EECS 70 Discrete Mathematics and Probability Theory
Fall 2014 Anant Sahai

Homework 7

This homework is due October 20, 2014, at 12:00 noon.

1. Section rollcall!

In your self-grading for this question, give yourself a 10, and write down what you wrote for parts (a) and (b) below as a comment. You can optionally put the answers in your written homework as well.

(a) What discussion did you attend on Monday last week? If you did not attend section on that day, please tell us why.

(b) What discussion did you attend on Wednesday last week? If you did not attend section on that day, please tell us why.

2. Polynomials Lab

In this week's Virtual Lab, we will dive deeper into Polynomials and their applications. We will start by generalizing our implementation of degree 2 polynomial's Lagrange Interpolation for any degree d polynomial in a finite field. We will then implement two simple Secret Sharing and Erasure Error clients using Lagrange Interpolation and Polynomial properties studied in lecture.

Finally, to wrap up our brief Matplotlib tutorial series, we will learn how to plot histograms (in addition to curves and bar charts from previous labs).

You will find in the skeleton a sample implementation of a `Polynomial` class. You don't need to understand the details of the implementation, but please read the docstring of each function carefully.

Starting from this Virtual Lab, we will ask you to write down something for every part in your written homework. This is done to reduce self-grading confusion, where students often are not sure which parts require written answers and which do not. For questions that involve plotting something, you will need to embed your plot(s) in your written homework. You can save your plots using `plt.savefig()` or Right Click > Save Image As. If you normally handwrite your homework, we recommend that you type up the lab report (similar in style to an EE/Physical Science one) to make inserting plots easier.

Please download the IPython starter code from Piazza or the course webpage, and answer the following questions.

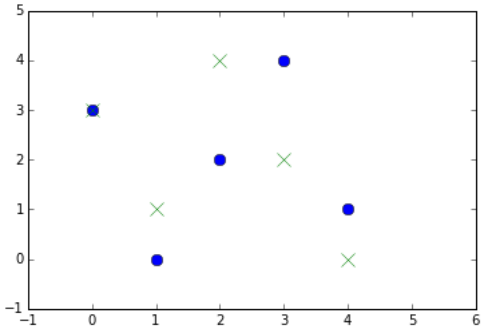
(a) *Warm up*: Run the code cells in the skeleton to have a better idea of how our implementation of `Polynomial` works. For example, to create a polynomial $f(x) = 2x + 1 \pmod 7$, we write: `f = Polynomial([1, 2], 7)`. Notice that the coefficients go from lower power of x to higher power of x.

Make sure you understand the basics of the `Polynomial` class and play around with the given examples in the skeleton before moving on. For this part, please write down the result of $(4x^3 + 3x^2 + 2x + 1)/7 \bmod 5$.

Solutions:

$$\begin{aligned} & (4x^3 + 3x^2 + 2x + 1)/7 \bmod 5 \\ & 3(4x^3 + 3x^2 + 2x + 1) \bmod 5 \\ & \equiv 2x^3 + 4x^2 + x + 3 \bmod 5. \end{aligned} \tag{1}$$

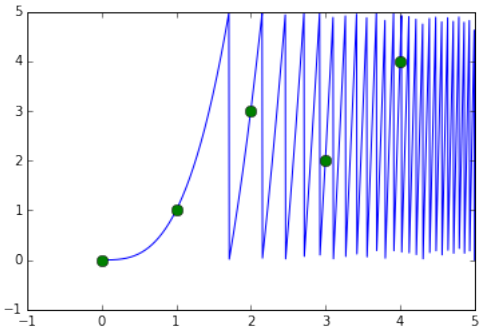
(b) Create and plot the polynomials $p(x) = 2x+3$ and $q(x) = 3x-2$ with all numbers reduced mod 5. Make sure you use the correct symbol ('o' or 'x') to represent the points so that your result looks similar to the figure on page 5 of Note 7 (there's no need to display the legend on the side). Your plot should look very similar to the figure below.



Save your plot, give it a title (to make it different from ours), and put it inside your written homework. Please do the same for other plotting questions from now on.

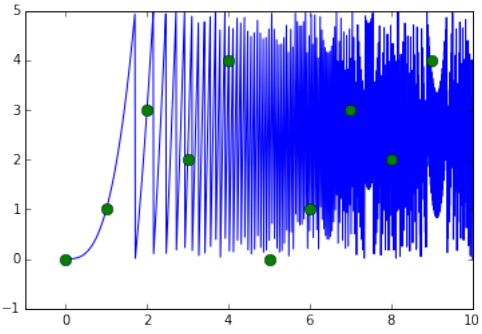
(c) Plot $x^3 \pmod 5$ for $x \in [0,5]$ in the reals. For example, $2.3 \equiv 7.3 \pmod 5$. On the same graph, plot $x^3 \pmod 5$ for integer values of $x = 0, 1, \ldots, 5$.

