6.966 Algebra and Computation	September 16, 1998
Lecture 3	
Lecturer: Madhu Sudan	*Scribe: Yevgeniy Dodis*

In this lecture we will study an application of polynomials to coding theory (Reed-Solomon code), GCD algorithm and its applications, construction of finite fields, Chinese remainder theorem (CRT) and begin our study of resultant.

1 Application of Polynomials to Coding Theory

Coding theory is concerned with the problem of reliable communication over a noisy channel. Assume we are given a source message $m \in \{0,1\}^k$, that we would like to transmit reliably in the presence of communication errors. We can try to encode the message by adding some redundancy to it in such a manner that one can reconstruct the original message even when large parts of the encoded message get corrupted. More formally, we wish to design an *encoding function* $E : \{0,1\}^k \to \{0,1\}^n$ where $n > k$. This function E defines a *code* $\mathcal{C} = \{E(m) | m \in \{0,1\}^k\}$. We would like all the words of the code (called the *codewords*) to be very far "apart" so that even a large corruption of a codeword enables one to reconstruct the message. Given $a, b \in \{0,1\}^n$ we let the *Hamming distance* $d(a,b)$ between a and b be the number of symbols they differ in. We define the *distance* of the code to be

$$d(\mathcal{C}) = \min_{c,c' \in \mathcal{C}, c \neq c'} d(c,c') = \min_{m,m' \in \{0,1\}^k, m \neq m'} d(E(m), E(m'))$$

The main problem of the theory of Error Correcting Codes is to construct a code with a small length n (compared to k), but a large distance d. Such constructions are quite non-trivial and algebra often comes as the main tool in designing such codes. The algebraic constructions do not directly give codes over a binary alphabet though, so we need to extend our definition of error-correcting codes to larger alphabets and in particular to encoding functions $E : \mathbb{F}^k \to \mathbb{F}^n$ over a field $\mathbb{F}$. (In many standard constructions of *binary* error-correcting codes, *non-binary* codes such as the one described below are used as starting points.)

1.1 Reed-Solomon (RS) Code

Let us view the source message as specifying the coefficients of some polynomial of degree at most $(k-1)$. So given $\bar{m} = (m_1, \ldots, m_k)$, $m_i \in \mathbb{F}$, we define the associated polynomial $M(x) = \sum_{i=1}^{k} m_i x^{i-1}$. Fix arbitrary n *distinct* elements $\alpha_1, \ldots, \alpha_n$ of $\mathbb{F}$ (thus, we must assume that $|\mathbb{F}| \geq n$). The encoding function $E^{RS}_{\alpha_1,\ldots,\alpha_n}(\bar{m}) = \langle M(\alpha_1), \ldots, M(\alpha_n)\rangle$. We claim that this is a great (in fact, "perfect") code. More specifically, let us compute its distance. Let $\bar{m}_1 \neq \bar{m}_2$. Then $M_1(x) \not\equiv M_2(x)$. Since encoding of $\bar{m}_i$ is a value of $M_i(x)$ at n points, and two non-equal polynomials of degree at most $k-1$ can agree in at most $k-1$ points, the encodings differ in at least $n-(k-1) = n-k+1$ places, so $d(\mathcal{C}) \geq n-k+1$. It is easy to show that the distance is exactly $n-k+1$; in fact, it follows from the more general fact below.

Exercise 1 *Show that for any $E : \mathbb{F}^k \to \mathbb{F}^n$ there are 2 distinct messages m_1 and m_2 s.t. $d(E(m_1), E(m_2)) \leq n-k+1$. Hence $d(\mathcal{C}) \leq n-k+1$.*

Hence, RS codes are the best possible codes in terms of minimal distance. Their only disadvantage is the requirement that $|\mathbb{F}| \geq n$.

2 GCD algorithm

Our first look at a non-trivial algebraic algorithm is the Euclidean algorithm for finding the greatest common divisor (GCD). We start by presenting the algorithm to find the GCD of two integers. We then extend it to finding GCD's of polynomials.

2.1 GCD for Integers

It is very easy to find GCD of two integers if one knows their factorization. However, factoring integers is believed to be hard (from computational perspective). The remarkable fact about the algorithm is the ability to find the GCD without factoring. Here is the algorithm.

