CS 70 Discrete Mathematics and Probability Theory
Fall 2019 Alistair Sinclair and Yun S. Song

HW 5

Due: Thursday, October 10, 2019 at 10 PM

Sundry

Before you start writing your final homework submission, state briefly how you worked on it. Who else did you work with? List names and email addresses. (In case of homework party, you can just describe the group.)

Note: *This homework consists of two parts. The first part (questions 1-4) will be graded and will determine your score for this homework. The second part (questions 5-6) will be graded if you submit them, but will not affect your homework score in any way. You are strongly advised to attempt all the questions in the first part. You should attempt the problems in the second part only if you are interested and have time to spare.*

For each problem, justify all your answers unless otherwise specified.

Part 1: Required Problems

1 Recursively Defined Polynomials

Let's define two polynomials $f_1(x) = x - 2$, $f_2(x) = x^2 + 3$, and for each natural number $n \geq 2$, recursively define $f_n(x) = xf_{n-1}(x) - f_{n-2}(x)$.

(a) Compute $f_3(x)$ and $f_4(x)$.

(b) What is the largest number of roots that $f_n(x)$ can have over $\mathbb{R}$? What is the smallest number of roots it can have? Both answers should depend only on the degree of $f_n(x)$, and one will depend on whether n is even or odd.

(c) Prove that $f_n(2) = 0 \mod 7$ for every n.

2 Equivalent Polynomials

This problem is about polynomials with coefficients in GF(q) for some prime $q \in \mathbb{N}$. We say that two such polynomials f and g are *equivalent* if $f(x) = g(x)$ for every $x \in \text{GF}(q)$.

(a) Use Fermat's Little Theorem to find a polynomial equivalent to $f(x) = x^5$ over GF(5); then find one equivalent to $g(x) = 1 + 3x^{11} + 7x^{13}$ over GF(11).

(b) Prove that whenever $f(x)$ has degree $\geq q$, it is equivalent to some polynomial $\tilde{f}(x)$ with degree $< q$.

3 Secret Veto

In the usual secret-sharing scenario we consider (for instance) a secret vault at the United Nations, which we want to design with the property that any k representatives can pool their information and open it, but any smaller number has no hope of doing so. Assume that the solution in the notes has been implemented, so that the key is some number s, and each member has been assigned a number $f(i) \pmod q$ for some degree $k-1$ polynomial f with coefficients in GF(q) and satisfying $f(0) = s$.

(a) A group of $k+\ell$ representatives get together to discuss opening the vault. What will happen if ℓ representatives are opposed to opening the vault and, instead of revealing their true numbers, secretly reveal some *different* numbers from GF(q)? Will the group be able to open the vault? If so, how long will it take?

(b) Repeat part (a) in the event that only $\ell/2$ of the ℓ representatives in opposition reveal different numbers than they were assigned—assume that ℓ is even.

4 Berlekamp-Welch Algorithm

In this question we will send the message $(m_0, m_1, m_2) = (1, 1, 4)$ of length $n = 3$. We will use an error-correcting code for $k = 1$ general error, doing arithmetic over GF(5).

(a) Construct a polynomial $P(x) \pmod 5$ of degree at most 2, so that

$$P(0) = 1, \qquad P(1) = 1, \qquad P(2) = 4.$$

What is the message $(c_0, c_1, c_2, c_3, c_4)$ that is sent?

(b) Suppose the message is corrupted by changing c_0 to 0. Set up the system of linear equations in the Berlekamp-Welch algorithm to find $Q(x)$ and $E(x)$.

(c) Assume that after solving the equations in part (b) we get $Q(x) = 4x^3 + x^2 + x$ and $E(x) = x$. Show how to recover the original message from Q and E.

Note: *This concludes the first part of the homework. The problems below are optional, will not affect your score, and should be attempted only if you have time to spare.*

Part 2: Optional Problems

5 Repeated Roots

Let $p(x) = a_k x^k + \cdots + a_0$ be a polynomial in the variable x, where k is a positive integer and the coefficients $a_0, \ldots, a_k$ are from some field F (here, F can be $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$, or GF(p) for some prime p). Let's define this polynomial's **derivative** to be the polynomial $p'(x) := ka_k x^{k-1} + \cdots + a_1 = \sum_{j=1}^{k} j a_j x^{j-1}$. We say that α is a **repeated root of** p if $p(x)$ can be factored as $(x-\alpha)^2 q(x)$ for some polynomial q. Show that α is a repeated root of p if and only if $p(\alpha) = p'(\alpha) = 0$.

[Note: You may be familiar with the derivatives of polynomials from studying calculus, but we are not using any calculus here, because it does not really make sense to perform calculus on finite fields! Think of the polynomial's derivative as a formal definition, i.e., in this context, it has nothing to do with rate of change, etc. In particular, you should not use any calculus rules such as the product rule without proof.]

6 Green Eggs and Hamming

The *Hamming distance* between two length-n bit strings b_1 and b_2 is defined as the minimum number of bits in b_1 you need to flip in order to get b_2. For example, the Hamming distance between 101 and 001 is 1 (since you can just flip the first bit), while the Hamming distance between 111 and 000 is 3 (since you need to flip all three bits).

(a) Sam-I-Am has given you a list of n situations, and wants to know in which of them you would like green eggs and ham. You are planning on sending him your responses encoded in a length n bit string (where a 1 in position i says you would like green eggs and ham in situation i, while a 0 says you would not), but the channel you're sending your answers over is noisy and sometimes corrupts a bit. Sam-I-Am proposes the following solution: you send a length $n+1$ bit string, where the $(n+1)$st bit is the XOR of all the previous n bits (this extra bit is called the parity bit). If you use this strategy, what is the minimum Hamming distance between any two valid bit strings you might send? Why does this allow Sam-I-Am to detect an error? Can he correct the error as well?

(b) If the channel you are sending over becomes more noisy and corrupts two of your bits, can Sam-I-Am still detect the error? Why or why not?

(c) If you know your channel might corrupt up to k bits, what Hamming distance do you need between valid bit strings in order to be sure that Sam-I-Am can detect when there has been a corruption? Prove as well that that your answer is tight—that is, show that if you used a smaller Hamming distance, Sam-I-Am might not be able to detect when there was an error.

(d) Finally, if you want to *correct* up to k corrupted bits, what Hamming distance do you need between valid bit strings? Prove that your condition is sufficient.