Discrete Mathematics: Homework #5

Due on March 1, 2025 at 6:00pm

Professor Satish Rao

Zachary Brandt
zbrandt@berkeley.edu

Problem 1: Equivalent Polynomials

This problem is about polynomials with coefficients in GF(p) for some prime $p \in \mathbb{N}$. We say that two such polynomials f and g are *equivalent* if $f(x) \equiv g(x) \pmod{p}$ for every $x \in \text{GF}(p)$.

A) Show that $f(x) = x^{p-1}$ and $g(x) = 1$ are **not** equivalent polynomials under GF(p).

B) Use Fermat's Little Theorem to find a polynomial with degree strictly less than 5 that is equivalent to $f(x) = x^5$ over GF(5); then find a polynomial with degree strictly less than 11 that is equivalent to $g(x) = 4x^{70} + 9x^{11} + 3$ over GF(11).

C) In GF(p), prove that whenever $f(x)$ has degree $\geq p$, it is equivalent to some polynomial $\tilde{f}(x)$ with degree $< p$.

Part A

To show $f(x) \not\equiv g(x) \pmod{p}$, consider every $x \in \text{GF}(p)$. When $x = 0$, $f(0) = 0^{p-1} = 0$ and $g(0) = 1$, which are not equivalent under modulo p. Since $f(x)$ and $g(x)$ are not equivalent for all $x \in \text{GF}(p)$, they are not equivalent polynomials under GF(p).

Part B

Using exponent rules in modular arithmetic in conjuction with Fermat's Little Theorem, $f(x) \equiv x^5 \equiv x^4 \cdot x \equiv x \pmod{5}$, and x is a polynomial with degree strictly less than 5. For the second part, $g(x) \equiv 4x^{70} + 9x^{11} + 3 \equiv 4(x^{10})^7 + 9x^{10}x + 3 \equiv 9x + 7 \pmod{11}$, and $9x + 7$ is a polynomial with degree strictly less than 11.

Part C

If $f(x)$ is a polynomial under GF(p) with degree $\geq p$, then by recursively applying Fermat's Little Thereom to every term x^k, where $k \geq p$, can be reduced to $x^k \equiv 1 \cdot x^r \pmod{p}$, since $x^k = x^{l(p-1)+r}$ where $l, r \in \mathbb{Z}$. This results in a polynomial $\tilde{f}$ with degree $< p$ under in GF(p) and is true for any $f(x)$ with degree $\geq p$ over GF(p).

Problem 2: Secret Sharing

Suppose the Oral Exam questions are created by 2 TAs and 3 Readers. The answers are all encrypted, and we know that:

A) Two TAs together should be able to access the answers

B) Three Readers together should be able to access the answers

C) One TA and one Reader together should also be able to access the answers

D) One TA by themself or two Readers by themselves should not be able to access the answers.

Design a Secret Sharing scheme to make this work.

Solution

Create a polynomial of degree 6, give each TA four points and give each Reader three points. If both TAs collaborate they can reconstruct the polynomial to find the secret since they have at least 7 points. If all Readers collaborate they can also reconstruct the polynomial to find the secret with 2 points to spare. If one TA and one Reader collaborate they can reconstruct the polynomial with no points to spare. Since one TA only has four points and two Readers only have 6 points (1 point away from being able to reconstruct the polynomial), this Secret Sharing scheme satisfies the condition.

Problem 3: To The Moon!

A secret number s is required to launch a rocket, and Alice distributed the values $(1, p(1)), (2, p(2)), \ldots, (n+1, p(n+1))$ of a degree n polynomial $p(x)$ to a group of \$GME holders $\text{Bob}_1, \ldots, \text{Bob}_{n+1}$. As usual, she chose $p(x)$ such that $p(0) = s$. Bob_1 through Bob_{n+1} now gather to jointly discover the secret. However, Bob_1 is secretly a partner at Melvin Capital and already knows s, and wants to sabotage $\text{Bob}_2, \ldots, \text{Bob}_{n+1}$, making them believe that the secret is in fact some fixed $s' \neq s$. How could he achieve this? In other words, what value should he report (in terms variables known in the problem, such as s', s or $p(1)$) in order to make the others believe that the secret is s'?

