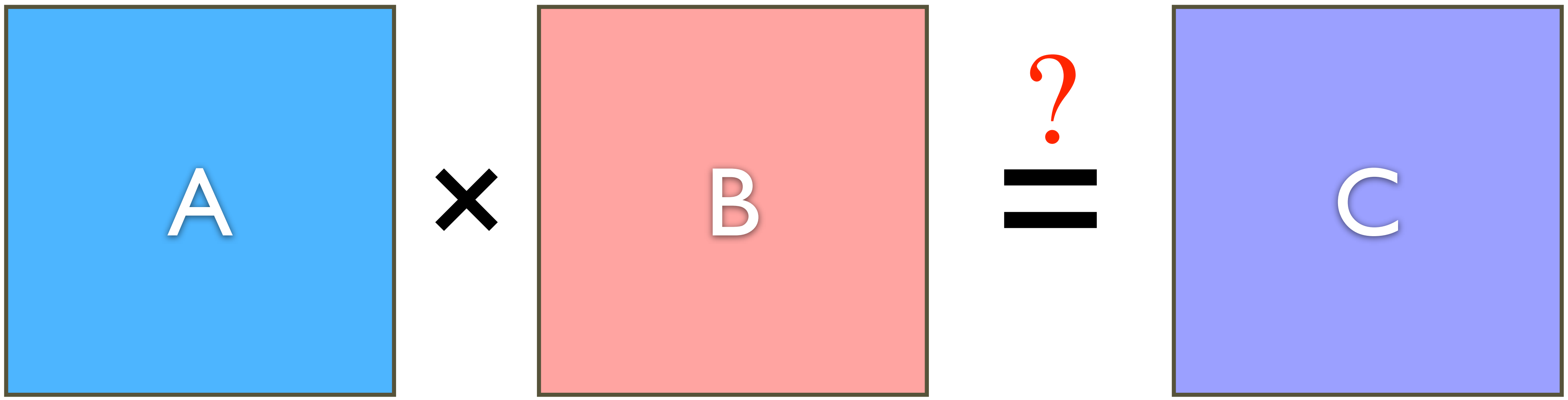
Advanced Algorithms

Fingerprinting

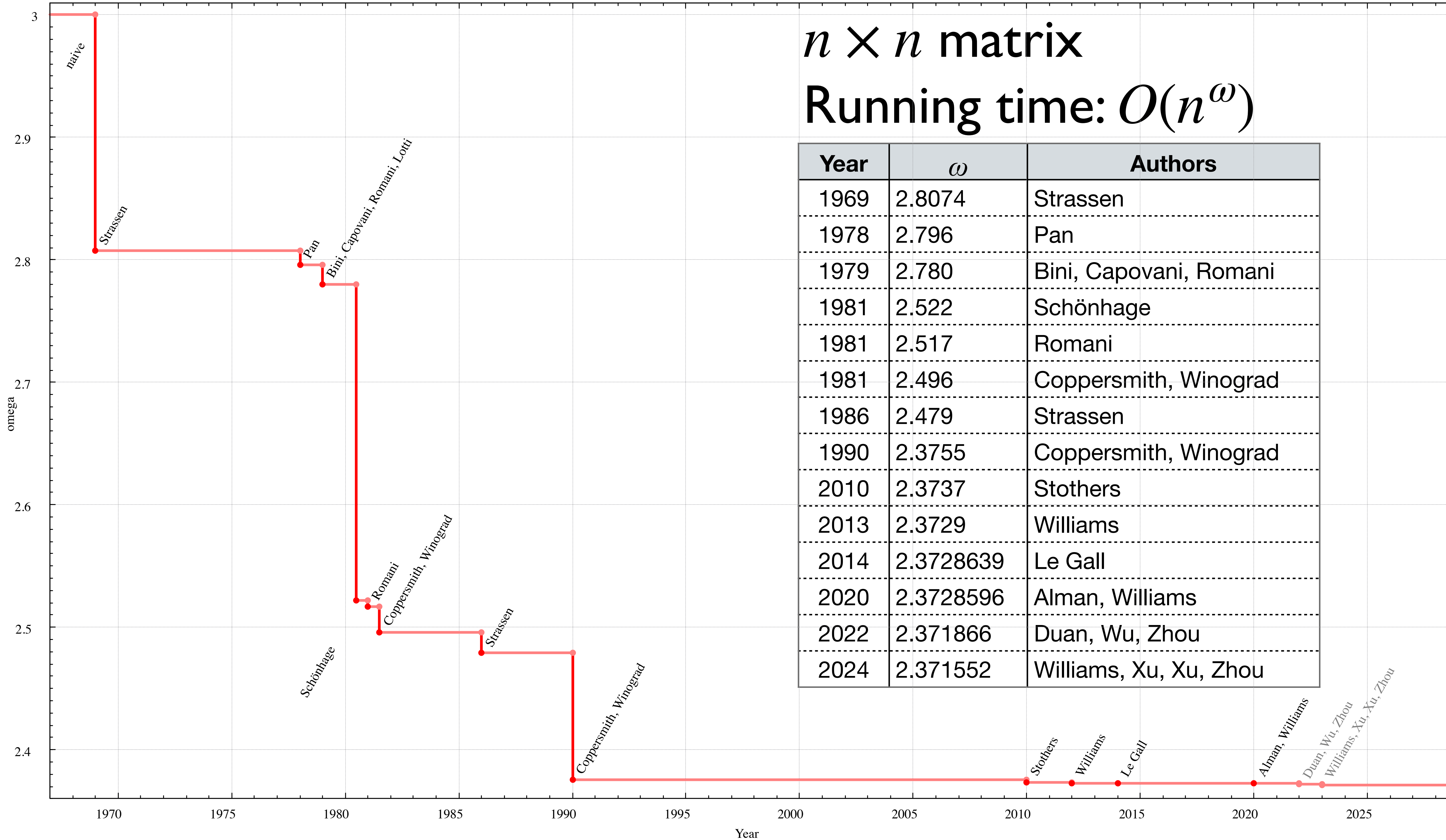
尹一通 Nanjing University, 2024 Fall

Checking Matrix Multiplication

- three $n \times n$ matrices A, B, C:



Matrix Multiplication Algorithms



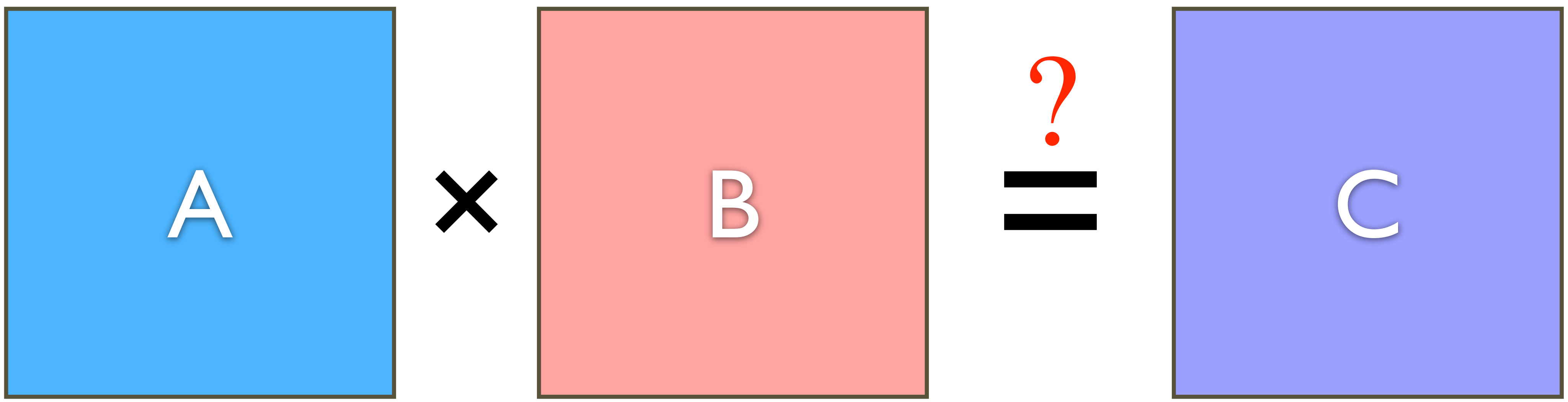
$n \times n$ matrix

Running time: $O(n^{\omega})$

Year	ω	Authors
1969	2.8074	Strassen
1978	2.796	Pan
1979	2.780	Bini, Capovani, Romani
1981	2.522	Schönhage
1981	2.517	Romani
1981	2.496	Coppersmith, Winograd
1986	2.479	Strassen
1990	2.3755	Coppersmith, Winograd
2010	2.3737	Stothers
2013	2.3729	Williams
2014	2.3728639	Le Gall
2020	2.3728596	Alman, Williams
2022	2.371866	Duan, Wu, Zhou
2024	2.371552	Williams, Xu, Xu, Zhou

Checking Matrix Multiplication

- three $n \times n$ matrices A, B, C:



