

(1) Fix a field \mathcal{F} and choose an element $r \in \mathcal{F}$ uniformly at random from \mathcal{F} :

* *Verifying polynomial identities:* Given two polynomials $f(x) = \prod_{i=1}^n (x - a_i)$ (in this form) and $g(x) = \sum_{i=0}^n c_i x^i$ (in this form), devise a Monte Carlo algorithm to verify whether $f(x) = g(x)$ in $O(n)$ expected time.

- Algorithm: Fix a field of integers $[1, j]$ and choose an element $r \in [1, j]$ uniformly at random. (The value of j is to be set shortly.) If $f(r) = g(r)$ then print "input polynomials are equal" else print "input polynomials are not equal".

While it is immediate $f(r)$ can be computed in $O(n)$ time, by applying Horner's rule, $g(r)$ is also computable in linear time.

- This algorithm errs only if $f(x) \neq g(x)$ but $f(r) = g(r)$. Hence, it is a Monte Carlo algorithm with one-sided error. Specifically, algorithm's output is incorrect whenever r is a root of $f(x) - g(x) = 0$ but $f(x) \neq g(x)$. Since the number of roots of $f(x) - g(x) = 0$ is at most n , by setting $j = 1000n$, this algorithm returns a wrong answer with probability at most $\frac{n}{1000n} = \frac{1}{1000}$.

Noting that choosing a $r \in [1, 1000n]$ takes $O(\lg n)$ time in expectation, the time complexity of this algorithm is $O(n)$ in expectation.

- Using the abundance of witnesses paradigm, choose $r_1, r_2, \dots, r_k \in [1, 1000n]$ uniformly at random. If $f(r_i) = g(r_i)$ for every $i \in [1, k]$, then print "input polynomials are equal" else print "input polynomial are not equal". Then, the probability algorithm outputs $f(x) = g(x)$ while $f(x) \neq g(x)$ is at most $(\frac{1}{1000})^k$.

The prob of error could also be reduced by increasing the value of j , however, the value to be used is limited by the precision available. Besides, since $\text{rand}(1, k)$ takes $O(\lg k)$ time in expectation (assuming $\text{rand}(0, 1)$ takes $O(1)$ worst-case time), the overhead due to $\text{rand}(1, j)$ needs to be accounted.

* *Verifying matrix multiplication:* Given three $n \times n$ matrices A, B , and C , devise a Monte Carlo algorithm to verify whether $AB = C$ in $O(n^2)$ time in the worst-case.

- Algorithm: Compute a vector $r = (r_1, r_2, \dots, r_i)$ from the field $\{0, 1\}^n$, while choosing each r_i uniformly at random from $\{0, 1\}$. In $O(n^2)$ time, compute $X = A(Br)$; in $O(n^2)$ time, compute $Y = Cr$. If $X \neq Y$, then print ' $AB \neq C$ ' else print ' $AB = C$ '.
- The above algorithm outputs an incorrect answer only if $AB \neq C$ but $X = Y$. ——— (1)

That is, $AB \neq C \Leftrightarrow D = (AB - C) \neq 0 \Leftrightarrow D$ must have some nonzero entry, say d_{ij} .

$$X = Y \Leftrightarrow (AB - C)r = 0 \Rightarrow \sum_{k=1}^n d_{ik}r_k = 0 \Leftrightarrow r_j = \frac{-(\sum_{k=1}^n d_{ik}r_k) + d_{ij}r_j}{d_{ij}}. \text{ ——— (2)}$$

Suppose r_j is chosen last among all r_i s. By the time r_j is chosen uniformly at random from $\{0, 1\}$, equality holds whenever r_i is set to exactly one of $\{0, 1\}$. That is, (1) holds whenever r_i is chosen so that (2) is satisfied. Therefore, (1) holds with probability at most $\frac{1}{2}$.

