# Chapter 9. ALGEBRAIC TECHNIQUES

Some of the most interesting results in the design of efficient computations were obtained using algebraic techniques combined with randomization.

## Fingerprinting technique

**Example:** Decide equality of two elements $x, y$ drawn from a large universe $U$.

Complexity under any reasonable model seems to be $\Omega(\lg |U|)$.

**Alternative approach:** Pick a random mapping

$$f : U \longrightarrow V \qquad |U| >> |V|.$$

such that there is a good chance that

$$f(x) = f(y) \text{ implies } x = y$$

and declare that $x = y$ $(x \neq y)$ if $f(x) = f(y)$ $(f(x) \neq f(y))$.

Complexity under any reasonable model is now $\Omega(\lg |V|)$

## FINGERPRINTING METHOD - BASICS

**The basic idea is to find out whether two given objects are equal (given their full representations), by comparing their fingerprints (incomplete representations).**

An important point of this method is that a fingerprint should always be chosen randomly from a possible set of fingerprints.

Another point behind is an observation that well chosen fingerprints could preserve some essential differences between objects they represent.

Two essential requirements fingerprints have to satisfy:

- **Fingerprints have to be short, simple, and easy to obtain.**
- **A fingerprint of an object has to contain as much of the essential information about the object as possible.**

## BASIC SCHEME

Given is a set of objects $\mathcal{O}$, find a set $\mathcal{F}$ of fingerprints and a set M of mappings

$$f : \mathcal{O} \leftrightarrow \mathcal{F}$$

such that for any two different objects $o_1$ and $o_2$ from $\mathcal{O}$ there is a lot of such mappings $f \in M$ that

$$f(o_1) \neq f(o_2)$$

Fingerprint method can therefore be seen as a special case of the method of abundance of the witnesses discussed in the next chapter.

# FREIVALDS TECHNIQUE

**Matrix multiplication** for matrices of degree $n$:

there is a classical (school) algorithm with complexity — $O(n^3)$

there is a sophisticated algorithm with complexity— $O(n^{2.376})$

**Problem:** to check whether AB = C for iven $n$-dimensional matrices $A$, $B$ and $C$.
**Method:** choose a random column vector $r \in \{0,1\}^n$.

Compute: $x = \mathrm{B}r, y = \mathrm{A}x, z = \mathrm{C}r$ – $\{O(n^2)$ steps$\}$ - and declare $AB = C$ iff $y = z$.

What is probability that conclusion is wrong - zero if $AB = C$ and less then $\frac{1}{2}$ if $AB \neq C$

**Theorem** Let A, B, C be $n \times n$ matrices, such that AB $\neq$ C. Then for randomly chosen $r \in \{0,1\}^n$ it holds that $Pr[\mathrm{AB}r = \mathrm{C}r] \leq \frac{1}{2}$.
**Proof:** Denote D = AB − C $\neq 0$. We wish to upper-bound probability that $y = z$ $(\Leftrightarrow \mathrm{D}r = 0)$

Without loss of generality, we may assume that the first row in D has a non-zero element and that all its non-zero elements are first. Let $d$ be the first row of D with the first $k$ elements $\neq 0$. We concentrate on probability that $Dr \neq 0$. This yields a lower bound on the probability that $y \neq z$.

$$Dr = 0 \qquad \text{iff} \qquad r_1 = \frac{-\sum_{i=2}^{k} d_i r_i}{d_1} \qquad (*).$$

Invoking the Principle of Deferred Decision we assume that $r_2, \ldots, r_n$ are chosen (randomly) before $r_1$. Since $r_1$ is also chosen randomly, the probability of $Dr \neq 0$ is $= \frac{1}{2}$.

# VERIFICATION of POLYNOMIAL IDENTITIES

**Polynomial product verification problem**

**Given two polynomials of degree $n$ – $P_1(x), P_2(x)$ – and one of degree $2n$ – $P_3(x)$ – verify whether $P_1(x) \cdot P_2(x) = P_3(x)$**

| | | |
|---|---|---|
| **polynomial** | **evaluation complexity** | $O(n)$ |
| | **multiplication complexity** | $O(n \lg n)$ |

**Randomized verification:** **Let $S$ be a set of integers and $|S| \geq 2n+1$.**

**Pick $r \in S$ uniformly and randomly and evaluate $P_1(r), P_2(r), P_3(r)$.**

**Declare the identity $P_1(x) \cdot P_2(x) = P_3(x)$ as correct unless $P_1(r) \cdot P_2(r) \neq P_3(r)$.**

**Analysis: Polynomial $Q(x) = P_1(x) \cdot P_2(x) - P_3(x)$ has degree $\leq 2n$, and has therefore at most $2n$ roots.**

**Unless $Q(x) \equiv 0$, for at most $2n$ random choices of $r \in S$, it holds $Q(r) = 0$.**

**The probability of error of the above verification is therefore at most $\frac{2n}{|S|}$.**

**The above verification procedure can be extended to verify any polynomial identity $P_1(x) = P_2(x)$, where $P_1, P_2$ are given implicitly.**