$\gcd(a, b)$, $0 \leq b \leq a$.
  IF $b = 0$ THEN OUTPUT a,
  ELSE EXPRESS $a = b \cdot q + r$, WHERE $0 \leq r < b$, AND OUTPUT $\gcd(b, r)$.

The correctness is clear as any common factor of a and b must divide r, and any common factor of r and b must divide a. We only need to argue the efficiency.

Claim 2 $r \leq a/2$

Indeed, as $q \geq 1$, we have $a = bq + r \geq b + r \geq 2r$. Thus, the algorithm terminates in $\log(ab)$ iterations as at each iteration the product ab decreases by at least a factor of 2.

Notice that if $g = \gcd(a, b)$, there are some $u, v \in \mathbb{Z}$ s.t. $g = u \cdot a + v \cdot b$. As we will see, it is often convenient to have these u and v. A trivial modification of our algorithm allows us to achieve it. This algorithm is called the *extended GCD* algorithm.

EXTENDED-$\gcd(a, b)$, $0 \leq b \leq a$.
  IF $b = 0$ THEN OUTPUT $(a, 1, 0)$,
  ELSE EXPRESS $a = b \cdot q + r$, WHERE $0 \leq r < b$,
  LET $(g, u', v') =$ EXTENDED-$\gcd(b, r)$, AND OUTPUT $(g, v', u' - v' \cdot q)$.

To verify correctness notice that if $g = u'b + v'r$, then $g = u'b + v'(a - qb) = v'a + (u' - v'q)b$.

2.2 GCD for Polynomials

We now address the question of extending our GCD algorithm to polynomials. Integers had natural ordering based on their magnitude. The natural ordering for the polynomials is based on their degree: $a(x) \leq b(x)$ iff $\deg(a) \leq \deg(b)$. With this in mind, the above algorithm works as it is for the polynomials. The main step to verify is that the division step above does have an appropriate analog in the case of polynomials since if $a(x) \geq b(x)$ then we can uniquely express $a(x) = q(x)b(x) + r(x)$, where $\deg(r) < \deg(b)$. Here the invariant after one iteration is that $\deg(r) < \deg(a)$, so that the sum of the degrees of a and b decreases. Thus, we have termination in at most $\deg(a) + \deg(b)$ iterations.

3 Applications of GCD

We give just a few handy applications of the extended GCD algorithm.

3.1 Inverse Computations in $\mathbb{Z}_p$

Addition and multiplication in $\mathbb{Z}_p$ are very easy to implement. We show that the same holds for the inverse operation as well. One clever way to compute the inverse in polynomial time is to use Fermat little theorem stating that for any $a \in \mathbb{Z}_p^*$, $a^{p-1} \equiv 1 \bmod p$, Thus $a^{-1} \equiv a^{p-2}$. Since exponentiation is easy via repeated squaring, we can compute $a^{-1} \bmod p$ using $O(\log p)$ multiplications. Using GCD, we obtain an alternative way to compute the inverse. It will have an advantage to generalize to the situation of computing $a^{-1} \bmod n$ for composite n, where $a \in \mathbb{Z}_n^*$, i.e. $\gcd(a, n) = 1$. Fermat's trick does not work here. Even though by Euler's theorem $a^{\varphi(n)} \equiv 1 \bmod n$, it is believed to be a hard problem to compute $\varphi(n)$ from n alone.

Without further delay, here is the algorithm to compute the inverse of a even for composite n, whenever $\gcd(a, n) = 1$. Use the extended GCD to find u, v s.t. $1 = ua + vn$. Then $ua \equiv 1 \bmod n$, so $u \bmod n = a^{-1} \bmod n$. Notice that this process takes $O(\log n)$ divisions.

