EECS 70 Discrete Mathematics and Probability Theory
Fall 2014 Anant Sahai

Midterm 2

Exam location: 10 Evans, Last name starting with A-B or R-T

PRINT your student ID: _____

PRINT AND SIGN your name: _____, _____ _____
(last) (first) (signature)

PRINT your Unix account login: cs70-_____

PRINT your discussion section and GSI (the one you attend): _____

Name of the person to your left: _____

Name of the person to your right: _____

Name of the person in front of you: _____

Name of the person behind you: _____

Section 0: Pre-exam questions (3 points)

1. **What is your favorite part of 70 so far? (1 pt)**

2. **Describe how you would feel in your dream stress-free vacation. (2 pts)**

Do not turn this page until the proctor tells you to do so. You can work on Section 0 above before time starts.

PRINT your name and student ID: _____

Section 1: Straightforward questions (30 points)

Unless told otherwise, you must show work to get credit. You get one drop: do 5 out of the following 6 questions. (We will grade all 6 and keep only the best 5 scores) However, there will be essentially no partial credit given in this section. Students who get all 6 questions correct will receive some bonus points.

3. Casting Out 9's

Let a number $N = a_0 + a_1 * 10 + a_2 * 10^2 + \cdots + a_k * 10^k$. Prove that $N \equiv a_0 + a_1 + a_2 + \cdots + a_k \pmod 9$.

Since $10 \equiv 1 \pmod 9$, we can write

$$\begin{aligned} & a_0 + a_1 * 10 + a_2 * 10^2 + \cdots + a_k * 10^k \pmod 9 \\ \equiv & a_0 + a_1 * 1 + a_2 * 1^2 + \cdots + a_k * 1^k \pmod 9 \\ \equiv & a_0 + a_1 + a_2 + \cdots + a_k \pmod 9. \end{aligned}$$

PRINT your name and student ID: _____

4. Interpolate!

Find the lowest-degree polynomial $P(x)$ that passes through the points $(1,4),(2,3),(5,0)$ in mod 7.

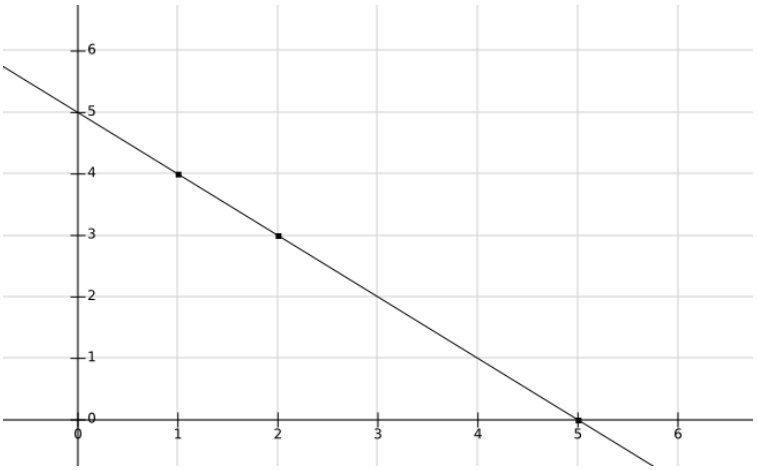
First, observe that we don't need to compute $\Delta_5(x)$, since it will be multiplied by 0 anyway.

$$\Delta_1(x) \equiv \frac{(x-2)(x-5)}{(1-2)(1-5)} \equiv \frac{x^2-7x+10}{4} \equiv 2\cdot(x^2+3) \equiv 2x^2+6 \pmod 7$$

$$\Delta_2(x) \equiv \frac{(x-1)(x-5)}{(2-1)(2-5)} \equiv \frac{x^2-6x+5}{-3} \equiv 2\cdot(x^2-6x+5) \equiv 2x^2+2x+3 \pmod 7$$

$P(x) \equiv y_1\Delta_1(x)+y_2\Delta_2(x) \equiv 4\cdot(2x^2+6)+3\cdot(2x^2+2x+3) \equiv 14x^2+6x+33 \equiv \boxed{6x+5} \pmod 7$.

