随机指纹

用少量信息完成快速验证

随机算法部分的第一项主题

第一讲借助图割勾勒了课程的三条主线；从本讲开始，我们正式进入随机算法与 随机化技术。贯穿全讲的做法是：不再完整搬运或比较一个庞大对象，而是随机 选取一个很短的“指纹”。相同对象在任何随机选择下都得到相同指纹；不同对象 虽可能碰撞，但碰撞概率可以严格控制。矩阵乘法检验、多项式恒等检验、通信 复杂度、完美匹配、字符串匹配和流式算法，都将由这一条思路串联起来。

高级算法课程教学团队

南京大学

学习目标与阅读建议

完成本讲后，你应当能够：

1. 证明 Freivalds 矩阵乘法检验的单侧错误界，并进行独立重复放大；
2. 区分形式多项式与有限域上的多项式函数，完整证明 Schwartz–Zippel 定理；
3. 为相等性通信问题设计只需 $O(\log n)$ 位的随机协议；
4. 识别行列式、滚动余数和根多项式这三类代数指纹；
5. 在陈述结论时同时交代随机性的来源、错误方向、域大小以及计算模型。

第一次阅读可先掌握sections 1 and 2、 定理 3.3的证明、第 4 节中的多项式协议，再从完美匹配、 Karp–Rabin 与多重集合三节中选择一节精读。随机模素数的两种算法是补充视角。附录只收录概率、素数计数、有限域和多项式代数等数学工具，所有高级算法都在 正文中讲完。

除非另有说明，代数算法的时间按“域运算次数”计算；涉及通信、空间与字长 RAM 时，则会单独说明每个数需要多少位。通信和空间界中的对数以 2 为底。 对正整数 n，记 $[n] = \{1, \dots, n\}$。

Contents

1 先丢掉信息，再保留判断能力

设 X, Y 来自一个很大的对象空间 $\mathcal{U}$。随机指纹并不是无损压缩：从 指纹通常无法恢复原对象。它只希望保留当前问题真正需要的信息——“两者是否 相等”。形式化地说，随机种子 s 从一个指纹函数族中选出 $h_s : \mathcal{U} \to \mathcal{R}$，并满足：

(i) 若 $X = Y$，则对每个 s 都有 $h_s(X) = h_s(Y)$；

(ii) 若 $X \neq Y$，则 $\mathbb{P}_s[h_s(X) = h_s(Y)] \leq \varepsilon$；

(iii) 种子、指纹和求值过程都远小于原对象的完整表示。

指纹值域小，就不可能对所有对象保持一一对应；碰撞不可避免。随机化的作用 不是消灭碰撞，而是让事先固定的一对不同对象难以预知本次使用的碰撞方式。 本文默认输入先固定、随机种子后抽取。若对手可以看到种子后再选择输入，就需要 使用新的或保密的种子，并重新证明碰撞界。

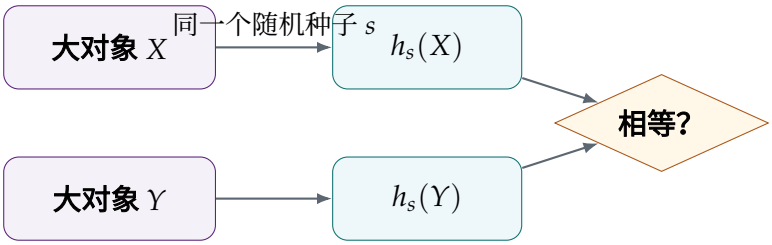

图 1: 随机指纹把一次大对象比较变成两次求值和一次短值比较。证明的核心是 数清楚哪些随机选择会造成碰撞。

对象或断言	随机指纹	碰撞为何稀少
AB **与** C	$(AB)r$ 与 Cr	非零矩阵的一行给出一个非平凡线性方程
多项式 f **与零多项式**	$f(r_1, \ldots, r_m)$	非零低次多项式在网格中的零点有限
二进制串 x, y	多项式取值或模素数余数	差多项式根少，差整数的素因子少
多重集合 A, B	$\prod_{a \in A}(r - a)$	元素重数变成根的重数

表 1: 每个应用都先找出“不相等”的代数见证，再估计随机选择隐藏该见证的概率。

先说清楚哪一边可能出错

比较两个对象时，相同对象绝不会被判成不同；不同对象却可能碰撞，所以只能 “误报相等”。完美匹配一节的错误方向恰好反过来：没有完美匹配的图一定被拒绝， 有完美匹配的图却可能在随机赋值后得到零行列式而被漏报。只写“以高概率正确” 会掩盖算法最重要的结构。

