CS 70 Discrete Mathematics and Probability Theory
Summer 2020 Course Notes **DIS 3D**

1 Berlekamp-Welch Algorithm

In this question we will send the message $(m_0, m_1, m_2) = (1, 1, 4)$ of length $n = 3$. We will use an error-correcting code for $k = 1$ general error, doing arithmetic over GF(5).

(a) Construct a polynomial $P(x) \pmod 5$ of degree at most 2, so that

$$P(0) = 1, \qquad P(1) = 1, \qquad P(2) = 4.$$

What is the message $(c_0, c_1, c_2, c_3, c_4)$ that is sent?

(b) Suppose the message is corrupted by changing c_0 to 0. Set up the system of linear equations in the Berlekamp-Welch algorithm to find $Q(x)$ and $E(x)$.

(c) Assume that after solving the equations in part (b) we get $Q(x) = 4x^3 + x^2 + x$ and $E(x) = x$. Show how to recover the original message from Q and E.

Solution:

(a) We use Lagrange interpolation to construct the unique quadratic polynomial $P(x)$ such that $P(0) = m_0 = 1, P(1) = m_1 = 1, P(2) = m_2 = 4$. Doing all arithmetic over GF(5), so that i.e. $2^{-1} = 3 \pmod 5$,

$$\begin{aligned}
\Delta_0(x) &= \frac{(x-1)(x-2)}{(0-1)(0-2)} = \frac{x^2-3x+2}{2} \equiv 3(x^2-3x+2) \pmod 5 \\
\Delta_1(x) &= \frac{(x-0)(x-2)}{(1-0)(1-2)} = \frac{x^2-2x}{-1} \equiv 4(x^2-2x) \pmod 5 \\
\Delta_2(x) &= \frac{(x-0)(x-1)}{(2-0)(2-1)} = \frac{x^2-x}{2} \equiv 3(x^2-x) \pmod 5 \\
P(x) &= m_0\Delta_0(x) + m_1\Delta_1(x) + m_2\Delta_2(x) \\
&= 1\Delta_0(x) + 1\Delta_1(x) + 4\Delta_2(x) \\
&\equiv 4x^2 + x + 1 \pmod 5
\end{aligned}$$

For the final message we need to add 2 redundant points of P. Since 3 and 4 are the only points in GF(5) that we have not used yet, we compute $P(3) = 0, P(4) = 4$, and so our message is $(1, 1, 4, 0, 4)$.

(b) The message received is $(c'_0, c'_1, c'_2, c'_3, c'_4) = (0,1,4,0,4)$. Let $R(x)$ be the function such $R(i) = c'_i$ for $0 \le i < 5$. Let $E(x) = x + b_0$ be the error-locator polynomial, and $Q(x) = P(x)E(x) = a_3x^3 + a_2x^2 + a_1x + a_0$. Since $Q(i) = P(i)E(i) = R(i)E(i)$ for $1 \le i < 5$, we have the following equalities $\pmod 5$

$$\begin{aligned} Q(0) &= 0E(0) \\ Q(1) &= 1E(1) \\ Q(2) &= 4E(2) \\ Q(3) &= 0E(3) \\ Q(4) &= 4E(4) \end{aligned}$$

which can be rewritten as a system of linear equations

$$\begin{array}{rcrcrcrcrcl} & & & & & & a_0 & & & = & 0 \\ a_3 & + & a_2 & + & a_1 & + & a_0 & - & b_0 & = & 1 \\ 8a_3 & + & 4a_2 & + & 2a_1 & + & a_0 & - & 4b_0 & = & 8 \\ 27a_3 & + & 9a_2 & + & 3a_1 & + & a_0 & & & = & 0 \\ 64a_3 & + & 16a_2 & + & 4a_1 & + & a_0 & - & 4b_0 & = & 1 \end{array} .$$

(c) From the solution, we know

$$\begin{aligned} Q(x) &= 4x^3 + x^2 + x, \\ E(x) &= x + b_0 = x. \end{aligned}$$

