CS 70 Discrete Mathematics and Probability Theory
Summer 2025 Tate

HW 02

1 Celebrate and Remember Textiles

Note 6

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 on the row lengths of each of the stitch patterns:

- Alternating Link: Multiple of 8, plus 3
- Double Helix: Multiple of 3, plus 1
- Crossover: Multiple of 7, plus 6

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.

Solution: 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:

$$\begin{aligned} x &\equiv 3 \pmod{8} \\ x &\equiv 1 \pmod{3} \\ x &\equiv 6 \pmod{7}. \end{aligned}$$

We have $M = 8 \cdot 3 \cdot 7 = 168$, $r_1 = 3$, $m_1 = 8$, $b_1 = M/m_1 = 3 \cdot 7 = 21$, $r_2 = 1$, $m_2 = 3$, $b_2 = M/m_2 = 8 \cdot 7 = 56$, and $r_3 = 6$, $m_3 = 7$, $b_3 = M/m_3 = 8 \cdot 3 = 24$. We need to solve for the multiplicative inverse of b_i modulo m_i for $i \in \{1,2,3\}$:

$$\begin{aligned} b_1 a_1 &\equiv 1 \pmod{m_1} \\ 21 a_1 &\equiv 1 \pmod{8} \\ 5 a_1 &\equiv 1 \pmod{8} \\ \rightarrow a_1 &= 5, \end{aligned}$$

$$\begin{aligned} b_2 a_2 &\equiv 1 \pmod{m_2} \\ 56 a_2 &\equiv 1 \pmod{3} \\ 2 a_2 &\equiv 1 \pmod{3} \\ \rightarrow a_2 &= 2, \end{aligned}$$

and

$$\begin{aligned} b_3a_3 &\equiv 1 \pmod{m_3} \\ 24a_3 &\equiv 1 \pmod{7} \\ 3a_3 &\equiv 1 \pmod{7} \\ \rightarrow a_3 &= 5. \end{aligned}$$

Therefore,

$$\begin{aligned} x &\equiv 3\cdot 21\cdot 5 + 1\cdot 56\cdot 2 + 6\cdot 24\cdot 5 \pmod{168} \\ &\equiv 1147 \equiv 139 \pmod{168}, \end{aligned}$$

so the smallest x that satisfies all three congruences is 139. Therefore we should cast on 139 stitches in order to be able to knit all three patterns into the blanket.

2 RSA with CRT

Note 7 Inspired by the efficiency of solving systems of modular equations with CRT, Alice decides to use CRT to speed up RSA!

She first generates the public key (e,N) and private key d as normal, keeping track of the primes $pq = N$. Recall that e is chosen to be coprime to $(p-1)(q-1)$, and d is then defined as $e^{-1} \pmod{(p-1)(q-1)}$. Next, she stores the following values:

$$\begin{aligned} d_p &\equiv d \pmod{p-1} \\ d_q &\equiv d \pmod{q-1} \end{aligned}$$

After receiving an encrypted message $c = m^e \pmod{N}$ from Bob, Alice computes the following expressions:

$$\begin{aligned} x &\equiv c^{d_p} \pmod{p} \\ x &\equiv c^{d_q} \pmod{q} \end{aligned}$$

The message m then calculated as the solution to the above modular system.

(a) Show that this algorithm is correct, i.e. that $x \equiv m$ is the only solution $\pmod{N}$ to the above modular system.

(b) Emboldened by her success in using CRT for RSA, Alice decides to invent a new cryptosystem. To generate her keypair, she first generates $N = pq$. Then, she chooses three numbers g, r_1, r_2 and publishes the public key $(N, g_1 = g^{r_1(p-1)} \pmod{N}, g_2 = g^{r_2(q-1)} \pmod{N})$. Her private key is (p, q).

To encrypt a message, Bob chooses two numbers s_1, s_2 and sends $c_1 = mg_1^{s_1}, c_2 = mg_2^{s_2}$.

Alice decrypts this message by solving the modular system

$$\begin{aligned} x &\equiv c_1 \pmod{p} \\ x &\equiv c_2 \pmod{q} \end{aligned}$$

Show that this algorithm is correct, i.e. show that $x \equiv m$ is the only solution $\pmod{N}$ to the above modular system.