2 验证未必需要重算：Freivalds 算法

给定 $A, B, C \in \mathbb{F}^{n \times n}$，任务是检查 $AB = C$。经典矩阵乘法需要 $O(n^3)$ 次域运算。Freivalds 的观察是：若两个矩阵相等，它们作用在任意向量 上都应得到相同结果；为了发现一个错误等式，我们不必一次检查所有向量 [3]。

Freivalds(A, B, C)

1. 从 $\{0,1\}^n$ 中均匀选取随机向量 r；
2. 依次计算 $u = Br$、$v = Au$ 与 $w = Cr$；
3. 若 $v = w$ 则通过检验，否则拒绝。

这里不能先算 AB。按照括号 $A(Br)$ 计算，只需三次稠密矩阵—向量乘法，即 $O(n^2)$ 次域运算；额外存储 $O(n)$ 个域元素，并消耗 n 个随机比特。读取三 个稠密矩阵本身已经需要 $\Theta(n^2)$ 个字。

定理 2.1 (Freivalds 算法的正确性)。 在任意域上，若 $AB = C$，算法以概率 1 通过；若 $AB \neq C$，错误通过的概率 至多为 $1/2$。

证明. 若 $AB = C$，则对每个 r 都有 $A(Br) = (AB)r = Cr$。

下面设 $M = AB - C \neq 0$。从 M 中选取一个非零元素 M_{ij}。固定 r_j 之外 的所有随机坐标后，方程 $(Mr)_i = 0$ 化为

$$M_{ij} r_j = -\sum_{k \neq j} M_{ik} r_k.$$

由于 M_{ij} 在域中可逆，该方程至多允许 r_j 取一个值。0 与 1 在任意 域中都是不同元素，因此均匀随机的 $r_j \in \{0,1\}$ 使命题成立的条件概率至多 为 $1/2$。算法通过还要求整个向量 Mr 都为零，概率不会更大。 □

推论 2.2 (独立重复)。 用相互独立的随机向量执行 t 次，并且只在每次都通过时接受。错误乘积被接受的 概率至多为 2^{-t}，代价是 $O(tn^2)$ 次域运算。若目标错误率为 δ，取 $t = \lceil \log_2(1/\delta) \rceil$ 即可。

证明真正使用了什么

把每个坐标从有限集合 $S \subseteq \mathbb{F}$ 中独立均匀选取，同样的固定坐标论证给出 $1/|S|$ 的错误界。关键不是矩阵的各行彼此独立，而是固定其他坐标之后，一个 非零系数只允许当前坐标取很少的值。

精确域运算与浮点数不是一回事

上述单侧错误保证依赖精确等式。若使用浮点数和容差判断，舍入误差会改变 “通过” 的含义，定理不能原样照搬。对有理数运算，若讨论位复杂度，还必须估计 分子和分母的增长，而不能只数域运算次数。

3 随机取值：多项式恒等检验的代数核心

Freivalds 算法检验的是线性映射是否相同。多项式恒等检验（PIT）把问题推广为：

$$\text{给定 } f \in \mathbb{F}[x_1, \ldots, x_m], \qquad f \equiv 0 \text{ 吗?}$$

符号 $f \equiv 0$ 表示所有形式系数都为零。它不等价于 “在域中每个点都取零”。 例如 $x^p - x$ 在 $\mathbb{F}_p[x]$ 中不是零多项式，却在 $\mathbb{F}_p$ 的每个元素上取值为零。 因此，黑盒 PIT 的定理必须同时交代形式多项式、次数和可选取的域元素数目。

我们假设可以高效计算 f 在指定点的值。它可能由算术电路、行列式或乘积公式 给出；把它完全展开成单项式之和反而可能非常昂贵。多元多项式的总次数，是其 非零单项式中指数之和的最大值。

例 3.1 (同一个多项式的紧凑表示)。 对 Vandermonde 矩阵 $V_{ij} = x_i^{j-1}$，有

$$\det V = \prod_{1 \le j < i \le m} (x_i - x_j).$$

行列式和乘积式都能直接求值；若按行列式定义完全展开，却要遍历 $m!$ 个排列。 黑盒 PIT 只计算给定表示在随机点的值，不要求先写出全部系数。

3.1 一元多项式的根数界

引理 3.2 (根数界)。 域 $\mathbb{F}$ 上次数为 d 的非零一元多项式，至多有 d 个不同的根。

证明. 对 d 归纳。非零常数没有根。若正次数多项式 f 以 a 为根，因式定理给出 $f(x) = (x-a)g(x)$，且 $\deg g = d-1$。f 的其他根都必须是 g 的根，因此总数 至多为 $1 + (d-1) = d$。 □

