CS 70 Discrete Mathematics and Probability Theory
Spring 2017 Rao

# HW 6

## 1 Sundry

Before you start your homework, write down your team. Who else did you work with on this homework? List names and email addresses. (In case of homework party, you can also just describe the group.) How did you work on this homework? Working in groups of 3-5 will earn credit for your "Sundry" grade.

Please copy the following statement and sign next to it:

*I certify that all solutions are entirely in my words and that I have not looked at another student's solutions. I have credited all external sources in this write up.*

I certify that all solutions are entirely in my words and that I have not looked at another student's solutions. I have credited all external sources in this write up. (Signature here)

## 2 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 $\alpha$ of lost packets (where $0 < \alpha < 1$). At least how many packets do we need to send (as a function of $n$ and $\alpha$)?

(b) Repeat part (a) for the case of general errors.

**Solution:**

(a) Suppose we send a total of $m$ packets (where $m$ is to be determined). Since at most a fraction $\alpha$ of these are lost, the number of packets received is at least $(1-\alpha)m$. But in order to reconstruct the polynomial used in transmission, we need at least $n$ packets. Hence it is sufficient to have $(1-\alpha)m \geq n$, which can be rearranged to give $m \geq n/(1-\alpha)$.

(b) Suppose we send a total of $m = n+2k$ packets, where $k$ is the number of errors we can guard against. The number of corrupted packets is at most $\alpha m$, so we need $k \geq \alpha m$. Hence $m \geq n+2\alpha m$. Rearranging gives $m \geq n/(1-2\alpha)$.

**Note**: Recovery in this case is impossible if $\alpha \geq 1/2$.

# 3 Polynomials in One Indeterminate

We will now prove a fundamental result about polynomials: every non-zero polynomial of degree $n$ (over a field $F$) has at most $n$ roots. If you don't know what a field is, you can assume in the following that $F = \mathbb{R}$ (the real numbers).

(a) Show that for any $\alpha \in F$, there exists some polynomial $Q(x)$ of degree $n-1$ and some $b \in F$ such that $P(x) = (x-\alpha)Q(x) + b$.

(b) Show that if $\alpha$ is a root of $P(x)$, then $P(x) = (x-\alpha)Q(x)$.

(c) Prove that any polynomial of degree 1 has at most one root. This is your base case.

(d) Now prove the inductive step: if every polynomial of degree $n-1$ has at most $n-1$ roots, then any polynomial of degree $n$ has at most $n$ roots.

**Solution:**

(a) Let

$$P(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots a_1 x + a_0.$$

We need to show that there is a polynomial

$$Q(x) = b_{n-1}x^{n-1} + b_{n-2}x^{n-2} + \ldots b_1 x + b_0$$

and $b \in F$ such that $Q(x)(x-\alpha) + b = P(x)$.

$$\begin{aligned} Q(x)(x-\alpha) + b = b_{n-1}x^n + (b_{n-2} - \alpha b_{n-1})x^{n-1} + (b_{n-3} - \alpha b_{n-2})x^{n-2} \\ + \cdots + (b_0 - \alpha b_1)x - \alpha b_0 + b \end{aligned}$$

Therefore if we set

$$\begin{aligned} b_{n-1} &= a_n \\ b_{n-2} &= a_{n-1} + \alpha b_{n-1} \\ b_{n-3} &= a_{n-2} + \alpha b_{n-2} \\ &\vdots \\ b_0 &= a_1 + \alpha b_1 \\ b &= a_0 + \alpha b_0 \end{aligned}$$

we get the desired equality.

(b) If $\alpha$ is a root of $P(x)$, $0 = P(\alpha) = (\alpha - \alpha)Q(\alpha) + b = 0 \cdot Q(\alpha) + b = b$ (where we used the theorem for general fields that $a0 = 0$). Hence $P(x) = (x-\alpha)Q(x)$.

(c) **Base case:** Consider a nonzero polynomial $P(x) = a_1x + a_0$. If there exists a root of the polynomial $\alpha$, $P(\alpha) = 0$. That is:

$$\begin{aligned} a_1\alpha + a_0 &= 0 \\ a_1\alpha &= -a_0 \end{aligned}$$

If $a_1 \neq 0$ multiplying both sides by $a_1^{-1}$ yields $\alpha = a_1^{-1}(-a_0)$, so there is exactly one possible value for $\alpha$. If $a_1 = 0$, then $a_1\alpha = 0$ (here we're again using the theorem $a0 = 0$). So the equation can be satisfied only if $a_0 = 0$, but we said $P(x)$ is the nonzero polynomial, so there is no such $\alpha$. In both cases there is at most one possible value for $\alpha$.