Solutions:



(d) Now repeat (c) for $x \in [0,10]$ and $x = 0, 1, \ldots, 10$. What do you observe? What happens to each plot as x gets larger? Why do you think this is happening?

Solutions:



We see the reals plot "wrapping" around faster and faster as x gets larger and the integer version repeating itself periodically.

The reals plot is wrapping around faster for large x because $f(x) = x^3$ gets significantly larger as x increases. Because of this, it might look like the plot is not even wrapping, and consists of many parallel straight lines, which is not correct. This can be viewed as a kind of "aliasing" that students will understand better conceptually after taking EECS 20, 120, and 123.

(e) Last week, we implemented Lagrange Interpolation for a degree 2 polynomial. This week, we will do so again for any degree d polynomial in a finite field. Implement the function `interpolate`, which takes a list of points and a prime modulus n, and returns a polynomial that passes through the points in $GF(n)$.

In the skeleton, we have given you most of the code you need for this function, except for three blanks. In your written homework, tell us what you fill in for these blanks and why.

Solutions: The first blank is `Polynomial([-pt2[0], 1], n)`. This is simply the numerator in the formula for Δ_i, where we multiply by $x - x_j$.

The second blank is `(pt[0] - pt2[0])`. Again, this is the denominator in Δ_i's formula.

The last blank is `pt[1] * delta_i`, where we add $y_i\Delta_i$ to the final polynomial $p(x)$.

(f) In the skeleton, you will find an incomplete implementation of a simple Secret Sharing class. First, implement the method `add_share`, which simply adds a share to the list of shares.

After adding enough shares, you can then reconstruct the polynomial with Lagrange Interpolation (part (b)), and evaluate it at 0 to find the secret. Implement the method `find_secret`, which basically conveys the idea described above.

Test your implementation by carrying out the example on page 7/8 of Note 7. Use the same shares in the example (assuming officials 3, 4, and 5 get together to recover the secret).

Now, choose a different set of 3 shares and recover the secret. What do you get? Is it consistent with the above part? Why is that the case?

Solutions: The secret is 1, which is consistent with the above part. This Secret Sharing scheme is designed so that any 3 or more officials are able to figure out the secret, so it doesn't matter which three come together.

(g) In the skeleton, you will find a simple implementation of an Erasure Error client. It has the ability to add/send packets and drop deterministic or random packet(s).

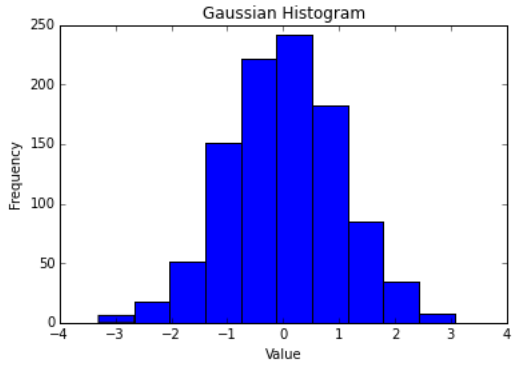
Your task is to carry out the example between Alice and Bob on page 2 and 3 of Note 8 from scratch using `ErasureError`. You will need to add the packets, find the polynomial constructed by interpolating the packets, and drop two packets mentioned in the note. Finally, construct another polynomial using the remaining packets, and make sure that the two polynomials match.

Now, add the packets that were dropped back, and drop two *random* packets. Can you still recover the original message like the first case? In one sentence, explain why. How about if we drop one random packet? Three or more random packets?

Solutions: We can still recover the original message since Erasure Error is designed to counter any random drop – the drop doesn't have to be deterministic. If we drop one packet, we can still recover the message since we technically have more points than needed to reconstruct the polynomial. If three or more packets are dropped, we can't reconstruct the polynomial, and hence won't be able to recover the original message.

(h) In the last couple of labs, we learned how to plot a simple curve and a bar chart. To wrap up our intro to Matplotlib series, we will learn another important plot today: the histogram.

What we will use for our data is 1000 random numbers, drawn from a Gaussian distribution. This is the common "normal" distribution, or the "bell curve" that occurs so frequently in nature. We will use a Gaussian centered about zero, with a standard deviation of 1.0 (this is the default for `np.random.normal`).



Listing 1: Gaussian Histogram Code

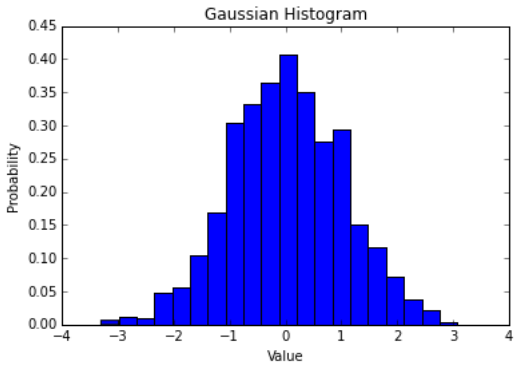
```
gaussian_numbers = np.random.normal(size=1000)
plt.hist(gaussian_numbers)
plt.title("Gaussian Histogram")
plt.xlabel("Value")
plt.ylabel("Frequency")
```

First, we want to plot a probability distribution instead of just frequency counts. In other words, we want to scale the values appropriately so that rather than showing how many numbers in each bin, we instead have a probability of finding a number in that bin.