这是一条对任意域都成立的代数事实，并不是关于复数分解的“代数基本定理”。 于是，对有限集合 $S \subseteq \mathbb{F}$ 中均匀随机的 r，

$$f \neq 0 \quad \Longrightarrow \quad \mathbb{P}[f(r) = 0] \le \frac{d}{|S|}.$$

当 $|S| \le d$ 时，这个上界仍然正确，却没有信息量。 根数界也给出一个确定性基准：若域中至少有 $d+1$ 个不同元素，则次数至多为 d 的多项式恒等为零，当且仅当它在任意 $d+1$ 个不同点上都取零。随机算法把 这 $d+1$ 次求值换成一次较大集合中的随机求值，并承担可控的错误概率。

3.2 Schwartz–Zippel 定理

多元版本源于 DeMillo–Lipton、Schwartz 与 Zippel 的工作 [1, 6, 8]。

定理 3.3 (Schwartz–Zippel)。 设 $f \in \mathbb{F}[x_1, \dots, x_m]$ 是总次数至多为 d 的非零多项式。 $S \subseteq \mathbb{F}$ 有限，$r_1, \dots, r_m$ 从 S 中相互独立且均匀选取，则

$$\mathbb{P}[f(r_1, \dots, r_m) = 0] \le \frac{d}{|S|}.$$

等价地，f 在网格 S^m 中至多有 $d|S|^{m-1}$ 个零点。

证明. 对变量个数 m 归纳。$m = 1$ 时结论就是引理 3.2。当 $m > 1$ 时，把 f 看成关于最后一个变量的多项式：

$$f(x_1, \dots, x_m) = \sum_{i=0}^{k} x_m^i f_i(x_1, \dots, x_{m-1}), \qquad f_k \neq 0.$$

因为 f 的总次数至多为 d，所以 $\deg f_k \le d - k$。记事件 $E = \{f_k(r_1, \dots, r_{m-1}) = 0\}$。归纳假设给出

$$\mathbb{P}(E) \le \frac{d-k}{|S|}.$$

若 E 不发生，前 $m-1$ 个变量代入后，剩下的是关于 x_m 的一个非零 k 次多项式。由于 r_m 在条件化后仍然均匀分布于 S，根数界给出

$$\mathbb{P}[f(r_1,\dots,r_m)=0 \mid E^c] \le \frac{k}{|S|}.$$

无需估计 E 发生后会怎样，直接把“最高次系数消失”和“剩余一元多项式取零” 两种可能合并：

$$\mathbb{P}[f(r)=0] \le \mathbb{P}(E) + \mathbb{P}[f(r)=0 \mid E^c] \le \frac{d-k}{|S|} + \frac{k}{|S|} = \frac{d}{|S|}.$$

□

黑盒多项式恒等检验

给定次数至多为 d 的多项式 f 的求值方法，以及有限集 $S \subseteq \mathbb{F}$：

1. 独立均匀选取 $r_1,\dots,r_m \in S$；
2. 若 $f(r_1,\dots,r_m)=0$，则回答“恒等为零”，否则回答“不恒为零”。

零多项式永远通过；非零多项式被误判为零的概率至多为 $d/|S|$。一次运行的 时间是一次求值所需的时间，不能仅凭次数界推出。

三个容易被省略的条件

Schwartz–Zippel 从一个**非零形式多项式**出发，使用它的**总次数**，并要求各坐标**独立抽样**。若基础域太小，可以在允许的计算模型中转到 扩域；仅对一个小域中的所有点求值，未必能区分非零形式多项式与零函数。

4 少说几句话也能判断相等：通信复杂度

Alice 持有 $x \in \{0,1\}^n$，Bob 持有 $y \in \{0,1\}^n$。双方只通过通信共同 计算

$$\mathrm{EQ}(x,y) = \begin{cases} 1, & x = y, \\ 0, & x \neq y. \end{cases}$$

通信复杂度模型由 Yao 提出[7]；局部计算不计代价，只统计双方交换 的比特数。

先看单向确定性协议：若 Bob 发出的消息少于 n 位，消息种类少于 2^n，必有 两个不同字符串 y, y' 产生同一条消息。Alice 在输入 $x=y$ 时必须接受这条消息， 于是面对不相等的输入对 (y,y') 也会接受。发送完整字符串需要 n 位，所以 单向确定性通信复杂度恰为 n。

即使允许双方交互，精确确定性下界仍为 n。确定性协议的每条完整通信记录都 对应一个输入矩形 $A \times B$。一个接受矩形不可能同时包含两个不同的对角输入 (x,x) 与 (y,y)，否则矩形性质还会把不相等的 (x,y) 包含进来。于是 2^n 个对角输入需要至少 2^n 条接受记录；最坏通信量为 c 的协议至多有 2^c 条记录，所以 $c \ge n$。这里约定由一位指定参与者输出答案。

