Randomized Algorithms

南京大学

尹一通
Polynomial Identity Testing (PIT)

**Input:** \( f, g \in \mathbb{F}[x_1, x_2, \ldots, x_n] \) of degree \( d \)

**Output:** \( f \equiv g \)?

\( \mathbb{F}[x_1, x_2, \ldots, x_n] \): ring of \( n \)-variate polynomials over field \( \mathbb{F} \)

\( f \in \mathbb{F}[x_1, x_2, \ldots, x_n] \):

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

**degree of** \( f \): maximum \( i_1 + i_2 + \cdots + i_n \) with \( a_{i_1, i_2, \ldots, i_n} \neq 0 \)
Input: $f, g \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ of degree $d$

Output: $f \equiv g$?

equivalently:

Input: $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ of degree $d$

Output: $f \equiv 0$?

$f$ is given as block-box: given any $\vec{x} = (x_1, x_2, \ldots, x_n)$

returns $f(\vec{x})$

or as product from: e.g. Vandermonde determinant

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

$$f(\vec{x}) = \det(M) = \prod_{j<i} (x_i - x_j)$$
Input: $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ of degree $d$

Output: $f \equiv 0?$

**fix an arbitrary** $S \subseteq \mathbb{F}$

pick random $r_1, r_2, \ldots, r_n \in S$

uniformly and independently at random;

check whether $f(r_1, r_2, \ldots, r_n) = 0$;

$f \equiv 0 \implies f(r_1, r_2, \ldots, r_n) = 0$
A degree polynomial has at most roots.

**Fundamental Theorem of Algebra:**
A degree $d$ polynomial has at most $d$ roots.

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

**Output:**
- $f \equiv 0$?

1. fix an arbitrary $S \subseteq \mathbb{F}$
2. pick a uniform random $r \in S$;
3. check whether $f(r) = 0$;

$$f \equiv 0 \implies f(r) = 0$$

$$f \not\equiv 0 \implies \Pr[f(r) = 0] \leq \frac{d}{|S|}$$
Input: \( f \in F[x_1, x_2, \ldots, x_n] \) of degree \( d \)
Output: \( f \equiv 0? \)

- fix an arbitrary \( S \subseteq F \)
- pick random \( r_1, r_2, \ldots, r_n \in S \)
  - uniformly and independently at random;
- check whether \( f(r_1, r_2, \ldots, r_n) = 0 \);

\[
f \equiv 0 \quad \rightarrow \quad f(r_1, r_2, \ldots, r_n) = 0
\]

Schwartz-Zippel Theorem

\[
f \neq 0 \quad \rightarrow \quad \Pr[f(r_1, r_2, \ldots, r_n) = 0] \leq \frac{d}{|S|}
\]
Schwartz-Zippel Theorem

\[ f \neq 0 \implies \Pr[f(r_1, r_2, \ldots, r_n) = 0] \leq \frac{d}{|S|} \]

**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

\[ f \neq 0 \implies \Pr[f(r_1, r_2, \ldots, r_n) = 0] \leq \frac{d}{|S|} \]

induction step:

\[ k: \text{ highest power of } x_n \text{ in } f \implies \begin{cases} f_k \neq 0 \\ \text{degree of } f_k \leq d - k \end{cases} \]

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

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

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

\[ f \neq 0 \quad \rightarrow \quad \Pr[f(r_1, r_2, \ldots, r_n) = 0] \leq \frac{d}{|S|} \]

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

\[ \left\{ \begin{array}{l}
 f_k \neq 0 \\
 \text{degree of } f_k \leq d - k
\end{array} \right. \quad \text{highest power of } x_n \text{ in } \bar{f} < k \]

Law of total probability:

\[ \Pr[f(r_1, r_2, \ldots, r_n) = 0] = \Pr[f(\bar{r}) = 0 \mid f_k(r_1, \ldots, r_{n-1}) = 0] \cdot \Pr[f_k(r_1, \ldots, r_{n-1}) = 0] \]

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

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

where \[ g_{x_1, \ldots, x_{n-1}}(x_n) = f(x_1, \ldots, x_n) \]
Schwartz-Zippel Theorem

\[ f \neq 0 \quad \Rightarrow \quad \Pr[f(r_1, r_2, \ldots, r_n) = 0] \leq \frac{d}{|S|} \]

\[
\Pr[f(r_1, r_2, \ldots, r_n) = 0] \leq \frac{d - k}{|S|} + \frac{k}{|S|} = \frac{d}{|S|}
\]
Input: \( f \in \mathbb{F}[x_1, x_2, \ldots, x_n] \) of degree \( d \)

