Algebraic Techniques

Wafi AlBalawi and Yan Liu
LDCSEE,
West Virginia University,
Morgantown, WV
{wafi,yanliu}@csee.wvu.edu

1 Introduction

In this chapter we describe some basic randomization techniques with an underlying algebraic flavor. We begin by describing Freivalds' technique for the verification of identities involving matrices and polynomials. We describe how this generalizes to the Schwartz-Zippel technique for identities involving multivariate polynomials and we illustrate this technique by applying it to the problem of detecting the existence of perfect matchings in graphs.

Most of these techniques and their applications involve a `fingerprinting` mechanism, which can be described as follows. Consider the problem of deciding the equality of two elements x and y drawn from a large universe U. Under any "reasonable" model of computation, testing the equality of x and y then has a deterministic complexity of at least $\log |U|$. An alternative approach is to pick a random mapping from U into a significantly smaller universe V in such a way that there is a good chance that x and y are identical if and only if their images are identical. The images of x and y are their fingerprints, and their equality can be verified in $\log |V|$ time by comparing the `fingerprints`.

2 Fingerprinting and Freivalds' Technique

We illustrate fingerprinting by describing a technique for verifying matrix multiplication. The fastest known algorithm for matrix multiplication runs in time $O(n^{2.376})$, which improves significantly on the obvious $O(n^3)$ time algorithm but has the disadvantage of being extremely complicated. Suppose we are given an implementation of this algorithm and would like to verify its correctness. Since program verification is a difficult task, a reasonable goal might be to verify the correctness of the output produced on specific executions of the algorithm.

Given $n \times n$ matrices $\mathbf{A}, \mathbf{B}$, we can calculate matrix $\mathbf{C} = \mathbf{AB}$ in $O(n^{2.376})$. The problem: if you are given $\mathbf{A}, \mathbf{B}$, and $\mathbf{C}$ over the field $\mathcal{F}$, we would like to verify if indeed $\mathbf{AB} = \mathbf{C}$. We would like to avoid computing $\mathbf{A} \cdot \mathbf{B}$, since doing so (in $O(n^3)$ time) defeats the purpose. The following technique, known as Freivalds' technique (see algorithm 2.1), provides an elegant solution which takes $O(n^2)$ time with bounded error probability.

```
1: Choose a random vector $\vec{\mathbf{r}} \in \{0,1\}^n$; each component of $\vec{\mathbf{r}}$ is chosen independently and uniformly at random
   from $\{0,1\}$, the additive and multiplicative identities of the field $\mathcal{F}$.
2: Compute $\vec{\mathbf{x}} = \mathbf{B}\vec{\mathbf{r}}$, $\vec{\mathbf{y}} = \mathbf{A}\vec{\mathbf{x}} = \mathbf{AB}\vec{\mathbf{r}}$, $\vec{\mathbf{z}} = \mathbf{C}\vec{\mathbf{r}}$ in $O(n^2)$ time.
3: If $\vec{\mathbf{y}} = \vec{\mathbf{z}}$ then $\mathbf{AB} = \mathbf{C}$
```

Algorithm 2.1: Freivalds' Algorithm

Theorem: 2.1 *Let $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ be $n \times n$ matrices over $\mathcal{F}$ such that $\mathbf{AB} \neq \mathbf{C}$. Then for $\vec{\mathbf{r}}$ chosen uniformly at random from $\{0,1\}^n$:*

$$\mathbf{Pr}[\vec{\mathbf{y}} = \vec{\mathbf{z}} | \mathbf{AB} \neq \mathbf{C}] = \mathbf{Pr}[\mathbf{AB}\vec{\mathbf{r}} = \mathbf{C}\vec{\mathbf{r}} | \mathbf{AB} \neq \mathbf{C}] \leq \frac{1}{2}$$

Proof: *Let* $\mathbf{D} = \mathbf{AB} - \mathbf{C}$, $\mathbf{D} \neq \mathbf{Z}$ *where* $\mathbf{Z}$ *is the 0-matrix.*

Permute $\mathbf{D}$ *so that the first row has non-zero entries and all non-zero occur before zeros. We can do this without affecting correctness.*

Let $\vec{\mathbf{d}}$ *be the first row in* $\mathbf{D}$, *the first entry in* $\mathbf{D}\vec{\mathbf{r}}$ *is* $\vec{\mathbf{d}}^T\vec{\mathbf{r}}$.