Solution

For a polynomial $p'(x)$ constructed by the holders using Bob_1's $p'(1)$ (instead of his true $p(1)$), $p(x)$ and $p'(x)$ can be expressed as follows from Lagrange Interpolation. Both have their first terms of the sum seperated to highlight where Bob_1 fixes the result.

$$p(x) = p(1)\Delta_1(x) + \sum_{i=2}^{n+1} p(i)\Delta_i(x)$$
$$p'(x) = p'(1)\Delta_1(x) + \sum_{i=2}^{n+1} p(i)\Delta_i(x)$$

For obtaining the secret, $p(x)$ is evaluated at $x = 0$ to obtain s. Similarly, $p'(x)$ evaluated at $x = 0$ should then equal $p'(0) = s'$. Although we know that, for a basis polynomial $\Delta_i(x_j)$, when $i \neq j$, $\Delta_i(x_j) = 0$ and 1 otherwise for $i, j \in \{1, 2, \ldots, n+1\}$, we are not sure what $\Delta_i(x)$ evaluated at $x = 0$ is out of the gate.

$$s = p(0) = p(1)\Delta_1(0) + \sum_{i=2}^{n+1} p(i)\Delta_i(0)$$
$$s' = p'(0) = p'(1)\Delta_1(0) + \sum_{i=2}^{n+1} p(i)\Delta_i(0)$$

Solving for $p'(1)$ using the above equations for both the true and fixed secret results in a solution for $p'(1)$

$$p'(1) = \frac{s' - \sum_{i=2}^{n+1} p(i)\Delta_i(0)}{\Delta_1(0)}$$
$$p'(1) = \frac{s' - (s - p(1)\Delta_1(0))}{\Delta_1(0)}$$
$$p'(1) = \frac{s' - s}{\Delta_1(0)} + p(1)$$

From the defintion of the basis polynomials in Lagrange Interpolation, $\Delta_1(x) = \frac{\prod_{j=2}^{n+1}(x - x_j)}{\prod_{j=2}^{n+1}(1 - x_j)}$, which for $x = 0$ equals

$$\Delta_1(0) = \frac{\prod_{j=2}^{n+1}(0 - j)}{\prod_{j=2}^{n+1}(1 - j)} = \frac{\cancel{2} \cdot \cancel{3} \cdot \cdots \cdot \cancel{n} \cdot (n+1)}{1 \cdot \cancel{2} \cdot \cdots \cdot \cancel{n}} = \frac{n+1}{1}$$

Therefore, $p'(1) = \frac{s'-s}{n+1} + p(1)$.

Problem 4: Lagrange? More like Lamegrange.

A) Let's say we wanted to interpolate a polynomial through a single point, (x_0, y_0). What would be the polynomial that we would get? (This is not a trick question. A degree 0 polynomial is fine.)

B) Call the polynomial from the previous part $f_0(x)$. Now say we wanted to define the polynomial $f_1(x)$ that passes through the points (x_0, y_0) and (x_1, y_1). If we write $f_1(x) = f_0(x) + a_1(x - x_0)$, what value of a_1 causes $f_1(x)$ to pass through the desired points?

C) Now say we want a polynomial $f_2(x)$ that passes through (x_0, y_0), (x_1, y_1), and (x_2, y_2). If we write $f_2(x) = f_1(x) + a_2(x - x_0)(x - x_1)$, what value of a_2 gives us the desired polynomial?

D) Suppose we have a polynomial $f_i(x)$ that passes through the points (x_0, y_0), ..., (x_i, y_i) and we want to find a polynomial $f_{i+1}(x)$ that passes through all those points and also (x_{i+1}, y_{i+1}). If we define $f_{i+1}(x) = f_i(x) + a_{i+1}\prod_{j=0}^{i}(x - x_j)$, what value must a_{i+1} take on?

Part A

If we interpolate a polynomial through a single point (x_0, y_0), we get a 0 degree polynomial, i.e., the point itself for all x: $f_0(x) = y_0$.