(d) **Inductive step:** Suppose every polynomial of degree $n$ has at most $n$ roots. Consider a polynomial $P(x)$ of degree $n+1$. We need to show that it has at most $n+1$ roots. We'll prove it by contradiction. Suppose $P(x)$ has at least $n+2$ distinct roots: $\alpha_1, \alpha_2, \ldots, \alpha_{n+2}$. By parts (a) and (b) $P(x) = (x-\alpha_1)Q(x)$ for some polynomial $Q(x)$ of degree $n$. Plugging for $x$ the values $\alpha_2, \alpha_3, \ldots, \alpha_{n+2}$, and using the fact that they are roots of $P(x)$ we get

$$0 = P(\alpha_i) = (\alpha_i - \alpha_1)Q(\alpha_i)$$

for $i = 2, 3, \ldots, n+2$. Since $\alpha_i \neq \alpha_1$, $\alpha_i - \alpha_1 \neq 0$ (this is easy to prove by contradiction by adding $\alpha_1$ to both sides). So we can multiply both sides by $(\alpha_i - \alpha_1)^{-1}$ and get

$$Q(\alpha_i) = 0(\alpha_i - \alpha_1)^{-1} = 0.$$

Therefore $\alpha_i$ for $i = 2, 3, \ldots, n$ are all roots of $Q(x)$. There are $n+1$ of them, but we know that since Q(x) is of degree $n$ it has at most $n$ roots by the induction hypothesis, so we've reached a contradiction.

# 4 Properties of GF($p$)

(a) Show that, if $p(x)$ and $q(x)$ are polynomials over the reals (or complex, or rationals) and $p(x) \cdot q(x) = 0$ for all $x$, then either $p(x) = 0$ for all $x$ or $q(x) = 0$ for all $x$ or both. (*Hint*: You may want to prove first this lemma, true in all fields: The roots of $p(x) \cdot q(x)$ is the union of the roots of $p(x)$ and $q(x)$.)

(b) Show that the claim in part (a) is false for finite fields GF($p$).

**Solution:**

(a) First, notice that if $r$ is a root of $p(x)$ such that $p(r) = 0$, then $r$ must also be a root of $p(x) \cdot q(x)$, since $p(r) \cdot q(r) = 0 \cdot q(r) = 0$. The same is true for any roots of $q(x)$. Also notice that if some value $s$ is neither a root of $p(x)$ nor $q(x)$, such that $p(s) \neq 0$ and $q(s) \neq 0$, then $s$ cannot be a root of $p(x) \cdot q(x)$ since $p(s) \cdot q(s) \neq 0$. We therefore conclude that the roots of $p(x) \cdot q(x)$ is the union of the roots of $p(x)$ and $q(x)$.

Now we will show the contrapositive. Suppose that $p(x)$ and $q(x)$ are both non-zero polynomials of degree $d_p$ and $d_q$ respectively. Then $p(x) = 0$ for at most $d_p$ values of $x$ and $q(x) = 0$ for at most $d_q$ values of $x$. This implies that $p(x) \cdot q(x)$ has at most $d_p + d_q$ roots. Since there are an infinite number of values for $x$ (because we are using complex, real, or rational numbers) we can always find an $x$, call it $x_{\text{not zero}}$, for which $p(x_{\text{not zero}}) \neq 0$ and $q(x_{\text{not zero}}) \neq 0$. This gives us $p(x_{\text{not zero}}) \cdot q(x_{\text{not zero}}) \neq 0$ so $pq$ is non-zero.

(b) In GF($p$) where $p$ is prime, $x^{p-1} - 1$ and $x$ are both non zero polynomials, but their product, $x^p - x$ is zero for all $x$ by Fermat's Little Theorem.

Examples for a specific p are also acceptable. For example, for GF(2), $p(x) = x$ and $q(x) = x + 1$.