We also want to increase the number of bins. `plt.hist` defaults to 10 bins, and since we have 1000 points, that seems a bit too small. Change the number of bins to 20.

You need to look into the documentation of `plt.hist` and find the argument(s) that help you accomplish the tasks above.

Your final plot should look very similar to the figure below.



In your written homework, insert your plot and state which arguments you use.

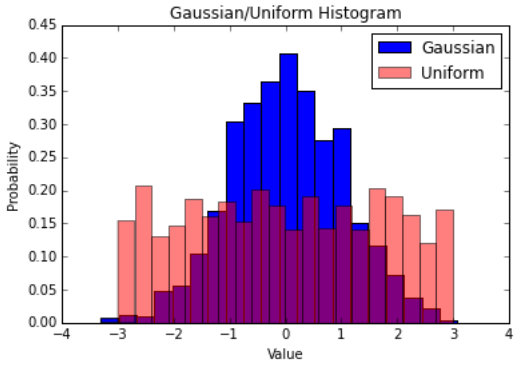
Solutions: `bins=X` specifies that we want X bins in our histogram.

`normed=True` specifies that the counts are normalized to form a probability density.

(i) Add a histogram of 1000 uniformly distributed random numbers from -3 to 3 in red with 50% transparency over top the blue Gaussian from the first task.

Also, give each of your histogram a label. Then, at the very end, you can call `plt.legend()`, which will display the legends nicely.

Your final plot should look very similar to the figure below.



In your written homework, insert your plot and state which arguments you use.

Solutions: To create the uniform distribution histogram, we use `np.random.uniform(low=-3, high=3, size=1000)`. The parameters are self-explanatory.

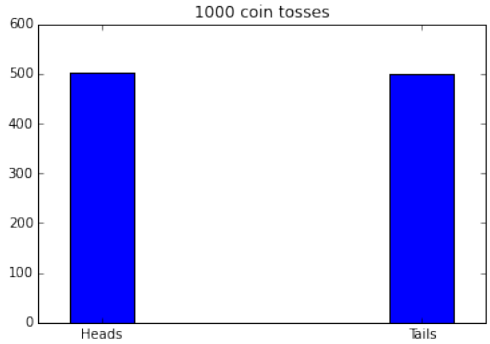
`bin` and `normed` are used again. We also use `color` to specify the plot's color, and `label` to give each plot a label for the legend. In the uniform distribution histogram, we also use `alpha=0.5`, which specifies that this plot will have 50% transparency.

(j) This last question is meant to warm you up for next week's lab on randomness. Imagine you are tossing a fair coin k times, and you would like to count the number of heads. Implement the function `count_heads`, which takes in k, the number of tosses, and returns the number of heads in k tosses.

Do 1000 coin tosses. Plot a bar chart of how many heads you got v.s. how many tails. What do you observe? Does it match your expectation?

Hint: In Python, `random.randint(a, b)` returns a random integer x such that $a \leq x \leq b$. Can you think of a way to use this function to simulate the fair coin tosses? You don't necessarily have to use `randint`, but you will definitely need a function from the `random` module.

Solutions:



The number of heads is approximately equal to the number of tails. This is the expected behavior, since each side has a one-half chance of coming up in tossing a fair coin.

Next week, we will begin our journey into the world of randomness and probability. Please make sure to review the plotting questions on the Virtual Labs, as well as the Matplotlib tutorial that was posted on Piazza.

Reminder: When you finish, don't forget to convert the notebook to pdf and merge it with your written homework. Please also zip the ipynb file and submit it as `hw7.zip`.

3. Polynomials in finite fields

Show your work clearly and use the same notations as in Lecture Note 7.

(a) Find a polynomial $h(x) = ax^2 + bx + c$ of degree at most 2 such that $h(0) \equiv 3 \pmod 7$, $h(1) \equiv 6 \pmod 7$, and $h(2) \equiv 6 \pmod 7$.