Freivald's Algorithm:

pick a uniform random $r \in \{0,1\}^n$;

check whether $A(Br) = Cr$;

time: $O(n^2)$

if $AB = C$: always correct

if $AB \neq C$:

Freivald's Algorithm:

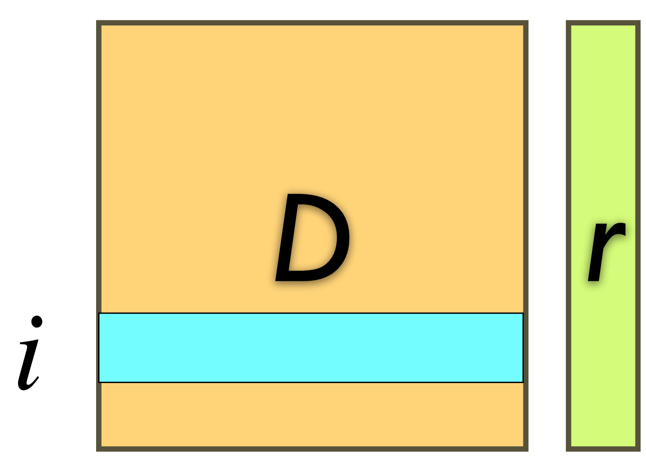
pick a uniform random $r \in \{0,1\}^n$;

check whether $A(Br) = Cr$;

if $AB \neq C$: let $D = AB - C \neq \mathbf{0}_{n\times n}$

suppose $D_{ij} \neq 0$

$$\Pr[ABr = Cr] = \Pr[Dr = \mathbf{0}] \leq \frac{2^{n-1}}{2^n} = \frac{1}{2}$$



$$(Dr)_i = \sum_{k=1}^{n} D_{ik} r_k = 0$$

$$r_j = -\frac{1}{D_{ij}} \sum_{k \neq j} D_{ik} r_k$$

Freivald's Algorithm:

pick a uniform random $r \in \{0,1\}^n$;

check whether $A(Br) = Cr$;

if $AB = C$: always correct

Theorem (Feivald 1979).

For $n \times n$ matrices A, B, C, if $AB \neq C$, for uniform random $r \in \{0,1\}^n$,

$$\Pr[ABr = Cr] \leq \frac{1}{2}$$

repeat independently for $O(\log n)$ times

Total running time: $O(n^2 \log n)$

Correct *with high probability* (w.h.p.).

Polynomial Identity Testing (PIT)

Input: two polynomials $f, g \in \mathbb{F}[x]$ of degree d.
Output: $f \equiv g$?

$\mathbb{F}[x]$: **polynomial ring** in x over **field** $\mathbb{F}$

$f \in \mathbb{F}[x]$ of degree d: $f(x) = \sum_{i=0}^{d} a_i x^i$ where $a_i \in \mathbb{F}$

field

Input: a polynomial $f \in \mathbb{F}[x]$ of degree d.
Output: $f \equiv 0$?

f is given as **black-box**

Input: a polynomial $f \in \mathbb{F}[x]$ of degree d.
Output: $f \equiv 0$?

- Deterministic algorithm (polynomial interpolation):

pick arbitrary *distinct* $x_0, x_1, \ldots, x_d \in \mathbb{F}$;
check if $f(x_i) = 0$ for all $0 \le i \le d$;

Fundamental Theorem of Algebra.

Any non-zero d-degree polynomial $f \in \mathbb{F}[x]$ has at most d roots.

- Randomized algorithm (fingerprinting):

pick a uniform random $r \in S$;
check if $f(r) = 0$;

let $S \subseteq \mathbb{F}$ be arbitrary
(whose size to be fixed later)

Input: a polynomial $f \in \mathbb{F}[x]$ of degree d.

Output: $f \equiv 0$?

pick a uniform random $r \in S$;

check if $f(r) = 0$;

let $S \subseteq \mathbb{F}$ be arbitrary

~~(whose size to be fixed later)~~

$$|S| = 2d$$

if $f \equiv 0$: always correct

if $f \not\equiv 0$:

$$\Pr[f(r) = 0] \le \frac{d}{|S|} = \frac{1}{2}$$

Fundamental Theorem of Algebra.

Any non-zero d-degree polynomial $f \in \mathbb{F}[x]$ has at most d roots.

Checking Identity

北京

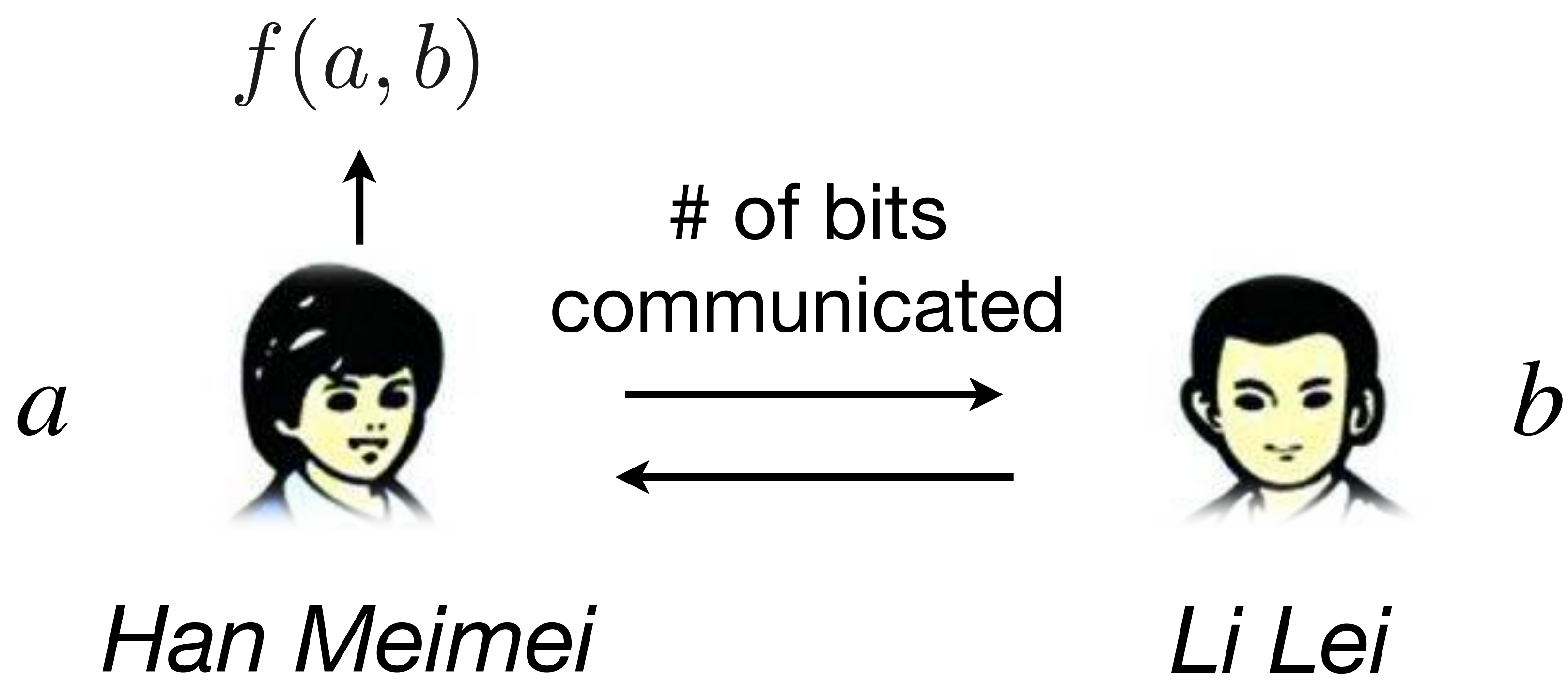
database 1

Are they identical?