Alternatively, you can graph the points in $GF(7)$ and observe that they all lie on $y=-x+5$, which is equivalent to $\boxed{6x+5}$ mod 7.



5. RSA Compute

Cindy is sending a message to Tommy with RSA. If she uses the public key $e = 3, N = 55$, what is the value of d that Tommy must use to decrypt the message?

We can easily figure out the primes p, q since $N = 55 = 5 \times 11$.

[2mm] We want $d = e^{-1} \bmod (p-1)(q-1)$, or $d = 3^{-1} \bmod 40$. Running the extended-gcd algorithm gives us

$$\begin{aligned} 1 &= 40 - 3(13) \\ 1 &= 40(1) + (3)(-13) \end{aligned} \tag{1}$$

and hence $3^{-1} \equiv -13 \equiv 27 \pmod{40}$. Now we have $\boxed{d = 27}$.

Alternatively, you only need to test 3 numbers* that are 1 mod 40: 41, 81, 121. From here, we see that 81 is divisible by 3, or that 27 is the number we are looking for.

[3mm]* Realize that the results repeat every 3×40, or that for example the result for 81 would be the same as that for 201. For 81, we have ed = 81 = 1 mod 40, or $d = 27$. For 201, we have ed = 201 = 1 mod 40, or $d = 67 = 27$ mod 40.

PRINT your name and student ID: _____

6. Packets

(a) You would like to send a message of length n packets to your friend. You know that at most k errors can occur during transmission. However, you are guaranteed that the first $n-1$ packets you send will arrive uncorrupted. How many packets must you send to guarantee successful transmission of your entire message? Why?

$\boxed{n+2k}$ packets.

We need to send $n+2k$ packets to protect against k general errors. If the first $n-1$ packets are guaranteed to be correct, the problem reduces to sending 1 packet with k possible general errors. Thus, we need to send $n-1+1+2k = n+2k$ total packets.

(b) You would like to send a message of length n over a channel. You know that at most k packets may be dropped along the way, and of the packets that are not dropped, at most j may be corrupted. How many packets should you send to guarantee successful decoding? Why? (No proof required)

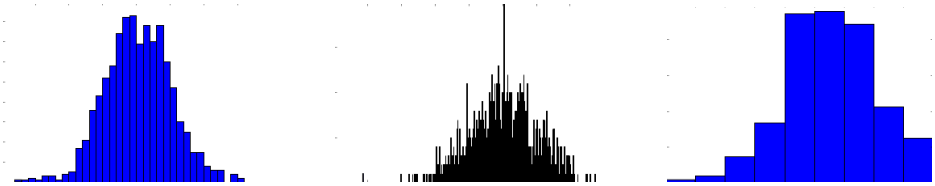
$\boxed{n+k+2j}$ packets.

Considering the worst case, where k packets are dropped and j packets are corrupted, we will need k more packets to protect from the erasure errors, and then $2j$ more packets to protect from the general errors. Any other edge case must be upper bounded by this number (for example, if 0 packets dropped and j packets corrupted, then we need $n+2j$ total packets).

PRINT your name and student ID: _____

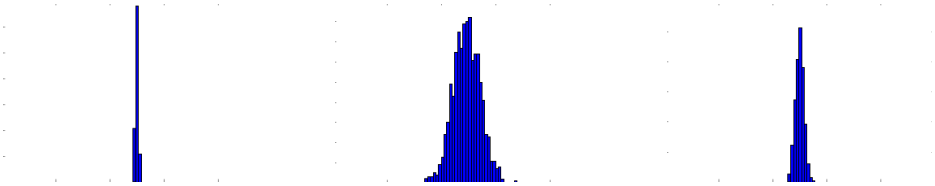
7. This Looks Familiar

(a) Excited about his upcoming concert, Justin Bieber decides to toss a large number of fair coins to help him calm his nerves. A trial consists of him flipping k such coins, and an experiment consists of 1000 such trials. He does 3 such experiments, with k being 10, 100, and 10000, respectively. (Justin Bieber can count coins faster than mere mortals.) For each experiment, he plots a histogram of the number of heads in each trial, using a horizontal axis scaled to include all the samples. **Which histogram below corresponds to which value of k?**