Since $Q(x) = P(x)E(x)$, the recipient can compute $P(x) = Q(x)/E(x) = 4x^2 + x + 1$, the polynomial $P(x)$ from part (a) used by the sender. The error locating polynomial $E(x)$ is degree one, so there is only one error, and as $E(x) = x = x - 0$, the corrupted bit was the first one. To correct this error we evaluate $P(0) = 1$ and combine this with the two uncorrupted bits m_1, m_2, to get the original message

$$(m_0, m_1, m_2) = (1,1,4).$$

Note: initially there was a typo in this part of the problem, asking that we assume $Q(x) = 2x^3 + 2x^2$. From this assumption, we deduce that $P(x) = Q(x)/E(x) = 2x^2 + 2x$. The intended message would then be

$$(m_0, m_1, m_2) = (P(1), P(1), P(2)) = (0,4,2)$$

2 Secret Veto

In the usual secret-sharing scenario we consider (for instance) a secret vault at the United Nations, which we want to design with the property that any k representatives can pool their information and open it, but any smaller number has no hope of doing so. Assume that the solution in the notes has been implemented, so that the key is some number s, and each member has been assigned a number $f(i) \mod q$ for some degree $k-1$ polynomial f with coefficients in $\mathrm{GF}(q)$ and satisfying $f(0) = s$.

(a) A group of $k+\ell$ representatives get together to discuss opening the vault. What will happen if ℓ representatives are opposed to opening the vault and, instead of revealing their true numbers, secretly reveal some *different* numbers from $\mathrm{GF}(q)$? Will the group be able to open the vault? If so, how long will it take?

(b) Repeat part (a) in the event that only $\ell/2$ of the ℓ representatives in opposition reveal different numbers than they were assigned—assume that ℓ is even.

Solution:

(a) In this situation, the polynomial interpolating the $k+\ell$ points will be some polynomial $\tilde{f}$, differant then f, but with the property that $\tilde{f}(i) = f(i)$, for any representative i who gave the correct number. In fact, we claim that $\tilde{f}$ must have degree strictly larger than that of f. This is because Lagrange interpolation produces the *unique* polynomial of lowest degree passing through a given set of points: if f and $\tilde{f}$ had the same degree, they would have to be the same, since both pass through the k points $(i, f(i))$ of the representatives who answered truthfully.

In other words, when group sees that the polynomial has degree strictly larger than $k-1$, they will know that someone has given faulty information, but not who did so. By trying all possible subsets of k of the $k+\ell$ representatives, it would be possible to both recover the key and discover which members were engaging in the subterfuge. However, there are exponentially many such subsets in ℓ.

(b) If only $\ell/2$ representatives give incorrect information, the resulting polynomial will be of degree at most $k+\ell/2-1$. This time, however, the group can use the Berlekamp-Welch algorithm to efficiently figure out who was responsible and what the true polynomial is! Recall that in the Berlekam-Welch setting you receive a list $f(1), f(2), \ldots, f(k+\ell)$, of which up to $\ell/2$ entries may have been corrupted; the algorithm allows you to recover the locations of the errors and the true polynomial f. This setting is functionally the same: the group has access to $k+\ell$ evaluations of a polynomial, and they know that $k+\ell/2$ of them are correct. The only difference is that the polynomial is not evaluated at the points $1, 2, \ldots, k+\ell$, but rather at $k+\ell$ arbitrary points among the numbers $1, \ldots, n$.

3 Berlekamp-Welch Algorithm with Fewer Errors

In class we derived how the Berlekamp-Welch algorithm can be used to correct k general errors, given $n+2k$ points transmitted. In real life, it is usually difficult to determine the number of errors that will occur. What if we have less than k errors? This is a follow up to the exercise posed in the notes.

Suppose Alice wants to send 1 message to Bob and wants to guard against 1 general error. She decides to encode the message with $P(x) = 4$ (on $\mathrm{GF}(7)$) such that $P(0) = 4$ is the message she want to send. She then sends $P(0), P(1), P(2) = (4, 4, 4)$ to Bob.

(a) Suppose Bob receives the message $(4, 5, 4)$. Without performing Gaussian elimination explicitly, find $E(x)$ and $Q(x)$.

(b) Now, suppose there were no general errors and Bob receives the original message $(4,4,4)$. Show that the $Q(x), E(x)$ that you found in part (a) still satisfies $Q(i) = r_i E(i)$ for all $i = 0,1,2$.

(c) Verify that $E(x) = x$, $Q(x) = 4x$ is another possible set of polynomials that satisfies $Q(i) = r_i E(i)$ for all $i = 0,1,2$.

