CSM 70 — Discrete Mathematics and Probability Theory

Spring 2016 — Error Correcting, Counting?

Worksheet 6

Key Terms

Erasure Errors	Berlekamp Welsh
General Errors	Countably infinite
Solomon Reed Codes	Halting

Goal: Alice wants to send Bob n packets over a noisy channel, where each packet is a number modulo q

Polynomial Facts:

1) There is a unique polynomial of degree n-1 such that $P(i) = m_i$ for each packet $m_1, \dots, m_n$
2) To account for errors we send $c_1=P(1), \dots, c_{n+j}=P(n+j)$
3) If polynomial P(x) has degree n - 1 then we can uniquely reconstruct it from any n distinct points.
4) If a polynomial P(x) has degree n - 1 then it can be uniquely described by its n coefficients

I. <u>Erasure Errors</u>

We want to send n packets and we know that k packets could get lost.

$$\boxed{3}\ \boxed{1}\ \boxed{5}\ \boxed{0} \longrightarrow \boxed{}\ \boxed{1}\ \boxed{5}\ \boxed{}$$

How many more points does Alice need to send to account for k possible errors? k

What degree will the resulting polynomial be? n-1

How large should q be if Alice is sending n packets with k erasure errors, where each packet has b bits?

Modulus should be larger than n+k and larger than 2^b and be prime

What would happen if Alice instead send n + k - 1? Why will Bob be unable to recover the message?

Bob will receive n - 1 distinct points and needs to reconstruct a polynomial of degree n - 1. By Fact #3 this is impossible. There are q polynomials of at most degree n - 1 in GF(q) that go through the n - 1 points that Alice sent.

Exercises

1. Suppose A = 1, B = 2, C = 3, D = 4, and E = 5. Assume we want to send a message of length 3. Recover the lost part of the message, or explain why it can not be done.
    a. C_AA
    P(0) = 3, P(2) = 1, P(3) = 1. Once we interpolate the polynomial over mod 7, as E is 5, we get 3x^2 + 3x + 3. Now, once we evaluate this at 1, we get 2. So, in the end, it's CBAA.
    b. CE_ _
    Impossible. In order to get the original degree 2 polynomial, we need at least 3 > 2 points.

2. Suppose we want to send n packets, and we know p = 20% of the packets will be erased. How many extra packets should we send? What happens if p increases (say to 90%)?

    We want to have (1-p)(n + k) = n, where k is the number of additional packets we send. Solving for k, we get (n / (1 - p)) - n. When p is large, we have to send many times the number of original packets.
    (fraction packets not erased)*(how many packets are sent) = (number packets in original message)

II. <u>General Errors (Berlekamp and Welch)</u>

Now instead of losing packets, we know that k packets are corrupted. Furthermore, we do not know which k packets are changed. Instead of sending k additional packets, we will send an additional 2k.

$$\boxed{3}\ \boxed{1}\ \boxed{5}\ \boxed{0} \longrightarrow \boxed{4}\ \boxed{1}\ \boxed{5}\ \boxed{1}$$

Solomon-Reed Codes

1. Identical to erasure errors: Alice creates n - 1 degree polynomial P(x).
    $P(x) = p_0 + p_1x + \dots + p_{n-1}x^n$
2. Alice sends P(1), …, P(n + 2k)
3. Bob receives R(1), …, R(n + 2k)

For how many points does R(x) = P(x)? n + k

True or false: P(x) is the unique degree n - 1 polynomial that goes through at least n + k of the received points.

Write the matrix view of encoding the points P(1), …, P(n + 2k)

$$\begin{bmatrix} P(1) \\ P(2) \\ P(3) \\ \vdots \\ P(n+2k) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1^2 & \cdot & 1 \\ 1 & 2 & 2^2 & \cdot & 2^{n-1} \\ 1 & 3 & 3^2 & \cdot & 3^{n-1} \\ \vdots & . & \vdots & \vdots & \\ 1 & (n+2k) & (n+2k)^2 & (n+2k)^{n-1} & \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_{n-1} \end{bmatrix}$$

Berlekamp Welch

How do we find the original polynomial P(x)?

Suppose that $m_1, \ldots, m_k$ are the corrupted packets. Let $E(x) = (x - m_1) \ldots (x - m_k)$

Then $P(i)\,E(i) = r_i\,E(i)$ for any i greater than 1 and less than n + 2k. Why?

Case 1: i is corrupted

$E(i) = 0 \rightarrow 0 = 0$

Case 2: i is not corrupted

$P(i) = ri \rightarrow P(i)\,E(i) = r_i\,E(i)$

Let $Q(i) = P(i)\,E(i)$

So we have $Q(i)= P(i)\,E(i) = r_i\,E(i)$ where $1 \le i \le 2k + n$

What degree is Q(i)? n + k - 1

How many coefficients do we need to describe Q(i)? n + k

What degree is(i)? k

How many unknown coefficients do we need to describe E(i)? k (we know that the first coefficient has to be 1)

We can write $Q(i) = r_i\,E(i)$ for every i that is $1 \le i \le 2k + n$.