3.2 Construction and Arithmetic of any Finite Field

Theorem 3 *For any prime p and any $n \geq 1$ there exists a unique field of order p^n. This field is denoted $GF(p^n)$ (Galois Field of order p^n). And these are the only finite fields.*

We prove this theorem completely later in the course, but now we show a part of it by demonstrating how to construct these fields. Suppose $f(x) \in \mathbb{Z}_p[x]$ is an irreducible polynomial of degree k. We will prove later that such f exists. Let $\mathbb{F} = \{q(x) \in \mathbb{Z}_p[x] | \deg q < k\}$. Define the addition on $\mathbb{F}$ in the natural way and let multiplication on $\mathbb{F}$ be the multiplication in $\mathbb{Z}_p[x]$ modulo $f(x)$. Thus, if $q_1(x)q_2(x) = \alpha(x)f(x) + \beta(x)$, where $\deg \beta < \deg f = k$, we let $q_1 \star q_2 = \beta$. By the uniqueness of the division procedure, the multiplication $\star$ is well defined. All the properties of the field are clear for $\mathbb{F}$ except the existence of inverses. Take any non-zero $q \in \mathbb{F}$. As $\deg q < \deg f$, we must have that $\gcd(f, q) = 1$ as f is irreducible. By the extended GCD, we can find polynomials g and r s.t. $1 = gf + rq$. But then $rq \equiv 1 \bmod f$, so $r \bmod q$ is the (easily seen to be unique) inverse of q. Thus, $\mathbb{F}$ is indeed a field. Moreover, all field operations are easily computable in polynomial time (given f). The order of $\mathbb{F}$ is p^k as there are k free coefficients in choosing a polynomial of degree at most $k - 1$.

3.3 Chinese Remainder theorem (CRT)

Theorem 4 (CRT) *Let $m_1, \ldots, m_n$ be mutually relatively prime integers, i.e.* $\gcd(m_i, m_j) = 1$ *for $i \neq j$. Then for any $\alpha_1, \ldots, \alpha_n$, $\alpha_i \in \mathbb{Z}_{m_1}$, there exists a* unique *integer* $0 \leq \alpha \leq m_1 \cdot m_2 \ldots \cdot m_n$ *s.t. for any $1 \leq i \leq n$,*

$$\alpha \equiv \alpha_i \bmod m_i$$

Moreover, this α can be found in polynomial time.

Clearly, it suffices to show the theorem for $n = 2$ only. Thus, given $m_1, m_2, \alpha_1, \alpha_2$ s.t. $\gcd(m_1, m_2) = 1$, we wish to find $0 \leq \alpha < m_1m_2$ s.t. $\alpha \equiv \alpha_i \bmod m_i$. We use the extended GCD to find u and v s.t. $um_1 + vm_2 = 1$. Let $M_1 = vm_2$, $M_2 = um_1$. Then $M_1 \equiv 1 \bmod m_1$, $M_1 \equiv 0 \bmod m_2$, $M_2 \equiv 0 \bmod m_1$, $M_2 \equiv 1 \bmod m_2$. Let $\alpha = \alpha_1 M_1 + \alpha_2 M_2 \bmod m_1 m_2$. Then α clearly satisfies the conditions. The uniqueness of α is clear as if there is another $\beta \neq \alpha$ like that, we would have $0 < |\alpha - \beta| < m_1 m_2$ and both m_1 and m_2 divide $|\alpha - \beta|$. As $\gcd(m_1, m_2) = 1$, this is impossible.

In other words, the theorem claims the following bijection (in fact, isomorphism):

$$\alpha \Longleftrightarrow (\alpha \bmod m_1, \ldots, \alpha \bmod m_n)$$

between $\mathbb{Z}_{m_1 m_2 \ldots m_n}$ and $\mathbb{Z}_{m_1} \times \mathbb{Z}_{m_2} \ldots \times \mathbb{Z}_{m_n}$.