$\mathbf{Pr}[\vec{\mathbf{d}}^T\vec{\mathbf{r}}] \neq 0$ *is a lower bound on* $\mathbf{Pr}[\mathbf{D}\vec{\mathbf{r}}] = \vec{\mathbf{0}}$.

Now the inner product, $\vec{\mathbf{d}}^T\vec{\mathbf{r}} = 0$ *if and only if* $r_1 = \frac{-\sum_{i=2}^{k} d_i r_i}{d_1}$.

By the principle of deferred decisions, the right hand side of the above expression has a predetermined value v, *which is 0 or 1 with uniform probability.*

Consequently $\mathbf{Pr}[\vec{\mathbf{r}}_1 = v] \leq \frac{1}{2}$.

□

3 Verifying Polynomial Identities

In this section we show that Freivalds' technique applies to the verification of identities involving polynomials. Two polynomials $P(x)$ and $Q(x)$ are said to be equal if they have the same coefficients for corresponding powers of x. Verifying identities of integers, or, in general, strings over any fixed alphabet, is a special case since we can represent any string of length n as a polynomial of degree n. This is achieved by treating the k^{th} element in the string as the coefficient of the k^{th} power of a symbolic variable.

3.1 Verifing Polynomial Identities (Univariate)

We first consider the `polynomial product verification` problem:

$$\boxed{\begin{array}{l}\textit{Given polynomials } P_1(x), P_2(x), P_3(x) \in \mathcal{F}[X] \textit{ where } \mathcal{F} \textit{ is a field,} \\ \textit{verify that } P_1(x) \times P_2(x) = P_3(x).\end{array}}$$

Assume that the polynomials $P_1(x)$ and $P_2(x)$ are of degree at most n; then $P_3(x)$ cannot have degree exceeding $2n$. Polynomials of degree n can be multiplied in $O(n \log n)$ time using *Fast Fourier Transforms*, whereas the evaluation of a polynomial at a fixed point requires $O(n)$ time.

Theorem: 3.1 *Let* $Q(x)$ *be a polynomial of degree* d *over field* $\mathcal{F}$. *Let* $\boldsymbol{S} \subseteq \mathcal{F}$ *be a finite subset of* $\mathcal{F}$. *Let* $r \in \boldsymbol{S}$ *be chosen uniformly at random, then* $\mathbf{Pr}[Q(r) = 0 \mid Q(x) \neq 0] \leq \frac{d}{|\boldsymbol{S}|}$

Proof: *From the fundamental theorem of theorem of algebra, we know that a polynomial of degree d has at most d roots.* □

Let $Q(x) = P_1(x) \times P_2(x) - P_3(x)$. Thus, for verifying that $P_1(x) \times P_2(x) = P_3(x)$, we draw a random point $\vec{\mathbf{r}}$ from $\mathbf{S}$ and say "yes", if $Q(\vec{\mathbf{r}}) = 0$. Clearly, the probablity of error is at most $\frac{2 \cdot n}{|\mathbf{S}|}$, where n represents the degree of both $P_1(x)$ and $P_2(x)$.

The extension of the univariate case to the multi-variate case is known as the Schwartz-Zippel theorem.

Theorem: 3.2 *Let* $Q(x_1, ..., x_n) \in \boldsymbol{F}[x_1, ..., x_n]$ *be a multivariate of total degree* d. *Fix any finite set* $\boldsymbol{S} \subseteq \boldsymbol{F}$, *and let* $r_1, ..., r_n$ *be chosen independently and uniformly at random form* $\boldsymbol{S}$. *Then*

$$\mathbf{Pr}[Q(r_1, ..., r_n) = 0 \mid Q(x_1, ..., x_n) \neq 0] \leq \frac{d}{|\boldsymbol{S}|}$$

Proof: *We provide an inductive proof, based on the number of distinct variables. If* $n = 1$, *the Schwartz-Zippel theorem follows from Theorem (3.1). Assume that the theorem is true for a multivariate polynomial with at most* $n - 1$ *distinct variables. Consider the polynomial* $Q(x_1, .., x_n)$ *and factor out* x_1. *Thus, we can write*

$$Q(x_1, .., x_n) = \sum_{i=0}^{k} X_1^i Q_i(x_2, ..., x_n),$$

where k is the largest exponent of x_1 in Q. We assume that x_1 affects Q, so that $k > 0$. The coefficient of x_1^k, viz. Q_k is not identically 0, by our choice of k. Since the total degree of Q_k is at most $d-k$, the induction hypotheis implies that the probability of $Q_k(r_1, r_2, \ldots, r_n) = 0$ is at most $\frac{d-k}{|S|}$.