4.1 多项式指纹协议

双方事先固定一个满足 $n^2 < p < 2n^2$ 的素数 p；$n \ge 2$ 时 Bertrand 定理 保证它存在。将字符串编码成

$$f_x(z) = \sum_{i=0}^{n-1} x_{i+1} z^i \in \mathbb{F}_p[z].$$

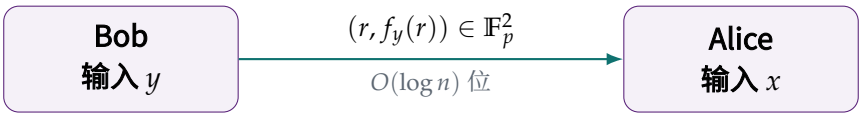

图 2: 单向私有随机协议：随机点由 Bob 选择，因此需要连同指纹值一起发送。

Bob 从 $\mathbb{F}_p$ 中均匀选择 r，发送 $(r, f_y(r))$。Alice 计算 $f_x(r)$ 并比较。若 $x = y$，每个随机点都通过。若 $x \neq y$，差多项式 $h = f_x - f_y$ 至少有一个系数为 1 或 -1，因此在 $\mathbb{F}_p$ 中仍非零，且次数至多为 $n-1$。由根数界，

$$\mathbb{P}[f_x(r) = f_y(r)] \leq \frac{n-1}{p} < \frac{1}{n}.$$

随机点和取值各占 $\lceil \log p \rceil$ 位，总通信量为 $O(\log n)$。这不是把对方输入“压缩后发来”，而是只传递完成相等性判断所需的随机证据。

注 4.1 (公开随机性与私有随机性)。 若双方共享随机点 r，Bob 只需发送 $f_y(r)$；上面的协议由 Bob 私下选择 r，因此必须把 r 也发给 Alice。两种模型的通信量在渐近意义下相同，但协议描述不能混用。

4.2 随机模素数校验

另一种办法把字符串直接看作 $[0, 2^n)$ 中的整数。令 $K = n^3$，Bob 从所有不超过 K 的素数中均匀选择 p，发送 $(p, y \bmod p)$；Alice 与自己的余数比较。

若 $x \neq y$，令 $z = |x - y|$。碰撞恰好意味着 $p \mid z$。由于 $0 < z < 2^n$，z 的不同素因子少于 n 个。记 $\pi(K)$ 为不超过 K 的素数数目，由 $\pi(K) = \Theta(K/\log K)$，

$$\mathbb{P}[x \equiv y \pmod{p}] \leq \frac{n}{\pi(n^3)} = O\left(\frac{\log n}{n^2}\right) = O(1/n).$$

消息仍然只有 $O(\log n)$ 位。注意“随机选取不超过 K 的素数”不等于从 $\{0, 1, \ldots, K\}$ 中随便选一个整数；均匀素数采样与素性检验也是本地计算的一部分，不能凭空视为一次常数时间操作。

5 完美匹配：行列式提供随机见证

设二分图 $G = (L, R, E)$ 的两侧分别为 $L = \{\ell_1, \ldots, \ell_n\}$ 与 $R = \{r_1, \ldots, r_n\}$。给每条边一个独立不定元，构造 Edmonds 矩阵[2]：

$$M_G(i, j) = \begin{cases} x_{ij}, & (\ell_i, r_j) \in E, \\ 0, & (\ell_i, r_j) \notin E. \end{cases}$$

定理 5.1 (Edmonds 行列式判据)。 G 存在完美匹配，当且仅当形式多项式 $\det M_G$ 不恒为零。

证明. 按排列展开行列式：

$$\det M_G = \sum_{\sigma \in S_n} \operatorname{sgn}(\sigma) \prod_{i=1}^{n} M_G(i, \sigma(i)).$$

其中 S_n 表示 $[n]$ 上的所有排列。 排列 σ 对应的乘积非零，当且仅当边 $\{(\ell_i, r_{\sigma(i)}) : i \in [n]\}$ 构成完美匹配。不同排列选出的边变量集合 不同，因而产生不同单项式；即使符号相反也不会抵消。所以展开式中存在非零项， 恰好等价于图中存在完美匹配。 □

现在选取有限域 $\mathbb{F}_q$，把每条边的不定元独立地赋为均匀随机域元素，再对所得 数值矩阵做高斯消元。

定理 5.2 (随机完美匹配检测)。 算法使用 $O(n^3)$ 次域运算。若图没有完美匹配，它必定回答“不存在”；若图有 完美匹配，它错误回答“不存在”的概率至多为 n/q。