Output: \( f \equiv 0? \)

fix an arbitrary \( S \subseteq \mathbb{F} \)

pick random \( r_1, r_2, \ldots, r_n \in S \) \( \text{uniformly and independently at random; } \)

check whether \( f(r_1, r_2, \ldots, r_n) = 0 \) ;

\[
\begin{align*}
f \equiv 0 & \implies f(r_1, r_2, \ldots, r_n) = 0
\end{align*}
\]

Schwartz-Zippel Theorem

\[
\begin{align*}
f \not\equiv 0 & \implies \Pr[f(r_1, r_2, \ldots, r_n) = 0] \leq \frac{d}{|S|}
\end{align*}
\]
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

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

determine whether a bipartite graph has PM:

- Hall’s theorem: enumerates all subset of \([n]\)
- Hungarian method: \(O(n^3)\)
- Hopcroft-Karp algorithm: \(O(m\sqrt{n})\)
- efficient parallel algorithms?
Proof of perfect matchings in bipartite graphs.

Consider a bipartite graph with bi-

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

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

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

**Edmond matrix:**

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

\[ \det(A) = \sum_{\pi \in S_n} \text{sgn}(\pi) \prod_{i \in [n]} A(i, \pi(i)) = \sum_{\pi \in S_n} \text{sgn}(\pi) \prod_{i \in [n]} x_{i,\pi(i)} \]

\[ \neq 0 \quad \text{if and only if } \exists \text{ a perfect matching} \]
Proof of every node being given as an expression that can be evaluated in time if the corresponding edges (namely, succinctly describes the problem as iterating over all permutations and checking whether there exists a perfect matching).

Consider a bipartite graph with biadjacency matrix \( A \), where
\[
A(i, j) = \begin{cases} 
  x_{i,j} & \text{if } (i, j) \in E \\
  0 & \text{if } (i, j) \notin E
\end{cases}
\]

\[\det(A) \neq 0\]

if and only if \( \exists \) a perfect matching.

\( \det(A) \) is an \( n \)-variate degree-\( n \) polynomial:

- use Schwartz-Zippel to check whether \( \det(A) \neq 0 \)
- there are fast parallel algorithms for computing determinants of numerical matrices (Chistov’s algorithm)
Isomorphism of Rooted Trees
Isomorphism of Rooted Trees

associate each tree $T$ a polynomial:

$T$ is of height $k$

subtrees rooted by children of the root of $T$: $T_1, T_2, \ldots, T_m$

\[
 f_T = (x_k - f_{T_1})(x_k - f_{T_2}) \cdots (x_k - f_{T_m})
\]

by induction and the uniqueness of polynomial factorization:

\[
 f_T \equiv f_{T'} \quad \iff \quad T \cong T'
\]
Fingerprinting

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

\[ X = Y \quad ? \]
\[ \downarrow \quad \downarrow \]
\[ \text{FING(} X \text{)} = \text{FING(} Y \text{)} \quad ? \]
Checking Matrix Multiplication

three \( n \times n \) matrices \( A, B, C \):

\[
\begin{array}{ccc}
A & \times & B \\
\downarrow & & \downarrow \\
\text{?} & = & C
\end{array}
\]

Freivald’s Algorithm

pick a \textbf{uniform} random \( r \in \{0,1\}^n \);
check whether \( A(Br) = Cr \);

matrix \( M \):

\[
\text{FING}(M) = Mr \text{ for uniform random } r \in \{0,1\}^n
\]
Polynomial Identity Testing (PIT)

Input: \( f \in \mathbb{F}[x_1, x_2, \ldots, x_n] \) of degree \( d \)

Output: \( f \equiv 0? \)

- fix an arbitrary \( S \subseteq \mathbb{F} \)
- pick random \( r_1, r_2, \ldots, r_n \in S \) \textit{uniformly and independently} at random;
- check whether \( f(r_1, r_2, \ldots, r_n) = 0 \);

polynomial \( f \):

\[
\text{FING}(f) = f(r_1, r_2, \ldots, r_n) \text{ for uniform\&independent } r_1, \ldots, r_n \in S
\]
Communication Complexity

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

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

FING(a) = FING(b)?

• FING( ) is a function: \( a = b \Rightarrow FING(a) = FING(b) \)

• if \( a \neq b \), \( \Pr[FING(a) = FING(b)] \) is small.