# 5 Poker Mathematics

A *pseudo-random number generator* is a way of generating a large quantity of random-looking numbers, if all we have is a little bit of randomness (known as the *seed*). One simple scheme is the *linear congruential generator*, where we pick some modulus $m$, some constants $a, b$, and a seed $x_0$, and then generate the sequence of outputs $x_0, x_1, x_2, x_3, \ldots$ according to the following equation:

$$x_{t+1} = ax_t + b \pmod{m}$$

(Notice that $0 \leq x_t < m$ holds for every $t$.)

You've discovered that a popular web site uses a linear congruential generator to generate poker hands for its players. For instance, it uses $x_0$ to pseudo-randomly pick the first card to go into your hand, $x_1$ to pseudo-randomly pick the second card to go into your hand, and so on. For extra security, the poker site has kept the parameters $a$ and $b$ secret, but you do know that the modulus is $m = 2^{31} - 1$ (which is prime).

Suppose that you can observe the values $x_0$, $x_1$, $x_2$, $x_3$, and $x_4$ from the information available to you, and that the values $x_5, \ldots, x_9$ will be used to pseudo-randomly pick the cards for the next person's hand. Describe how to efficiently predict the values $x_5, \ldots, x_9$, given the values known to you.

**Solution:**

**Answer 1:** We know:

$$\begin{aligned} x_1 &\equiv ax_0 + b \pmod{m} \\ x_2 &\equiv ax_1 + b \pmod{m} \end{aligned}$$

Because we know $x_0$, $x_1$, and $x_2$, this is a system of two equations with two unknowns (namely, $a$ and $b$). So we can solve for $a$ and $b$. More explicitly, by subtracting the first equation from the second, we get

$$x_2 - x_1 \equiv a(x_1 - x_0) \pmod{m}.$$

If $x_0 = x_1$ then by induction on $n$ we see $x_n = x_0$ for all $n$ which allows us to immediately calculate $x_5, x_6, x_7, x_8, x_9$. So suppose $x_0 \neq x_1$. Then $x_1 - x_0$ is invertible modulo $m$ (because $m$ is prime,

therefore $\gcd(x_0 - x_1, m) = 1$), and we see

$$a \equiv (x_2 - x_1)(x_1 - x_0)^{-1} \pmod{m}.$$

Once we know $a$, we can plug in the known value of $a$ into the first equation and solve for $b$:

$$b \equiv x_1 - ax_0 \equiv x_1 - x_0(x_2 - x_1)(x_1 - x_0)^{-1} \pmod{m}$$

Since we know $a$ modulo $m$ and $b$ modulo $m$, we can compute $x_5, x_6, x_7, x_8$ and $x_9$.

**Answer 2:** Alternatively, we could start by solving for $b$. Multiplying the first equation by $x_1$, multiplying the second equation by $x_0$, and subtracting gives $x_1^2 - x_2x_0 \equiv b(x_1 - x_0) \pmod{m}$, and then

$$b \equiv (x_1^2 - x_0x_2)(x_1 - x_0)^{-1} \pmod{m}.$$

Now we can plug in the known value of $b$ into the first equation and solve for $a$, and continue as before.

**Note:** For both answers, It is important to consider the case when $x_1 \equiv x_0 \pmod{m}$. Solutions that didn't address that were incomplete, as $x_1 - x_0$ would not necessarily have an inverse modulo $m$.

**Comment:** This homework exercise was loosely modelled after a real-life story, where a group of computer scientists discovered serious flaws in the pseudorandom number generator used by several real poker sites. They could have used this to make thousands of dollars off everyone else who played at that site, but rather than cheat, they instead revealed the flaw to the poker site and the public. For more, see their paper "How We Learned to Cheat at Online Poker: A Study in Software Security" (`http://www.cigital.com/papers/download/developer_gambling.php`).

## 6 Secret Sharing with Spies

An officer stored an important letter in her safe. In case she is killed in battle, she decides to share the password (which is a number) with her troops. However, everyone knows that there are 3 spies among the troops, but no one knows who they are except for the three spies themselves. The 3 spies can coordinate with each other and they will either lie and make people not able to open the safe, or will open the safe themselves if they can. Therefore, the officer would like a scheme to share the password that satisfies the following conditions:

- When $M$ of them get together, they are guaranteed to be able to open the safe even if they have spies among them.
- The 3 spies must not be able to open the safe all by themselves.