(c) This system is woefully insecure. Show how anyone with access to the public key can recover p, q, given that $g_1 \not\equiv 1 \pmod{q}$.

Solution:

(a) Intuitively, note that $x = c^{d_p} \equiv m \pmod{p}$, and $x = c^{d_p} \equiv m \pmod{q}$. Therefore, the solution to the modular system must satisfy both constraints, which leaves m as the only solution.

(b) Similarly to the previous question, we have

$$\begin{aligned} x &\equiv mg_1^{s_1} \pmod{p} \\ x &\equiv mg_2^{s_2} \pmod{q} \end{aligned}$$

Key to this subpart is the fact that $g_1^{s_1} = g^{s_1 r_1(p-1)} \equiv 1 \pmod{p}$, and $g_2^{s_2} = g^{s_2 r_2(q-1)} \equiv 1 \pmod{q}$. Therefore, this system reduces to

$$\begin{aligned} x &\equiv m \pmod{p} \\ x &\equiv m \pmod{q} \end{aligned}$$

By the previous subpart, we know that $x \equiv m \pmod{N}$.

(c) We are given a value $g_1 = g^{r_1(p-1)} \pmod{p}$ (as part of the public key) that is $1 \pmod{p}$ (by FLT) but not $1 \pmod{q}$. It follows that $g_1 - 1$ is a multiple of p, and we can find $\gcd(g_1 - 1, N) = p$. From there, we can find $q = \frac{N}{p}$. Note that if $g_1 \equiv 1 \pmod{q}$, this won't work, since then $g_1 - 1$ is a multiple of N and $\gcd(g_1 - 1, N) = N$. However, then $c_1 = m$ for all encryptions, making it insecure regardless.

3 RSA with Just One Prime

Given the message $x \in \{0, 1, \ldots, N-1\}$ and $N = pq$, where p and q are prime numbers, conventional RSA encrypts x with $y = E(x) \equiv x^e \pmod{N}$. The decryption is done by $D(y) \equiv y^d \pmod{N}$, where d is the inverse of $e \pmod{(p-1)(q-1)}$.

Alice is trying to send a message to Bob, and as usual, Eve is trying to decipher what the message is. One day, Bob gets lazy and tells Alice that he will now use $N = p$, where p is a 1024-bit prime number, as part of his public key. He tells Alice that it's okay, since Eve will have to try out 2^{1024} combinations to guess x. It is very likely that Eve will not find out the secret message in a reasonable amount of time! In this problem, we will see whether Bob is right or wrong. Assume that Eve has found out about this new setup and that she knows the public key.

Similar to the original method, for any message $x \in \{0, 1, \ldots, N-1\}$, $E(x) \equiv x^e \pmod{p}$, and $D(y) \equiv y^d \pmod{p}$. Choose e such that it is coprime with $p-1$, and choose $d \equiv e^{-1} \pmod{p-1}$.

(a) Prove that the message x is recovered after it goes through your new encryption and decryption functions, $E(x)$ and $D(y)$.

(b) Can Eve compute d in the decryption function? If so, by what algorithm and approximately how many iterations does it take for it to terminate?

(c) Given part (b), how would Eve recover x and what algorithm would she use? Approximately how many iterations does it take to terminate?

(d) Based on the previous parts, can Eve recover the original message in a reasonable amount of time? Explain.

Solution:

(a) We want to show x is recovered by $E(x)$ and $D(y)$, such that $D(E(x)) = x$. In other words, $x^{ed} \equiv x \pmod{p}\ \forall x \in \{0, 1, \ldots, N-1\}$.
Proof: By construction of d, we know that $ed \equiv 1 \pmod{p-1}$. This means we can write $ed = k(p-1)+1$, for some integer k, and $x^{ed} = x^{k(p-1)+1}$.

- x is a multiple of p: Then this means $x = 0$, and indeed, $x^{ed} \equiv 0 \pmod{p}$.
- x is not a multiple of p: Then $x^{ed} \equiv x^{k(p-1)+1} \equiv x^{k(p-1)}x \equiv 1^k x \equiv x \pmod{p}$, by using FLT.