南京

database 2

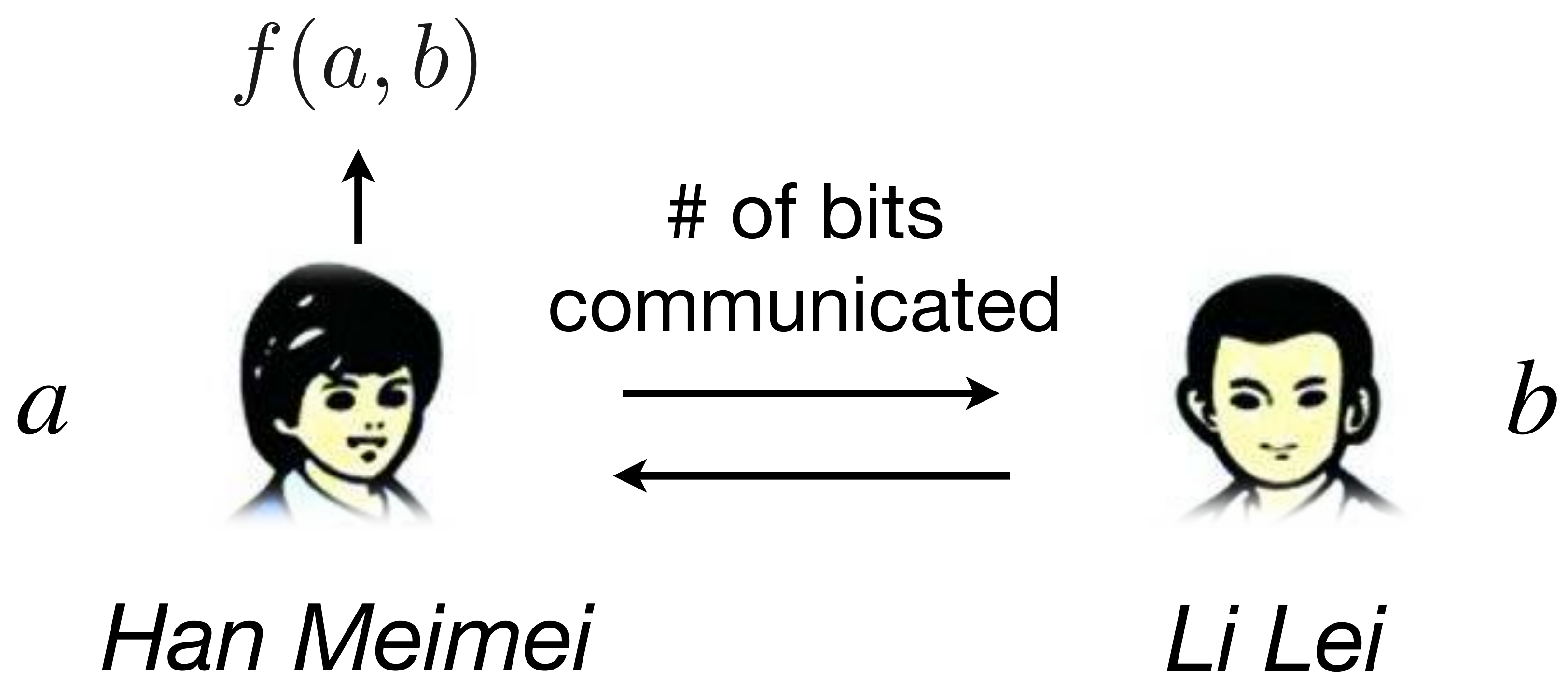
Communication Complexity



$$\mathrm{EQ} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$$

$$\mathrm{EQ}(a,b) = \begin{cases} 1 & a = b \\ 0 & a \neq b \end{cases}$$

Communication Complexity



$$\mathrm{EQ} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$$

Theorem (Yao 1979)**.**

Every deterministic communication protocol solving EQ communicates n bits in the worst-case.

Communication Complexity

$$f = \sum_{i=0}^{n-1} a_i x^i$$

$f(r) = g(r)$?

$$g = \sum_{i=0}^{n-1} b_i x^i$$

$a \in \{0,1\}^n$

$r, g(r)$ ←

$b \in \{0,1\}^n$

pick uniform random $r \in [2n]$

by **PIT**:

one-sided error $\leq \frac{1}{2}$

# of bit communicated: **too large!**

Communication Complexity

$$f = \sum_{i=0}^{n-1} a_i x^i$$

$f(r)=g(r)$?

$$g = \sum_{i=0}^{n-1} b_i x^i$$

$a \in \{0,1\}^n$

$r, g(r)$ ←

O(log n) bits

$b \in \{0,1\}^n$

pick uniform random $r \in [p]$

- choose a **prime** $p \in [n^2, 2n^2]$
- let $f, g \in \mathbb{Z}_p[x]$
- by **PIT**: one-sided error is $\frac{n}{p} = O\left(\frac{1}{n}\right)$ (correct w.h.p.)

Polynomial Identity Testing (PIT)

Input: a polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ of degree d.
Output: $f \equiv 0$?

$\mathbb{F}[x_1, \ldots, x_n]$: ring of n-variate polynomials in $x_1, \ldots, x_n$ over field $\mathbb{F}$

$f \in \mathbb{F}[x_1, \ldots, x_n]$:

$$f(x_1, \ldots, x_n) = \sum_{i_1, \ldots, i_n \geq 0} a_{i_1, i_2, \ldots, i_n} x_1^{i_1} x_2^{i_2} \cdots x_n^{i_n}$$

Degree of f: maximum $i_1 + i_2 + \cdots + i_n$ with $a_{i_1, i_2, \ldots, i_n} \neq 0$

Polynomial Identity Testing (PIT)

Input: a polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ of degree d.
Output: $f \equiv 0$?

$$f(x_1, \ldots, x_n) = \sum_{\substack{i_1, \ldots, i_n \geq 0 \\ i_1 + \cdots + i_n \leq d}} a_{i_1, i_2, \ldots, i_n} x_1^{i_1} x_2^{i_2} \cdots x_n^{i_n}$$

f is given as **black-box**: given any $\vec{x} \in \mathbb{F}^n$, return $f(\vec{x})$

or as product form: e.g. Vandermonde determinant

$$M = \begin{bmatrix} 1 & x_1 & x_1^2 & \ldots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \ldots & x_2^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \ldots & x_n^{n-1} \end{bmatrix}$$

$$f(\vec{x}) = \det(M) = \prod_{j<i} (x_i - x_j)$$

Polynomial Identity Testing (PIT)

Input: a polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ of degree d.
Output: $f \equiv 0$?

f is given as product form

if ∃ a *poly-time deterministic* algorithm for **PIT**:

⇒ either: $\mathbf{NEXP} \neq \mathbf{P/poly}$
or: $\mathbf{\#P} \neq \mathbf{FP}$

Input: a polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ of degree d.

Output: $f \equiv 0$?

Fix an arbitrary $S \subseteq \mathbb{F}$:

pick $r_1, \ldots, r_n \in S$ uniformly and independently at random;

check if $f(r_1, \ldots, r_n) = 0$;

$$f \equiv 0 \implies f(r_1, \ldots, r_n) = 0$$

Schwartz-Zippel Theorem.

$$f \not\equiv 0 \implies \Pr\left[f(r_1, \ldots, r_n) = 0\right] \leq \frac{d}{|S|}$$

# of roots for any $f \not\equiv 0$ in any cube S^n is $\leq d \cdot |S|^{n-1}$

Schwartz-Zippel Theorem.

$$f \not\equiv 0 \implies \Pr\left[f(r_1, \ldots, r_n) = 0\right] \le \frac{d}{|S|}$$

$$f(x_1, x_2, \ldots, x_n) = \sum_{\substack{i_1, i_2, \ldots, i_n \ge 0 \\ i_1 + i_2 + \cdots + i_n \le d}} a_{i_1, i_2, \ldots, i_n} x_1^{i_1} x_2^{i_2} \cdots x_n^{i_n}$$

f can be treated as a single-variate polynomial of x_n:

$$f(x_1, x_2, \ldots, x_n) = \sum_{i=0}^{d} x_n^i f_i(x_1, x_2, \ldots, x_{n-1})$$

$$= g_{x_1, x_2, \ldots, x_{n-1}}(x_n)$$

$$\Pr[f(r_1, r_2, \ldots, r_n) = 0] = \Pr[g_{r_1, r_2, \ldots, r_{n-1}}(r_n) = 0]$$

$$g_{r_1, r_2, \ldots, r_{n-1}} \not\equiv 0?$$

done?

Schwartz-Zippel Theorem.

$$f \not\equiv 0 \implies \Pr\left[f(r_1, \ldots, r_n) = 0\right] \leq \frac{d}{|S|}$$

induction on n :

basis: $n=1$ single-variate case, proved by the *fundamental Theorem of algebra*

I.H.: Schwartz-Zippel Thm is true for all smaller n

Schwartz-Zippel Theorem.

$$f \not\equiv 0 \implies \Pr\left[f(r_1, \ldots, r_n) = 0\right] \le \frac{d}{|S|}$$

induction step:

k: highest power of x_n in f ⇨ $\begin{cases} f_k \not\equiv 0 \\ \text{degree of } f_k \le d - k \end{cases}$

$$f(x_1, x_2, \ldots, x_n) = \sum_{i=0}^{k} x_n^i f_i(x_1, x_2, \ldots, x_{n-1})$$

$$= x_n^k f_k(x_1, x_2, \ldots, x_{n-1}) + \bar{f}(x_1, x_2, \ldots, x_n)$$

where $\bar{f}(x_1, x_2, \ldots, x_n) = \sum_{i=0}^{k-1} x_n^i f_i(x_1, x_2, \ldots, x_{n-1})$

highest power of x_n in $\bar{f}$ $< k$

Schwartz-Zippel Theorem.