- Algorithm utilizing abundance of witnesses paradigm: Choose k random vectors, each from $\{0, 1\}^n$. If $A(Br') \neq Cr'$ for any such random vector r' , then print ' $AB \neq C$ ' else print ' $AB = C$ '. It is obvious that this reduces the error probability to at most $(\frac{1}{2})^k$. Obviously, this algorithm takes $O(kn^2)$ time in the worst-case.
- This technique of verifying matrix multiplication is known as the *Freivalds' technique*.

(2) Fix the point of evaluation a and choose a random field over which the evaluation of a is to be performed:

* *Verifying equality of strings*: Alice has string $A : a_1, \dots, a_n$, and Bob has string $B : b_1, \dots, b_n$, with every $a_i, b_j \in \{0, 1\}$, Bob determines whether $A = B$ but Alice can afford to transmit only $o(n)$ bits to Bob. Devise a Monte Carlo algorithm with polynomially small error wherein Alice does transmit only $o(n)$ bits to Bob.

- Define points of evaluation $a = \sum_{i=1}^n a_i 2^{i-1}$ and $b = \sum_{i=1}^n b_i 2^{i-1}$. Define the fingerprint function $f_p(x) = x \bmod p$ for a prime p chosen uniformly at random from $[2, \tau = n^2 \lg n]$.

Algorithm: if $f_p(a) \neq f_p(b)$ then print $A \neq B$ else print $A = B$.

Noting that $f_p(a) = f_p(b)$ iff p divides $|a - b|$, below, we upper bound $pr(f_p(a) = f_p(b) \mid a \neq b)$.

- **Observation**: Since $|a - b| < 2^n$, there can be at most n prime divisors of $|a - b|$. (Proof: If the number of distinct prime divisors of $|a - b|$ are more than n , since each prime divisor is at least 2, the $|a - b| > 2^n$.)

From the prime number theorem, the number of primes less than or equal to τ is asymptotically $\frac{\tau}{\ln \tau}$; again, since only n among these can be divisors of $|a - b|$, when probability is taken over the random choices of p ,

$$pr(f_p(a) = f_p(b) \mid a \neq b) = pr(p \text{ divides } |a - b| \mid a \neq b) \leq \frac{n}{\frac{\tau}{\ln \tau}} = \frac{n \ln \tau}{\tau} = \frac{n(\ln(n^2) + \ln(\lg(n)))}{n^2 \lg(n)} = \frac{1}{n} \left(\frac{2 \ln n}{\lg n} + \frac{\ln \lg n}{\lg n} \right) = O\left(\frac{1}{n}\right). \leftarrow \text{error is polynomially small, desirable since the error reduces as } n \text{ grows}$$

Further, since p is at most τ , the number of bits to be transmitted (at most p) from Alice to Bob is $O(\lg \tau)$, i.e., $O(\lg n)$.

Note that the most expensive operation in this algorithm is computing a $p \in [2, \tau]$. However, once a p is shared between Alice and Bob, fingerprints can be taken for any A and B .

- Significantly, in this problem, we fixed the points of evaluation ($a = \sum_{i=1}^n a_i 2^{i-1}$, $b = \sum_{i=1}^n b_i 2^{i-1}$) and for a random prime p of a reasonably small magnitude, fingerprints were obtained by evaluating a and b over the field Z_p .
- How many numbers one needs to choose in $[1, \tau]$ with replacement before the number chosen is prime?

View the process of randomly selecting a number and determining whether it is a prime as a Bernoulli trial. Further, we know, for a geometric random variable X that has value i if the success occurs at i^{th} trial, then $E[X] = \frac{1}{p}$. From the prime number theorem, $p = \frac{(\tau / \ln(\tau))}{\tau} = \frac{1}{\ln(\tau)}$. Hence, the expected number

of trials needed to obtain a prime number in $[1, \tau]$ is $\ln \tau$. Then, with AKS algorithm, the average-time to find a prime in $[1, \tau]$ is $O((\lg \tau)^7)$.