Solutions: Solve this using Lagrange interpolation. First we would find the three $\Delta_i(x)$ polynomials.

$$\begin{aligned}
\Delta_0(x) &\equiv \frac{(x-1)(x-2)}{(0-1)(0-2)} \pmod 7 \\
&\equiv 2^{-1} \cdot (x^2 - 3x + 2) \pmod 7 \\
&\equiv 4 \cdot (x^2 - 3x + 2) \pmod 7 \\
&\equiv 4x^2 - 12x + 8 \pmod 7 \\
&\equiv 4x^2 + 2x + 1 \pmod 7
\end{aligned}$$

$$\begin{aligned}
\Delta_1(x) &\equiv \frac{(x-0)(x-2)}{(1-0)(1-2)} \pmod 7 \\
&\equiv (-1)^{-1} \cdot (x^2 - 2x) \pmod 7 \\
&\equiv (-1) \cdot (x^2 - 2x) \pmod 7 \\
&\equiv -x^2 + 2x \pmod 7 \\
&\equiv 6x^2 + 2x \pmod 7
\end{aligned}$$

$$\begin{aligned}
\Delta_2(x) &\equiv \frac{(x-0)(x-1)}{(2-0)(2-1)} \pmod 7 \\
&\equiv 2^{-1} \cdot (x^2 - x) \pmod 7 \\
&\equiv 4 \cdot (x^2 - x) \pmod 7 \\
&\equiv 4x^2 - 4x \pmod 7 \\
&\equiv 4x^2 + 3x \pmod 7
\end{aligned}$$

Next we compute $h(x)$:

$$\begin{aligned}
h(x) &\equiv h(0)\Delta_0(x) + h(1)\Delta_1(x) + h(2)\Delta_2(x) \pmod 7 \\
&\equiv 3 \cdot (4x^2 + 2x + 1) + 6 \cdot (6x^2 + 2x) + 6 \cdot (4x^2 + 3x) \pmod 7 \\
&\equiv 72x^2 + 36x + 3 \pmod 7 \\
&\equiv 2x^2 + x + 3 \pmod 7
\end{aligned}$$

(b) Find a polynomial $h(x) = ax^2 + bx + c$ of degree at most 2 such that $h(0) \equiv 2 \pmod 7$, $h(1) \equiv 5 \pmod 7$, and $h(2) \equiv 1 \pmod 7$.

Solutions: Again, we would first find the three $\Delta_i(x)$ polynomials (note that these are the same $\Delta_i(x)$ polynomials that you already found in part (a)).

$$\Delta_0(x) \equiv \frac{(x-1)(x-2)}{(0-1)(0-2)} \equiv \frac{(x-1)(x-2)}{2} \equiv 4(x-1)(x-2) \equiv 4x^2 + 2x + 1 \pmod 7$$

$$\Delta_1(x) \equiv \frac{(x-0)(x-2)}{(1-0)(1-2)} \equiv \frac{(x-0)(x-2)}{-1} \equiv -(x)(x-2) \equiv 6x^2 + 2x \pmod 7$$

$$\Delta_2(x) \equiv \frac{(x-0)(x-1)}{(2-0)(2-1)} \equiv \frac{(x-0)(x-1)}{2} \equiv 4(x)(x-1) \equiv 4x^2 + 3x \pmod 7$$

Next we compute $h(x)$:

$$\begin{aligned} h(x) &\equiv 2\Delta_0(x) + 5\Delta_1(x) + 1\Delta_2(x) \pmod 7 \\ &\equiv 2(4x^2+2x+1) + 5(6x^2+2x) + 1(4x^2+3x) \pmod 7 \\ &\equiv x^2 + 4x + 2 + 2x^2 + 3x + 4x^2 + 3x \pmod 7 \\ &\equiv 3x + 2 \pmod 7 \end{aligned}$$

Notice that the resulting polynomial is of degree 1.

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 $q(x)$.)

Solutions: We will show the contrapositive. That is, we will prove the following statement "If both $p(x)$ and $q(x)$ are non-zero polynomials, then $p(x)q(x)$ is a non-zero polynomial."

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. 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 $x_{notzero}$, for which $p(x_{notzero}) \neq 0$ and $q(x_{notzero}) \neq 0$. This gives us $p(x_{notzero}) \cdot q(x_{notzero}) \neq 0$ so $p(x)q(x)$ is non-zero.

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

Solutions: We provide a counterexample. In $GF(p)$, $x^{p-1} - 1$ and x are both non zero polynomials, but their product, $x^p - x$ is zero for all x by Fermat's Little Theorem.

Another way to solve this problem is by constructing polynomials that are each zero over complementing parts of the domain; for instance, picking x and $x - 1 \pmod 2$.

5. GCD of Polynomials

Let $A(x)$ and $B(x)$ be polynomials (with coefficients in $\mathbb{R}$). We say that $\gcd(A(x), B(x)) = D(x)$ if $D(x)$ divides $A(x)$ and $B(x)$, and if every polynomial $C(x)$ that divides both $A(x)$ and $B(x)$ also divides $D(x)$. For example, $\gcd((x-1)(x+1), (x-1)(x+2)) = x - 1$. Notice this is the exact same as the normal definition of GCD, just extended to polynomials.

Incidentally, $\gcd(A(x), B(x))$ is the highest degree polynomial that divides both $A(x)$ and $B(x)$. In the subproblems below, you may assume you already have a subroutine `divide(`$P(x), S(x)$`)` for dividing two polynomials, which returns a tuple $(Q(x), R(x))$ of the quotient and the remainder, respectively, of dividing $P(x)$ by $S(x)$.