And for both cases, we have shown that x is recovered by $E(D(y))$.

(b) Since Eve knows the value of $N = p$, and the fact that $d \equiv e^{-1} \pmod{p-1}$, she can compute d using EGCD. Since EGCD decreases the largest number by at least a factor of two every two iterations, Eve needs at most $2n$ iterations, where n is the number of bits of the larger input. This means at most 2048 iterations.

(c) Since Eve now has d from part 3, and the encrypted message y, she can calculate x directly by using $D(y) = x \equiv y^d \pmod{p}$. She can now use exponentiation by repeated squaring, giving her no more than 1024 iterations.

(d) Assuming each recursive call in EGCD and exponentiation by squaring have reasonable operation time costs, Eve only needs at most 3×1024 iterations, which can easily be done with today's computing power.

4 Equivalent Polynomials

Note 7
Note 8

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 13 that is equivalent to $f(x) = x^{13}$ over GF(13); then find a polynomial with degree strictly less than 7 that is equivalent to $g(x) = 2x^{74} + 6x^7 + 3$ over GF(7).

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

Solution:

(a) For f and g to be equivalent, they must satisfy $f(x) \equiv g(x) \pmod{p}$ for all values of x, including zero. But $f(0) \equiv 0 \pmod{p}$ and $g(0) \equiv 1 \pmod{p}$, so they are not equivalent.

(b) Fermat's Little Theorem says that for any nonzero integer a and any prime number p, $a^{p-1} \equiv 1 \bmod p$. We're allowed to multiply through by a, so the theorem is equivalent to saying that $a^p \equiv a \bmod p$; note that this is true even when $a = 0$, since in that case we just have $0^p \equiv 0 \pmod{p}$.

The problem asks for a polynomial $\tilde{f}(x)$, different from $f(x)$, with the property that $\tilde{f}(a) \equiv a^{13} \bmod 13$ for any integer a. Directly using the theorem, $\tilde{f}(x) = x$ will work. We can do something similar with $g(x) = 2x^{74} + 6x^7 + 3$ modulo 7; since $x^7 \equiv x \pmod{7}$, we repeatedly substitute x^7 with x, effectively reducing the exponent by 6. We can only do this as long as the exponent remains greater than or equal to 7, so we end up with $\tilde{g}(x) = 2x^2 + 6x + 3$.

(c) One proof uses Fermat's Little Theorem. As a warm-up, let $d \geq p$; we'll find a polynomial equivalent to x^d. For any integer, we know

$$\begin{aligned} a^d &= a^{d-p}a^p \\ &\equiv a^{d-p}a \pmod{p} \\ &\equiv a^{d-p+1} \pmod{p}. \end{aligned}$$

In other words x^d is equivalent to the polynomial $x^{d-(p-1)}$. If $d-(p-1) \geq q$, we can show in the same way that x^d is equivalent to $x^{d-2(p-1)}$. Since we subtract $p-1$ every time, the sequence $d, d-(p-1), d-2(p-1), \ldots$ must eventually be smaller than p. Now if $f(x)$ is any polynomial with degree $\geq p$, we can apply this same trick to every x^k that appears for which $k \geq p$.

Another proof uses Lagrange interpolation. Let $f(x)$ have degree $\geq p$. By Lagrange interpolation, there is a unique polynomial $\tilde{f}(x)$ of degree at most $p-1$ passing through the points $(0, f(0)), (1, f(1)), (2, f(2)), \ldots, (p-1, f(p-1))$, and we know it must be equivalent to $f(x)$ because f also passes through the same p points.

5 Lagrange? More like Lamegrange.

Note 8

In this problem, we walk you through an alternative to Lagrange interpolation.

(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?

Solution:

(a) We want a degree zero polynomial, which is just a constant function. The only constant function that passes through (x_0, y_0) is $f_0(x) = y_0$.