- * *Pattern matching*: Let the alphabet Σ be $\{0, 1\}$. Given a text string $T \in \Sigma^*$ of length n and a pattern string $P \in \Sigma^*$ of length m , for $m < n$, devise a Monte Carlo algorithm to find the smallest value of shift s such that $T[s+1 \dots s+m] = P$.
- Algorithm: Choose a prime number p in $f_p(x) = x \bmod p$ uniformly at random from the set of numbers in the range $[1, \tau = n^2 m \lg(n^2 m)]$. If $f_q(T[s+1 \dots s+m]) = f_q(P)$ then output shift s and exit; otherwise, try with shift $s+1$.
- Since $|T[s+1 \dots s+m] - P|$ is an m -bit positive integer with value $< 2^m$, there can be at most m distinct prime divisors to $|T[s+1 \dots s+m] - P|$. The number of primes smaller than τ is asymptotically $\frac{\tau}{\ln \tau}$; again, only m among these can be divisors of $|T[s+1 \dots s+m] - P|$.

When probability is taken over the random choices of q ,

$$\begin{aligned} & \text{pr}((f_q(T[s+1 \dots s+m]) = f_q(P)) \mid (T[s+1 \dots s+m] \neq P)) \\ &= \text{pr}(q \text{ divides } (|T[s+1 \dots s+m] - P|) \mid (T[s+1 \dots s+m] \neq P)) \\ &\leq \frac{m}{\frac{\tau}{\ln \tau}} = O\left(\frac{m \lg \tau}{\tau}\right) = O\left(\frac{1}{n}\right). \end{aligned}$$

- The worst-case time, ignoring the number of times to iterate for finding a prime, is $O(n+m)$.

(3) Interpret the bit vectors a and b as the n -bit integers a and b ; fix a prime number $p > 2^n$; choose a random polynomial over the field Z_p , and obtain the fingerprints by evaluating this polynomial at the integers a and b , performing all arithmetic over the field Z_p , and then reducing the resulting values modulo a number of magnitude close to $\lg n$.

- Let U be the universe comprising keys and let T be a hash table. Also, let $|T| = m$ and $|U| > m$.

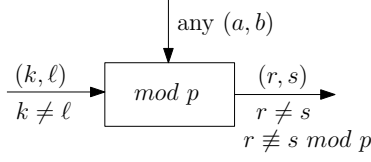
A collection of hash functions \mathcal{H} is called a *2-universal hash family* whenever (i) each function in \mathcal{H} is from U to T , and (ii) for any pair of distinct keys $k_i, k_j \in U$, the number of hash functions $h \in \mathcal{H}$ for which $h(k_i) = h(k_j)$ is at most $\frac{|\mathcal{H}|}{m}$.

A hash function h is a *2-universal hash function* if h is chosen, uniformly at random and independent of keys being stored in T , from a 2-universal hash family. (Choosing independent of keys being stored ensures lesser number of collisions in expectation, even if keys are chosen by an adversary.)

With any 2-universal hash function h , for any pair of distinct keys $k_i, k_j \in U$, $\text{pr}(h(k_i) = h(k_j)) \leq \frac{|\mathcal{H}|/m}{|\mathcal{H}|} = \frac{1}{m}$. That is, a 2-universal hash function obeys simple uniform hashing.

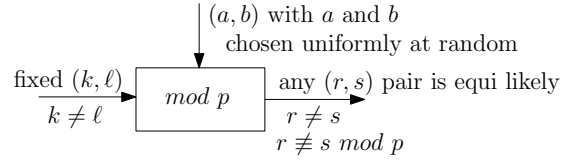
Below, we assume $a \in [1, p-1]$, $b \in [0, p-1]$, $U \in [0, p-1]$, and obviously every key $k \in U$. Let k, ℓ be two distinct keys, we define, $r = (ak + b) \bmod p$ and $s = (a\ell + b) \bmod p$. The following proofs are reproduced from pages 267-268 of [CLRS].

- Lemma 1: For (k, ℓ) with $k \neq \ell$, any fixed (a, b) , leads to (r, s) with $r \neq s$ and $r \not\equiv s \pmod p$.