(a) k=100 (b) k=10000 (c) k=10

(b) Justin Bieber now carries out 3 different experiments with k being 100, 1000, and 10000, respectively. For each experiment, he plots a histogram of the fraction of heads, on a horizontal axis from 0 to 1. **Which histogram below corresponds to which value of k?**



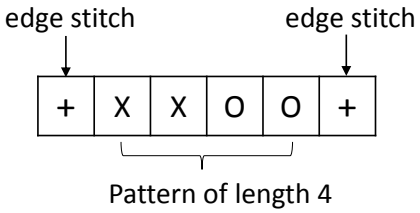
(d) k=10000 (e) k=100 (f) k=1000

PRINT your name and student ID: _____

8. Celebrate and Remember Textiles

Mathematics and computing both owe an immense debt to textiles, where many key ideas originated.

Instructions for knitting patterns will tell you to begin by "casting on" the needle some multiple of m plus r, where m is the number of stitches to create one repetition of the pattern and r is the number of stitches needed for the two edges of the piece. For example, in the simple rib stitch pattern below, the repeating pattern is of length $m = 4$, and you need $r = 2$ stitches for the edges.



Thus, to make the final piece wider, you can add as many multiples of the pattern of length 4 as you like; for example, if you want to repeat the pattern 3 times, you need to cast on a total of $3m + r = 3(4) + 2 = 14$ stitches (shown below).

+	X	X	O	O	X	X	O	O	X	X	O	O	+

You've decided to knit a 70-themed baby blanket as a gift for your cousin and want to incorporate rows from three different stitch patterns with the following requirements:

- Katie's stitch: Multiple of 7, plus 4
- Anant's lace: Multiple of 4, plus 2
- Gireeja's grid: Multiple of 5, plus 2

You want to be able to switch between knitting these different patterns without changing the number of stitches on the needle, so you must use a number of stitches that simultaneously meets the requirements of all three patterns. **Find the smallest number of stitches you need to cast on in order to incorporate all three patterns in your baby blanket.**

(HINT: There is a tool you learned in class that is perfect for this problem.)

[We know space is limited here, so feel free to use the next page as needed.]

Let x be the number of stitches we need to cast on. Using the Chinese Remainder Theorem, we can write the following system of congruences:

$$x \equiv 4 \pmod 7$$
$$x \equiv 2 \pmod 4$$
$$x \equiv 2 \pmod 5.$$

We have $M = 7 \cdot 4 \cdot 5 = 140$, $r_1 = 4$, $m_1 = 7$, $b_1 = M/m_1 = 4 \cdot 5 = 20$, $r_2 = 3$, $m_2 = 4$, $b_2 = M/m_2 = 7 \cdot 5 = 35$, and $r_3 = 2$, $m_3 = 5$, $b_3 = M/m_3 = 7 \cdot 4 = 28$. We need to solve for the multiplicative inverse of b_i modulo m_i for $i \in \{1,2,3\}$:

$$\begin{aligned} b_1 a_1 &\equiv 1 \pmod{m_1} \\ 20 a_1 &\equiv 1 \pmod{7} \\ 6 a_1 &\equiv 1 \pmod{7} \\ \rightarrow a_1 &= 6, \end{aligned}$$

$$\begin{aligned} b_2 a_2 &\equiv 1 \pmod{m_2} \\ 35 a_2 &\equiv 1 \pmod{4} \\ 3 a_2 &\equiv 1 \pmod{4} \\ \rightarrow a_2 &= 3, \end{aligned}$$

and

$$\begin{aligned} b_3 a_3 &\equiv 1 \pmod{m_3} \\ 28 a_3 &\equiv 1 \pmod{5} \\ 3 a_3 &\equiv 1 \pmod{5} \\ \rightarrow a_3 &= 2. \end{aligned}$$

Therefore,

$$\begin{aligned} x &\equiv 6 \cdot 20 \cdot 4 + 2 \cdot 35 \cdot 3 + 2 \cdot 28 \cdot 2 \pmod{140} \\ &\equiv 102 \pmod{140}, \end{aligned}$$

so the smallest x that satisfies all three congruences is 102. Therefore we should cast on 102 stitches in order to be able to knit all three patterns into the blanket.

