CS 70: Homework #6

Abhijay Bhatnagar

October 12, 2018

Problem 1: Polynomial Practice 2
Problem 2: The CRT and Lagrange Interpolation 4
Problem 3: Old secrets, new secrets 6
Problem 4: Berlekamp-Welch for General Errors 7
Problem 5: Error-Detecting Codes 8

Problem 1: Polynomial Practice

a) If f and g are non-zero real polynomials, how many roots do the following polynomials have at least? How many can they have at most? (Your answer may depend on the degrees of f and g.)

i.) $f + g$

Solution: max(degree of f, degree of g).

Addition cannot change polynomial exponents.

ii.) $f \cdot g$

Solution: degree of f + degree of g.

The highest ordered term in multiplication is a combination of the highest ordered terms of the polynomials.

iii.) f/g, assuming that f/g is a polynomial.

Solution: degree of f - degree of g.

The highest ordered term in division is a subtraction of the highest ordered terms of the polynomials.

b) Now let f and g be polynomials over GF(p).

i.) If $f \cdot g = 0$, is it true that either $f = 0$ or $g = 0$?

Solution: Not necessarily. Consider $f = x + 1$ and $g = x + 2$ in $GF(2)$. $f \cdot g = 0$, but $f(0) = 1$ and $g(1) = 1$.

ii.) If $\deg f \geq p$, show that there exists a polynomial h with $\deg h < p$ such that $f(x) = h(x)$ for all $x \in \{0, 1, ..., p - 1\}$.

Solution:

Proof. Within $GF(p)$, all solutions of $f(x)$ are cyclical, i.e. $S_f = \{f(1), f(2), ..., f(n) : 0 \leq n < p\}$, giving us p solutions. From those, we can construct p pairs of points $(i, f(i))$, and use Lagrangian interpolation to find a function of degree $p-1$ that exactly fits those points. $\square$

iii.) How many f of degree *exactly* $d < p$ are there such that $f(0) = a$ for some fixed $a \in \{0, 1, \ldots, p-1\}$?

Solution:

Proof. m^d polynomials. We are given only 1 point, $(0, a)$, and so we have d points of freedom. $\square$

c) Find a polynomial f over GF(5) that satisfies $f(0) = 1, f(2) = 2, f(4) = 0$. How many such polynomials are there?

Solution: $f = \frac{-3}{8}x + \frac{5}{4}x + 1$. We are given three points, and you need five points to describe a degree 4 polynomial, so we have 2 points of freedom, meaning there are $5^2 = 25$ different polynomials.

Problem 2: The CRT and Lagrange Interpolation

Let $n_1, \ldots n_k$ be pairwise coprime, i.e. n_i and n_j are coprime for all $i \neq j$. The Chinese Remainder Theorem (CRT) tells us that there exist solutions to the following system of congruences:

$$x \equiv a_1 \pmod{n_1} \tag{1}$$
$$x \equiv a_2 \pmod{n_2} \tag{2}$$
$$\vdots \tag{$\vdots$}$$
$$x \equiv a_k \pmod{n_k} \tag{k}$$

and all solutions are equivalent $\pmod{n_1 n_2 \cdots n_k}$. For this problem, parts (a)-(c) will walk us through a proof of the Chinese Remainder Theorem. We will then use the CRT to revisit Lagrange interpolation.

a) We start by proving the $k = 2$ case: Prove that we can always find an integer x_1 that solves (1) and (2) with $a_1 = 1, a_2 = 0$. Similarly, prove that we can always find an integer x_2 that solves (1) and (2) with $a_1 = 0, a_2 = 1$.

Solution:

Proof. Let $n = 2$.

(i) $a_1 = 1, a_2 = 0$

$$x_1 \equiv 1 \pmod{n_1} \tag{1}$$
$$x_1 \equiv 0 \pmod{n_2} \tag{2}$$
$$\tag{3}$$

$x_1 = n_2 * (n_1 + 1)$ will always satisfy those constraints. ($x_1 = n_2 \pmod{n_1 n_2}$)

(ii) $a_1 = 0, a_2 = 1$

$$x_2 \equiv 0 \pmod{n_1} \tag{4}$$
$$x_2 \equiv 1 \pmod{n_2} \tag{5}$$
$$\tag{6}$$

$x_2 = n_1 * (n_2 + 1)$ will always satisfy those constraints. It is also unique ($x_2 = n_1 \pmod{n_1 n_2}$)

□

b) Use part (a) to prove that we can always find at least one solution to (1) and (2) for any a_1, a_2. Furthermore, prove that all possible solutions are equivalent $\pmod{n_1 n_2}$.