$$f \not\equiv 0 \implies \Pr\left[f(r_1, \ldots, r_n) = 0\right] \le \frac{d}{|S|}$$

$$f(x_1, x_2, \ldots, x_n) = x_n^k f_k(x_1, x_2, \ldots, x_{n-1}) + \bar{f}(x_1, x_2, \ldots, x_n)$$

$$\begin{cases} f_k \not\equiv 0 \\ \text{degree of } f_k \le d-k \end{cases}$$

highest power of x_n in $\bar{f}$ $< k$

law of total probability:

$$\Pr[f(r_1, r_2, \ldots, r_n) = 0]$$

I.H. $\Rightarrow$ $\le \frac{d-k}{|S|}$

$$= \Pr[f(\vec{r}) = 0 \mid f_k(r_1, \ldots, r_{n-1}) = 0] \cdot \Pr[f_k(r_1, \ldots, r_{n-1}) = 0]$$

$$+ \Pr[f(\vec{r}) = 0 \mid f_k(r_1, \ldots, r_{n-1}) \neq 0] \cdot \Pr[f_k(r_1, \ldots, r_{n-1}) \neq 0]$$

$$= \Pr[g_{r_1, \ldots, r_{n-1}}(r_n) = 0 \mid f_k(r_1, \ldots, r_{n-1}) \neq 0] \le \frac{k}{|S|}$$

where $g_{x_1, \ldots, x_{n-1}}(x_n) = f(x_1, \ldots, x_n)$

Schwartz-Zippel Theorem.

$$f \not\equiv 0 \implies \Pr\left[f(r_1, \ldots, r_n) = 0\right] \leq \frac{d}{|S|}$$

$$\Pr[f(r_1, r_2, \ldots, r_n) = 0] \leq \frac{d-k}{|S|} + \frac{k}{|S|} = \frac{d}{|S|}$$

Input: a polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ of degree d.

Output: $f \equiv 0$?

Fix an arbitrary $S \subseteq \mathbb{F}$:

pick $r_1, \ldots, r_n \in S$ uniformly and independently at random;

check if $f(r_1, \ldots, r_n) = 0$;

$$f \equiv 0 \implies f(r_1, \ldots, r_n) = 0$$

Schwartz-Zippel Theorem.

$$f \not\equiv 0 \implies \Pr\left[f(r_1, \ldots, r_n) = 0\right] \le \frac{d}{|S|}$$

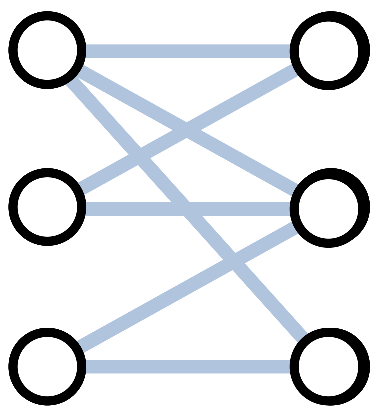
# of roots for any $f \not\equiv 0$ in any cube S^n is $\le d \cdot |S|^{n-1}$

Applications of *Schwartz-Zippel*

- test whether a graph has perfect matching;
- test isomorphism of rooted trees;
- distance property of Reed-Muller codes;
- proof of hardness vs randomness tradeoff;
- algebraic construction of ***probabilistically checkable proofs*** (**PCP**);
- ....

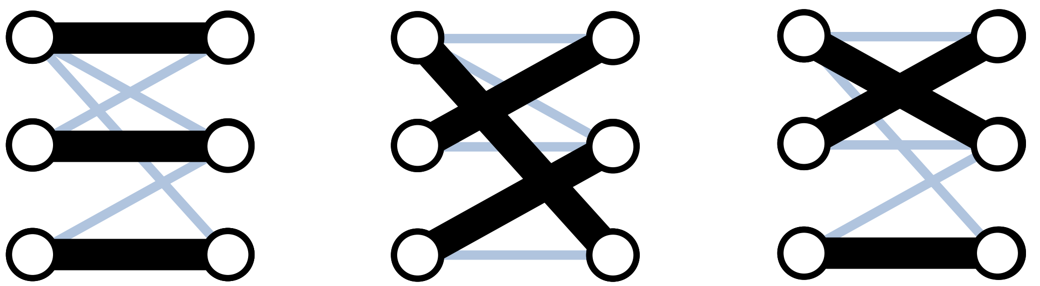
Bipartite Perfect Matching

bipartite graph

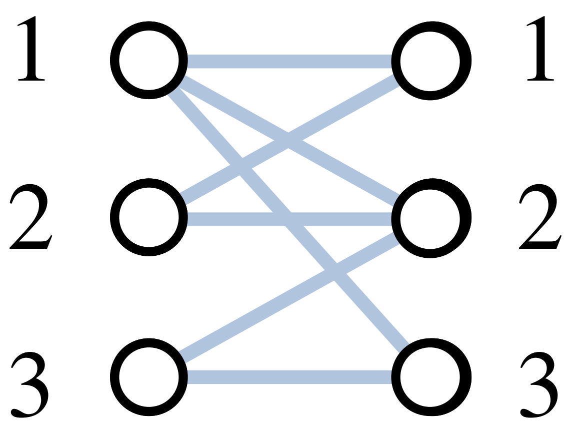


$G([n],[n],E)$

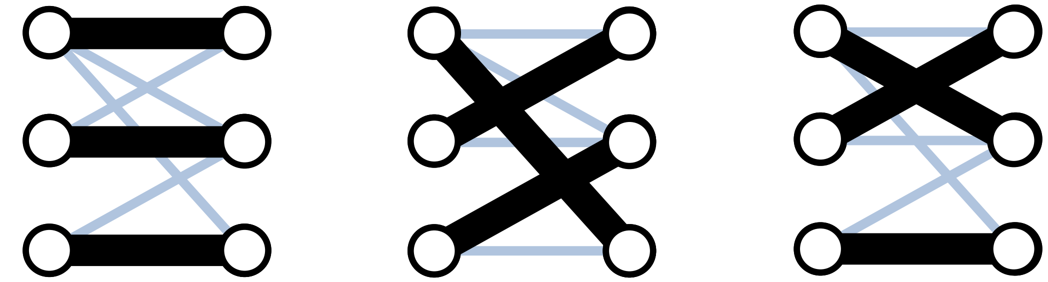
perfect matchings



- determine whether G has a perfect matching:
  - **Hall's theorem:** enumerates all subset of $[n]$
  - **Hungarian method:** $O(n^3)$
  - **Hopcroft-Karp algorithm:** $O(m\sqrt{n})$



$G([n],[n],E)$



$$A = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & 0 \\ 0 & x_{32} & x_{33} \end{bmatrix}$$

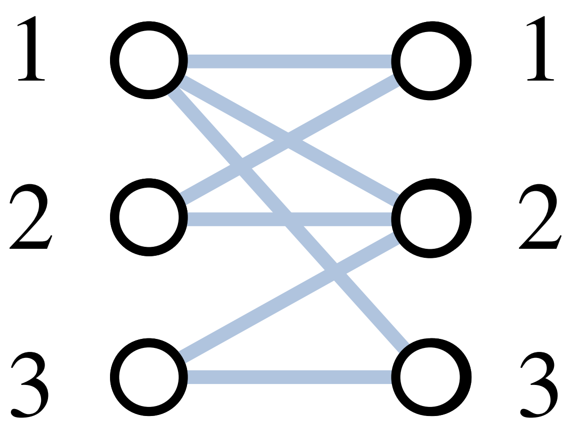
$$\det(A) = x_{11}x_{22}x_{33} + x_{13}x_{21}x_{32} - x_{12}x_{21}x_{33}$$

Edmonds matrix: an $n \times n$ matrix A defined as

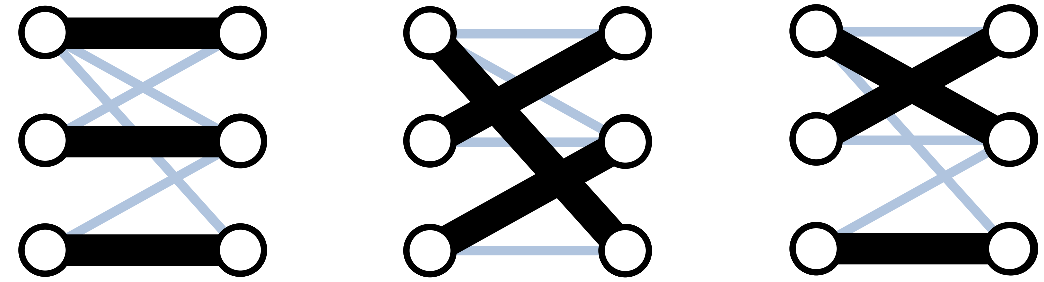
$$\forall i,j \in [n], \quad A(i,j) = \begin{cases} x_{i,j} & \text{if } (i,j) \in E \\ 0 & \text{if } (i,j) \notin E \end{cases}$$