PRINT your name and student ID: _____

[Extra page. If you want the work on this page to be graded, make sure you tell us on the problem's main page.]

PRINT your name and student ID: _____

Section 2: True/False (30 points)

For the questions in this section, determine whether the statement is true or false. If true, prove the statement is true. If false, demonstrate that it is false.

9. Prime or Composite

If positive integer m does not divide a and $a^{m-1} \not\equiv 1 \bmod m$, then m is composite.

Mark one: (TRUE) or FALSE.

Proof by contraposition: Suppose that m is 1 or m is prime, we want to show that $m|a$ or $a^{m-1} \equiv 1 \bmod m$.

When m is 1, both $1|a$ and $a^{1-1} \equiv 1 \bmod m$ hold.

When m is prime, if $m|a$, then we're done. If $m \not| a$, by Fermat's Little Theorem, we know that $a^{m-1} \equiv 1 \bmod m$.

Since the contraposition is true, the original statement must also hold.

Proof by contradiction: Recall that a statement $P \Rightarrow Q$ is equivalent to $\neg P \vee Q$. Example: "If you play loud music, I'll be angry." Equivalently, you can say "Don't play loud music, or I'll be angry." If you are not convinced, draw out the Truth Table for the two statements.

The negation of $P \Rightarrow Q$ is then logically equivalent to $\neg(\neg P \vee Q) = P \wedge \neg Q$ by De Morgan's Laws. Suppose that positive integer m does not divide a, $a^{m-1} \not\equiv 1 \bmod m$, and m is not composite (i.e. m is 1 or m is prime).

If $m \not| a$ and $m = 1$, then $a^{1-1} \equiv a^0 \equiv 1 \bmod m$. If $m \not| a$ and m is prime, by Fermat's Little Theorem, we know that $a^{m-1} \equiv 1 \bmod m$. We also have $a^{m-1} \not\equiv 1 \bmod m$, so this is a contradiction.

Hence, the supposition is incorrect, and the original statement must hold.

We decided not to take points off if your solution didn't include the case $m = 1$.

PRINT your name and student ID: _____

10. Remainder Riddles

There exists a polynomial (over $GF(7)$) $p(x)$ that has a remainder of 3 when divided by $x-1$, a remainder of 1 when divided by $x+1$, and a remainder of $2x+1$ when divided by x^2-1.

Mark one: TRUE or (FALSE).

Solution 1: Proof by contradiction. Assume that

$$p(x) = q(x)(x-1)+3,$$
$$p(x) = h(x)(x+1)+1, \text{ and}$$
$$p(x) = g(x)(x^2-1)+2x+1 = g(x)(x-1)(x+1)+2x+1.$$

for some polynomials $q(x)$, $h(x)$, and $g(x)$. Then plugging in $x=1$ to the first equation, $p(1)=3$ and plugging in $x=-1$ to the second equation, $p(-1)=1$. From the third equation, if we plug in $x=1$ we get $p(1)=3$, and if we plug in $x=-1$, we get $p(-1)=-1$. But this contradicts that $p(-1)=1$, since this would mean that $p(x)$ is not a polynomial function. Therefore the statement is false.