(b) By defining $f_1(x) = f_0(x) + a_1(x - x_0)$, we get that

$$f_1(x_0) = f_0(x_0) + a_1(x_0 - x_0) = y_0 + 0 = y_0.$$

So now we just need to make sure that $f_1(x_1) = y_1$. This means that we need to choose a_1 such that

$$f_1(x_1) = f_0(x_1) + a_1(x_1 - x_0) = y_1.$$

Solving this for a_1, we get that

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

(c) We apply similar logic to the previous part. From our definition, we know that

$$f_2(x_0) = f_1(x_0) + a_2(x_0 - x_0)(x_0 - x_1) = y_0 + 0 = y_0.$$

and that

$$f_2(x_1) = f_1(x_1) + a_2(x_1 - x_0)(x_1 - x_1) = y_1 + 0 = y_1.$$

Thus, we just need to choose a_2 such that $f_2(x_2) = y_2$. Putting in our formula for $f_2(x)$, we get that we need a_2 such that

$$f_1(x_2) + a_2(x_2 - x_0)(x_2 - x_1) = y_2.$$

Solving for a_2, we get that

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

(d) If we try to calculate $f_{i+1}(x_k)$ for $0 \leq k \leq i$, we know one of the $(x - x_j)$ terms (specifically the kth one) will be zero. Thus, we get that

$$f_{i+1}(x_k) = f_i(x_k) + a_{i+1}(0) = y_k + 0 = y_k.$$

So now we just need to pick a_i such that $f_{i+1}(x_{i+1}) = y_{i+1}$. This means that we need to choose a_{i+1} such that

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

Solving for a_{i+1}, we get that

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

The method you derived in this question is known as Newtonian interpolation. (The formal definition of Newtonian interpolation uses divided differences, which we don't cover in this class, but it's in effect doing the same thing.) This method has an advantage over Lagrange interpolation in that it is very easy to add in extra points that your polynomial has to go through (as we showed in part (c), whereas Lagrange interpolation would require you to throw out all your previous work and restart. However, if you want to keep the same x values but change the y values, Newtonian interpolation requires you to throw out all your previous work and restart. In contrast, this is fairly easy to do with Lagrange interpolation–since changing the y values doesn't affect the δ_is, you don't have to recalculate those, so you can skip most of the work.

6 Trust No One

Gandalf has assembled a fellowship of eight peoples to transport the One Ring to the fires of Mount Doom: four hobbits, two humans, one elf, and one dwarf. The ring has great power that may be of use to the fellowship during their long and dangerous journey. Unfortunately, the use of its immense power will eventually corrupt the user, so it must not be used except in the most dire of circumstances. To safeguard against this possibility, Gandalf wishes to keep the instructions a secret from members of the fellowship. The secret must only be revealed if enough members of the fellowship are present and agree to use it.

Requiring all eight members to agree is certainly a sufficient condition to know the instructions, but it seems excessive. However, we also know that the separate peoples (hobbits, humans, elf, and dwarf) do not completely trust each other so instead we decide to require members from at least two different peoples in order to use the ring. In particular, we will require a unanimous decision by all members of one group in addition to at least one member from a different group. That is, if only the four hobbits want to use the ring, then they alone should not have sufficient information to figure out the instructions. Same goes for the two humans, the elf, and the dwarf.

More explicitly, only four hobbits agreeing to use the ring is not enough to know the instructions. Only two humans agreeing is not enough. Only the elf agreeing is not enough. Only the dwarf agreeing is not enough. However, all four hobbits and a man agreeing is enough. Both humans and a dwarf agreeing is enough. Both the elf and the dwarf agreeing is enough.

Gandalf has hired your services to help him come up with a secret sharing scheme that accomplishes this task, summarized by the following points:

- There is a party of four hobbits, two humans, an elf, and a dwarf.
- There is a secret message that needs to be known if enough members of the party agree.
- The message must remain unknown to everyone if not enough members of the party agree.
- If only the members of one people agree, the message remains a secret.
- If all the members of one people agree plus at least one additional person, the message can be determined.

