EECS 70 Discrete Mathematics and Probability Theory
Fall 2014 Anant Sahai

Homework 8

This homework is due October 27, 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. Intro to Randomness Lab

We now have all the tools to start diving into the world of randomness. The following questions are designed to give you some intuition about concepts in probability. Some of the powerful ideas behind the plots will be further explored later in the course.

Starting from this Virtual Lab, please leave your answers in the empty Markdown cells in the skeleton. The converted pdf will be your entire answer to this question – there's no need to individually save figures or write down anything for the lab.

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

(a) Last week, we did 1000 coin tosses and plotted a bar chart of how many heads we got v.s. how many tails. This week, we will do the same thing again 1000 times.

Plot a histogram of how many times you got N heads, where $0 \leq N \leq 1000$. What do you observe about N and its frequency?

Hint: Implement the function `count_heads_in_runs(k, n)` in the skeleton, which returns a list of length n, where each element is the number of heads in k flips. In this case, $k = n = 1000$.

(In general, always try to parameterize the values instead of hard-coding it. This will make your code reusable for later parts.)

(b) Let k be a parameter that tells how many coins you toss in one experiment. Do part (a) again for the following sequence of ks: 2, 5, 10, 100, 1000, 10000. What do you observe as k gets larger?

(c) Notice that the horizontal axis has different scales as k varies. Suppose you wanted to "center" these histogram plots. How should you change your code for the above part to center the plots around the origin (0) as k varies?

Plot the resulting histogram of $N - f(k)$ where N is the number of heads in a particular run of fair coin tosses and $f(k)$ is the shift that you have chosen.

(d) Repeat the plots of the previous part except this time, choose a common set of units for the x-axes (so the x-axes in all plots will have the same range). What range did you choose, and why?

Hint: You can use `plt.xlim()` to set the x-axis's range.

(e) Repeat the plots of the previous part except this time, choose a normalized set of units, corresponding to the fraction of heads appearing (from 0 to 1).

The left-most point should correspond to the case of tossing all tails. And the right-most point should correspond to the case of tossing all heads. How is this set of plots different from the previous ones?

(f) Comment on what you observed in the three sets of plots above. Five or less sentences should be sufficient.

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 `hw8.zip`.

3. Chinese Remainder Theorem for Polynomials

(a) Prove that the remainder of polynomial $p(x)$ divided by $(x-c)$ is $p(c)$.

(b) Consider extending the notion of "modding" to polynomials. Similar to how x mod 5 is the remainder when x is divided by 5 (or the equivalence class of $\{x+5k : k \in Z\}$), let $p(x)$ mod $(x-1)$ be the remainder when $p(x)$ is divided by $(x-1)$ (or the equivalence class of... *what?*).

Solve the following system for all polynomials $p(x)$ over $GF(5)$ which satisfy:

$$\begin{aligned} p(x) &\equiv 3 && \mod (x-1) \\ p(x) &\equiv 3 && \mod (x+2) \\ p(x) &\equiv 1 && \mod (x) \end{aligned}$$

(Hint: Interpret the system using the result of part (a))

(c) From the previous HW, we know $gcd((x-1),(x+2)) = 1$. Still considering polynomials over $GF(5)$, does $(x+2)$ have a multiplicative inverse mod $(x-1)$?

(d) Show how to solve the system of congruences in part (b) using the explicit form of the Chinese Remainder Theorem (by analogy to the usual CRT). Comment on the similarities/differences to how you solved (b) previously.

(e) *(optional)* Now that we can take the inverse of certain polynomials in the "mod polynomial" universe (as in part (c)), let's see how far this takes us (unrelated to the CRT). Consider the set of all polynomials over $GF(3)$, modulo (x^2+1). How many distinct elements are there? How many of them have multiplicative inverses? Does this construction form a field?

4. Guardians of the Galaxy Rendez-Vous

The Guardians of the Galaxy are back, and this time they need your help! The band has gone their separate ways, and Peter Quill needs to organize an urgent meeting to alert his friends about the latest threat to the galaxy. Unfortunately, communication across the galaxy still isn't perfect – stray radiation can erase parts of messages!