Solution:

Proof. Any solution of the form $x = (n_1 + a_1)(n_2 + a_2) \pmod{n_1}$ will work.

$\square$

c) Now we can tackle the case of arbitrary k: Use part (b) to prove that there exists a solution x to (1)-(k) and that this solution is unique $\pmod{n_1 n_2 \cdots n_k}$.

Solution: Using the proof from (b) [if I had one],

d) For two polynomials $p(x)$ and $q(x)$, mimic the definition of $a \bmod b$ for integers to define $p(x) \bmod q(x)$. Use your definition to find $p(x) \bmod (x-1)$.

e) Define the polynomials $x - a$ and $x - b$ to be coprime if they have no common divisor of degree 1. Assuming that the CRT still holds when replacing x, a_i and n_i with polynomials (using the definition of coprime polynomials just given), show that the system of congruences

$$p(x) \equiv y_1 \pmod{(x - x_1)} \tag{1'}$$
$$p(x) \equiv y_2 \pmod{(x - x_2)} \tag{2'}$$
$$\vdots \tag{$\vdots$}$$
$$p(x) \equiv y_k \pmod{(x - x_k)} \tag{k'}$$

has a unique solution $\pmod{(x-x_1)\cdots(x-x_k)}$ whenever the x_i are pairwise distinct. What is the connection to Lagrange interpolation?

Problem 3: Old secrets, new secrets

In order to share a secret number s, Alice distributed the values $(1, p(1)), (2, p(2)), \ldots, (n+1, p(n+1))$ of a degree n polynomial p with her friends $\text{Bob}_1, \ldots, \text{Bob}_{n+1}$. As usual, she chose p such that $p(0) = s$. Bob_1 through Bob_{n+1} now gather to jointly discover the secret. Suppose that for some reason Bob_1 already knows s, and wants to play a joke on $\text{Bob}_2, \ldots, \text{Bob}_{n+1}$, making them believe that the secret is in fact some fixed $s' \neq s$. How can he achieve this?

Solution: Bob uniquely knows $p(1)$, while all others know the x_1 and $\Delta_1(x)$ which implies $y_1 = p(1)$ is the only thing he can manipulate. The final solution is the polynomial constructed by $\sum_i y_i \Delta_i$, implying he can manipulate the $y_1 \Delta_1(x)$ term.

We can look at the final solution in term of the deltas, i.e.:

$$\begin{aligned} P(x) &= a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + s \\ &= p(1)\Delta_1(x) + p(2)\Delta_2(x) + \ldots + p(n+1)\Delta_{n+1}(x) \\ &= p(1)\Delta_1(x) + \sum_{i \neq 1} p(i)\Delta_i(x) \end{aligned}$$

From here we see how the polynomial is a sum of its Lagrangian parts:

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + s = p(1)\Delta_1(x) + \sum_{i \neq 1} p(i)\Delta_i(x)$$

However, since we are only concerned with altering the $P(0)$ term, all we need to do is consider the constants of the sub-polynomials, i.e.:

$$P(0) = s = p(1)\Delta_1^c(x) + \sum_{i \neq 1} p(i)\Delta_i^c(x)$$

We can determine $\Delta_1^c(x) = \frac{(x-2)(x-3)\cdot \ldots \cdot (x-(n+1))}{(1-2)(1-3)\cdot \ldots \cdot (1-(n+1))} = \frac{(n+1)!}{n!} = n+1$.

From here we need to construct a $p'(1)$ value s.t. $p'(1)\Delta_1^c(x) = p(1)\Delta_1^c(x) + s - s'$ (add s' and subtract s from both sides to make the algebra work out).

$\implies p'(1) = \frac{p(1)\Delta_1^c(x) + s - s'}{\Delta_1^c(x)} = p(1) + \frac{s-s'}{n+1}$.

We simply provide that $P'(1)$ value to deceive the others.