## Example

$$M = \begin{vmatrix} 1 & x_1 & x_1^2 & \dots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \dots & x_2^{n-1} \\ \vdots & & & & \\ 1 & x_n & x_n^2 & \dots & x_n^{n-1} \end{vmatrix}$$

**Task: Verify that $Det(M) = \prod_{i<j}(x_j - x_i)$.**

## SCHWARZ-ZIPPEL THEOREM

**Theorem** Let $Q(x_1, \ldots, x_n)$ be a polynomial of total degree $d$. Fix any finite set $S$ of reals, and let $r_1, \ldots, r_n$ be chosen independently and randomly from $S$. Then

$$Pr[Q(r_1, \ldots, r_n) = 0 \,|\, Q(x_1, x_2, \ldots, x_n) \not\equiv 0] \leq \frac{d}{|S|}$$

**Proof** By induction on $n$.

The case $n = 1$ was actually already discussed on the previous page.

**Induction step**: Let $Q(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{k} x_1^i Q_i(x_2, \ldots, x_n)$

where $Q_k \not\equiv 0$. Since the total degree of $Q_k$ is at most $d - k$, the induction hypothesis shows, that the probability that $Q_k(r_2, \ldots, r_n) = 0$ is at most $\frac{d-k}{|S|}$.

Suppose that $Q_k(r_2, \ldots, r_n) \neq 0$. Consider the following polynomial

$$q(x_1) = Q(x_1, r_2, \ldots, r_n) = \sum_{i=0}^{k} x_1^i Q_i(r_2, \ldots, r_n)$$

Degree of $q$ is $k$ and $q(x_1) \not\equiv 0$. The base case implies that the probability that $q(r_1) = Q(r_1, \ldots, r_n) = 0$ is at most $\frac{k}{|S|}$.

Hence

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

$$Pr\left[Q(r_1, r_2, \ldots, r_n) = 0 \mid Q_k(r_2, \ldots, r_n) \neq 0\right] \leq \frac{k}{|S|}$$

$$\Longrightarrow$$

$$Pr\left[Q(r_1, \ldots, r_n) = 0\right] \leq \frac{d}{|S|}$$

because for any two events $\mathcal{E}_1$ and $\mathcal{E}_2$, it holds

$$\Pr(\mathcal{E}_1) \leq \Pr(\mathcal{E}_1|\bar{\mathcal{E}}_2) + \Pr(\mathcal{E}_2).$$

# TESTING SIMILARITIES of MATRICES

**Definition Two $n \times n$ matrices $A$ and $B$ are called similar if there exists a non-singular matrix $T$ such that $TAT^{-1} = B$.**

To decide whether two given matrices $A$ and $B$ are similar, one has to decide whether $TA = BT$ for a matrix $T$ such that $\det(T) \neq 0$.

We can see entries of $T$ as variables and denote by $\bar{t}$ the vector of $n^2$ such variables.

In such a case equality $TA = BT$ can be seen as a system of $n^2$ linear equations, with entries of $\bar{t}$ as variables. These equations can be expressed as $C\bar{t} = 0$, where $C$ is an $n^2 \times n^2$ matrix.

This way we get a homogeneous system of equations and it is possible to find, in polylog time on a parallel computer, a basis for the solution space.

Let $\{\bar{t}_1, \ldots, \bar{t}_k\}$ denote such a basis. In such a case any solution matrix has the form

$$T = c_1T_1 + c_2T_2 + \ldots + c_kT_k,$$

where the matrix $T_i$ corresponds to the basis vector $\bar{t}_i$.

We are interested in a non-singular solution. Such a solution exists iff there is a $c_1, \ldots, c_k$ vector for which $\det(T) \not\equiv 0$. That is if $\det(T)$ as a polynomial in the variables $c_1, \ldots, c_k$ is not identically zero. As we already know, we can checked efficiently whether $\det(T) \not\equiv 0$.

# PERFECT MATCHING of GRAPHS

Let $G = (U, V, E)$ be a bipartite graph, $|U| = |V| = n$. A **matching of $G$** is a collection of edges $M \subset E$ such that each node occurs at most once in an edge of $M$. A matching of size $n$ is **perfect**. (A note: $\{$Each perfect matching defines a permutation of the set $\{1, \ldots, n\}\}$)

**Theorem** Let $A$ be an $n \times n$ matrix obtained from $G$ as follows

$$A_{ij} = \left\{ \begin{array}{lll} x_{ij} & if & (u_i, v_j) \in E \\ 0 & if & (u_i, v_j) \notin E \end{array} \right\}$$

Then $G$ has a perfect matching iff $det(A) \not\equiv 0$.

