CS 70 Discrete Mathematics and Probability Theory
Spring 2016 Rao and Walrand HW 6

Due Thursday March 3rd at 10PM

Warmup

1. **(2/2/2/2/2) Reviewing Lagrange**

   (a) Prove the following: If p is a prime and $y_1, \ldots, y_n \in \mathbb{N}$ are all different from 0 modulo p, then $y_1 \times \cdots \times y_n$ is also different from 0 modulo p.

   (b) Prove the following: Given a prime p and two integers a, b, it is always possible to find a polynomial $f(x)$ of degree at most 1 such that $f(0) \equiv a \pmod{p}$ and $f(1) \equiv b \pmod{p}$.

   (c) You are given a prime p and a positive number $n < p$. Show how to find a polynomial $f(x)$ of degree at most n satisfying $f(0) \equiv f(1) \equiv \cdots \equiv f(n-1) \equiv 0 \pmod{p}$ and $f(n) \equiv 1 \pmod{p}$. In other words, the polynomial f should be congruent to zero at the points $x = 0, \ldots, n-1$; at $x = n$ the polynomial should be 1 mod p.
   Hint: Consider $F(x) = (x-0)(x-1)(x-2)\cdots(x-(n-1))$; what can you say about it?

   (d) You are given p and n as before, but now you are also given an index j with $0 \le j \le n$. Show how to find a polynomial $g_j(x)$ of degree at most n satisfying

   $$g_j(i) \equiv \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases} \pmod{p} \qquad \text{for each } i = 0, 1, \ldots, n.$$

   In other words, the polynomial g_j should be congruent to zero at the points $x = 0, \ldots, n$, except that at $x = j$ it should be congruent to 1 mod p.

   (e) You are given a prime p, a number n with $0 < n < p$, and a sequence of values $a_0, a_1, \ldots, a_n \pmod{p}$. Describe an efficient algorithm to find a polynomial $h(x)$ of degree at most n satisfying $h(0) \equiv a_0 \pmod{p}$, $h(1) \equiv a_1 \pmod{p}$, …, $h(n) \equiv a_n \pmod{p}$.
   Hint: What can you say about the polynomial $3g_0(x) + 7g_1(x)$, where $g_0(x), g_1(x)$ are as defined in part (d)? Does this give you any ideas?

Polynomials

2. **(2/2/2/2/2) Representing Polynomials**

   Let f be a polynomial of degree at most d. The *coefficient representation* of f is the sequence $(a_0, a_1, \ldots, a_d)$ of coefficients of f. A *point-value representation* of f is a collection $\{(x_1, f(x_1)), (x_2, f(x_2)), \ldots, (x_t, f(x_t))\}$ of the values of f at any t points $x_1, x_2, \ldots, x_t$, where $t \ge d+1$. (Recall from Lecture Note 7 that a polynomial of degree d is completely determined by its values at any $d+1$ points. Note that t may be greater than $d+1$, so more points than necessary may be given.)

   In the following questions, let f and g by any two real polynomials of degree at most d.

(a) What is the maximum degree of the product polynomial fg?

(b) Given coefficient representations of f and g, explain how to compute the coefficient representation of fg using $O(d^2)$ arithmetic operations (additions/subtractions/multiplications/divisions) over real numbers.

(c) Now suppose that f and g are specified by point-value representations at t points for some $t \geq d+1$, i.e., f is specified as $(x_1, f(x_1)), (x_2, f(x_2)), \ldots, (x_t, f(x_t))$, and g as $(x_1, g(x_1)), (x_2, g(x_2)), \ldots, (x_t, g(x_t))$. With a suitable value of t (which you should specify), show how to compute a point-value representation of fg using only $O(d)$ arithmetic operations.

(d) Suppose that polynomial g divides polynomial f, and that f, g are given in point-value representation as in part (c) with $t = d+1$. Show how to compute a point-value representation for the quotient f/g using $O(d)$ arithmetic operations, and justify your algorithm carefully.

(e) Suppose you are given f in coefficient representation, and you want to compute a point-value representation for f at $t = d+1$ points. Show how to do this using $O(d^2)$ arithmetic operations. [HINT: Show how to evaluate f at one point using $O(d)$ operations; to do this, consider writing f in the form $f(x) = a_0 + xh(x)$, where h is a polynomial of degree at most $d-1$, and iterating.]

Error Correcting Codes

3. **(5/5) Error-correcting codes: an optimization**

In class, we saw an error-correcting code where the n message packets $m_1, \ldots, m_n$ are encoded to the $n+k$ encoded packets $c_1, \ldots, c_{n+k}$ by setting $P(x) = m_n x^{n-1} + \cdots + m_2 x + m_1$, then defining $c_i = P(i)$ (all this is in $GF(q)$, where q is prime and larger than $n+k$, so each packet is a number in the range $0 \ldots q-1$). However, one possible criticism of this error-correcting code is that decoding always requires a Lagrange interpolation step, even if no packets are lost.