(a) Write a recursive program to compute $gcd(A(x), B(x))$.

Solutions: Specifically, we wish to find a gcd of two polynomials $A(x)$ and $B(x)$, assuming that $\deg A(x) \geq \deg B(x) > 0$. Here, $\deg A(x)$ denotes the degree of $A(x)$.

We can find two polynomials $Q_0(x)$ and $R_0(x)$ by polynomial long division (see lecture note 7) which satisfy

$$A(x) = B(x)Q_0(x) + R_0(x), \qquad 0 \leq \deg R_0(x) < \deg B(x)$$

Notice that a polynomial $C(x)$ divides $A(x)$ and $B(x)$ if and only if it divides $B(x)$ and $R_0(x)$.

[Proof (also from Discussion 7W): $C(x)$ divides $A(x)$ and $B(x)$, there $\exists S(x)$ and $S'(x)$ s.t. $A(x) = C(x)S(x)$ and $B(x) = C(x)S'(x)$, so $R_0(x) = A(x) - B(x)Q_0(x) = C(x)(S(x) - S'(x)Q_0(x))$, therefore $C(x)$ divides $R_0(x)$ or $R_0(x) = 0$.]

We deduce that

$$gcd(A(x), B(x)) = gcd(B(x), R(x))$$

and set $A_1(x) = B_1(x), B_1(x) = R_0(x)$; we then repeat to get new polynomials $Q_1(x), R_1(x), A_2(x), B_2(x)$ and so on. The degrees of the polynomials keep getting smaller and will eventually reach a point at which $B_N(x) = 0$; and we will have found our gcd:

$$gcd(A(x), B(x)) = gcd(A_1(x), B_1(x)) = \cdots = gcd(A_N(x), 0) = A_N(x)$$

Here, we have the function that can perform the polynomial long division on $A(x)$ and $B(x)$ and return both the quotient $Q(x)$ and the remainder $R(x)$, i.e. [$Q(x)$, $R(x)$] = div($A(x), B(x)$). The algorithm can be extended from the original integer-based GCD as follows:

Algorithm 1 gcd(A(x), B(x))

if $B(x) = 0$ **then**
  return $A(x)$
else if $degA(x) < degB(x)$ **then**
  return $gcd(B(x), A(x))$
else
  $(Q(x), R(x)) = divide(A(x), B(x))$
  return $gcd(B(x), R(x))$
end if

(b) Write a recursive program to compute $extended\text{-}gcd(A(x), B(x))$.

Solutions: We will return a triple of polynomials $(d(x), g(x), h(x))$ such that $d(x) = gcd(A(x), B(x))$ and $d(x) = g(x) * A(x) + h(x) * B(x)$. The algorithm can be extended from the polynomial-based GCD algorithm in part (a) by returning two more polynomials, $g(x)$ and $h(x)$.

We first recursively compute a triple of polynomials $(d(x), g(x), h(x))$ such that

$$d(x) = gcd(B(x), A(x) - Q(x)B(x)) = gcd(B(x), R(x))$$

and

$$d(x) = g(x)B(x) + h(x)R(x).$$

As discussed in part (a), $d(x)$ will be $gcd(A(x), B(x))$. As the goal of the algorithm, we need to update polynomials $g(x)$ and $h(x)$ by $g'(x)$ and $h'(x)$ such that

$$d(x) = g'(x)A(x) + h'(x)B(x).$$

By rearranging the equation:

$$\begin{aligned} d(x) &= g(x)B(x) + h(x)R(x) \\ &= g(x)B(x) + h(x)(A(x) - Q(x)B(x) \\ &= h(x)A(x) + (g(x) - h(x)Q(x)B(x) \end{aligned}$$

Thus,

$$g'(x) = h(x) \text{ and } h'(x) = g(x) - h(x)Q(x).$$

Therefore, the algorithm can be extended as follows:

Algorithm 2 extended-gcd(A(x),B(x))

if $B(x) = 0$ **then**
  return $(A(x), 1, 0)$
else
  $(Q(x), R(x)) = divide(A(x), B(x))$
  $(d(x), g(x), h(x)) =$ extended-gcd$(B(x), R(x))$
  return $(d(x), h(x), g(x) - Q(x)h(x))$
end if

6. How many packets?

(a) The error-correcting code for erasure errors 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 $0 < \alpha < 1$). How many packets do we need to send in total (as a function of n and α)?

Solutions: Suppose we send a total of $n+k$ packets (where k is to be determined). Since at most a fraction α of these are lost, the number of packets received is at least $(1-\alpha)(n+k)$. But in order to reconstruct the polynomial used in transmission, we need at least n packets. Hence it is sufficient to have $(1-\alpha)(n+k) \geq n$, which can be rearranged to give $k \geq \frac{\alpha}{1-\alpha}n$, so $k = \lceil \frac{\alpha n}{1-\alpha} \rceil$ will suffice. Therefore the total number of packets we must send is $n+k = \lceil \frac{n}{1-\alpha} \rceil$ packets.