Theorem: $\det(A) \not\equiv 0 \iff \exists$ a perfect matching in G

$$\det(A) = \sum_{\pi \in S_n} \mathrm{sgn}(\pi) \prod_{i \in [n]} A(i,\pi(i)) = \sum_{\pi \in S_n} \mathrm{sgn}(\pi) \begin{cases} \prod_{i \in [n]} x_{i,\pi(i)} & \pi \text{ is a P.M.} \\ 0 & \text{otherwise} \end{cases}$$



$G([n],[n],E)$



$$A = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & 0 \\ 0 & x_{32} & x_{33} \end{bmatrix}$$

$$\begin{aligned} \det(A) = & \ x_{11}x_{22}x_{33} \\ & + x_{13}x_{21}x_{32} \\ & - x_{12}x_{21}x_{33} \end{aligned}$$

Edmonds matrix: an $n \times n$ matrix A defined as

$$\forall i, j \in [n], \quad A(i,j) = \begin{cases} x_{i,j} & \text{if } (i,j) \in E \\ 0 & \text{if } (i,j) \notin E \end{cases}$$

Theorem: $\det(A) \not\equiv 0 \iff \exists$ a perfect matching in G

- $\det(A)$ is an m-variate degree-n polynomial:
  - Use *Schwartz-Zippel* to check whether $\det(A) \not\equiv 0$
  - Computing determinants is generic and can be done in parallel (Chistov's algorithm)

Fingerprinting

$$X \quad = \quad Y \quad ?$$

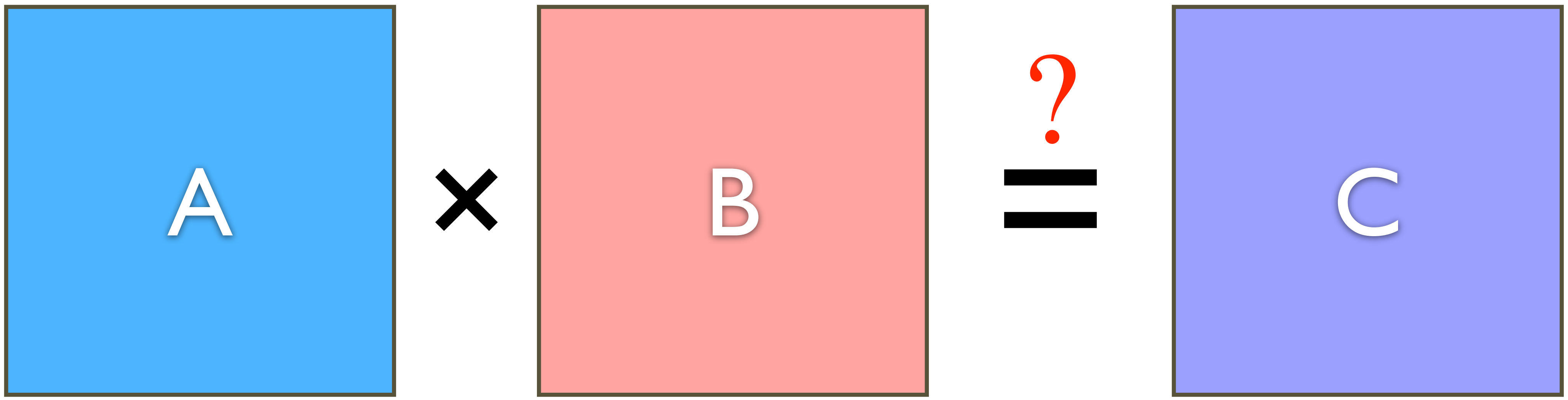
$$\downarrow \qquad\qquad \downarrow$$

$$\text{FING}(X) \quad = \quad \text{FING}(Y) \quad ?$$

- FING( ) is a function: $X = Y \implies \text{FING}(X) = \text{FING}(Y)$
- if $X \neq Y$, $\Pr[\ \text{FING}(X) = \text{FING}(Y)\ ]$ is small.
- Fingerprints are easy to compute and compare.

Checking Matrix Multiplication

- three $n \times n$ matrices A, B, C:



Freivald's Algorithm:

pick a uniform random $r \in \{0,1\}^n$;

check whether $A(Br) = Cr$;

For an $n \times n$ matrix M:

$$\text{FING}(M) = Mr \text{ for uniform random } r \in \{0,1\}^n$$

Polynomial Identity Testing (PIT)

Input: a polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ of degree d.
Output: $f \equiv 0$?

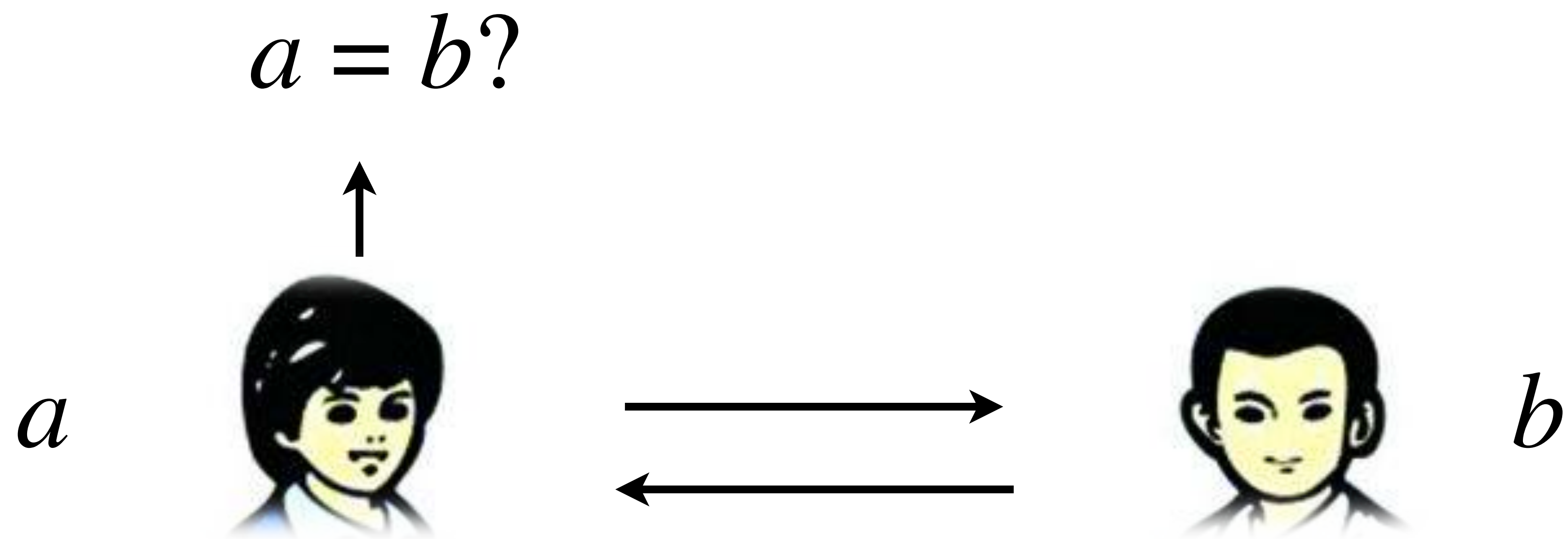
Fix an arbitrary $S \subseteq \mathbb{F}$:

pick $r_1, \ldots, r_n \in S$ uniformly and independently at random;
check if $f(r_1, \ldots, r_n) = 0$;

For a polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$:

FING$(f) = f(r_1, \ldots, r_n)$ for uniform independent $r_1, \ldots, r_n \in S$

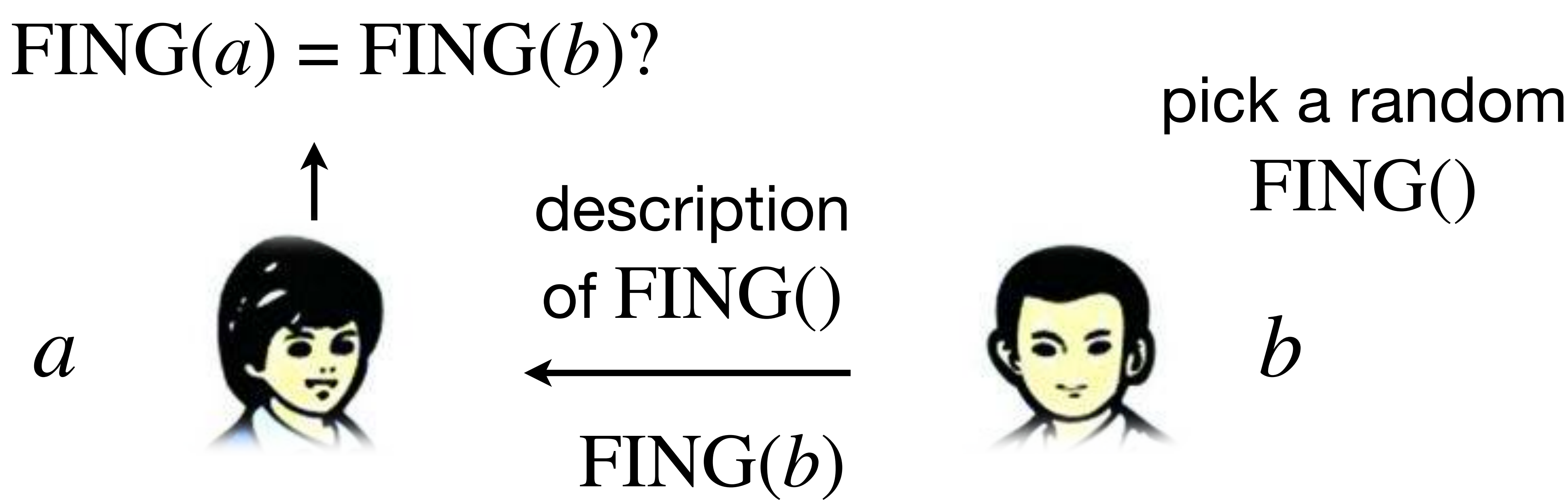
Communication Complexity



$$\mathrm{EQ} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$$

$$\mathrm{EQ}(a,b) = \begin{cases} 1 & a = b \\ 0 & a \neq b \end{cases}$$

Fingerprinting



- FING( ) is a function: $a = b \implies$ FING(a) = FING(b)
- if $a \neq b$, $\Pr[$ FING(a) = FING(b) $]$ is small.
- Fingerprints are short.