(a) In this part, you will develop a scheme that addresses this criticism. Let's preserve the basic approach where $c_i = Q(i)$ (for $i = 1, 2, \ldots, n+k$), for some appropriately chosen polynomial $Q(x)$ which encodes the entire message, and which has degree at most $n-1$. (As before, we'll work in $GF(q)$, where $q > n+k$ and q is prime.) At the same time, let's ensure $c_1 = m_1$, $c_2 = m_2$, ..., $c_n = m_n$, so that if no packets are lost, we can just use the first n encoded packets to immediately read off the message. Describe how to choose $Q(x)$ with this desired property, given $m_1, \ldots, m_n$. In other words, describe an efficient algorithm we can use for encoding.

(b) For your scheme from part 1, if some packets are lost, the recipient can use Lagrange interpolation to recover $Q(x)$. Describe how the recipient could recover $m_1, \ldots, m_n$ from $Q(x)$.

4. **(5/5) List decoding**

(a) Consider a n character message encoded into m characters over the field $GF(p)$ using polynomials. Consider that one recieves $n-1$ of the m packets. Give a method to find a list of size at most p of possible messages. Your running time must be polynomial in terms of p, m, and n.

(b) Consider a n character message encoded into $m = n+2k$ characters over the field $GF(p)$ using polynomials. Consider that $k+1$ of the m received packets are corrupted. Give a method to find a list of possible messages which contains the original message. What is the size of the list for your scheme. It should be a small polynomial in p. Your running time must be polynomial in terms of p, m, and n.

5. **(3/3/4) Reed-Solomon and Reliable Computation**
In this question, we will see how error correction can help with faulty computations. Let us first establish two useful facts.

a) For a communication system that uses Reed-Solomon codes, what is the minimum number of additional packets to transmit for an intended message of n packets, knowing that the communication channel will corrupt at most a fraction $0 \leq f < \frac{1}{2}$ of the packets? (e.g. If $f = \frac{1}{4}$, that means that 3 out of 4 transmitted packets will be received flawlessly, but one quarter of them might have their contents corrupted in an arbitrary manner.)

b) 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.

c) 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. More specifically, you need to define functions $E : GF(p)^n \to GF(p)^m$ and $D : GF(p)^m \to GF(p)^n$ such that $D(M(E(a_1, ..., a_n), E(b_1, ..., b_n)) = (a_1 + b_1, ..., a_n + b_n)$, even if M makes errors on up to $1/4$ of its additions.

Secret Sharing

6. **(3/3/4) Secret Sharing Pirate**

After a long and illustrious career as a buccaneer, Captain Flint passed away in the year 1754. He had split the gold accumulated over years of terrorizing ships into two batches, and just before his death he told his five faithful pirates the locations of the the two batches using a secret sharing scheme.

The captain chose polynomials $P(x)$ and $Q(x)$ over $GF(7)$, of degrees 1 and 2 respectively, with the secrets being the values $P(0)$ and $Q(0)$. For $1 \leq i \leq 5$, Pirate i received the two numbers $P(i)$ and $Q(i)$, but was not told which was which. The pirates cursed the Captain as they could not figure out how to recover the secrets, and the treasure lay undiscovered for many years.

On the eve of the 10th anniversary of the Captain's demise, the pirates captured a small vessel and encountered Monsieur Lagrange (who, having forsaken the ennui of land-life in favor of the vicissitudes of the life on the high seas, was himself an aspiring pirate). Lagrange offered his mathematical prowess in solving the pirates' problems, in exchange for a share of the treasure.

The secret shares received by the five pirates were $\{0,5\},\{1,4\},\{3,4\},\{0,4\}$ and $\{0,3\}$ respectively. (So, for example, Pirate 2's share was $\{1,4\}$, meaning that either $P(2)=1$ and $Q(2)=4$, or $P(2)=4$ and $Q(4)=1$.) Trace the following steps to see how M. Lagrange helped the pirates to solve the mystery.

(a) Find $P(0)+Q(0)$.

(b) Find $P(0)Q(0)$.

(c) Using parts (a) and (b), find the two secrets $P(0)$ and $Q(0)$ that were hidden by Captain Flint, assuming that $P(0)<Q(0)$.

Counting

7. **(3/3/4) Counting Subsets**

Consider the set S of all (possibly infinite) subsets of $\mathbb{N}$.

(a) Show that there is a bijection between S and $T=\{f:\mathbb{N}\to\{0,1\}\}$ (the set of all functions that map each natural number to 0 or 1).

(b) Prove or disprove: S is countable.

(c) Say that a function $f:\mathbb{N}\to\{0,1\}$ has *finite support* if it is non-zero on only a finite set of inputs. Let F denote the set of functions $f:\mathbb{N}\to\{0,1\}$ with finite support.
Prove that F is countably infinite.

8. **(2/2/2/2/2) More Countability**

Given:

- A is a countable set, non-empty set. Forall $i\in A$, S_i is an uncountable set.
- B is an uncountable set. Forall $i\in B$, Q_i is a countable set.

For each of the following, decide if the expression is "Always Countable", "Always Uncountable", "Sometimes Countable, Sometimes Uncountable."

For the "Always" cases, prove your claim. For the "Sometimes" case, provide two examples – one where the expression is countable, and one where the expression is uncountable.

(a) $\cup_{i\in A}S_i$

(b) $\cap_{i\in A}S_i$

(c) $\cup_{i\in B}Q_i$

(d) $\cap_{i\in B}Q_i$

(e) $A\cap B$