(b) How many packets do we need to send in total (as a function of n and α) in the case of general errors?
Solutions: Suppose we send a total of $n+k$ packets (where k is to be determined). The number of uncorrupted packets received is at least $(1-\alpha)(n+k)$ and the number of errors is at most $\alpha(n+k)$. We need to have two redundant packets to be able to absorb the malicious corruption of a single packet. Thus, redundant packets $k \geq 2\alpha(n+k)$ Rearranging the equation, we have $k \geq \frac{2\alpha}{1-2\alpha}n$, so $k = \lceil \frac{2\alpha n}{1-2\alpha} \rceil$ will suffice. Therefore we must send a total of $n+k = \lceil \frac{n}{1-2\alpha} \rceil$ packets.

7. Alice wants to talk

Alice has a message of length $n = 3$ for Bob. She also has another message of length $n = 3$ for Charles. Her message for Bob is $a_0 = 4$, $a_1 = 3$, and $a_2 = 2$. And her message for Charles is $a_0 = 1$, $a_1 = 2$, and $a_2 = 2$.

(a) If Alice accounts for $k = 1$ general errors, then what are Alice's augmented messages to Bob and Charles (each modulo 5)?

Solutions:

You could have solved this problem either using interpolation encoding, where Alice encodes her message as points of some polynomial, or coefficient encoding, where Alice encodes her message as the coefficients of some polynomial. We will provide solutions for both. To avoid confusion, we will use $P(x)$ to denote the polynomial Alice uses to send messages to Bob and $R(x)$ to denote the polynomial Alice uses to send messages to Charles.

Using interpolation encoding:

Since Alice wants to guard against $k = 1$ general errors, she will need to send an augmented message of length $n + 2k = 5$. She finds the polynomial that describes her message to Bob, i.e., a polynomial $P(x)$ of degree at most 2 that passes through $(0,4)$, $(1,3)$, and $(2,2)$. Using Lagrange interpolation

modulo 5,

$$\Delta_0 \equiv \frac{(x-1)(x-2)}{(0-1)(0-2)} \equiv \frac{x^2+2x+2}{2} \equiv 3x^2+x+1 \pmod 5,$$
$$\Delta_1 \equiv \frac{x(x-2)}{(1-0)(1-2)} \equiv \frac{x^2+3x}{-1} \equiv \frac{x^2+3x}{4} \equiv 4(x^2+3x) \equiv 4x^2+2x \pmod 5,$$
$$\Delta_2 \equiv \frac{x(x-1)}{(2-0)(2-1)} \equiv \frac{x^2+4x}{2} \equiv 3(x^2+4x) \equiv 3x^2+2x \pmod 5,$$

so

$$\begin{aligned} P(x) &\equiv 4(3x^2+x+1)+3(4x^2+2x)+2(3x^2+2x) \pmod 5 \\ &\equiv 4x+4 \pmod 5. \end{aligned}$$

She knows she needs to augment with 2 additional packets, the polynomial evaluated at $x=3$ and $x=4$. So she finds $P(3) \equiv 4\cdot 3+4 \equiv 1 \pmod 5$ and $P(4) \equiv 4\cdot 4+4 \equiv 0 \pmod 5$. The message she sends to Bob is thus $(4,3,2,1,0)$.

Now, Alice must do the same thing for the message she wants to send to Charles. She needs to find another polynomial $R(x)$ of degree at most 2 that passes through $(0,1)$, $(1,2)$, and $(2,2)$. Alice recognizes that the polynomials she finds for Δ_0, Δ_1, and Δ_2 will be the same as those she found for Bob, since we are still working with the x values 0, 1, and 2. The only thing that changes are the y values (the second number in the tuples). She can then compute

$$\begin{aligned} R(x) &\equiv 1(3x^2+x+1)+2(4x^2+2x)+2(3x^2+2x) \pmod 5 \\ &\equiv 2x^2+4x+1 \pmod 5. \end{aligned}$$

Again, she knows she needs to augment with 2 additional packets, the polynomial evaluated at $x=3$ and $x=4$. So she finds $R(3) \equiv 2\cdot 3^2+4\cdot 3+1 \equiv 1 \pmod 5$ and $R(4) \equiv 2\cdot 4^2+4\cdot 4+1 \equiv 4 \pmod 5$. The message she sends to Charles is thus $(1,2,2,1,4)$.