$$f = \sum_{i=0}^{n-1} a_i x^i$$

$f(r)=g(r)$?

$$g = \sum_{i=0}^{n-1} b_i x^i$$

$a \in \{0,1\}^n$

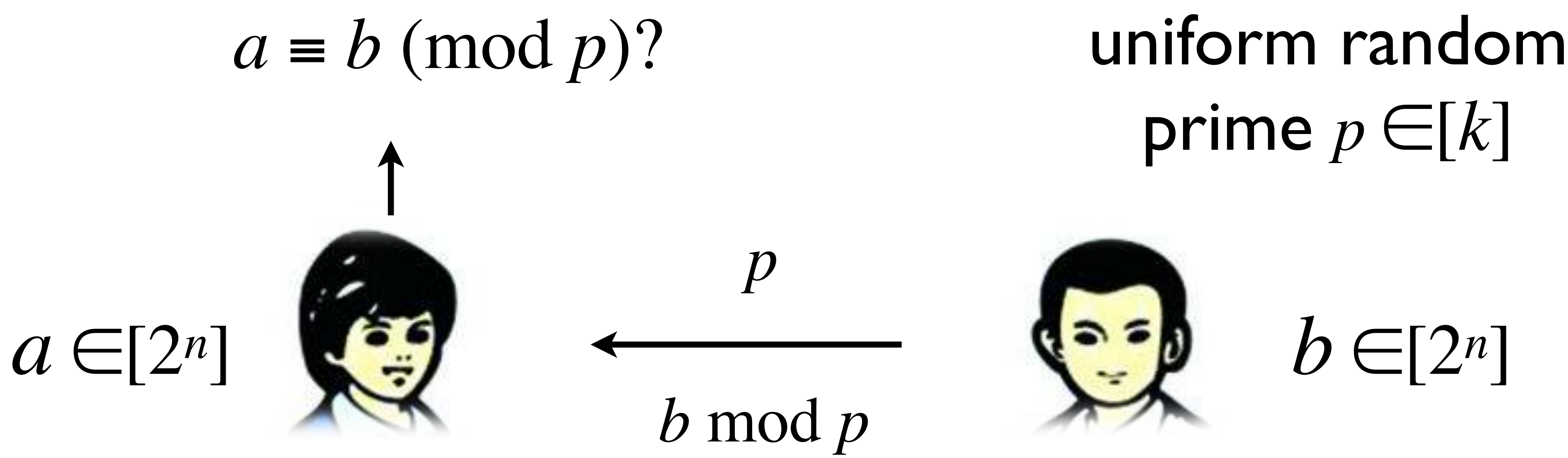
$r, g(r)$

$b \in \{0,1\}^n$

pick uniform
random $r \in [p]$

$f, g \in \mathbb{Z}_p[x]$ for a prime $p \in [n^2, 2n^2]$

$$\text{FING}(b) = \sum_{i=0}^{n-1} b_i r^i \text{ for random } r$$



FING(x) = $x \bmod p$ for uniform random prime $p \in [k]$

communication complexity: $O(\log k)$

if $a = b$ ⇒ $a \equiv b \pmod{p}$

if $a \neq b$: $\Pr[a \equiv b \pmod{p}] \leq ?$

for a $z = |a - b| \neq 0$: $\Pr[z \bmod p = 0] \leq ?$

uniform random prime $p \in [k]$

for a $z = |a - b| \neq 0$: $\Pr[z \bmod p = 0] \leq ?$

$\in [2^n]$

each prime divisor ≥ 2 } ⇒ # of prime divisors of $z \leq n$

$$\Pr[z \bmod p = 0] = \frac{\text{\# of prime divisors of } z}{\text{\# of primes in } [k]} \quad \begin{matrix} \leq n \\ = \pi(k) \end{matrix}$$

$\pi(N)$: # of primes in $[N]$

Prime Number Theorem (PNT):

$$\pi(N) \sim \frac{N}{\ln N} \text{ as } N \to \infty$$

$a \equiv b \pmod{p}$?

uniform random prime $p \in [k]$

$a \in [2^n]$

p

$\longleftarrow$

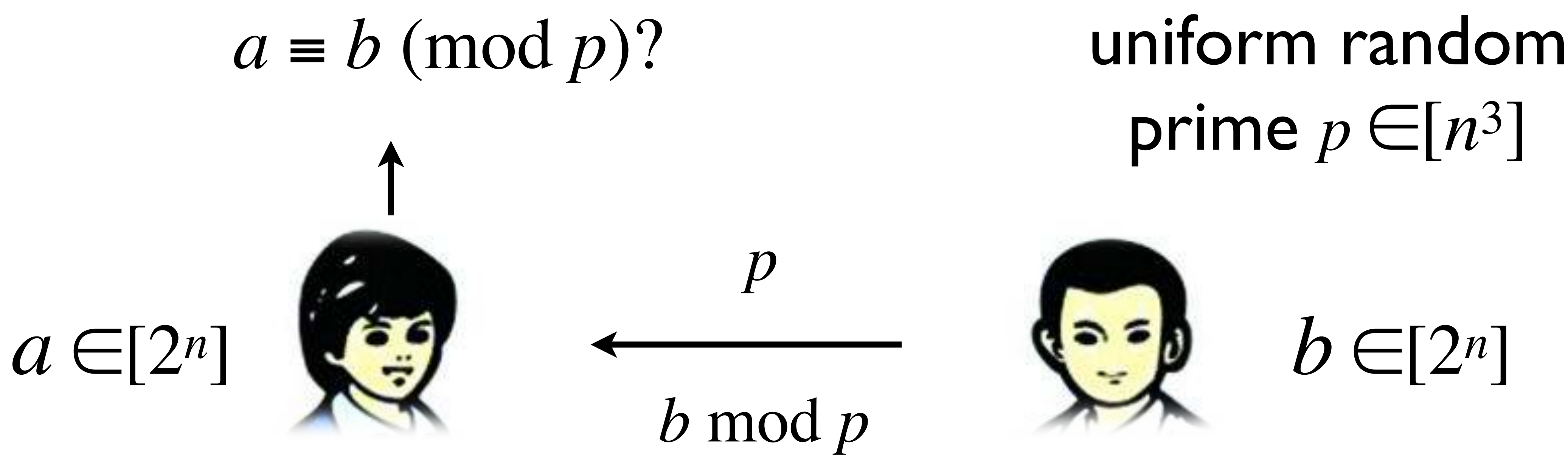
$b \bmod p$

$b \in [2^n]$

for a $z = |a - b| \neq 0$: $\Pr[z \bmod p = 0] \leq$?

$$\Pr[z \bmod p = 0] = \frac{\text{\# of prime divisors of } z}{\text{\# of primes in } [k]} \quad \begin{matrix} \leq n \\ = \pi(k) \end{matrix}$$

choose $k = n^3$ $\quad \leq \dfrac{n \ln k}{k} = \dfrac{3 \ln n}{n^2} = O\left(\dfrac{1}{n}\right)$