Solution 2: From the problem, we have

$$p(x) = q(x)(x-1)+3 \text{ and}$$
$$p(x) = h(x)(x+1)+1$$

for some polynomials $q(x)$ and $h(x)$, so we know $p(1)=3$ and $p(-1)=1$. We want to find $r(x)$ where

$$p(x) = g(x)(x^2-1)+r(x) = g(x)(x-1)(x+1)+r(x)$$

for some polynomial $g(x)$. We know that the degree of $r(x)$ is necessarily less than the degree of x^2-1, so the degree of $r(x)$ is at most 1. Therefore we can write $r(x)=ax+b$ and solve for a and b. We can write the equations

$$p(1) = r(1) = a+b = 3 \quad \text{and} \quad p(-1) = r(-1) = b-a = 1.$$

We can solve this by a simple substitution. From the second equation we can write $b=1+a$, and substituting into the first, $2a+1=3$, so $a=1$. Then substituting back into the second equation, $b-1=1$, so $b=2$. Therefore the remainder must be $r(x)=x+2 \neq 2x+1$. Therefore the statement is false.

Common mistakes: Many students answered that the statement was false because x^2-1 is not coprime to $x+1$ and $x-1$, and therefore the Chinese Remainder Theorem can not be used. It is true that the Chinese Remainder Theorem can not be used on the three equations as stated in the problem, but as shown in Solution 2 above, this does not prevent us from determining the remainder of $p(x)$ divided by x^2-1. Another common mistake was stating that the statement was false based on some reasoning that the remainder of $p(x)$ divided by x^2-1 could not be of degree 1. Again, as shown in Solution 2 above, the remainder is indeed of degree 1, so this is not a valid line of reasoning.

PRINT your name and student ID: _____

[Extra page. If you want the work on this page to be graded, make sure you tell us on the problem's main page.]

PRINT your name and student ID: _____

Section 3: Free-form Problems (65 points)

11. Secret Sharing with Spies (20 points)

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.

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 $4 + 2 * 3 = 10$ 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.

PRINT your name and student ID: _____

[Extra page. If you want the work on this page to be graded, make sure you tell us on the problem's main page.]

PRINT your name and student ID: _____

12. T/F: A Different Code (25 points)

An n-character message $\vec{a} = (a_0, \ldots, a_{n-1})$, with $a_i \in \text{GF}(p)$, is encoded into a polynomial of degree-d as follows. $P_{\vec{a}}(x) = a_0 x^d + a_1 x^{d-1} + \cdots + a_{n-1} x^{d-n+1}$. A codeword of length $n+2k$ is generated by evaluating $P_{\vec{a}}(x)$ as follows. $\vec{c}(\vec{a}) = (P_{\vec{a}}(1), P_{\vec{a}}(2), \ldots, P_{\vec{a}}(n+2k))$. If $p > d > n+2k$ and $k > 0$, then this code has a minimum Hamming distance of $2k+1$.

(Recall that the Hamming distance between two codewords is the number of positions they are different. The minimum distance of a code is the minimum distance between two distinct codewords.)

Mark one: TRUE or FALSE.

If true, prove the statement is true. If false, demonstrate that it is false.

The statement is $\boxed{TRUE}$.

We will reduce the problem into coefficient encoding for Reed-Solomon code. We observe that the difference between this problem and the RS code is that everything is multiplied by x^{d-n+1}. That is,

$$\begin{aligned} P_{\vec{a}}(x) &= a_0 x^d + a_1 x^{d-1} + \cdots + a_{n-1} x^{d-n+1} \\ &= (a_0 x^{n-1} + a_1 x^{n-2} + \cdots + a_{n-1}) x^{d-n+1}. \end{aligned}$$

We are working over a finite field GF(p), and the encoding does not use $x = 0$, so a unique multiplicative inverse must exist for $x = 1, 2, \ldots, n+2k$. Thus, at the received end, we can multiply each received position by the appropriate number to the order of $(-d+n-1)$. Therefore the codeword:

$$\vec{c'}(\vec{a}) = [P_{\vec{a}}(1) * 1^{(-d+n-1)}, P_{\vec{a}} * (2) 2^{(-d+n-1)}, \ldots, P_{\vec{a}}(n+2k) * (n+2k)^{(-d+n-1)}]$$

is the codeword of $(a_{n-1}, \ldots, a_0)$ encoded using coefficient-encoding for RS code. Therefore, two codewords of the proposed encoding will differ in a position if and only if their corresponding RS codewords differ in exactly that position. Or in other words, if u, v are two codewords at position x, $u = v$ if and only if $x^k u = x^k v$. Thus, the Hamming distance is preserved when transforming the RS code to the proposed code.