**Proof** Let $S_n$ be the set of all permutations of $\{1, 2, \ldots, n\}$

$$det(A) = \sum_{\pi \in S_n} sgn(\pi) A_{1\pi(1)} \cdots A_{n\pi(n)}$$

Since each indeterminate $x_{ij}$ occurs at most once in $A$ there can be no cancellation of terms in the above sum.

$$\Longrightarrow$$

$det(A) \not\equiv 0$ iff there exist a $\pi \in S_n$ such that $A_{1\pi(1)}, \ldots, A_{n\pi(n)} \neq 0$
iff there is a perfect matching

**Implications:** There is a simple randomized algorithm for testing the existence of a perfect matching in bipartite graphs

## VERIFICATION of EQUALITY of STRINGS - A REPETITION

**Problem story:** Both Alice and Bob maintain a copy of database. Periodically they have to verify its consistency. How to do that efficiently?

**Alice's data:** $a = (a_1, \ldots, a_n) \in \{0, 1\}^n$. **Bob's data:** $(b = b_1, \ldots, b_n) \in \{0, 1\}^n$.
**Solution: To use a fingerprinting mapping:** Define first

$$\mathbf{num}(a) = \sum_{i=1}^{n} a_i 2^{i-1}, \quad \mathbf{num}(b) = \sum_{i=1}^{n} b_i 2^{i-1}$$

**and then define the fingerprinting mapping:** $F_p(x) = x \mod p$, where $x$ is an integer and $p$ is a prime.
**Aim:** We would like that it holds: $a \neq b \Rightarrow F_p(\mathbf{num}(a)) \neq F_p(\mathbf{num}(b))$.

At this technique the number of bits transmitted is $O(\lg p)$. This strategy can be easily fooled by an adversary.

**Way to get around:** to choose $p$ randomly

**Lemma:** The number of distinct prime divisors of any integer less than $2^n$ is at most $n$.

The fingerprint comparison fails if $p$ divides $|\mathbf{num}(a) - \mathbf{num}(b)|$. Choose a threshold $\tau > n$, and choose $p < \tau$.

**Theorem** $Pr[F_p(\mathbf{num}(a)) = F_p(\mathbf{num}(b)) \mid a \neq b\ ] \leq \frac{n}{\pi(\tau)}$ and if $\tau = tn \ln tn$, then

$$Pr[F_p(\mathbf{num}(a)) = F_p(\mathbf{num}(b)) \mid a \neq b\ ] \leq O(\frac{1}{t}),$$

where $\pi(x) = \frac{x}{\ln x}$ is the number of primes smaller than $x$.

# PATTERN MATCHING

Given: **text** $X = x_1, \ldots, x_n$, **pattern** $Y = y_1, \ldots, y_m$, both over the alphabet $\{0, 1\}$, and $m < n$. Find the leftmost occurrence of $Y$ in $X$.

**Trivial solution - by exhaustive comparisons:** $O(nm)$.
**Knuth-Morris algorithm solution:** $O(n + m)$.

We present now a **Monte Carlo and Las Vegas algorithms** as another solutions that are based on the fingerprinting technique.

**Notation:** $X(j) = x_j x_{j+1}, \ldots, x_{j+m-1}$, $\mathrm{num}(X(j)) = \sum_{k=j}^{j+m-1} x_k 2^{j+m-k-1}$.

**Basic idea:** Interpret any $m$ - bit string $x$ as an integer $\mathrm{num}(x)$ and use as the fingerprint function $F_p(x) = \mathrm{num}(x) \pmod p$, where $p \leq \tau$ – a chosen threshold.

$$Pr[F_p(Y) = F_p(X(j)) \mid Y \neq X(j)\ ] \leq \frac{m}{\pi(\tau)} = O\left(\frac{m \ln \tau}{\tau}\right)$$

Taking $\tau = nm \ln nm$ we get

$$Pr[\text{a false match occurs}] = O(\frac{1}{n})$$

## Monte Carlo algorithm

Compare, for $j = 1, \ldots, n - m$, fingerprints of $Y$ and $X(j)$, and output the smallest $j$ at which a match occurs.

# COMPLEXITY ANALYSIS – $O(m+n)$

**Key point:** The cost of computing $F_p(X(j+1))$ from $F_p(X(j))$ is only $O(1)$ operations.

Indeed, for $1 \leq j \leq n-m+1$,

$$\mathrm{num}(X(j+1)) = 2\left[\mathrm{num}((X(j)) - 2^{m-1}x_j\right] + x_{j+m}$$

$$\Longrightarrow$$

$$F_p(\mathrm{num}(X(j+1))) = (2[\mathrm{num}(F_p(X(j))) - 2^{m-1}x_j] + x_{j+m}) \bmod p$$

## Conversion into a Las Vegas algorithm

Whenever a match occurs between the fingerprints of $Y$ and $X(j)$, we compare strings $Y$ and $X(j)$ in $O(m)$ time. If this is a false match, we abandon the whole process in favor of using a brute-force $(O(nm))$ – algorithm.

Resulting algorithm does not make any error and has as the expected running time

$$O\left(m + (n)(1 - \frac{1}{n}) + nm(\frac{1}{n})\right) = O(n+m)$$