证明. 无完美匹配时，定理 5.1说明行列式是零多项式，任何赋值都得到零。 有完美匹配时，行列式是总次数为 n 的非零多项式。Schwartz–Zippel 定理把 随机赋值后恰好取零的概率界为 n/q；数值行列式可用 $O(n^3)$ 次域运算计算。 □

取 $n^2 < q < 2n^2$ 的素数即可把错误率降到 $1/n$ 以下，并使域元素只需 $O(\log n)$ 位。这里的价值不在于宣称它优于所有确定性二分图匹配算法，而在于 展示一种可复用的转化：组合对象的存在性先编码成形式多项式是否为零，再交给 随机恒等检验。一次数值行列式只负责判断存在性，并不会直接输出匹配。

6 Karp–Rabin：用滚动指纹搜索字符串

给定文本串 $x = x_1 \cdots x_n \in \{0,1\}^n$ 和模式串 $y = y_1 \cdots y_m \in \{0,1\}^m$，其中 $m \le n$。逐位置完整比较需要 $O(nm)$ 时间。相邻窗口共享 $m-1$ 个字符，Karp–Rabin 算法利用这部分重叠， 只维护窗口代表整数的模素数余数[4]。

对长度为 m 的二进制串 $w = w_1 \cdots w_m$，定义

$$[w]_2 = \sum_{j=1}^{m} w_j 2^{m-j}.$$

第 i 个窗口记作 $X_i = x_i \cdots x_{i+m-1}$。从不超过 $K = mn^3$ 的素数中均匀选择 p，令 $H_p(w) = [w]_2 \bmod p$。

Karp–Rabin 的 Monte Carlo 版本

1. 计算 $H_p(y)$、$H_p(X_1)$ 以及 $a = 2^{m-1} \bmod p$；
2. 对 $i = 1, \ldots, n-m+1$：
   (a) 若 $H_p(X_i) = H_p(y)$，报告位置 i；
   (b) 若还有下一个窗口，使用

$$H_p(X_{i+1}) \equiv 2\big(H_p(X_i) - x_i a\big) + x_{i+m} \pmod{p}$$

   更新指纹。

更新式依次完成三件事：减去最高位、整体左移一位、接入新字符。算法始终只保存 模 p 余数，并不会存下可能有 m 位的窗口整数。

若 $X_i \neq y$，非零差值 $|[X_i]_2 - [y]_2|$ 小于 2^m，因而不同素因子少于 m 个。于是单个错误窗口发生指纹碰撞的概率满足

$$\mathbb{P}[H_p(X_i) = H_p(y)] \leq \frac{m}{\pi(mn^3)} = O\left(\frac{\log n}{n^3}\right).$$

对不超过 n 个窗口使用并合界，整次运行出现任何伪命中的概率为

$$O\left(\frac{\log n}{n^2}\right) = o(1/n).$$

真正匹配一定具有相同指纹，因此判定版本不会漏掉模式；未经核验的报告位置则有 上述小概率是错误的。也可以在每次指纹相等时逐字符复核，从而得到永不出错的 版本，但其最坏时间会受候选位置数量影响。

因为 $p \leq mn^3 \leq n^4$，所有算术量都只有 $O(\log n)$ 位。随机素数选定后， 预处理需 $O(m)$ 次字操作，窗口扫描需 $O(n)$ 次字操作；这里采用字长至少为 $\Theta(\log n)$、模运算视为常数时间的字长 RAM 模型。若按位操作计费，还要 加入模运算和素数生成的代价。若文本只能顺序到达，为了知道离开窗口的字符， 还需要保存一个长度为 m 的循环缓冲区。

7 多重集合：在数据流中比较相等性

设 $A = \{\{a_1, \ldots, a_n\}\}$、 $B = \{\{b_1, \ldots, b_n\}\}$ 是定义在 $[n]$ 上的两个多重集合。排序可以 比较它们，却要保存输入；完整频数表需要 $\Theta(n)$ 个计数器，最坏为 $\Theta(n \log n)$ 位。即使只做互异性检验，也要一个 n 位的出现标记向量。 Lipton 提出的乘积指纹对元素顺序天然不敏感[5]。

把每个元素写成一个根：

$$f_A(z) = \prod_{i=1}^{n} (z - a_i), \qquad f_B(z) = \prod_{i=1}^{n} (z - b_i).$$

只要域中 $1, \ldots, n$ 互不相同，多项式的唯一因子分解就说明：$f_A \equiv f_B$ 当且仅当 $A = B$。一个元素在多重集合中的重数，正好是对应线性因子的重数。