We have proved that interpolation-encoding has distance $2k+1$ in the lecture note. The set of polynomials obtained by coefficient encoding and the set of polynomials obtained by interpolation-encoding is the same set of polynomials — namely *all* polynomials of degree $n-1$. Since the set of codewords depend only on the set of polynomials, these two sets are identical. So, their distance properties are identical. Therefore, the minimum distance of coefficient-encoding is $2k+1$, so d_{min} in this problem is also $2k+1$.

PRINT your name and student ID: _____

[Extra page. If you want the work on this page to be graded, make sure you tell us on the problem's main page.]

PRINT your name and student ID: _____

13. Like RSA (20 points)

You want to come up with an RSA-like scheme with $N = pqrs$, where p, q, r, s are distinct primes and the public encrypting function is $x^e \bmod N$.

(a) (5 points) Specify how to choose e.

We need to choose an e which is relatively prime to $(p-1)(q-1)(r-1)(s-1)$. In addition, e cannot be 1, although this did not need to be specified. An explanation why is not necessary, but the reason is that it requires an a multiplicative inverse in that mod.

(b) (15 points) Use the Chinese Remainder Theorem to come up with a decryption procedure that works slightly faster than the usual RSA decryption. Show that your scheme works (*i.e.* you can recover encrypted messages by decrypting them.).

You can assume here that exponentiating smaller numbers to smaller powers mod smaller numbers is significantly faster. So $(ab)^{cd} \bmod ef$ is slower than calculating both $a^c \bmod e$ and $b^d \bmod f$.

(HINT: Your secret key should include p, q, r, s individually.)

A key sentence in this problem was this: "exponentiating **smaller** numbers to **smaller** powers mod **smaller** numbers is significantly faster", which did not say that when only two of the three numbers are smaller, exponentiation would still be faster. As such, full points was only granted to solutions that decreased all three numbers.

First, similarly to the regular RSA algorithm, a decryption key must be chosen. However, instead of calculating $d = e^{-1} \pmod{(p-1)(q-1)(r-1)(s-1)}$, we calculate separate decryption keys:

$$\begin{aligned} d_p &\equiv e^{-1} \pmod{p-1} \\ d_q &\equiv e^{-1} \pmod{q-1} \\ d_r &\equiv e^{-1} \pmod{r-1} \\ d_s &\equiv e^{-1} \pmod{s-1} \end{aligned}$$

For our private key, we thus store $(d_p, d_q, d_r, d_s, p, q, r, s)$. Once a message y is received, we first reduce

it:

$$\begin{aligned} y_p &:= y \pmod{p} \\ y_q &:= y \pmod{q} \\ y_r &:= y \pmod{r} \\ y_s &:= y \pmod{s}. \end{aligned}$$

Now, to decrypt, we calculate separately using the repeated-squaring mod-exp algorithm (Note 5):

$$\begin{aligned} x_p &\equiv y_p^{d_p} \pmod{p} \\ x_q &\equiv y_q^{d_q} \pmod{q} \\ x_r &\equiv y_r^{d_r} \pmod{r} \\ x_s &\equiv y_s^{d_s} \pmod{s}. \end{aligned}$$

Note that this exponentiation step involves all of smaller bases, smaller exponents, and smaller mods, than calculating $y^{e^{-1} \bmod (p-1)(q-1)(r-1)(s-1)} \pmod{pqrs}$. Finally, we use CRT to combine and get the original decrypted message:

$$\begin{aligned} x &\equiv x_p \pmod{p} \\ x &\equiv x_q \pmod{q} \\ x &\equiv x_r \pmod{r} \\ x &\equiv x_s \pmod{s}. \end{aligned}$$