Suppose that $Q_k(r_2, r_3, \ldots, r_n) \neq 0$. Consider the following univariate polynomial:

$$q(x_1) = \sum_{i=0}^{k} x_1^i Q_i(r_2, r_3, \ldots, r_n)$$

The polynomial $q(x_1)$ is not identically zero, since the coefficient of x_1^k is $Q_k(r_2, r_3, \ldots, r_n)$. The base case now implies that the probability that $q(r_1) = Q(r_1, r_2, \ldots, r_n)$ evaluates to 0 is at most $\frac{k}{|S|}$. We have thus shown that

$$\begin{aligned}\mathbf{Pr}[Q_k(r_2, r_3, \ldots, r_n) = 0] &\leq \frac{d-k}{|S|} \\ \mathbf{Pr}[Q(r_1.r_2, \ldots, r_n) = 0 | Q_k(r_2, r_3, \ldots, r_n) \neq 0] &\leq \frac{k}{|S|}\end{aligned}$$

Use "A Probabilistic Identity" on the Class Website the conclude the hypothesis of the theorem. □

4 Perfect Matchings in Graphs

Consider a bipartite graph $G(U, V, E)$ with the independent sets of vertices $U = u_1, ..., u_n$ and $V = v_1, ..., v_n$. A *matching* is a collection of edges $M \subseteq E$ such that each vertex occurs at most once in M. A *perfect matching* is a matching of size n. Each perfect matching M in G can be viewed as a permutation from U into V. More precisely, the perfect matchings in G can be put into a one-to-one correspondence with the permutations in $\mathbf{S_n}$ ($\mathbf{S_n}$ is the permutation group on n elements), where the matching corresponding to a permutation $\pi \in \mathbf{S_n}$ is given by the pairs $(u_i, v_{\pi(i)})$, for $1 \leq i \leq n$.

Theorem (4.1) called Edmonds' Theorem establishes a connection between determinants and matchings.

***Theorem:** 4.1 Let A be the $n \times n$ matrix obtained from $G(U, V, E)$ as follows:*

$$\begin{aligned}A_{ij} &= x_{ij} \quad (u_i, v_j) \in E \\ &= 0 \quad (u_i, v_j) \notin E\end{aligned}$$

Define the multivariate polynomial $Q(x_1, x_2, ..., x_n)$ as being equal to $det(A)$. Then, G has a perfect matching if and only if $Q \neq 0$.

<u>Proof</u>: *The determinant of A is given by $det(A) = \sum_{\pi \in \mathbf{S_n}} sgn(\pi) A_{1,\pi(1)}, A_{2,\pi(2)}, ...A_{n,\pi(n)}$. (Check this our in any book on Abstract Algebra!) Since each indeterminate x_{ij} occurs at most once in A, there can be no cancellation of the terms in the summation. Therefore the determinant is not identically zero if and only if there is a permutation π for which the corresponding term in the summation is non-zero. The latter happens if and only if each of the entries $A_i, \pi_(i)$, for $1 \leq i \leq n$, is non-zero. This is equivalent to having a perfect matching in G.* □

5 Verifying Equality in Strings

5.1 Problem

The problem of testing the equality of two strings we present here can be reduced to that of verifying polynomial identities which we have seen in the previous lectures. However, the new type of fingerprint introduced here has important benefits when extended to the pattern matching problem discussed later. The string equality verification problem can be described as the following:

Suppose that Alice maintains a large database of information. Bob maintains a second copy of the database. Periodically they must compare their databases for consistency. Denote Alice's data by the sequence of bits $(a_1, a_2, ..., a_n)$ and Bob's by the sequence $(b_1, b_2, ..., b_n)$. The very simple deterministic strategy for checking the consistency will be transmitting of n bits between them. Furthermore, it is obvious to see that any deterministic consistency check that transmits fewer than n bits will fail if any adversary could decide which bits of either database to modify.