Problem 4: Berlekamp-Welch for General Errors

Suppose that Hector wants to send you a length $n = 3$ message, m_0, m_1, m_2, with the possibility for $k = 1$ error. For all parts of this problem, we will work mod 11, so we can encode 11 letters as shown below:

A	B	C	D	E	F	G	H	I	J	K
0	1	2	3	4	5	6	7	8	9	10

Hector encodes the message by finding the degree ≤ 2 polynomial $P(x)$ that passes through $(0, m_0)$, $(1, m_1)$, and $(2, m_2)$, and then sends you the five packets $P(0), P(1), P(2), P(3), P(4)$ over a noisy channel. The message you receive is

$$\text{DHACK} \Rightarrow 3, 7, 0, 2, 10 = r_0, r_1, r_2, r_3, r_4$$

which could have up to 1 error.

a) First, let's locate the error, using an error-locating polynomial $E(x)$. Let $Q(x) = P(x)E(x)$. Recall that

$$Q(i) = P(i)E(i) = r_i E(i), \quad \text{for} \quad 0 \leq i < n + 2k.$$

What is the degree of $E(x)$? What is the degree of $Q(x)$? Using the relation above, write out the form of $E(x)$ and $Q(x)$ in terms of the unknown coefficients, and then a system of equations to find both these polynomials.

Solution: E is a polynomial of degree 1. Q is a polynomial of degree (3+1-1=3).

$$E(x) = (x - e_1) = x + b_0$$

$$Q(x) = a_3 x^3 + a_2 x^2 + a_1 + x^1$$

I ran out of time but the rest of the problem is pretty easy. We can construct a system of equations using the $Q(i) = r_i E(i)$ property, solve the system of equations to get $Q(x)$ and $E(x)$, then we can solve $P(x) = Q(x)/E(x)$ and recompute the message.

b) Solve for $Q(x)$ and $E(x)$. Where is the error located?

c) Finally, what is $P(x)$? Use $P(x)$ to determine the original message that Hector wanted to send.

Problem 5: Error-Detecting Codes

Suppose Alice wants to transmit a message of n symbols, so that Bob is able to *detect* rather than *correct* any errors that have occurred on the way. That is, Alice wants to find an encoding so that Bob, upon receiving the code, is able to either

i.) tell that there are no errors and decode the message, or

ii.) realize that the transmitted code contains at least one error, and throw away the message.

Assuming that we are guaranteed a maximum of k errors, how should Alice extend her message (i.e. by how many symbols should she extend the message, and how should she choose these symbols)? You may assume that we work in GF(p) for very large prime p. Show that your scheme works, and that adding any lesser number of symbols is not good enough.

Solution: We want to be able to send a message of length n with a maximum of k errors s.t. Bob is able to detect the presence of errors and decode if not. We would need to extend the original message by k to send a final message of length $(n + k)$ that satisfies those requirements.

Proof. We claim we need to extend a message by at least k to be able to detect a maximum of k errors in the original message. The proof continues in two parts:

i.) Extending the message by k allows us to detect errors:

ii.) Extending the message by some $k' < k$ is insufficient.

For both parts of the proof, let us begin by defining the encoding from Alice. We can denote the message as $m_1, ..., m_n : m_i \in GF(q)$ where q is some prime number larger than the range of m_i. From there, we can find the unique polynomial of degree $n - 1$ s.t. $P(i) = m_i$ for $1 \leq i \leq n$, i.e. it passes through all points (i, m_i) for $1 \leq i \leq n$. Then we set the extended portion of the message to be equivalent to $(j, P(j))$ for $n < j \leq (n + k)$.

For part (i) of the proof, we have to show this encoding allows us to detect if up to k errors exist in a received message M (of length $n + k$). As Bob, we construct a polynomial P' that fits the original message length, i.e. the first n terms of (i, m_i). We are then able to detect if no errors exist by validating that all points (j, m_j) for $n < j \leq (n + k)$ lie in P'. Otherwise we can discard the message.

For part (ii) of the proof, we have to show that an extension of a lesser length would not be sufficient. The logic in its simple form is that if there are k errors but fewer redundant symbols, there could exist a scenario in which all redundant symbols are modified as well as a symbols in the original message, and those errors 'cancel out' in the detection algorithm. Formally, upon intercepting a message M, we could corrupt some symbol in $m_i : i \leq n$, construct a new polynomial $P_e(x)$ that fits this corrupted message, mutate each extended bit to fit this new polynomial, i.e., $m_{j'} = P_e(j) : n < j < n + k$. This new message has k corruptions, yet it would pass the detection algorithm. $\square$