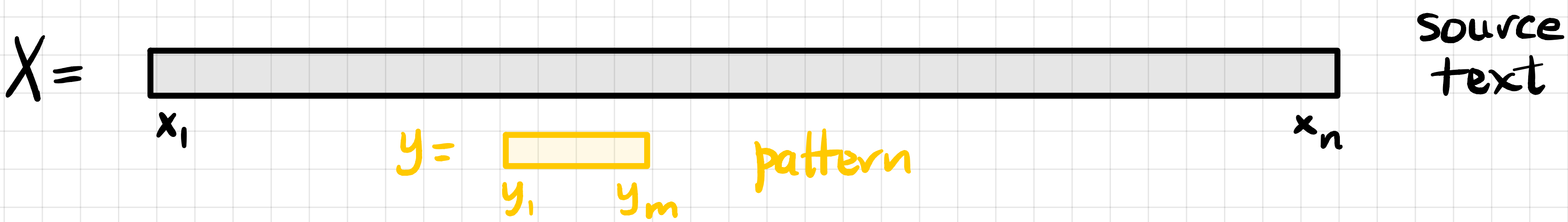
$T = 2^{64}$ ($\Rightarrow$ fingerprints are 64-bit words)

Then $\Pr[\text{error}] \leq \frac{L \ln T}{T} \leq 2^{33} \times \frac{64}{2^{64}}$

$= 2^{-25}$

$\approx \boxed{3.4 \times 10^{-7}}$

3. Pattern Matching



Question: Does Y occur as a contiguous substring in X?
i.e., is $Y = X(i) := x_i \, x_{i+1} \cdots x_{i+m-1}$ for some i?

Simple algorithm:

```
for i := 1 to n-m+1
    if Y = X(i) output "match found" & halt
output "no match"
```

Running time: $O(nm)$

Clever randomized algorithm: use fingerprints!

Pick a prime $p \in [2, .., T]$ u.a.r.

Compute $F_p(Y) := Y \bmod p$

for $i := 1$ **to** $n-m+1$

  Compute $F_p(X(i)) := X(i) \bmod p$

  if $F_p(Y) = F_p(X(i))$ **output** "match found" & halt

output "no match"

Clever randomized algorithm: use fingerprints!

Pick a prime $p \in [2, .., T]$ u.a.r.

Compute $F_p(y) := y \bmod p$

for $i := 1$ **to** $n-m+1$

  Compute $F_p(X(i)) := X(i) \bmod p$

  if $F_p(y) = F_p(X(i))$ **output** "match found" & halt

output "no match"

$$\Pr[\text{error}] = \Pr[\exists i : y \neq X(i) \ \& \ y - X(i) = 0 \pmod p]$$

union bound $\longrightarrow$ $$\leq \sum_{i=1}^{n-m+1} \Pr[y \neq X(i) \ \& \ y - X(i) = 0 \pmod p]$$

$$\lesssim n \times \frac{m \ln T}{T}$$

[$y - X(i)$ has m bits $\Rightarrow \leq m$ prime factors]

So if we set $\boxed{T = 4nm\ln(nm)}$ then $\boxed{\Pr[\text{error}] \leq \frac{1}{2}}$

So if we set $\boxed{T = 4nm\ln(nm)}$ then $\boxed{\Pr[\text{error}] \le \frac{1}{2}}$

With this choice of T, number of bits in p is

$$O(\log(nm)) = O(\log n).$$

So can assume arithmetic mod p is fast.

Running time of algorithm?

- compute $F_p(Y)$ — $O(m)$
- compute $F_p(X(1))$ — $O(m)$
- n iterations:

  compute $F_p(X(i))$ — $O(1)$

  compare $F_p(Y) = F_p(X(i))$ — $O(1)$

  $\Big\}\ O(n)$

Total: $O(n)$ [much faster than $O(nm)$]

Computing $F_p(X(i+1))$ from $F_p(X(i))$

$$\overbrace{x_i \;\; \underbrace{x_{i+1} \;\; x_{i+2} \; - \; - \; - \;\; x_{i+m-1} \;\; \rlap{x_{i+m}}\phantom{x_{i+m}}}_{X(i+1)}}^{X(i)}$$

$$X(i+1) = 2\left(X(i) - 2^{m-1}x_i\right) + x_{i+m}$$

$$\Rightarrow F_p(X(i+1)) = 2\left(F_p(X(i)) - 2^{m-1}x_i\right) + x_{i+m} \quad (\text{mod } p)$$

requires four arithmetic ops. mod p $\rightarrow$ $O(1)$ time

Example:

Searching for a pattern in a DNA sequence

Say $n = 2^{28}$, $m = 2^{11}$ (search for 1000-bp pattern in 100M-bp chromosome)

Take $T = 2^{64}$ (64-bit words)

$$\Pr[\text{error}] \le \frac{nm \ln T}{T} \le 2^{39} \cdot \frac{64}{2^{64}}$$

$$= 2^{-19}$$

$$\approx 0.000005$$

Note: Deterministic $O(n)$ algorithms do exist, but they're much harder to implement and have overheads

Life After CS70

CS 170 — Algorithms

CS 172 — Complexity & Computability

CS 174 — Randomized Algorithms

CS171 — Cryptography

CS176 — Computational Biology

EECS126 — Probability & Random Processes

EECS127 — Optimization

<u>Course Evaluations</u>: course-evaluations.berkeley.edu