5.2 Randomized Strategy

The randomized strategy we present is to detect an inconsistency with high probability while transmitting far fewer than n bits of data. To reach the high probability of detection, we use the following fingerprint mechanism:

1. Interpret the data as n-bit integers a and b, by defining $a = \sum_{i=1}^{n} a_i 2^{i-1}$ and $b = \sum_{i=1}^{n} b_i 2^{i-1}$.
2. Define the fingerprint function $F_p(x) = x \bmod p$ for a prime p.
3. Alice transmits $F_p(a)$ to Bob, who will compare it with $F_p(b)$.
4. In case $F_p(a) \neq F_p(b)$, then it can be concluded that $a \neq b$.

We shall now show that if $F_p(a) \neq F_p(b)$, then the probability that $a \neq b$ is small. From this strategy, we can see that the bits needed to be transmitted will be reduced to $O(\log p)$, which is much smaller than n for a small prime p. However, this strategy can be easily foiled by an adversary for any fixed choice of p since, for any p and b, there exist many choices of a for which $a \equiv b (\text{mod } p)$. Therefore, we get around this problem by choosing p at random.

Theorem: 5.1 *For any number k, let $\pi(k)$ be the number of distinct primes less than k. $\pi(k)$ is asymptotically $\frac{k}{\ln k}$.*

Proof: *See [2].* □

Let $c = |a - b|$, c is a non-negative integer. The fingerprint defined above fails only when $c \neq 0$ and p divides c. Let $N = 2^n$, since $a \leq N$ and $b \leq N$, we know that $c \leq N$.

Lemma: 5.1 *The number of distinct prime divisors of any number less than 2^n is at most n.*

Proof: *Each prime number is greater than 1. If $N = 2^n$ has more than t distinct prime divisors, then $N \geq 2^t$.* □

Choose a threshold τ that is larger than $n = \log N$. The number of primes smaller than τ is $\pi(\tau) \sim \frac{\tau}{\ln \tau}$. Of these, at most n can be divisors of c and cause our fingerprint strategy to fail. Therefore, we pick a random prime p smaller than τ for defining F_p. The number of bits needed for transmission is $O(\log \tau)$. Choose $\tau = tn \log tn$, for large t. Then we can easily conclude that:

Theorem: 5.2

$$\mathbf{Pr}[F_p(a) = F_p(b) | a \neq b] \leq \frac{n}{\pi(\tau)} = O(\frac{1}{t})$$

Proof: *Since the probability is taken over the random choice of p, it is easy to see that $\mathbf{Pr}[F_p(a) = F_p(b) | a \neq b] \leq \frac{n}{\pi(\tau)}$. We also know that $\pi(\tau) \sim \frac{\tau}{\ln \tau}$ and $\tau = tn \log tn$, therefore,*

$$\frac{n}{\pi(\tau)} = \frac{n \ln \tau}{tn \log tn} = \frac{\ln \tau}{t \log tn} = \frac{(\ln tn + \ln \log tn)}{t \log tn}$$

$$= \frac{1}{t} + \frac{\ln \log tn}{t \log tn} = O(\frac{1}{t}).$$

□

Thus, we get an error probability of at most $O(\frac{1}{t})$, and the number of bits to be transmitted is $O(\log t + \log n)$. Choosing $t = n$ gives us an excellent strategy for this problem.

6 Pattern Matching

6.1 Problem

Pattern Matching is a typical problem of that finding an occurrence (if any) of a given pattern in the text. Here, a *text* is a string $X = x_1 x_2 ... x_n$ and a *pattern* is a string $Y = y_1 y_2 .. y_m$ both over a fixed finite alphabet Σ, such that $m \leq n$.

In the following lectures, without loss of generality, we restrict ourselves to the case $\Sigma = \{0, 1\}$. The pattern occurs in the text if there is a $j \in 1, 2, ..., n - m + 1$ such that for $1 \leq i \leq m$, $x_{j+i-1} = y_i$.