Now, we must justify why x is the original message sent. First, we will examine why $x_p = x \pmod{p}$. There are two possible cases: either $x = 0 \pmod{p}$, or $x \neq 0 \pmod{p}$. In the first case, we have

$$x_p = y^{d_p} \equiv (x^e)^{d_p} \equiv (0^e)^{d_p} = 0 = x \pmod{p}$$

In the second case, we know that $x^{p-1} \equiv 1 \pmod{p}$ by FLT, and that $e * d_p = 1 \pmod{p-1}$ so $e * d_p = k_p * (p-1) + 1$ for some integer k, so

$$x_p = y^{d_p} \equiv (x^e)^{d_p} = x^{e*d_p} \equiv x^{k_p*(p-1)+1} = x * (x^{p-1})^k \equiv x * 1^k = x \pmod{p}$$

Thus, we know that in both cases, $x_p \equiv x \pmod{p}$. From symmetry, we know that the other three equations also hold, and, since p, q, r, and s are coprime, we can thus CRT to calculate $x \pmod{pqrs}$.

PRINT your name and student ID: _____

[Extra page. If you want the work on this page to be graded, make sure you tell us on the problem's main page.]

PRINT your name and student ID: _____

14. (optional) Oblivious Updates: File Sync (30 points)

Alice and Bob start with identical versions of a file. Assume the original file is n symbols from $GF(p)$, denoted $\vec{m}$. Assume that $2n < p$. Throughout this problem, Alice can send messages (noiselessly) to Bob, but Bob cannot send anything to Alice.

(a) (5 points) Suppose that one of the symbols in Bob's file gets erased. Bob knows which position was erased, but not what it was before it was erased. Alice only knows that one symbol was erased but not which one.

Alice would like to send Bob a short message, such that he can recover the original file $\vec{m}$. But noiseless communication is expensive – can you devise a scheme for Alice to do this, better than re-sending her n-long entire file $\vec{m}$? Argue why this works.

Let $m_i \in GF(p)$ be the i^{th} symbol of Alice's file. Say the j^{th} symbol in Bob's file was erased.

Method 1:

Alice sends the single symbol $s = \sum_i m_i$ to Bob. From his intact symbols, Bob can compute $s' = \sum_{i \neq j} m_i$, and can subtract to recover $m_j = s - s'$.

Method 2:

Let $S = \{0, 1, \ldots, (n-1)\}$ (the "evaluation set"). Alice constructs a $(n-1)$-degree polynomial $P(x)$, such that $P(i) = m_i$ for all $i \in S$. She then sends the single symbol $P(n)$ to Bob. Using his intact symbols, Bob has access to the $n-1$ points $P(i) = m_i$ for $i \in S, i \neq j$. Combining this with the received symbol $P(n)$, Bob has access to n evaluations of $P(x)$, which he can interpolate to find the full polynomial $P(x)$.

He then evaluates $P(j)$ to recover the symbol.

Remarks: In general, both solution methods took advantage of the "shared information" between Alice and Bob to reduce the communication required. Notice that for Method 2, it was crucial that the polynomial was constructed by **interpolation encoding**, not coefficient encoding (coefficient encoding can be made to work, but would require a modification of Berlekamp-Welch). This allowed Bob to use his intact data to reconstruct the polynomial.

Full credit was given for correct schemes that were communication-optimal (1-symbol). Many students used some version of Method 1. (We gave credit for sending the "XOR" of all symbols, even though strictly speaking we are not working in a power-of-two field, so converting to binary representation may be slightly suboptimal).

Varying amounts of partial credit was awarded for reasonable but suboptimal (or only partially correct) solutions. For example, sending the product of all symbols instead of the sum (in Method 1) would fail if any symbol is zero (ie, almost always).

Surprisingly many students constructed a polynomial from Alice's data, then tried to "send the polynomial" to Bob – claiming that this requires less communication than sending the underlying data. This is incorrect, and received no credit.