Solution:

Solution 1

There will be two parts to this secret: a unanimity secret U and a multi-people secret M. U ensures that at least all members of one peoples are in agreement while M ensures that members of at least two peoples are in agreement. We will discuss these two in order below. Once both U and M are recovered, they can then be combined to reveal the original secret: each will be a point of the degree-1 polynomial $R(x)$ whose y-intercept contains the secret of the ring.

The *unanimity secret* involves creating a separate secret for each people. We will require all members of that people to join forces in order to reveal the secret. For example, the hobbits will each have distinct points of a degree-3 polynomial and the humans will each have distinct points of a degree-1 polynomial. When all members of a people come together, they will reveal U (encoded, for example, as the y-intercept of each of these polynomials). Note that the elf and the dwarf each know U already since they are the only members of their people.

The *multi-people secret* involves creating a degree-1 polynomial $P_m(x)$ and giving one point to all members of each people. For example, the hobbits may each get $P_m(1)$ while the elf gets $P_m(2)$ and the humans each get $P_m(3)$. In this way if members of any two peoples are in agreement, they can reveal M (encoded, for example, as the y-intercept of $P_m(x)$).

Once U and M are each known, they can be *combined* to determine the final secret. U and M allow us to uniquely determine $R(x)$ and thus $R(0)$, the secret of the ring.

This scheme is an example of hierarchical secret sharing. Let's work out a specific example.

Example: Suppose the secret is $s = 4$, $M = 3$, and $U = 2$. From now on, we can work in GF(5) since $s < 5$ and $n < 5$ (n is the number of people who have pieces of the secret).

First we need to create a degree-1 polynomial $R(x)$ such that $R(0) = s = 4$, $R(1) = M = 3$, and $R(2) = U = 2$. By inspection, $R(x) = 4x + 4$ has these properties (e.g. $R(1) = 4 \cdot 1 + 4 = 8 \equiv 3$).

Now we can create the multi-people secret M. We choose degree-1 polynomial $P_m(x) = x + 3$ and tell each hobbit $P_m(1) = 4$, the elf $P_m(2) = 5 \equiv 0$, each of the humans $P_m(3) = 6 \equiv 1$, and the dwarf $P_m(4) = 7 \equiv 2$. Now any two members of distinct peoples can determine $P_m(x)$ and thus $P_m(0)$ by interpolating their two values.

When creating the unanimity secret U, we first note that each of the dwarf and the elf will be told U directly since they are the only members of their respective people. On the other hand, the humans will each have a point on the degree-1 polynomial $P_{humans}(x)$. Suppose $P_{humans}(x) = 2x+2$. Then the first human receives $P_{humans}(1) = 4$ and the second receives $P_{humans}(2) = 4+2 = 6 \equiv 1$. When they interpolate using these values, they will discover the original polynomial and therefore $P_{humans}(0) = U = 2$. The hobbits will have a similar secret but with a degree-3 polynomial (e.g. $P_{hobbit}(x) = 4x^3 + x^2 + 2$).

Now suppose that two humans and one hobbit come together. The two humans work together to determine U as described above. Together the three of them also know $P_m(3) = 6$ and $P_m(1) = 4$, from which they can find $P_m(x)$ and thus $P_m(0) = M = 3$. Now that they have U and M, they can interpolate to find $R(x)$ and thus $R(0) = s = 4$.

Solution 2

Alternatively, we can construct a single degree 6 polynomial and distribute 1 point to each hobbit, 3 points to each human, 6 points to the elf, and 6 points to the dwarf. We can see that if all the hobbits agree, they will need 3 more points in order to interpolate successfully and each member of all the other peoples are given at least 3 points. Moreover, each of the other peoples have 6 points in total, meaning that if all the humans, the elf, or the dwarf agree, they'll only need one more point which can be provided by any additional member of the party outside their people. On the other hand, the most amount of points that could be obtained from an agreeing group that does not satisfy the requirements would be 6, from the group consisting of one human and all the hobbits. This would be insufficient to interpolate the polynomial so therefore, the scheme fulfills the requirements.

7 Error-Correcting Codes

Note 9

(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.

Solution:

(a) Suppose we send a total of m packets (where m is to be determined). Since at most a fraction α 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 αm, 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 $\alpha \geq 1/2$.