Part B

When $x = x_0$, $f_1(x)$ is equal to $f_1(x_0) = f_0(x_0) + a_1(x_0 - x_0) = y_0$. Then for $x = x_1$, $f_1(x_1) = f_0(x_1) + a_1(x_1 - x_0) = y_0 + a_1(x_1 - x_0)$. $f_1(x_1)$ should equal y_1, so

$$y_1 = y_0 + a_1(x_1 - x_0)$$
$$a_1 = \frac{y_1 - y_0}{x_1 - x_0} = \frac{y_1 - f_0(x_1)}{x_1 - x_0}$$

Part C

For $f_2(x)$ evaluated at $x = x_0$ and $x = x_1$, we know that $f_1(x)$ will output our desired values of y_0 and y_1, and since the rest of the terms in $f_2(x)$ evaluate to zero, we know $f_2(x)$ passes through the first two of our points. For $x = x_2$, $f_2(x)$ is $f_2(x_2) = f_1(x_2) + a_2(x_2 - x_0)(x_2 - x_1)$. Therefore, a_2 must be

$$a_2 = \frac{y_2 - f_1(x_2)}{(x_2 - x_0)(x_2 - x_1)}$$

Part D

Like in the previous parts, for values $x_0, x_1, \ldots, x_i$, $f_{i+1}(x)$ will output $y_0, y_1, \ldots, y_i$ since the product $\prod_{j=0}^{i}(x - x_j)$ will end up being zero and $f_{i+1}(x)$ collapses to $f_i(x)$. Like before, consider $f_{i+1}(x_{i+1}) = f_i(x_{i+1}) + a_{i+1}\prod_{j=0}^{i}(x_{i+1} - x_j)$ and that it should equal y_{i+1}. Therefore, a_{i+1} is

$$y_{i+1} = f_i(x_{i+1}) + a_{i+1}\prod_{j=0}^{i}(x_{i+1} - x_j)$$
$$a_{i+1} = \frac{y_{i+1} - f_i(x_{i+1})}{\prod_{j=0}^{i}(x_{i+1} - x_j)}$$

Problem 5: Error-Correcting Codes

A) Recall from class the error-correcting code for erasure errors, which protects against up to k lost packets by sending a total of $n + k$ packets (where n is the number of packets in the original message). Often the number of packets lost is not some fixed number k, but rather a *fraction* of the number of packets sent. Suppose we wish to protect against a fraction α of lost packets (where $0 < \alpha < 1$). At least how many packets do we need to send (as a function of n and α)?

B) Repeat part (A) for the case of general errors.

Part A

For a total amount of packets sent l, if α are lost, the amount recieved is $(1 - \alpha) \cdot l$ packets. To reconstruct the polynomial to find the missing m's, the reciever needs to get at the very least n packets. So $n \geq (1-\alpha) \cdot l$ packets and l, the number of packets we need to send as a function of n and α, is $\frac{n}{1-\alpha}$, at least this amount.

Part B

For the case of general errors, when we know that k amount of packages are corrupted, we must send $n + 2k$ to recover the message. If l is again the total amount of packages sent, then we need to protect against $\alpha \cdot l$ errors. So, $l \geq n + 2l \cdot \alpha$, or $l \geq \frac{n}{1-2\alpha}$.

Problem 6: Alice and Bob

A) Alice decides that instead of encoding her message as the values of a polynomial, she will encode her message as the coefficients of a degree 2 polynomial $P(x)$. For her message $[m_1, m_2, m_3]$, she creates the polynomial $P(x) = m_1x^2 + m_2x + m_3$ and sends the five packets $(0, P(0))$, $(1, P(1))$, $(2, P(2))$, $(3, P(3))$, and $(4, P(4))$ to Bob. However, one of the packet y-values (one of the $P(i)$ terms; the second attribute in the pair) is changed by Eve before it reaches Bob. If Bob receives

$$(0, 1), (1, 3), (2, 0), (3, 1), (4, 0)$$

and knows Alice's encoding scheme and that Eve changed one of the packets, can he recover the original message? If so, find it as well as the x-value of the packet that Eve changed. If he can't, explain why. Work in mod 7. Also, feel free to use a calculator or online systems of equations solver, but make sure it can work under mod 7.

B) Bob gets tired of decoding degree 2 polynomials. He convinces Alice to encode her messages on a degree 1 polynomial. Alice, just to be safe, continues to send 5 points on her polynomial even though it is only degree 1. She makes sure to choose her message so that it can be encoded on a degree 1 polynomial. However, Eve changes two of the packets. Bob receives $(0, 5)$, $(1, 7)$, $(2, x)$, $(3, 5)$, $(4, 0)$. If Alice sent $(0, 5)$, $(1, 7)$, $(2, 9)$, $(3, -2)$, $(4, 0)$, for what values of x will Bob not uniquely be able to determine Alice's message? Assume that Bob knows Eve changed two packets. Work in mod 13. Again, feel free to use a calculator or graphing calculator software.