Using coefficient encoding:

Alice sets the message she wants to send as the coefficients of the polynomial and then evaluates the polynomial at $x=0,1,2,3,4$ to encode her message. For Bob, her polynomial is $P(x)=2x^2+3x+4$, and for Charles the polynomial is $R(x)=2x^2+2x+1$. So Alice sends to Bob the message

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

and the message she sends to Charles is

$$(R(0),R(1),R(2),R(3),R(4)) = (1,0,3,0,1).$$

(b) Alice now transmits the augmented message intended for Bob over an erasure channel. Bob receives only $P(0)$, $P(2)$, and $P(4)$. The rest are erased. How does Bob recover Alice's message? Show your work in detail.

Solutions:

Using interpolation encoding:

Since Alice has sent an augmented message of length 5, 2 erasure errors can be tolerated. Bob can the recover Alice's original message by finding the polynomial that passes through $(0,4)$, $(2,2)$, and

$(4,0)$. He uses Lagrange interpolation:

$$\Delta_0 \equiv \frac{(x-2)(x-4)}{(0-2)(0-4)} \equiv \frac{x^2+4x+3}{3} \equiv 2(x^2+4x+3) \equiv 2x^2+3x+1 \pmod 5$$
$$\Delta_2 \equiv \frac{x(x-4)}{(2-0)(2-4)} \equiv \frac{x^2+x}{1} \equiv x^2+x \pmod 5.$$

Bob recognizes that since the y value is 0 for the point $(4,0)$, he doesn't need to calculate Δ_4 since $0 \cdot \Delta_4 = 0$. He then recovers the polynomial by computing

$$P(x) \equiv 4(2x^2+3x+1)+2(x^2+x) \equiv 4x+4 \pmod 5.$$

He then finds $P(1)$ to recover Alice's message: $P(1) \equiv 8 \equiv 3 \pmod 5$, so Alice's original message was $(4,3,2)$.

Using coefficient encoding:

In this case, Bob has the points $(0,4),(2,3)$, and $(4,3)$. He uses Lagrange interpolation to find $P(x)$:

$$\Delta_0 \equiv \frac{(x-2)(x-4)}{(0-2)(0-4)} \equiv \frac{x^2+4x+3}{3} \equiv 2(x^2+4x+3) \equiv 2x^2+3x+1 \pmod 5$$
$$\Delta_2 \equiv \frac{x(x-4)}{(2-0)(2-4)} \equiv \frac{x^2+x}{1} \equiv x^2+x \pmod 5$$
$$\Delta_4 \equiv \frac{x(x-2)}{(4-0)(4-2)} \equiv \frac{x^2+3x}{3} \equiv 2x^2+x \pmod 5.$$

He then recovers the polynomial by computing

$$P(x) \equiv 4(2x^2+3x+1)+3(x^2+x)+3(2x^2+2) \equiv 2x^2+3x+4 \pmod 5.$$

He then looks at the coefficients of $P(x)$ to recover Alice's message, so Alice's original message was $(a_0,a_1,a_2) = (4,3,2)$.

(c) Alice then transmits the augmented message intended for Charles over a noisy channel. Charles receives the entire message but now $P(2)$ is corrupted to $P(2)+2 \pmod 5$. Charles doesn't know where the error is but he does know that at most one error has occurred. How will Charles recover Alice's message?

Solutions:

Using interpolation encoding:

Charles receives $(1,2,4,1,4)$. He will use the Berklekamp-Welch method to locate the error and find the original polynomial $R(x)$.

He knows that since at most one error has occurred, the error locating polynomial will have the form $E(x) = x + b_0$. He also knows that there exists a polynomial $Q(x)$ of degree at most $n+k-1 = 3$ such that $Q(x) = R(x)E(x)$. Thus he knows $Q(x)$ will have the form $Q(x) = a_3x^3 + a_2x^2 + a_1x + a_0$. Using the relationship $Q(i) = r_iE(i)$, where r_i are the values of the messages he received, he writes the system of linear equations

$$\begin{aligned} a_0 &= b_0 \\ a_3+a_2+a_1+a_0 &= 2(1+b_0) \\ 8a_3+4a_2+2a_1+a_0 &= 4(2+b_0) \\ 27a_3+9a_2+3a_1+a_0 &= 3+b_0 \\ 64a_3+16a_2+4a_1+a_0 &= 4(4+b_0). \end{aligned}$$

Then reducing modulo 5 and rearranging so all unknowns are on the left-hand side, he gets the system of equations

$$\begin{aligned} a_0 + 4b_0 &= 0 \\ a_3 + a_2 + a_1 + a_0 + 3b_0 &= 2 \\ 3a_3 + 4a_2 + 2a_1 + a_0 + b_0 &= 3 \\ 2a_3 + 4a_2 + 3a_1 + a_0 + 4b_0 &= 3 \\ 4a_3 + 1a_2 + 4a_1 + a_0 + b_0 &= 1. \end{aligned}$$

To solve this, he could either use repeated substitution to eliminate variables, or put these equations in augmented matrix form and use Gaussian elimination. After solving he finds that $b_0 = 3$, $a_0 = 3$, $a_1 = 3$, $a_2 = 0$, and $a_3 = 2$. Therefore $E(x) \equiv x + 3 \equiv x - 2 \pmod 5$, so he knows the error is located in $R(2)$. He also knows that $Q(x) = 2x^3 + 3x + 3$, and he can find $R(x)$ by dividing $Q(x)$ by $E(x)$, which gives $R(x) = 2x^2 + 4x + 1$. He then computes $R(2) = 2$ to recover the corrupted packet, and determines that the original message Alice sent was $(1, 2, 2)$.