• Fingerprints are easy to compute and compare.
\[ f = \sum_{i=0}^{n-1} a_i x^i \quad f(r)=g(r) \ ? \quad g = \sum_{i=0}^{n-1} b_i x^i \]

\[ a \in \{0, 1\}^n \quad r, g(r) \quad b \in \{0, 1\}^n \]

pick uniform random \( r \in [2n] \)

\[ f, g \in \mathbb{Z}_p[x] \]

\[ k = \lfloor \log_2(2n) \rfloor \]

\[ p \in [2^k, 2^{k+1}] \]

\[ \text{FING}(b) = \sum_i b_i r^i \text{ for random } r \]
\[ a \equiv b \pmod{p} \text{?} \]

Uniform random prime \( p \in [k] \)

\[ a \in [2^n] \quad \quad \quad b \in [2^n] \]

\[ \text{FING}(x) = x \mod p \text{ for uniform random prime } p \in [k] \]

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 \( z = |a - b| \neq 0 \):

\[ \Pr[z \mod p = 0] \leq ? \]
uniform random prime $p \in [k]$

for a $z = |a - b| \neq 0$:

$$\Pr[z \mod p = 0] \leq ?$$

$\in [2^n]$

# of prime divisors of $z \leq n$

each prime divisor $\geq 2$

$$\Pr[z \mod p = 0] = \frac{\text{# of prime divisors of } z}{\text{# of primes in } [k]} = \pi(k)$$

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

**Prime Number Theorem (PNT)**

$$\pi(n) \sim \frac{N}{\ln N} \quad \text{as } N \to \infty$$
for a $z = |a - b| \neq 0$: $\Pr[z \mod p = 0] \leq \frac{\# \text{ of prime divisors of } z}{\# \text{ of primes in } [k]} = \pi(k)$

choose $k = n^2 \leq \frac{n \ln k}{k} = \frac{2 \ln n}{n}$
$a \equiv b \pmod{p}$?

$a \in [2^n]$  
$b \in [2^n]$  
uniform random prime $p \in [n^2]$  

$FING(b) = b \mod p$ for uniform random prime $p \in [n^2]$  

communication complexity: $O(\log n)$  

if $a = b$  
$a \equiv b \pmod{p}$  

if $a \neq b$  
$\Pr[a \equiv b \pmod{p}] \leq \left(2 \ln n\right) / n$
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
- the Knuth-Morris-Prat algorithm: $O(m+n)$ time
Pattern Matching

\[ x(i) = y? \]

\[ y : \begin{array}{cccc} y_1 & y_2 & \cdots & y_m \end{array} \in \{0,1\}^m \]

\[ x : \begin{array}{cccc} x_1 & \cdots & x_i & x_{i+1} & \cdots & x_{i+m-1} & \cdots & x_n \end{array} \in \{0,1\}^n \]

\[ \text{denote } x(i) = x_i x_{i+1} \ldots x_{i+m-1} \]

FING(a) = a \mod p

pick a random FING();
for \( i=1, 2, \ldots, n-m+1; \)
if FING(x(i)) = FING(y) then return “match!”;
return “not match!”;
Pattern Matching

$x(i) = y$?

$y : \ y_1 \ y_2 \ \cdots \ y_m \ \in \{0,1\}^m$

$x : \ \begin{array}{cccccc}
    x_1 & \cdots & x_i & x_{i+1} & \cdots & x_{i+m-1} & \cdots & x_n
  \end{array} \ \in \{0,1\}^n$

$x(i)$

denote $x(i) = x_i x_{i+1} \ldots x_{i+m-1}$

**Karp-Rabin Algorithm:**

1. pick a uniform random prime $p \in [mn^2]$;
2. for $i=1, 2, \ldots, n-m+1$;
   - if $x(i) \equiv y \pmod{p}$ then return "match!";
3. return "not match!";

$\text{FING}(a) = a \mod p$
\[ y : \quad y_1 \ y_2 \ \cdots \ y_m \in \{0,1\}^m \]

\[ x : \quad x_1 \ \cdots \ x_i \ x_{i+1} \ \cdots \ x_{i+m-1} \ \cdots \ x_n \in \{0,1\}^n \]

**Karp-Rabin Algorithm:**

- pick a uniform random prime \( p \in [mn^2] \);
- for \( i=1, 2, \ldots, n-m+1 \);
  - if \( x(i) \equiv y \pmod{p} \) then return "match!";
- return "not match!";

for each \( i \), if \( x(i) \neq y \)

\[
\Pr[ x(i) \equiv y \pmod{p} ] = m \ln(mn^2) / mn^2 = o(1/n)
\]

\[
\Pr[ \text{mistake} ] = \Pr[ \exists i, x(i) \neq y \land x(i) \equiv y \pmod{p} ] = o(1)
\]

union bound
Karp-Rabin Algorithm:

pick a uniform random prime \( p \in [mn^2] \);
for \( i = 1, 2, \ldots, n-m+1; \)
    if \( x(i) \equiv y \pmod{p} \) then return "match!";
return "not match!"

\[ FING(a) = a \mod p \]

\[ x(i+1) = x_{i+m} + 2(x(i) - 2^{m-1}x_i) \]

\[ FING(x(i+1)) = (x_{i+m} + 2(FING(x(i)) - 2^{m-1}x_i)) \mod p \]

"\( x(i) \equiv y \pmod{p} \)" can be tested in constant time