Rocket the Raccoon has told Peter about a strategy to get his message across:

1. There are only 105 possible safe planets Peter would use to meet his friends, and he labels them 0 to 104. His friends also know which planet each number ID refers to.
2. He takes the unique ID and finds the remainder mod 3, 5, 7, 11 and 13.
3. This is the message he sends across, as a five-number tuple.
4. The receiver makes clever use of the CRT to find the unique message.

For example, if Peter wants to go planet 51, he sends the message $(0,1,2,7,12)$. If the first symbol is erased in transit, the received message will be $(X,1,2,7,12)$.

(a) Gamora gets the message $(X,4,X,6,3)$! What planet does she thinks Peter is at?

(b) Although Peter wants to use this scheme, he's still a little paranoid everyone will make the rendezvous. Describe the "clever use" of the CRT more precisely. How many radiation-erasures can this scheme tolerate, in the worst case? Prove it.

(c) Even with Ronan gone, Thanos and the Ravagers are still out to get Peter! They will do their best to corrupt his message. Groot gets the message $(2,1,2,2,10)$, which may have at most one corruption. Can he/it determine where Peter wants to meet?

5. Magic!

In this problem we will investigate what happens when in error-correcting codes there are fewer errors than the decoding algorithm is able to handle. For the entire problem we are working in $GF(7)$.

Assume that we wish to transfer a message of length 2, which we denote by (m_1,m_2). Each m_i is a member of $GF(7)$. We also wish to be able to correct up to $k=1$ error. Using the error-correcting codes we learned in class, we have to first find a polynomial $P(x)$ of degree at most 1 such that $P(1)=m_1$ and $P(2)=m_2$. Then we have to extend the message we send by $2k$ symbols, i.e., we will send $(P(1),P(2),P(3),P(4))$ to the recipient.

(a) Consider an example where $(m_1,m_2)=(4,2)$. What are the four symbols that are transmitted?

(b) Now let's work with a different message of length 2. Assume that you have received these numbers: $(5,0,2,4)$, i.e., if there were no errors then we would have $P(1)=5,P(2)=0,P(3)=2,P(4)=4$. Write down the linear equations that help decode error-correcting codes: $Q(i)=P(i)E(i)=r_iE(i)$, for $1\leq i\leq 4$.

(c) Try to solve the linear equations you got in the previous section. You should observe that there are multiple solutions to these equations. Pick two different solutions and for each one write down the error-locating polynomial $E(x)$ and the polynomial $Q(x)$. In each of the two solutions, divide $Q(x)$ by $E(x)$ to get the original polynomial. Do you get the same polynomial in both cases?

(d) Do both $E(x)$'s have the same root? What does that tell you about the position of error in the transmitted message?

(e) Consider accounting for $k=2$ general errors instead of $k=1$. If we want to send a message of length 2, we have to send $2+2k=6$ packets. Suppose only one packet is corrupted, then you will see multiple possible $Q(x)$'s and $E(x)$'s again. What do you expect in common between these $E(x)$'s? Give a brief justification. *(Hint: Think about the positions of errors they point to again.)*

6. Po(l)lynomial Pranks

Alex and Barb talk to each other via Polly. Knowing her tendency to prank, they use polynomials to ensure they can recover their original messages in case she decides to erase (i.e., replace with a blank) or change some of the packets.

(a) Never the one to run out of ideas, Polly adds an integer offset c to the x values of the packets instead, i.e., each packet (x,y) becomes $(x+c,y)$. Will Alex and Barb be able to get their original messages back without knowing c beforehand? Do they need to modify their scheme to handle this prank? If so, describe the method briefly.

(b) Realizing what you just showed, Polly adds the integer offset c to the y values of the packets instead, i.e., each packet (x,y) becomes $(x,y+c)$. Can Alex and Barb get their original messages back using their current scheme? If not, propose a modified scheme that will work.

(c) Polly changes her mind again. Adding a constant offset c is too simple. She now picks a random po(l)lynomial $N(x)$ of degree $\leq d$ and adds it to the y values instead, i.e., each packet (x,y) becomes $(x,y+N(x))$. Note that $N(x)$ can have higher degree than $P(x)$. Suppose Alex and Barb know d, provide a scheme for them to reliably communicate using the minimum number of packets. You only need to give a brief justification.

(d) Extend the scheme in part (c) to account for an additional k_e erasure errors. Does your scheme still use the minimum number of packets? If not, come up with a new scheme that does. You should give a brief justification.

(e) Modify the scheme in part (d) to account for an additional k_g general errors instead of erasure errors.

