EECS 70 Discrete Mathematics and Probability Theory
Spring 2013 Anant Sahai

HW 6

Due Mar 4

1. $d+2$ points vs. a polynomial of degree d

1. Given 3 points $(0,1)$, $(1,1)$, and $(2,3)$, use Lagrange interpolation to construct the degree-2 polynomial which goes through these points.
2. Given 4 points $(0,1)$, $(1,1)$, $(2,3)$, and $(-1,3)$, does there exist a degree-2 polynomial which goes through these points? If yes, find the polynomial; if no, explain why none exists.
3. Given 4 points $(0,1)$, $(1,1)$, $(2,3)$, and $(-1,0)$, does there exist a degree-2 polynomial which goes through these points? If yes, find the polynomial; if no, explain why none exists.
4. Design a machine (i.e. give the pseudocode for an algorithm) with the following function: given four points $(x_1,y_1),(x_2,y_2),(x_3,y_3),(x_4,y_4)$ with all the x_i distinct, the machine outputs YES if there exists a polynomial $p(x)$ of degree at most 2 such that $p(x_i)=y_i$ for all i; otherwise, it outputs NO.

2. Working with polynomials

1. List all possible functions from GF(3) to itself. (Hint: think about 3 digit numbers in base 3).
2. List all degree-0 polynomials with coefficients from GF(3).
3. List all degree-1 polynomials with coefficients from GF(3).
4. List all degree-2 polynomials with coefficients from GF(3).
5. Treating the polynomials in parts (2)-(4) as functions from GF(3) to itself, establish a correspondence between the functions you listed in part (1) and the polynomials identified so far. Comment.
6. Consider the polynomial x^3 as a function from GF(3) to itself. Is this a new function? If not, which polynomial from parts (2)-(4) is it?
7. Give a rule that transforms an arbitrary degree polynomial with coefficients from GF(3) into an equivalent polynomial of degree 2 or lower. Equivalent in the sense that they are the same functions on GF(3) to itself.
8. Is it possible to multiply two nonzero polynomials together (with coefficients from GF(3)) and get a polynomial that is equivalent to the 0 polynomial? If so, give an example.
9. Suppose I take arbitrary polynomials with coefficients from GF(3) and formally mod them by x^2+1. How many different results can I get? List all the "remainders" that are possible.
10. Take two nonzero degree-1 polynomials with coefficients drawn from GF(3) and multiply them together. Do you think you can get a result that is divisible by x^2+1? Explain.
11. Can you generalize Euclid's algorithm to compute the GCD of polynomials?
12. Can you generalize the E-GCD in the same way that you generalized the GCD?
13. Do you think you could use the result of the previous part to compute multiplicative inverses for polynomials mod x^2+1 ?

14. Comment on what you think this means for viewing the degree-1 polynomials with coefficients drawn from GF(3) as a little universe of "numbers." Can you add them mod x^2+1? Subtract them? Can you multiply them mod x^2+1? Divide by nonzero elements?

Preview: Any similarities that you've noticed between polynomials and numbers is not coincidental. There are some very deep mathematical properties that connect them. Those interested are encouraged to take abstract algebra, Math (H)113.

3. How many errors?

Suppose that the message we want to send consists of 10 numbers. We find a polynomial of degree 9 which goes through all these points and evaluate it on 25 points. How many errors can we recover from?

4. Linearity of Reed-Solomon codes

In this problem, you will verify that Reed-Solomon codes are a linear function of the messages.

1. Verify that the all zero-message will be encoded as all zeros.
2. Verify that if $m' = \alpha m$, then the encoding of m' is just α times the encoding of m.
3. Verify that if $m'' = m + m'$, then the encoding of m'' is the sum of the encodings of the other two.