6.2 Monte Carlo Algorithm

We can simply solve this problem by giving an algorithm of $O(nm)$ running time by trivially trying for a match at all possible locations i. The best deterministic algorithm to solve this problem has running time $O(n + m)$. However, this algorithm is extremely complicated and difficult to implement. Here we describe a very simple Monte Carlo algorithm which also runs in expected time of $O(n + m)$; later, we will convert this into a Las Vegas algorithm.

Define $X(j) = x_j x_{j+1} .. x_{j-m+1}$ as the substring of length m in X that starts at position j. A match occurs if there exists a j, for $1 \leq j \leq n - m + 1$, we have $Y = X(j)$. Now, instead of finding any match, we change the problem into finding the smallest value of j such that $X(j) = Y$. This makes the solution unique. To find such a match, we use a fingerprint function F and compare $F(Y)$ with each $F(X(j))$. An error occurs if $F(Y) = F(X(j))$ but $Y \neq X(j)$. Therefore, we would like to use a function F that has a small probability of error and can be efficiently computed. The function we choose is as below:

For any string $Z \in \{0, 1\}^m$, interpret Z as an m-bit integer and define $F_p(Z) = Z \bmod p$.

Assume that p is chosen uniformly at random from the set of primes smaller than a threshold τ. Suppose that we interpret the strings Y and $X(j)$ as m-bit integers, and then compare $F_p(Y)$ and $F_p(X(j))$, the only possible error will occur when we get $F_p(Y) = F_p(X(j))$ while $Y \neq X(j)$. By Theorem (1.2) , we can bound the probability of such a *false match* as follows:

$$\mathbf{Pr}[F_p(Y) = F_p(X(j)) | Y \neq X(j)] \leq \frac{m}{\pi(\tau)} = O(\frac{m \log \tau}{\tau})$$

Then, the probability that a false match occurs for any of the at most n values of j is $O((nm \log \tau)/\tau)$. By choosing $\tau = n^2 m \log(n^2 m)$, we will get:

$$\mathbf{Pr}[\textit{a false match occurs}] = O(\frac{nm \log(n^2 m \log(n^2 m))}{n^2 m \log(n^2 m)}) = O(\frac{1}{n} + \frac{\log \log(n^2 m)}{n \log(n^2 m)}) = O(\frac{1}{n})$$

Our Monte Carlo algorithm just simply compares the $F_p(Y)$ with all $F_p(X(j))$ and output the first j for which a match occurs. The computation of expected running time of this algorithm is as claimed.

For $1 \le j \le n - m + 1$,

$$X(j+1) = 2[X(j) - 2^{m-1}x_j] + x_{j+m}.$$

From this, we can find that

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

We can see that given $F_p(X(j))$, the incremental cost of computing $F_p(X(j+1))$ is $O(1)$. Furthermore, since each x_j is either 0 or 1, there is no extra computation needed for division or multiplication. Thus, the total running time required for this algorithm is $O(n+m)$.

Theorem: 6.1 *The Monte Carlo algorithm for pattern matching requires $O(n+m)$ expected time and has a probability of error of $O(\frac{1}{n})$.*

6.3 Las Vegas Algorithm

In our Las Vegas algorithm, when a false match occurs, we detect it and abandon the whole process and then use the $O(nm)$ running time deterministic algorithm to find the match. The expected running time of this algorithm is :

$$T(n) = (1 - \frac{1}{n})O(n+m) + \frac{1}{n}O(nm) = O(n+m) + O(m) = O(n+m).$$

An alternative version of the Las Vegas algorithm restarts the entire algorithm with a new random choice of p whenever a false match occurs. The probability of restarting once is $\frac{1}{n}$, thus the probability of restarting more than t times is bounded by $\frac{1}{n^t}$. Therefore, the variance of running time between restarting t times and $t+1$ times is bounded by $\frac{1}{n^t}O(n+m)$, which decreases exponentially while t increases. Compared to this algorithm, the first approach has a relatively high probability of $\frac{1}{n}$ of being forced to use the $O(nm)$ time algorithm and hence has a high variance in the running time.

References

[1] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms.* Cambridge University Press, Cambridge, England, June 1995.

[2] Ramanujachary Kumanduri and Cristina Romero. *Number Theory With Computer Applications.* Prentice Hall, August 1997.