7. Error-Detecting Codes

In the realm of error-correcting codes, we usually want to recover the original message if we detect any errors, and we want to provide a guarantee of being able to do this even if there are k general errors. Suppose that instead we are satisfied with detecting whether there is any error at all and do not care about the original message if we detect any errors. In class you saw that for recovering from at most k general errors when transmitting a message of length n you need to extend your message by $2k$ symbols and send a message of length $n+2k$. But since we don't require recovering the original message, it is conceivable that we might need less symbols.

Formally, suppose that we have a message consisting of n symbols that we want to transmit. We want to be able to detect whether there is any error if we are guaranteed that there can be at most k general errors. That is, your receiver should be able to say either 'this message is completely correct' and decode it, or say 'this message has at least one error' and throw it away. How should we extend our message (i.e. by how many symbols should we extend, and how should we get those symbols) in order to be able to detect whether our message has been corrupted on its way? You may assume that we work in $GF(p)$ for a very large prime number p. Show that your scheme works, and that adding any lesser number of symbols is not good enough.

8. Orpheus' Adventures in the Halls of Time

You're designing a new role-playing game for a mathematically themed production house. Your eccentric colleague comes to you with an idea for a key scene and he wants you to think about it.

The backstory is that the mortal Orpheus wants to gain knowledge of the dates of certain key events in the year to come: call these the prophecies of interest. He has heard that in the Halls of Time, these things are already known so he quests through the underworld until he comes upon them.

In the Halls of Time, he encounters the Guardians. They have access to the knowledge of the Fates.

The game behaves as follows. There are 12 guardians (corresponding to the 12 constellations of the Zodiac or the 12 months) and each knows all the prophecies, but they have a peculiar property. Half of them are honest and answer questions posed to them exactly. One quarter of them consider mortals to be beneath them and will simply say "Begone mortal!" And one quarter despise mortals and will answer maliciously.

But mortals do not know the secret forms of the guardians and so Orpheus doesn't know who he is talking to.

On this setting, Orpheus can only ask questions (he can invoke arithmetic operations in $GF(367)$ if he wants) whose answer is a number from $\{0,1,2,\ldots,366\}$.

(The prophecies he wants are answers to questions like: "When will my child be born?" The answers can be viewed as numbers: 1, ..., 365 for the days in the coming year. 0 for the past. 366 to represent the future beyond this coming year. Fortunately for Orpheus, 367 happens to be prime.)

All guardians are good at math and can answer any question as long as the answer is from 0 to 366 (not limited to just a simple answer to a prophecy). Orpheus can only ask any individual guardian one question. After that, that particular guardian will magically leave the room. He gets to question all 12 guardians.

How many prophecies can Orpheus reliably extract from the 12 guardians? How can he do it? (Be explicit) Why will this work?

9. Reed-Solomon and Reliable Computation

In this question, we will see how error correction can help with faulty computations. Let us first establish a useful fact:

(a) Suppose we are using a Reed-Solomon code over $GF(p)$ guarding for k transmission errors. Let $a = (a_1, \ldots, a_n)$ and $b = (b_1, \ldots, b_n)$ be two n-packet messages. Show that the Reed-Solomon codeword for message $a+b = (a_1+b_1, \ldots, a_n+b_n)$ is the same as the sum of the Reed-Solomon codewords of a and b. In other words, the RS codeword of the element-wise sum is the element-wise sum of the RS codewords.

Suppose you have invented a machine for doing additions extremely fast. Your invention takes a list of pairs of numbers as input and returns the list of the pairs sums. Although the machine is blazing fast, it is at the same time prone to mistakes. Luckily, you can bound the number of mistakes: over all of the n outputs returned, you know that at most $\max(1, \lfloor n/4 \rfloor)$ outputs have an error. For example, if we feed the machine $((2,3),(4,3),(0,7),(4,2))$ we might get back output $(5,7,4,6)$, where $4 \neq 0+7$ is a mistake.

You want to sell your invention, but none of your potential clients is interested in an error-prone device like this. They feel the speed benefit does not compensate for the unreliability of the results.

(b) Show that you can augment your machine with a Reed-Solomon encoding and decoding scheme such that no wrong outputs are ever returned. Your clients can use the machine the exact same way as before, but they no longer experience erroneous results.

10. 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?