$\text{FING}(b) = b \bmod p$ for uniform random prime $p \in [n^3]$

communication complexity: $O(\log n)$

if $a = b$ ⇨ $a \equiv b \pmod{p}$

if $a \neq b$ ⇨ $\Pr[\, a \equiv b \pmod{p}\,] = O\left(\frac{1}{n}\right)$

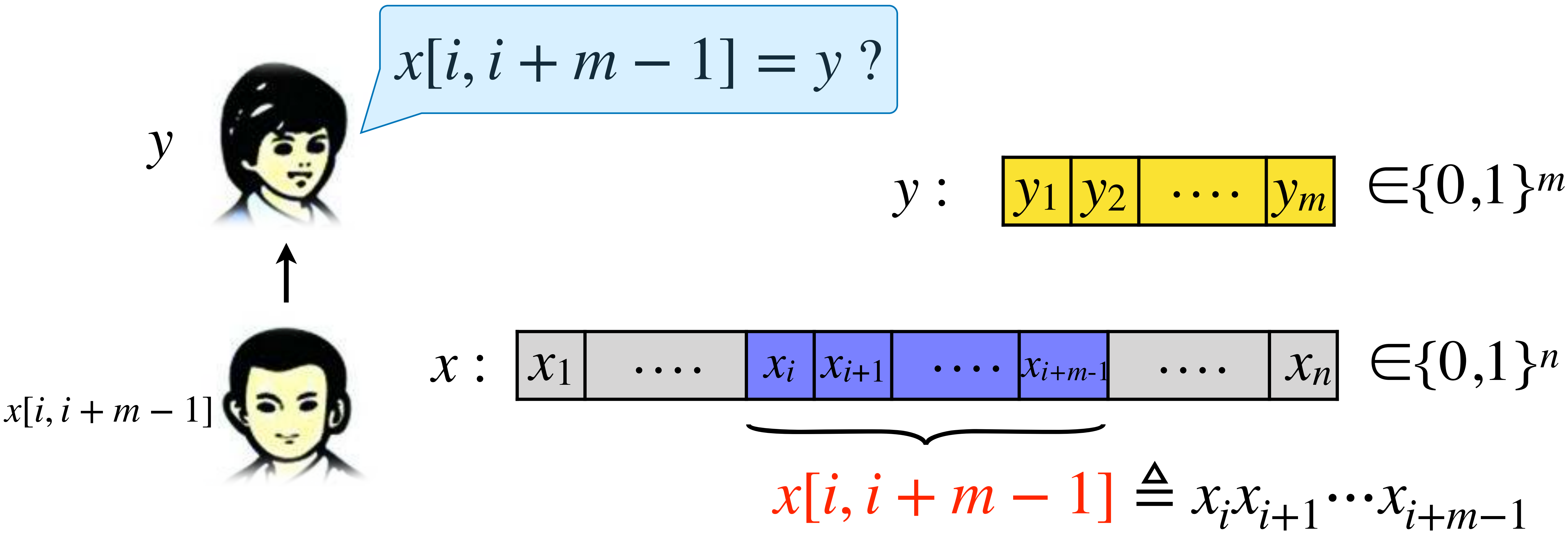
Pattern Matching

Input: string $x \in \{0,1\}^n$, pattern $y \in \{0,1\}^m$

Check whether y is a substring of x.

- naive algorithm: $O(mn)$ time
- **Knuth-Morris-Prat** (KMP) algorithm: $O(m+n)$ time
  - finite state automaton

Pattern Matching via Fingerprinting



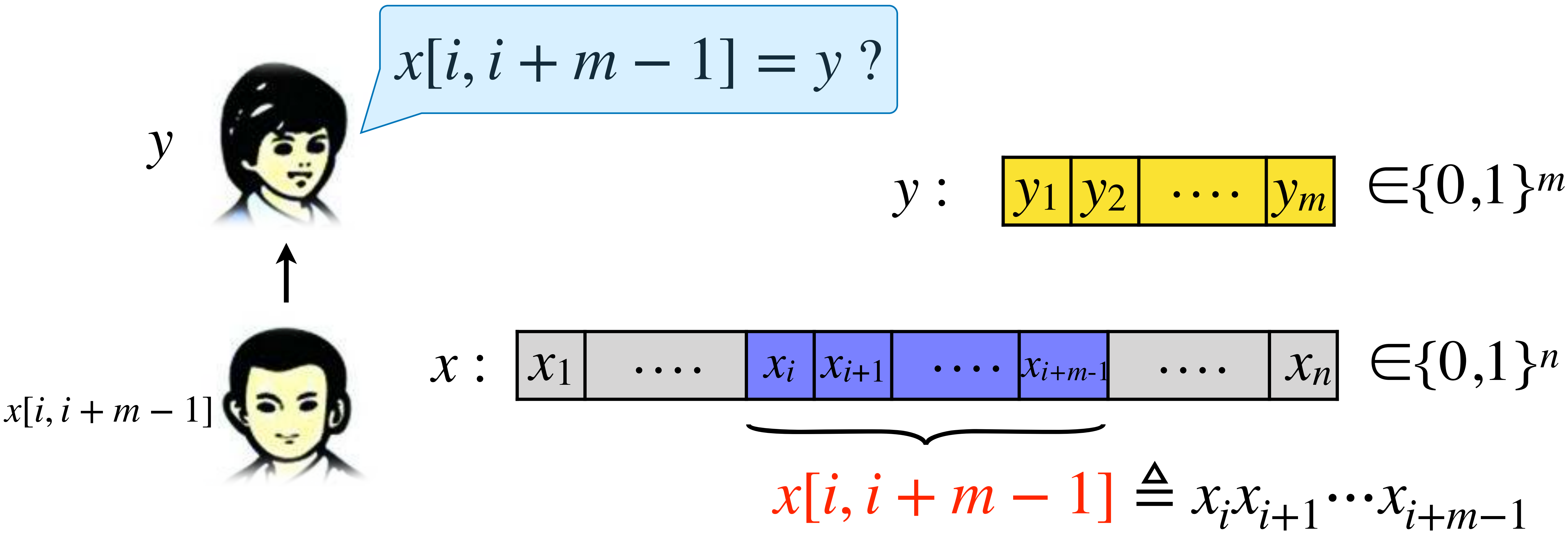
pick a random FING();

for $i = 1, 2, \ldots, n-m+1$ do:

  if FING($x[i, i+m-1]$) = FING(y) then return i;

return "*no match*";

Karp-Rabin Algorithm



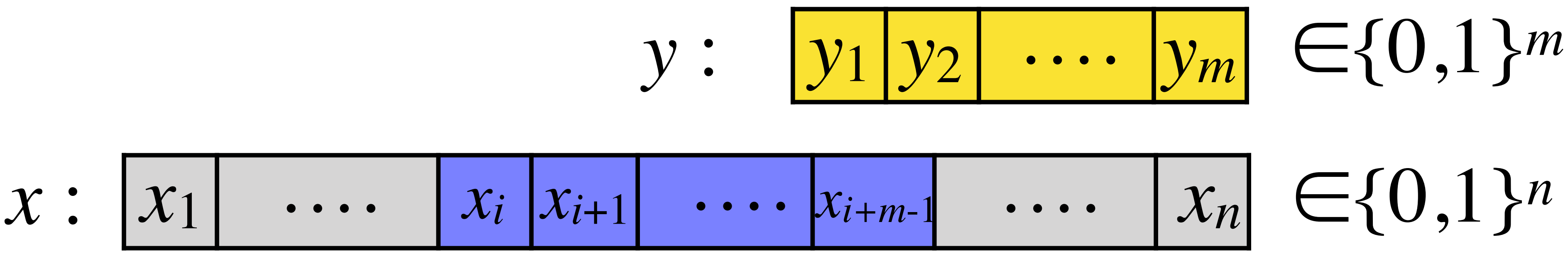
Karp-Rabin Algorithm: FING$(a) = a \bmod p$

pick a uniform random prime $p \in [mn^3]$;

for $i = 1,2,\ldots,n-m+1$ **do:**

  if $x[i, i+m-1] \equiv y \pmod{p}$ **then return** i;

return "*no match*";



Karp-Rabin Algorithm: FING$(a) = a \bmod p$

pick a uniform random prime $p \in [mn^3]$;

for $i = 1,2,\ldots,n-m+1$ **do:**

  if $x[i, i+m-1] \equiv y \pmod{p}$ **then return** i;

return "*no match*";

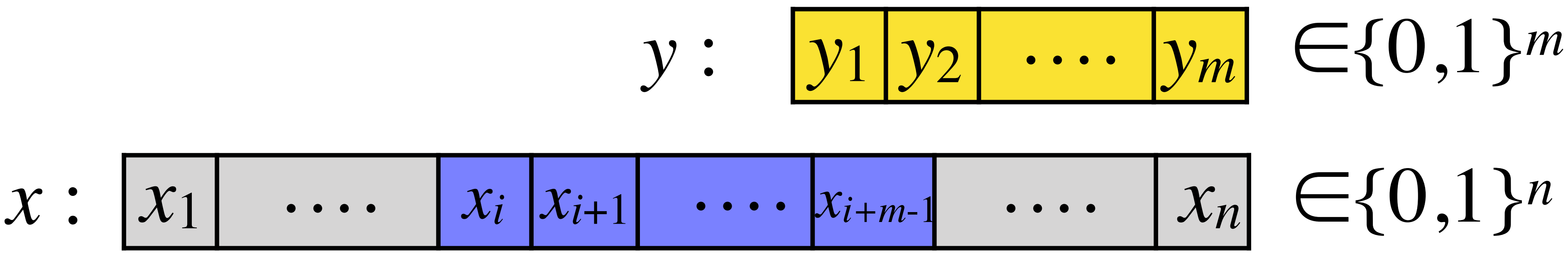
For each i, if $x[i, i+m-1] \neq y$:

$$\Pr\left[x[i, i+m-1] \equiv y \pmod{p}\right] \leq m \ln(mn^3)/mn^3 = o(1/n^2)$$