Hint: Observe that since Bob knows that Eve changed two packets, he's looking for a polynomial that passes through at least 3 of the given points. Think about what must happen in order for Bob to be unable to uniquely identify the original polynomial.

C) Alice wants to send a length n message to Bob. There are two communication channels available to her: Channel X and Channel Y. Only 6 packets can be sent through channel X. Similarly, Channel Y will only deliver 6 packets, but it will also corrupt (change the value) of one of the delivered packets. Using each of the two channels once, what is the largest message length n such that Bob so that he can always reconstruct the message?

Part A

To recover the original message, Bob will have to recover the polynomial $P(x)$ using the Berlekamp–Welch Algorithm. Since we know that Eve only changed one of the packets, the error locator polynomial is $E(x) = (x - e_1) = e + b_0$, as there is only one position that is in error. The system of equations to solve for b_0 is given by $Q(i) = P(i)E(i)$ for $0 \leq i \leq 4$ where $Q(x) = P(x)Q(x)$ is of the form $Q(x) = a_3 \cdot x^3 + a_2 \cdot x^2 + a_1 \cdot x + a_0$ since $P(x)$ is of degree 2 and $E(x)$ is of degree 1.

$$\begin{aligned}
(i = 0) \qquad & a_0 + 6b_0 \equiv 0 \pmod 7 \\
(i = 1) \qquad & a_3 + a_2 + a_1 + a_0 + 4b_0 \equiv 3 \pmod 7 \\
(i = 2) \qquad & a_3 + 4a_2 + 2a_1 + a_0 \equiv 0 \pmod 7 \\
(i = 3) \qquad & 6a_3 + 2a_2 + 3a_1 + a_0 + 6b_0 \equiv 3 \pmod 7 \\
(i = 4) \qquad & a_3 + 2a_2 + 4a_1 + a_0 \equiv 0 \pmod 7
\end{aligned}$$

Problem 6 continued on next page...

To solve this, we can row reduce a corresponding matrix for the system of equations under modulos 7.

$$\left[\begin{array}{ccccc|c} 0 & 0 & 0 & 1 & 6 & 0 \\ 1 & 1 & 1 & 1 & 4 & 3 \\ 1 & 4 & 2 & 1 & 0 & 0 \\ 6 & 2 & 3 & 1 & 6 & 3 \\ 1 & 2 & 4 & 1 & 0 & 0 \end{array}\right]$$

Solving this (on paper), the coefficients are $a_3 = 1, a_2 = 5, a_1 = 5, a_0 = 4$, and $b_0 = -3$. Therefore, the polynomial $Q(x)$ is $Q(x) = x^3 + 5x^2 + 5x + 4$ and the error location is $e_1 = -b_0 = 3$. The secret message is the coefficients of $P(x)$ which is

$$P(x) = \frac{Q(x)}{E(x)} = \frac{x^3 + 5x^2 + 5x + 4}{x - 3} = x^2 + 8x + 29 \textit{ (R: 91)} \equiv x^2 + x + 1 \pmod 7$$

So the secret message is then $m_1 = 1, m_2 = 2, m_3 = 1$ and the error is located at $i = 3$.

Part B

If Alice is encoding her messags on a 1 degree polynomial, all the points she attempts to send Bob must fall on the same line. For the corrupted message, Bob can still find the 1 degree polynomial from the 3 of the 5 points that are uncorrupted and still fall on the same line. So that Bob can no longer *uniquely* determine a 1 degree polynomial with the corrupted packets, there must be at least two sets of three points that determine two different lines.

The points $(0, 5), (1, 7), (4, 0)$ fall on Alice's 1 degree polynomial under modulo 13. An x value that determines another line is $x = 10$. Although it falls on Alice's original line, in conjunction with $(4, 0)$ and the corrupted $(3, 5)$ it forms another line with slope $m = -5$. $(3, 5)$ is also at the same y-level as the point $(0, 5)$, and so if $x = 5$, an additional, horizontal line is determined with a set of three points. Finally, $x = 6$ determines another line with points $(1, 7)$ and $(3, 5)$ with slope $m = -1$.

Part C

In total, Alice can send Bob 12 packets, with one of the packets corrupted. For general errors, i.e., corruption, to protect against k errors, a sender must send $n + 2k$ packages to protect against the errors. Therefore, 2 of the 12 packets sent by Alice are used to protect against one packet corruption, and the largest message she can send Bob is $n = 10$.