7.1 有界值域下的简洁算法

固定满足 $n^2 < p < 2n^2$ 的素数 p，从 $\mathbb{F}_p$ 中均匀选择 r，并计算

$$\mathrm{FING}_r(A) = \prod_{i=1}^{n} (r - a_i) \pmod{p}, \qquad \mathrm{FING}_r(B) = \prod_{i=1}^{n} (r - b_i) \pmod{p}.$$

定理 7.1 (流式多重集合相等性检验)。 若 $A = B$，两个指纹对每个 r 都相等；若 $A \neq B$，则

$$\mathbb{P}_r[\mathrm{FING}_r(A) = \mathrm{FING}_r(B)] \leq \frac{n-1}{p} < \frac{1}{n}.$$

算法只需一遍读入、$O(n)$ 次模运算和 $O(\log n)$ 位工作空间。

证明. $p > n$ 保证 $[n]$ 中不同整数在 $\mathbb{F}_p$ 中仍然不同。若 $A \neq B$，两个首一根 多项式在 $\mathbb{F}_p[z]$ 中也不同；最高次项相消后， $h = f_A - f_B$ 是次数至多为 $n-1$ 的非零多项式。根数界给出碰撞概率。处理数据 流时只保留 p, r、计数器和当前乘积，每个量都只有 $O(\log n)$ 位。 □

若输入为 $a_1, \ldots, a_n \in [n]$，要检查它们是否恰好组成一个排列，只需把 输入指纹与 $\prod_{i=1}^{n}(r-i) \bmod p$ 比较。于是互异性检验也能在线性时间、 对数工作空间内完成，且只可能把非排列误判成排列。

7.2 什么时候还要随机化模数

在值域就是 $[n]$ 时，只要固定 $p > n$，不同元素就不会在模 p 后合并；原始 材料中的“随机素数”并非必要。不过，当记录是很大的整数而我们仍希望使用短模数 时，随机化 p 有其作用，同时也引入了两层不同的失败事件。

先把 $f_A - f_B$ 看作 $\mathbb{Z}[z]$ 中的非零多项式。从 $[L, 2L]$ 内的素数中均匀 选择 p，把系数模 p 化简，再从 $\mathbb{F}_p$ 中均匀选择 r：

(1) **模数不佳：**所有系数差都被 p 整除，非零整数多项式退化为 $\mathbb{F}_p[z]$ 中的零多项式；

(2) **取值碰撞：**模 p 后多项式仍非零，但随机点 r 恰好是它的根。

回到 $a_i, b_i \in [n]$ 的参数，每个根多项式的系数绝对值至多为 $(n+1)^n$， 因此可以固定一个满足 $0 < |c| \le 2(n+1)^n$ 的非零系数差。若第一种坏事件发生， 则 $p \mid c$；这样的不同素数只有 $O(n \log n)$ 个。另一方面，区间 $[L, 2L]$ 中有 $\Theta(L/\log L)$ 个素数。条件在第一种事件不发生时，根数界 把第二种事件的概率限制在 n/L。取 $L = \Theta((n \log n)^2)$，得到

$$\mathbb{P}[\text{指纹碰撞}] \le \mathbb{P}[\text{模数不佳}] + \mathbb{P}[\text{取值碰撞} \mid \text{模数良好}] = O(1/n).$$

Schwartz–Zippel 只能在确认模 p 后仍是非零多项式之后使用；把两层随机性混在 同一个“根的概率”里，会漏掉关键的可靠性条件。

8 设计随机指纹时应检查什么

六项检查表

1. **不相等的见证是什么？**非零矩阵行、非零多项式、非零整数差， 或发生变化的根重数。
2. **随机性在哪里？**随机坐标、求值点、素数模数，还是两者兼有； 哪些选择彼此独立？
3. **哪些坏选择会隐藏见证？**数的是线性方程的解、根，还是素因子？
4. **哪一边绝不出错？**明确写出可能误报相等，还是可能漏报存在性。
5. **指纹有多短？**扩大域会降低错误率，也会增加消息、存储和位运算 成本。
6. **时间界属于哪个模型？**域运算、机器字运算、位运算和流式遍数 不能互相替代。

9 习题

1. **扩大采样集合。**证明 Freivalds 算法从 S^n 中均匀选取向量时， 错误通过概率至多为 $1/|S|$。若各坐标不独立，证明的哪一步需要修改？
2. **长方形矩阵。**对 $A \in \mathbb{F}^{a\times b}$、$B \in \mathbb{F}^{b\times c}$、 $C \in \mathbb{F}^{a\times c}$ 设计相应检验，并给出域运算次数和随机比特数。
3. **紧例。**构造一组 $AB \neq C$ 的矩阵，使 Freivalds 算法恰以 $1/2$ 的概率错误通过。
4. **分别控制各变量次数。**若 f 关于 x_i 的次数至多为 d_i， 证明碰撞概率至多为 $\sum_i d_i/|S|$，并与总次数版本比较。