(b) (15 points) Assume that Bob has a correct version of $\vec{m}$. Alice makes a small change to her copy of the file, modifying one of the n symbols. Let this updated file be $\vec{m}'$. Bob later wants to update his copy to match Alice's copy – but Alice has forgotten which symbol she changed!

Once again, Alice would like to send Bob a short message, such that he can recover the updated file $\vec{m}'$. But noiseless communication is expensive – can you devise a scheme for Alice to do this, better than re-sending her n-long entire file $\vec{m}'$? Argue why your scheme works.

Method 1:

(Using the notation of the previous part.) Let Alice construct $P(x)$ as in the previous part (Method 2), such that $P(i) = m'_i$ for $i \in \{0, 1, \ldots (n-1)\}$. She then sends the two symbols $P(n), P(n+1)$ to Bob.

Since $m'_i = m_i$ for all $i \neq j$, Bob can use his symbols (m_i) to find n evaluations of P: $P(i) = m'_i$, with at most one error (at $i = j$). Together with the two received symbols, Bob has access to $n+2$ evaluations of the $(n-1)$-degree polynomial $P(x)$, with at most one error.

This is exactly equivalent to an interpolation-encoded Reed-Solomon code, with at most $k = 1$ general error. Thus Bob can use a Berlekamp-Welch decoder to find $P(x)$, and correct his file.

Method 2:

Alice sends the two symbols $s'_1 = \sum_i m'_i$ and $s'_2 = \sum_i i m'_i$ (that is, treating the index $i \in \{0, 1, \ldots, (n-1)\}$ as a symbol in $GF(p)$). Bob similarly computes $s_1 = \sum_i m_i$ and $s_2 = \sum_i i m_i$ from his file, and subtracts to find $\Delta = s'_1 - s_1 = m'_j - m_j$ and $K = j m'_j - j m_j$. Then, if $\Delta = 0$ he knows there is no change in the file. Otherwise, he computes $K\Delta^{-1}$ to find j, the changed position. He can then add Δ to his symbol m_j, to correct his file.

Remarks: The above solutions are actually similar, but Method 1 is simpler to generalize. Again, we exploited the shared information to reduce communication. In fact, it can be shown that no scheme for this problem can achieve better than 2 symbols of communication.

Full credit was awarded only for entirely correct schemes, with proper justification. Notice that Method 1 relies crucially on using the interpolation encoding, so Bob can use his file as "faulty evaluations" of Alice's polynomial. Simply saying "Alice constructs a polynomial from her file" (or similar) was insufficient, since the scheme intimately depends on the details of the construction. Varying amounts of partial credit was awarded for entirely correct but suboptimal solutions. No credit was awarded for incorrect solutions.

Many students also required Alice to send the x-coordinates of her evaluations (ie, packets of the form $(n, P(n))$). This is not required, as they can agree on the evaluation set beforehand (as is normally done when using Reed-Solomon codes).

(c) (10 points) What if Alice changes $z > 1$ symbols? How would you deal with this case? (The size of the message sent to Bob is allowed to depend on z.) Argue why this works.

Use the natural generalization of Method 1 from the previous part. Alice constructs the polynomial $P(x)$ as before, and sends the $2z$ symbols $P(n), P(n+1), \ldots, P(n+2z-1)$. Using his file and the received symbols, Bob has access to $n+2z$ evaluations of $P(x)$, with at most z errors. Thus he can use Berlekamp-Welch as before, to correct his file.

There is a minor issue here... if $n < 2z$, then Alice is better off just sending her entire file!

Remarks: Most students who correctly argued Method 1 on the previous part got this part. We awarded partial credit for promising extensions of part (b), even if part (b) was not presented/argued properly.

Only a few students caught the "minor issue", but we only subtracted a minor amount of points for the oversight.

PRINT your name and student ID: _____

[Extra page. If you want the work on this page to be graded, make sure you tell us on the problem's main page.]

PRINT your name and student ID: _____

[Doodle page! Draw us something if you want or give us suggestions or complaints. You can also use this page to report anything suspicious that you might have noticed.]