Please help the officer to design a scheme to share her password. What is the scheme? What is the smallest $M$? Show your work and argue why your scheme works and any smaller $M$ couldn't work.

**Solution:**

The key insight is to realize that both polynomial-based secret-sharing and polynomial-based error correction work on the basis of evaluating an underlying polynomial at many points and then trying to recover that polynomial. Hence they can be easily combined.

Suppose the password is $s$. The officer can construct a polynomial $P(x)$ such that $s = P(0)$ and share $(i, P(i))$ to the $i$-th person in her troops. Then the problem is: what should the degree of $P(x)$ be and what is the smallest $M$?

First, the degree of polynomial $d$ should not be less than 3. It is because when $d < 3$, the 3 spies can decide the polynomial $P(x)$ uniquely. Thus, $n$ will be at least 4 symbols.

Let's choose a polynomial $P(x)$ of degree 3 such that $s = P(0)$. We now view the 3 spies as 3 general errors. Then the smallest $M = 10$ since $n$ is at least 4 symbols and we have $k = 3$ general errors, leading us to a "codeword" of $4 + 2 \cdot 3 = 10$ symbols (or people in our case). Even though the 3 spies are among the 10 people and try to lie on their numbers, the 10 people can still be able to correct the $k = 3$ general errors by the Berlekamp-Welch algorithm and find the correct $P(x)$.

**Alternative solution:**
Another valid approach is making $P(x)$ of degree $M - 1$ and adding 6 public points to deal with 3 general errors from the spies. In other words, in addition to their own point $(i, P(i))$, everyone also knows the values of 6 more points, $(t+1, P(t+1)), (t+2, P(t+2)), \ldots, (t+6, P(t+6))$, where $t$ is the number of the troops. The spies have access to total of $3 + 6 = 9$ points so the degree $M - 1$ must be at least 9 to prevent the spies from opening the safe by themselves. Therefore, the minimum $M$ is 10.

# 7 Berlekamp-Welch Algorithm

In this question we will go through an example of error-correcting codes with general errors. We will send a message $(m_0, m_1, m_2)$ of length $n = 3$. We will use an error-correcting code for $k = 1$ general error, doing arithmetic modulo 5.

(a) Suppose $(m_0, m_1, m_2) = (4, 3, 2)$. Use Lagrange interpolation to construct a polynomial $P(x)$ of degree 2 (remember all arithmetic is mod 5) so that $(P(0), P(1), P(2)) = (m_0, m_1, m_2)$. Then extend the message to length $n + 2k$ by appending $P(3), P(4)$. What is the polynomial $P(x)$ and what is the message $(c_0, c_1, c_2, c_3, c_4) = (P(0), P(1), P(2), P(3), P(4))$ that is sent?

(b) Suppose the message is corrupted by changing $c_0$ to 0. We will locate the error using the Berlekamp-Welch method. Let $E(x) = x + b_0$ be the error-locator polynomial, and $Q(x) = P(x)E(x) = a_3x^3 + a_2x^2 + a_1x + a_0$ be a polynomial with unknown coefficients. Write down the system of linear equations (involving unknowns $a_0, a_1, a_2, a_3, b_0$) in the Berlekamp-Welch method. You need not solve the equations.

(c) The solution to the equations in part (b) is $b_0 = 0, a_0 = 0, a_1 = 4, a_2 = 4, a_3 = 0$. Show how the recipient can recover the original message $(m_0, m_1, m_2)$.

**Solution:**

(a) We use Lagrange interpolation to construct the unique quadratic polynomial $P(x)$ such that $P(0) = m_0 = 4, P(1) = m_1 = 3, P(2) = m_2 = 2$.

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

[Note that all arithmetic is $\bmod 5$, so for example $2^{-1} \equiv 3 \pmod 5$.] Then we compute $P(3) = 1$ and $P(4) = 0$, so our message is 43210.

(b) The message received is $(c'_0, c'_1, c'_2, c'_3, c'_4) = (0,3,2,1,0)$. 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 $\pmod 5$:

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

They lead to the following system of linear equations:

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

(c) From the solution, we know

$$\begin{aligned}
Q(x) &= a_3x^3 + a_2x^2 + a_1x + a_0 = -x^2 + 4x, \\
E(x) &= x + b_0 = x.
\end{aligned}$$

Since $Q(x) = P(x)E(x)$, the recipient can compute $P(x) = Q(x)/E(x) = -x+4$ [note that this is the same polynomial $P(x)$ from part (a) used by the sender]. The recipient may deduce the location of the error from $E(x)$ as follows. There is only one error at location $e_1$, we have $E(x) = (x - e_1) = x$, so $e_1 = 0$ and the error is at position 0. To correct the error we evaluate $P(0) = 4$. Since the other two positions $m_1, m_2$ of the message are uncorrupted, we recover the original message $(m_0, m_1, m_2) = (4,3,2)$.