Using coefficient encoding: Charles receives $(1, 0, 0, 0, 1)$. He will use the Berklekamp-Welch method to locate the error and find the original polynomial $R(x)$.

He knows that since at most one error has occurred, the error locating polynomial will have the form $E(x) = x + b_0$. He also knows that there exists a polynomial $Q(x)$ of degree at most $n + k - 1 = 3$ such that $Q(x) = R(x)E(x)$. Thus he knows $Q(x)$ will have the form $Q(x) = a_3x^3 + a_2x^2 + a_1x + a_0$. Using the relationship $Q(i) = r_iE(i)$, where r_i are the values of the messages he received, he writes the system of linear equations

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

Then reducing modulo 5 and rearranging so all unknowns are on the left-hand side, he gets the system of equations

$$\begin{aligned} a_0 + 4b_0 &= 0 \\ a_3 + a_2 + a_1 + a_0 &= 0 \\ 3a_3 + 4a_2 + 2a_1 + a_0 &= 0 \\ 2a_3 + 4a_2 + 3a_1 + a_0 &= 0 \\ 4a_3 + 1a_2 + 4a_1 + a_0 + 4b_0 &= 4. \end{aligned}$$

To solve this, he could either use repeated substitution to eliminate variables, or put these equations in augmented matrix form and use Gaussian elimination. After solving he finds that $b_0 = 3$, $a_0 = 3$, $a_1 = 2$, $a_2 = 3$, and $a_3 = 2$. Therefore $E(x) \equiv x + 3 \equiv x - 2 \pmod 5$, so he knows the error is located in $R(2)$. He also knows that $Q(x) = 2x^3 + 3x^2 + 2x + 3$, and he can find $R(x)$ by dividing $Q(x)$ by $E(x)$, which gives $R(x) = 2x^2 + 2x + 1$. He then looks at the coefficients of $R(x)$ to recover the uncorrupted message, and determines that the original message Alice sent was $(a_0, a_1, a_2) = (1, 2, 2)$.

8. Trust No One

Gandalf has assembled a fellowship of eight people to transport the One Ring to the fires of Mount Doom: four hobbits, two men, 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 races (hobbits, men, elf, and dwarf) do not completely trust each other so instead we decide to require members from at least two races in order to use the ring. In particular, we will require a unanimous decision by all members of one race in addition to at least one member of a different race. 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 men, the elf, and the dwarf.

More explicitly, some examples: only four hobbits agreeing to use the ring is not enough to know the instructions. Only two men agreeing is not enough. Only the elf agreeing is not enough. Only the dwarf agreeing is not enough. All four hobbits and a man agreeing is enough. Both men 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 men, 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 (except Gandalf) if not enough members of the party agree.
- If only the members of one race agree, the message remains a secret.
- If all the members of one race agree plus at least one additional person, the message can be determined.
- Other combinations of members (e.g. two hobbits and a man) can either determine the message or keep it a secret (it is up to your discretion).

Solutions:

There will be two parts to this secret: a unanimity secret U and a multi-race secret M. U ensures that at least all members of one races are in agreement while M ensures that members of at least two races 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 race. We will require all members of that race 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 men will each have distinct points of a degree-1 polynomial. When all members of a race 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 race.

The *multi-race secret* involves creating a degree-1 polynomial $P_m(x)$ and giving one point to all members of each race. For example, the hobbits may each get $P_m(1)$ while the elf gets $P_m(2)$ and the men each get $P_m(3)$. In this way if members of any two races 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-race 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 men $P_m(3) = 6 \equiv 1$, and the dwarf $P_m(4) = 7 \equiv 2$. Now any two members of distinct races 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 races. On the other hand, the men will each have a point on the degree-1 polynomial $P_{men}(x)$. Suppose $P_{men}(x) = 2x + 2$. Then the first man receives $P_{men}(1) = 4$ and the second receives $P_{men}(2) = 4 + 2 = 6 \equiv 1$. When they interpolate using these values, they will discover the original polynomial and therefore $P_{men}(0) = U = 2$. 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 men and one hobbit come together. The two men 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 $P_m(0) = M$. Now that they have U and M, they can interpolate to find $R(x)$ and thus $R(0) = s = 4$.

9. Write Your Own Problem

Write your own problem related to this week's material and solve it. You may still work in groups to brainstorm problems, but each student should submit a unique problem. What is the problem? How to formulate it? How to solve it? What is the solution?