应用	一次运行代价	单侧错误	典型参数
Freivalds	$O(n^2)$ 次域运算	错误等式通过，概率 $\le 1/2$	独立重复 t 次
次数 d 的 PIT	一次多项式求值	非零多项式被判为零，概率 $\le d/\lvert S\rvert$	$\lvert S\rvert \gg d$
相等性通信	$O(\log n)$ 位	误报相等，概率 $< 1/n$	$p \asymp n^2$
完美匹配	$O(n^3)$ 次域运算	漏报存在，概率 $\le n/q$	$q \asymp n^2$
Karp–Rabin	$O(n+m)$ 次字操作	报告错误位置，概率 $o(1/n)$	随机 $p \le mn^3$
多重集合数据流	$O(n)$ 次字操作，$O(\log n)$ 位	误报相等，概率 $< 1/n$	固定 $p \asymp n^2$

表 2: 本讲算法的定量保证。表中的渐近参数省略了无关常数和有限个小规模输入。

5. **小有限域。**验证 $x^p - x$ 在 $\mathbb{F}_p[x]$ 中非零，却在每个 $\mathbb{F}_p$ 元素上取零。说明为什么转到大小超过 p 的扩域后可能发现它非零。

6. **通信参数。**选择 p，使多项式指纹协议的错误率至多为 2^{-40}；进一步把通信量写成 n 与一般目标错误率 δ 的函数。

7. **公开随机点。**若 Alice 与 Bob 共享随机 r，重新计算协议的 通信量。消息中少了哪些比特？概率空间中仍有哪些随机变量？

8. **特征为二。**在特征为 2 的域上，行列式展开中的正负号消失。 证明二分图完美匹配的 Edmonds 判据仍然成立。

9. **核验候选位置。**在 Karp–Rabin 中逐字符核验每个指纹命中。 估计期望运行时间，并给出一个有大量真实匹配的位置的输入。

10. **插入—删除数据流。**数据流用 (a, Δ) 修改整数重数。尝试 用 $\prod_a (r-a)^{\Delta}$ 构造指纹；当 $\Delta < 0$ 时，需要排除什么事件？

11. **重用随机点。**两次多重集合比较共用同一个 r。并合界是否仍 可用？两次碰撞事件是否一定独立？分别给出理由。

12. **审计一句话。**某讲义写道：“$\mathbb{F}_p$ 上次数为 d 的非零 多项式，在随机点非零的概率至少为 $1 - d/p$。”列出要让这句话正确且有信息量 所需的全部限定条件。

A 概率放大与素数计数工具

A.1 条件概率与独立性

当 $\mathbb{P}(B) > 0$ 时，条件概率定义为 $\mathbb{P}(A \mid B) = \mathbb{P}(A \cap B)/\mathbb{P}(B)$。按事件 B 是否发生分解，得到全概率公式

$$\mathbb{P}(A) = \mathbb{P}(A \mid B)\mathbb{P}(B) + \mathbb{P}(A \mid B^c)\mathbb{P}(B^c).$$

随机变量独立，意味着知道其中一部分的取值不会改变其余变量的分布。 Schwartz–Zippel 的证明正是在固定 $r_1, \dots, r_{m-1}$ 后，继续使用 r_m 在 S 中均匀这一事实。

A.2 并合界与独立重复

任意事件 $E_1, \dots, E_k$ 都满足

$$\mathbb{P}\left[\bigcup_{i=1}^{k} E_i\right] \le \sum_{i=1}^{k} \mathbb{P}(E_i).$$

并合界不要求事件独立。因此 Karp–Rabin 可以让所有窗口共用同一个随机素数；即使各窗口的碰撞事件相关，概率之和仍是合法上界。

若每次试验独立，且单次失败概率至多为 ε，连续 t 次都失败的概率 至多为 ε^t。更一般地，若单次成功概率至少为 q，则

$$\mathbb{P}[t \text{ 次都不成功}] \le (1-q)^t \le e^{-qt}.$$

重复使用同一个随机样本不会产生上述概率放大。

A.3 一个整数能有多少不同素因子

若非零整数 z 有 k 个不同素因子，它们的乘积至少为 2^k，所以

$$\omega(z) \le \log_2 |z|.$$

这里 $\omega(z)$ 只数不同的素因子，不计重数。这条粗略上界已经足够分析随机 模素数校验和 Karp–Rabin。

记 $\pi(N)$ 为不超过 N 的素数个数。素数定理给出

$$\pi(N) \sim \frac{N}{\ln N}.$$

本讲只使用其较弱推论：当 N 充分大时，$\pi(N) = \Theta(N/\log N)$，并且 $\pi(2N) - \pi(N) = \Theta(N/\log N)$。渐近算法可以把有限个小输入吸收到常数 处理中。Bertrand 定理还保证每个整数 $N > 1$ 与 $2N$ 之间至少有一个素数；它只证明“存在”，不能替代随机素数分析所需的密度估计。