# 8 Countability Introduction

(a) Do $(0,1)$ and $\mathbb{R}_+ = (0,\infty)$ have the same cardinality? If so, give an explicit bijection (and prove that it's a bijection). If not, then prove that they have different cardinalities.

(b) Is the set of English strings countable? (Note that the strings may be arbitrarily long, but each string has finite length.) If so, then provide a method for enumerating the strings. If not, then use a diagonalization argument to show that the set is uncountable.

(c) Consider the previous part, except now the strings are drawn from a countably infinite alphabet $\mathscr{A}$. Does your answer from before change? Make sure to justify your answer.

**Solution:**

(a) Yes, they have the same cardinality. Consider the bijection $f : (0,1) \to (0,\infty)$ given by

$$f(x) = \frac{1}{x} - 1.$$

$f$ is one-to-one: suppose that $f(x) = f(y)$. Then,

$$\begin{aligned} \frac{1}{x} - 1 &= \frac{1}{y} - 1, \\ \frac{1}{x} &= \frac{1}{y}, \\ x &= y. \end{aligned}$$

Hence, $f$ is one-to-one.

$f$ is onto: take any $y \in \mathbb{R}_+$. Let $x = 1/(1+y) \in (0,1)$. Then,

$$f(x) = \frac{1}{1/(1+y)} - 1 = 1 + y - 1 = y,$$

so $f$ maps $x$ to $y$. Hence, $f$ is onto.

We have exhibited a bijection from $(0,1)$ to $\mathbb{R}_+$, so they have the same cardinality. (In fact, they are both uncountable.)

(b) Countable.

We can enumerate the strings as follows: list all strings of length 1, and then all strings of length 2, and then strings of length 3, and so forth. At each step, there are only finitely many strings of a particular length $\ell$, so this algorithm never gets "stuck" at any length.

(c) Yes, the strings are still countable, although the enumeration requires a little more finesse. Notice that if we tried to list all strings of length 1, we would be stuck forever, since the

alphabet is infinite! On the other hand, if we try to restrict our alphabet and only print out strings containing the first character $a \in \mathscr{A}$, we would also have a similar problem: the list

$$a, aa, aaa, \ldots$$

also does not end.

Let $\mathscr{A} = \{a_1, a_2, \ldots\}$ denote the alphabet. (We are making use of the fact that the alphabet is countably infinite when we assume there is such an enumeration.) The idea is to restrict *both* the length of the string and the characters we are allowed to use:

(a) List all strings containing only $a_1$ which are of length at most 1.

(b) List all strings containing only characters in $\{a_1, a_2\}$ which are of length at most 2.

(c) List all strings containing only characters in $\{a_1, a_2, a_3\}$ which are of length at most 3.

(d) Proceed onwards.

At each step, we have restricted ourselves to a finite alphabet with a finite length, so each step is guaranteed to terminate. To show that the enumeration is complete, consider any string $s$ of length $\ell$; since the length is finite, it can contain at most $\ell$ distinct $a_i$ from the alphabet. Let $k$ denote the largest index of any $a_i$ which appears in $s$. Then, $s$ will be listed in step $\max(k, \ell)$, so it appears in the enumeration.

In fact, the above idea should look somewhat familiar to you: this is the same method we used to show that $\mathbb{Z} \times \mathbb{Z}$ (and therefore $\mathbb{Q}$) is countable. We produce a table

| | $\{a_1\}$ | $\{a_1, a_2\}$ | $\cdots$ |
|---|---|---|---|
| 1 | $\swarrow$ | $\swarrow$ | $\swarrow$ |
| 2 | $\swarrow$ | $\swarrow$ | |
| $\vdots$ | $\swarrow$ | | |

where the column represents which characters we are allowed to use in the string, and the row represents the maximum length of the string. To enumerate all of the strings, we proceed diagonally through the table.