We can also define a good error correcting code using the CRT. Let $m_1 \leq m_2 \leq \ldots m_k \leq m_{k+1} \leq \ldots \leq m_{2k}$ be distinct l-bit primes. We can easily have $l \approx (1+\epsilon)\log k$ by the prime number theorem. Let $N = m_1 \cdot \ldots \cdot m_k$. For any $n < N$ define $E(m) = \langle (n \bmod m_1), \ldots (n \bmod m_{2k}) \rangle$. Let $\mathbb{F} = GF(2^l)$ and let us view E as a function $E : \mathbb{F}^k \to \mathbb{F}^{2k}$. Then E maps strings of length k over $\mathbb{F}$ into ones of length $2k$. Moreover, we claim that the minimum distance of this code is at least $k+1$. Indeed, let $E_i(n) = (n \bmod m_i)$. If $n_1 \neq n_2$, $n_1, n_2 < N$, then for any $I \subseteq [2k]$, $|I| = k$, there is $i \in I$ s.t. $E_i(n_1) \neq E_i(n_2)$. This is because equality for all $i \in I$ would imply by the CRT that $n_1 \equiv n_2 \bmod (\prod_{i \in I} m_i)$. As m_i's go in an increasing order, $\prod_{i \in I} m_i \geq N$, so we get that $n_1 = n_2$. Thus, out of any k components of the code, at least 1 of them must be distinct for any $n_1 \neq n_2$, so at most $k-1$ components can be the same. To summarize, our code has length $2k$, distance at least $k+1$ but is defined over a large field of size roughly $k^{1+\epsilon}$. (For people familiar with the notion of a "linear" code, we stress that this is not a linear code.)

4 Resultant

Let us now view the problem of determining whether $a(x)$ and $b(x)$ are relatively prime from a more algebraic point of view. Suppose $\gcd(a(x), b(x)) = 1$. Then there are $u(x)$ and $v(x)$ s.t.

$$u(x)a(x) + v(x)b(x) = 1 \tag{1}$$

We can also easily arrange that $\deg(u) < \deg(b)$ and $\deg(v) < \deg(a)$. Indeed, if $u'a + v'b = 1$ and $u' = qb + u$, where $\deg(u) < \deg(b)$, then $(qb+u)a + v'b = ua + (qa+v')b = ua + vb$, where $v = qa + v'$. We have by construction that $\deg(u) < \deg(b)$, so that $\deg(ua) < \deg(a) + \deg(b)$. This and (1) imply that $\deg(v) < \deg(a)$. Let $\deg(a) = m$, $\deg(b) = n$. Then the problem of determining whether $\gcd(a,b) = 1$ is equivalent to finding polynomials $u(x) = u_0 + u_1 x + \ldots u_{n-1}x^{n-1}$, $v(x) = v_0 + v_1 x + \ldots + v_{m-1}x^{m-1}$ s.t. identity (1) holds.

Let us view it as a linear system with unknowns $u_0, \ldots, u_{n-1}, v_0, \ldots, v_{m-1}$, where $u_0 a_0 + v_0 b_0 = 1$ and for any $i > 0$ the coefficient of x^i of the polynomial $u(x)a(x) + v(x)b(x)$ is 0. Hence, we have a system of linear equations (where $a_{-i} = b_{-i} = 0$ for $i > 0$):

$$a_0 u_0 + b_0 v_0 = 1 \tag{2}$$

$$\sum_{j=0}^{i} a_{i-j} u_j + \sum_{j=0}^{i} b_{i-j} v_j = 0 \qquad 1 \leq i \leq n+m-1 \tag{3}$$

In matrix form,

$$
m+n\left\{\begin{pmatrix}
\underbrace{\begin{matrix}
a_0 & & \\
a_1 & \ddots & \\
\vdots & \ddots \; \ddots & \\
a_m & \ddots & a_0 \\
 & \ddots & a_1 \\
 & \ddots & \vdots \\
 & & a_m
\end{matrix}}_{n}
&
\underbrace{\begin{matrix}
b_0 & & \\
b_1 & \ddots & \\
\vdots & \ddots & b_0 \\
\vdots & & b_1 \\
b_n & & \vdots \\
 & \ddots & \vdots \\
 & & b_n
\end{matrix}}_{m}
\end{pmatrix}\right.
\begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ \vdots \\ u_{n-1} \\ v_0 \\ v_1 \\ \vdots \\ v_{m-1} \end{pmatrix}
=
\begin{pmatrix} 1 \\ 0 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ 0 \end{pmatrix}
$$

The determinant of this matrix $M_{a,b}$ is called the *resultant* of $a(x)$ and $b(x)$: $\mathrm{Res}(a,b) \stackrel{\mathrm{def}}{=} \det(M_{a,b})$. The resultant is a classical example of an "algebraic invariant" and plays an important role in algorithmic algebra. We will study its properties next time; and in particular show that

$$\gcd(a,b) = 1 \iff \mathrm{Res}(a,b) \neq 0 \;\; (\iff \det(M_{a,b}) \neq 0)$$