By **union bound:** when y is not a substring of x

$$\Pr[\text{ the algorithm ever makes a mistake }]$$

$$\leq \Pr\left[\exists i,\ x[i, i+m-1] \equiv y \pmod{p}\right] = o(1/n)$$



$$x[i, i+m-1] \triangleq x_i x_{i+1} \cdots x_{i+m-1}$$

Karp-Rabin Algorithm: FING$(a) = a \bmod p$

pick a uniform random prime $p \in [mn^3]$;

for $i = 1,2,\ldots,n-m+1$ do:

  if $x[i, i+m-1] \equiv y \pmod{p}$ then return i;

return "*no match*";

Testable in O(1) time

Observe: $x[i+1, i+m] = x_{i+m} + 2\left(x[i, i+m-1] - 2^{m-1}x_i\right)$

$$\text{FING}(x[i+1, i+m]) = \left(x_{i+m} + 2\left(\text{FING}(x[i, i+m-1]) - 2^{m-1}x_i\right)\right) \bmod p$$

Checking Distinctness

Input: n numbers $x_1, x_2, \ldots, x_n \in \{1,2,\ldots,n\}$

Determine whether every number appears exactly once.

$$A = \{x_1, x_2, \ldots, x_n\}$$
$$B = \{1, 2, \ldots, n\}$$

Input: two multisets $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_n\}$ where $a_1, \ldots, a_n, b_1, \ldots, b_n \in \{1,\ldots,n\}$

Output: $A = B$ (as multisets)?

$A = B \iff \forall x$: # of times x appearing in A = # of times x appearing in B

Input: two multisets $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_n\}$
where $a_1, \ldots, a_n, b_1, \ldots, b_n \in \{1, \ldots, n\}$

Output: $A = B$ (as multisets)?

- naive algorithm: use $O(n)$ time and $O(n)$ space
- fingerprinting: random fingerprint function FING( )
  - check $\mathrm{FING}(A) = \mathrm{FING}(B)$?
  - time cost: time to compute and check fingerprints $O(n)$
  - space cost: space to store fingerprints $O(\log p)$

multisets $A=\{a_1, a_2, \ldots, a_n\}$ ⇒ $f_A(x) = \prod_{i=1}^{n}(x - a_i)$

$f_A \in \mathbb{Z}_p[x]$ for prime p (to be specified)

$\mathrm{FING}(A) = f_A(r)$ for uniform random $r \in \mathbb{Z}_p$

multisets $A=\{a_1, a_2, \ldots, a_n\}$

$B=\{b_1, b_2, \ldots, b_n\}$

where $a_i, b_i \in \{1, 2, \ldots, n\}$

$$\Rightarrow \begin{cases} f_A(x) = \prod_{i=1}^{n}(x - a_i) \\ f_B(x) = \prod_{i=1}^{n}(x - b_i) \end{cases}$$

$f_A, f_B \in \mathbb{Z}_p[x]$ for prime p (to be specified)

$$\left.\begin{array}{l} \mathrm{FING}(A) = f_A(r) \\ \mathrm{FING}(B) = f_B(r) \end{array}\right\}$$ for uniform random $r \in \mathbb{Z}_p$

$A \neq B \implies f_A \not\equiv f_B$ on reals $\mathbb{R}$

(but possibly $f_A \equiv f_B$ on finite field $\mathbb{Z}_p$)

if $A = B$: $\mathrm{FING}(A) = \mathrm{FING}(B)$

if $A \neq B$: $\mathrm{FING}(A) = \mathrm{FING}(B)$

- $f_A \equiv f_B$ on finite field $\mathbb{Z}_p$ $\Rightarrow$ in $f_A - f_B$ on $\mathbb{R}$: $\exists$ coefficient $c \neq 0$, $c \bmod p = 0$
- $f_A \not\equiv f_B$ on $\mathbb{Z}_p$ but $f_A(r) = f_B(r)$ $\Rightarrow$ (Schwartz-Zippel) with probability $\leq n/p$

multisets $A=\{a_1, a_2, \ldots, a_n\}$

$B=\{b_1, b_2, \ldots, b_n\}$

where $a_i, b_i \in \{1, 2, \ldots, n\}$

$$\Rightarrow \begin{cases} f_A(x) = \prod_{i=1}^{n}(x - a_i) \\ f_B(x) = \prod_{i=1}^{n}(x - b_i) \end{cases}$$

$f_A, f_B \in \mathbb{Z}_p[x]$ for uniform random prime $p \in [L, U]$

(L, U to be specified)

$$\left.\begin{array}{l} \mathrm{FING}(A) = f_A(r) \\ \mathrm{FING}(B) = f_B(r) \end{array}\right\}$$ for uniform random $r \in \mathbb{Z}_p$

if $A \neq B$: $\mathrm{FING}(A) = \mathrm{FING}(B)$

- $f_A \equiv f_B$ on finite field $\mathbb{Z}_p$ ⇒

  in $f_A - f_B$ on $\mathbb{R}$:
  ∃ coefficient $c \neq 0$
  $c \bmod p = 0$

$$\Pr[\, c \bmod p = 0 \,] \leq \frac{\#\text{ of prime factors of } c}{\#\text{ of primes in } [L, U]}$$

$|c| \leq n^n$ ⇒ $$\leq \frac{n \log_2 n}{\pi(U) - \pi(L)} \sim \frac{n \log_2 n}{U/\ln U - L/\ln L}$$

- $f_A \not\equiv f_B$ on $\mathbb{Z}_p$ but $f_A(r) = f_B(r)$ ⇒ (Schwartz-Zippel) with probability $\leq n/p \leq n/L$

multisets $A=\{a_1, a_2, \ldots, a_n\}$

$B=\{b_1, b_2, \ldots, b_n\}$

where $a_i, b_i \in \{1, 2, \ldots, n\}$

$$\begin{cases} f_A(x) = \prod_{i=1}^{n}(x - a_i) \\ f_B(x) = \prod_{i=1}^{n}(x - b_i) \end{cases}$$

$f_A, f_B \in \mathbb{Z}_p[x]$ for uniform random prime $p \in [L, U]$

with $U = 2L = (n \log n)^2$

$\left.\begin{array}{l} \text{FING}(A) = f_A(r) \\ \text{FING}(B) = f_B(r) \end{array}\right\}$ for uniform random $r \in \mathbb{Z}_p$

if $A \neq B$: $\text{FING}(A) = \text{FING}(B)$

- $f_A \equiv f_B$ on finite field $\mathbb{Z}_p$ → with probability $\leq \frac{n \log_2 n}{U/\ln U - L/\ln L} = O(1/n)$
- $f_A \not\equiv f_B$ on $\mathbb{Z}_p$ but $f_A(r) = f_B(r)$ → (Schwartz-Zippel) with probability $\leq n/p \leq n/L$ $= O(1/n)$

Input: two multisets $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_n\}$ where $a_1, \ldots, a_n, b_1, \ldots, b_n \in \{1, \ldots, n\}$

Output: $A = B$ (as multisets)?

Lipton's Algorithm (1989):

$$\mathrm{FING}(A) = \prod_{i=1}^{n} (r - a_i) \bmod p$$

$$\mathrm{FING}(B) = \prod_{i=1}^{n} (r - b_i) \bmod p$$

for uniform random prime $p \in [(n \log n)^2/2, (n \log n)^2]$ and uniform random $r \in \mathbb{Z}_p$

if $A \neq B$ as multisets:

$$f_A(x) = \prod_{i=1}^{n} (x - a_i) \bmod p \qquad f_B(x) = \prod_{i=1}^{n} (x - b_i) \bmod p$$

$$\Pr[\ \mathrm{FING}(A) = \mathrm{FING}(B)\ ]$$

$$\leq \Pr[\ f_A \equiv f_B\ ] + \Pr[\ f_A(r) = f_B(r) \mid f_A \not\equiv f_B\ ] = O(1/n)$$

Input: n numbers $x_1, x_2, \ldots, x_n \in \{1, 2, \ldots, n\}$

Determine whether every number appears exactly once.

Lipton's Algorithm (1989):

$$\mathrm{FING}(A) = \prod_{i=1}^{n} (r - a_i) \bmod p$$

check if:

$$\mathrm{FING}(A) = \prod_{i=1}^{n} (r - i) \bmod p?$$

for uniform random prime $p \in [(n \log n)^2/2, (n \log n)^2]$ and uniform random $r \in \mathbb{Z}_p$

- time cost: $O(n)$
- space cost: $O(\log n)$
- error probability (false positive): $O(1/n)$
- data stream: input comes one at a time