5. Again, find the liar

1. Let $q(x)$ be a degree-2 polynomial and suppose there are 4 people where the i-th person knows the value of $q(i)$ (you know who is the first person, who is the second person, and so on). However, there is one liar among them and the liar always tells a wrong value of $q(i)$. Can you use your machine designed in the previous question to find out who is the liar? Explain your answer.
2. Everything is the same (i.e. $q(x)$ is still a degree 2 polynomial and there is still one liar) except that there are 5 people. Can you use the machine to find out who is the liar? Explain your answer.
3. Suppose you have a better machine with the following function: given $d+2$ points with all the x_i distinct, the machine outputs YES if there exists a polynomial $p(x)$ of degree at most d such that $p(x_i) = y_i$; otherwise, it outputs NO. Now, $r(x)$ is a degree d polynomial and there are $d+3$ people where the i-th person knows the value of $r(i)$. There is still one liar among them. Can you use the machine to find out who is the liar? Explain your answer.
4. Let us try a very simple example with $d = 1$. Use the machine to find out who is the liar if the people tell you $(1,3),(2,2),(3,2),(4,0)$.
5. Given d, how many times do you need to use the machine (call the function) in the worst case? (You will learn a more efficient approach very soon.)

6. Construct a message

In this question we will go through an example of error-correcting codes. Since we will do this by hand, the message we will send is going to be short, consisting of $n = 3$ numbers, each modulo 5, and the number of errors will be $k = 1$.

Your task is to construct the message. Let $a_0 = 4$ and $a_1 = 3$, $a_2 = 2$; then use the polynomial interpolation formula to construct a polynomial $P(x)$ of degree 2 (remember that all arithmetic is mod 5) so that $P(0) = a_0$, $P(1) = a_1$, and $P(2) = a_2$; then extend the message to length $N + 2k$ by adding $P(3)$ and $P(4)$. What is the polynomial $P(x)$ and what are $P(3)$ and $P(4)$?

7. 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 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 are in agreeance.
- The message must remain unknown to everyone (except Gandalf) if not enough members of the party are in agreeance.
- If only the members of one race are in agreeance, the message remains a secret.
- If all the members of one race are in agreeance 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).

8. Multiplying Polynomials Quickly? Almost...

Suppose we have two degree n polynomials, where $n > 1$:

$$P(x) = \sum_{i=0}^{n} a_i x^i$$

$$Q(x) = \sum_{i=0}^{n} b_i x^i$$

1. Multiplying P and Q will result in a $2n$ degree polynomial, call it R.

$$R(x) = \sum_{i=0}^{2n} c_i x^i$$

In terms of the a_i's and b_i's, write down a formula for the c_i's. (Hint: work this out for very small n and spot the pattern.)

2. To calculate each c_i, how many multiplications and additions does it take? Your answer should depend on i and n.
3. How many multiplications and additions does it take to compute all the coefficients of R (i.e. all of the c_i's)? Your answer should only depend on n.
4. How many multiplications and additions does it take to evaluate P or Q at a given x value (using the above formula, without exponentiation by squaring)? Your answer should depend only on n.
5. We know that a degree-n polynomial is completely determined by $n+1$ points. This gives us an alternative algorithm for polynomial multiplication:
   (a) First we evaluate both P and Q at $2n+1$ points.
   (b) From this, we can determine the value of R at those $2n+1$ points (i.e. $R(x) = P(x)Q(x)$).
   (c) Finally, we use Lagrange interpolation to determine the coefficients of R.

   How many additions and multiplications does it take to individually do step 1, step 2, and step 3? How many does it take for the entire algorithm?
6. Is this faster or slower than the algorithm given in part 3 if n is very large?
7. Suppose we write out the polynomials P and Q in the following way:

$$P(x) = a_0 + x(a_1 + x(a_2 + \cdots + x(a_{n-1} + a_n x)\ldots))$$

$$Q(x) = b_0 + x(b_1 + x(b_2 + \cdots + x(b_{n-1} + b_n x)\ldots))$$

   Now how many multiplications and additions does it take to evaluate P or Q at a given x value?
8. Repeat the analysis you did in part 5 with this new algorithm for evaluating polynomials.
9. Is this new algorithm faster or slower for very large n?

Preview: In CS 170 (and EE 120, 123), you will learn a much faster algorithm for multiplying polynomials known as the Fast Fourier Transform. The idea behind the FFT is very similar to the method outlined in part 5, but utilizes a very clever trick for speeding up both the evaluation and interpolation step.