B 有限域与多项式代数工具

B.1 域、素域与扩域

域 $\mathbb{F}$ 上定义了加法与乘法，有加法和乘法单位元 $0, 1$；每个元素有加法逆元，每个非零元素有乘法逆元，并满足通常的结合、交换与分配律。整数模素数 p 构成 素域

$$\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}.$$

模合数运算可能出现非零零因子，非零元素也未必可除，所以一旦证明需要域结构，就不能随意把素数模数换成合数。

每个有限域的大小都是 p^k，其中 p 为素数、$k \ge 1$；反过来，每个素数幂 都对应一个有限域，并且在同构意义下唯一。若 $g \in \mathbb{F}_p[z]$ 是 k 次不可约多项式，则

$$\mathbb{F}_{p^k} \cong \mathbb{F}_p[z]/(g(z)).$$

扩域可以在保持特征不变的同时提供更多求值点，这正是基础域小于多项式次数时 常用的补救办法。

B.2 多项式除法与因式定理

$\mathbb{F}[x]$ 中的多项式是形式有限和 $a_0 + a_1 x + \cdots + a_d x^d$。对非零 g，存在唯一的 $q, r \in \mathbb{F}[x]$ 使

$$f = qg + r, \qquad r = 0 \text{ 或 } \deg r < \deg g.$$

令 $g = x - a$ 便得到因式定理：$f(a) = 0$ 当且仅当 $(x - a)$ 整除 f。逐次提出 线性因子，就得到正文使用的一元多项式根数界。

非常数多项式若不能分解为两个更低次的非常数多项式之积，就称为不可约。 用不可约多项式取商，正如用素数对整数取模，所得商环才是域。

B.3 形式多项式不等于取值函数

$\mathbb{F}[x_1, \ldots, x_m]$ 的元素首先是带系数的形式代数对象；逐点求值后，才得到 $\mathbb{F}^m \to \mathbb{F}$ 的函数。在无限域上，不同形式多项式诱导不同函数；在有限域上却 未必如此。由 Fermat 小定理，

$$x^p - x = 0 \qquad (\forall x \in \mathbb{F}_p),$$

但 $x^p - x$ 不是零多项式。Schwartz–Zippel 定理从“形式上非零”出发，并把 采样集合大小与次数比较，正是为了避免把这两种概念混为一谈。

C 文献说明

Freivalds 的矩阵乘法检验见于 1977 年 IFIP 论文；一些讲义把它写成 1979 年， 往往是与作者随后发表的相关论文混淆。多元根数界通常简称 Schwartz–Zippel 定理；DeMillo 与 Lipton 更早得到过紧密相关的结果，因此也 常见四人合称。本文的 Edmonds 矩阵是二分图的符号邻接矩阵，不是一般图匹配中 使用的斜对称 Tutte 矩阵。多重集合的乘积指纹由 Lipton 在 Princeton 技术报告 *Fingerprinting Sets* 中系统提出。

参考文献

[1] R. A. DeMillo and R. J. Lipton. A probabilistic remark on algebraic program testing. *Information Processing Letters*, 7(4):193–195, 1978. doi:10.1016/0020-0190(78)90067-4.

[2] J. Edmonds. Systems of distinct representatives and linear algebra. *Journal of Research of the National Bureau of Standards B*, 71B(4):241–245, 1967.

[3] R. Freivalds. Probabilistic machines can use less running time. In *Information Processing 77*, pages 839–842. North-Holland, 1977.

[4] R. M. Karp and M. O. Rabin. Efficient randomized pattern-matching algorithms. *IBM Journal of Research and Development*, 31(2):249–260, 1987. doi:10.1147/rd.312.0249.

[5] R. J. Lipton. Fingerprinting sets. Technical Report CS-TR-212-89, Princeton University, 1989.

[6] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, 27(4):701–717, 1980. doi:10.1145/322217.322225.

[7] A. C.-C. Yao. Some complexity questions related to distributive computing. In *Proceedings of the 11th ACM Symposium on Theory of Computing*, pages 209–213, 1979. doi:10.1145/800135.804414.

[8] R. Zippel. Probabilistic algorithms for sparse polynomials. In *EUROSAM 1979*, Lecture Notes in Computer Science 72, pages 216–226. Springer, 1979.