(d) Suppose you're actually trying to decode the received message $(4,4,4)$. Based on what you showed in the previous two parts, what will happen during row reduction when you try to solve for the unknowns?

(e) Prove that in general, no matter what the solution of $Q(x)$ and $E(x)$ are though, the recovered $P(x)$ will always be the same.

Solution:

(a) $E(x) = x - 1$ and $Q(x) = P(x)E(x) = 4x - 4$.

(b) This is true because there were no errors, so $P(i) = r_i$ for $i = 0,1,2$.

(c) Since $Q(x) = P(x)E(x)$ and $P(i) = r_i$ for $i = 0,1,2$, we must have $Q(i) = r_i E(i)$ for all $i = 0,1,2$.

(d) There are multiple solutions to the system of equations.

(e) Suppose we got two solutions $Q'(x), E'(x)$ and $Q(x), E(x)$. Since they are both solutions, by definition, we have $Q'(i) = r_i E'(i)$ and $Q(i) = r_i E(i)$ for $1 \le i \le n+2k$. Therefore, $Q'(i)E(i) = Q(i)E'(i) = r_i E(i)E'(i)$. However, $Q'(x)E(x) - Q(x)E'(x)$ is a degree $n+2k-1$ polynomial, which is 0 at $n+2k$ points. Thus, $Q'(x)E(x) = Q(x)E'(x)$ for all x, so we arrive at

$$\frac{Q'(x)}{E'(x)} = \frac{Q(x)}{E(x)}.$$

This proves that the final solution for $P(x)$ is the same.

4 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:

Since k bits can break, it seems reasonable to extend our message by k symbols for a total of $n+k$. And indeed, we show that this works: Let Alice generate her message $y_0, \dots y_{n-1}$ of length n by constructing the unique polynomial f of degree $\leq n-1$ that passes through (i, y_i) for $i \in \{0, \dots, n-1\}$, and add the k extra symbols $y_j = f(j)$, where $j \in \{n, \dots, n+k-1\}$. Now Bob receives the message $r_i, i \in \{0, \dots, n+k-1\}$, upon which he interpolates the unique degree $\leq n-1$ polynomial g that passes through the points $(0, r_0), \dots, (n-1, r_{n-1})$. We claim that the message is corrupted if and only if $g(i) \neq r_i$ for some $i \in \{n, \dots, n+k-1\}$.

The backward direction becomes clear when stated as its contrapositive: If the message contains no error, then $g(i)$ and $f(i)$ coincide on all of n points $\{0, \dots, n-1\}$. Since they are both of degree $n-1$, they must be the same polynomial and hence $g(i) = f(i) = r_i$ for all i.

Let us prove the forward direction: Since we know that at most k errors occured, there must exist a subset $A \subset \{0, \dots, n+k-1\}$ of size n on which $r_i = y_i$. Now either

1. $A = \{0, \dots, n-1\}$, in which case $g = f$ and at least one error must have occured for some $j_0 \in \{n, \dots, n+k-1\}$. But then $r_{j_0} \neq y_{j_0} = f(j_0) = g(j_0)$, which is what we wanted to show.

2. Or at least one error occured for an index $i \in \{0, \dots, n-1\}$ in which case $g \neq f$. But since g and f are of degree $n-1$ and $\mid A \mid = n$, f and g cannot take the same values on A, so there must be some element $j_0 \in A, j_0 \in \{n, \dots, n+k-1\}$ for which $g(j_0) \neq f(j_0) = y_{j_0} = r_{j_0}$.

Lastly, we need to show that our algorithm doesn't work if Alice extends her message by less than k symbols, which we can do by crafting a counterexample: Assume Alice sends $m < n+k-1$ symbols in the same fashion as above, then we may corrupt $y_{n-1}, \dots y_{m-1}$ by setting $r_{n-1} \neq y_{n-1}$ and $r_j = h(j)$ for $j \in \{n, \dots, m-1\}$, where h is the unique polynomial of degree $\leq n-1$ passing through $(0, y_0), \dots (n-2, y_{n-2}), (n-1, r_{n-1})$. Since Bob is going to reconstruct $g = h$, $g(j) = r_j$ for all $j \in \{n, \dots, m-1\}$ and he will not notice the corruption.