We know, $r - s \equiv a(k - \ell) \pmod p$. Since $a \neq 0$ (from the definition of \mathcal{H}_{pm}), since $a \not\equiv 0 \pmod p$ (as $a \in [1, p - 1]$), since $k - \ell \not\equiv 0 \pmod p$ (as every key $k \in [0, p - 1]$), and since their product must also be nonzero modulo p , $r - s \not\equiv 0 \pmod p$. Since p is a prime, this implies, $r \not\equiv s \pmod p$.

- Lemma 2: For (k, ℓ) with $k \neq \ell$, if (a, b) is chosen uniformly at random from $[1, p - 1] \times [0, p - 1]$, then the resulting pair (r, s) is equally likely to be any pair of distinct values modulo p .



Since $r = (ak + b) \pmod p$ and $s = (a\ell + b) \pmod p$,
 $a = (r - s)((k - \ell)^{-1} \pmod p) \pmod p$, and
 $b = (r - ak) \pmod p$.

From these, we can find a and b , given r and s . Hence, each of the possible $p(p - 1)$ choices for the pair (a, b) yields a different resulting pair (r, s) with $r \neq s$.

Since there are only $p(p - 1)$ possible pairs (r, s) with $r \neq s$ (from Lemma 1), there is a one-to-one correspondance between pairs (a, b) and pairs (r, s) with $r \not\equiv s \pmod p$.

Thus, for any given pair of inputs k and ℓ , if we pick (a, b) uniformly at random from $[1, p - 1] \times [0, p - 1]$, the pair (r, s) is equally likely to be any pair with $r \not\equiv s \pmod p$.

Choosing tuple (a, b) uniformly at random from any of $p(p - 1)$ number of tuples,
 $pr(r \neq s) = \frac{1}{p(p-1)}$. ——— (A)

- Theorem: Assuming every key $k \in [0, p - 1]$,
 $\mathcal{H}_{pm} = \{h_{ab}(k) = ((ak + b) \pmod p) \pmod m : a \in [1, p - 1], b \in [0, p - 1], \text{ for a prime } p > m\}$
 is a 2-universal hash family.

[Here, choosing $h \in \mathcal{H}_{pm}$ uniformly at random is equivalent to choosing a and b uniformly at random respectively from $[1, p - 1]$ and $[0, p - 1]$.
 Significantly, given the universe and the hash table are fixed, to save any function from \mathcal{H}_{pm} , one needs to store only a and b .]

$$\frac{(r, s)}{r \neq s \atop r \neq s \bmod p} \rightarrow \boxed{\text{mod } m} \rightarrow \frac{(r' = r \bmod m, s' = s \bmod m)}{\text{with } pr(r' \equiv s' \bmod m) \leq \frac{1}{m}}$$

For $k \neq \ell$, it is immediate that $pr(h_{ab}(k) = h_{ab}(\ell)) = pr(r \equiv s \bmod m)$.

From (A), we know $pr(r \neq s)$ is $\frac{1}{p(p-1)}$; leading to,

$$pr(h_{ab}(k) = h_{ab}(\ell)) = \frac{1}{p(p-1)} \cdot |\{(r, s) : r \neq s \text{ and } r \equiv s \bmod m\}|.$$

For a given value of r , of the $p-1$ possible remaining values for s , the number of values s such that $s \neq r$ and $s \equiv r \bmod m$ is at most $\lceil \frac{p}{m} \rceil - 1 \leq (\frac{p+m-1}{m}) - 1 = \frac{p-1}{m}$.

Since r can assume p number of values, $pr(h_{ab}(k) = h_{ab}(\ell)) \leq \frac{1}{p(p-1)} \cdot \frac{p(p-1)}{m} = \frac{1}{m}$.

References:

- Randomized Algorithms by R. Motawani and P. Raghavan. [Sections 7.1-7.2 and 7.4-7.6.]
- Introduction to Algorithms by T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, Third Edition. [pg 267-268]