How many equations do we have? How many unknowns? n + 2k

Once we have the above described equations, how do we determine what P(i) is?

Solve the equations to get the coefficients for E(i) and Q(i). Then divide E(i)/Q(i) to get P(i).

Exercises

1) Alice sends Bob a message of length 3 on the Galois Field of 5 (modular space of mod 5). Bob receives the following message: (3, 2, 1, 1, 1). Assuming that Alice is sending messages using the proper general error message sending scheme, set up the linear equations that, when solved, give you the Q(x) and E(x) needed to find the original P(x).

Set up 5 equations for the five values of x that we have, such that $Q(x) \bmod 5 = r_x(x\text{-}b) \bmod 5$ where $Q(x)= a_3{*}x_i^3 + a_2{*}x_i^2 + a_1{*}x_i + a_0$

(r_i = ith received number)

In the form, for $x = x_i$, $a_3{*}x_i^3 + a_2{*}x_i^2 + a_1{*}x_i + a_0 = r_i(x - b)$

x=1, $(a_3 + a_2 + a_1 + a_0) \bmod 5 = 3(1 - b) \bmod 5$

x=2, $(3a_3 + 4a_2 + 2a_1 + a_0) \bmod 5 = 2(2 - b) \bmod 5$

x=3, $2a_3 + 4a_2 + 3a_1 + a_0 \bmod 5 = 1(3 - b) \bmod 5$

x=4, $4a_3 + 1a_2 + 4a_1 + a_0 \bmod 5 = 1(4 - b) \bmod 5$

x=5, $0 + 0 + 0 + a_0 \bmod 5 = 1(0 - b) \bmod 5$

Solving for these equations, which you should give your students after setting up the above, you get the following:

b = 3, note that this is the index of the error.

$a_3 = 1$

$a_2 = 3$

$a_1 = 3$

$a_0 = 2$

What is the encoded message that Alice actually sent? What was the original message? Which packet(s) were corrupted?

Now we have P(x) = Q(x)/E(x). Plug in the coefficients for Q and E and divide (using polynomial long division) and you get P(x) = (x^2) + x + 1.
Using the P(x) that we found, we plug in the values 1, 2, 3, 4, 5 to find the encoded 5 packet message. P(1) = 3, P(2) = 2, P(3) = 3, P(4) = 1, P(5) = 1. The original message is the first 3 P(1), P(2), P(3). Using the value of b, we know that the 3rd packet was corrupted, which we can confirm in the message that was received.

2) You want to send a super secret message consisting of 10 packets to the space station through some astronauts. You're afraid that some malicious spy people are going to tell the wrong message and make the space station go spiraling out of orbit. Assuming that up to 5 of the astronauts are malicious, design a scheme so that the group of astronauts (including the malicious ones) still find the correct message that you want to send. You can send any number of astronauts, but try to make the number that you have to send as small as possible.

We can use the scheme provided in the notes for general errors. If there are up to 5 malicious astronauts, then we have up to 5 general errors in our transmission. To account for those errors and to send the original message we have to send at least the length of the original message plus the twice the possible number of general errors. Thus, we have to send 20 astronauts or more. Create a polynomial of degree 9 (which is the length of the message minus 1), such that P(0), P(1), … P(10) is the original message, and give each of the 20 astronauts unique points on that polynomial. Then use the methods we used in the previous question to find the message.

3) 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 erasure errors can we recover from? How about general errors?
Erasure: n + k = # points to send
    10 + k = 25
    k = 15 erasure errors
General: n + 2k = # points to send
    10 + 2k = 25
    k = 7 general errors.

III. <u>Countability</u>

1. What does it mean for a set to be countably infinite?
There is a bijection between the set and the natural numbers (or any other countable set)

2. Does N and Z^+ have the same cardinality? Does adding one element change cardinality?
Do the hotel argument. Just take each person starting at some arbitrary point k, and slide them over by one. In other words, map everybody at some point p >= k to p+1, and then slide the newcomer into spot k.

No, adding in one more point did not change the cardinality of the set.

Cantor-Bernstein Theorem: Suppose there is an injective function from set A to set B and there is an injective function from set B to set A. Then there is a bijection between A and B.
This is used when proving that rational numbers are countable.
We know that $|N| \leq |Q|$ because every natural number is a rational number.
Just need to show that $|Q| \leq |N|$. Do the spiral proof.

Exercises
1. Are these sets countably infinite/ uncountable infinite/ finite? If finite, what is the order of the set?

A. Finite bit strings of length n.
Finite. There are 2 choices (0 or 1) for each bit, and n bits, so there are 2 * 2 * … = 2^n such bit strings

B. All finite bit strings of length 1 to n.
Finite. By part (1), there are 2^1 bit strings of length 1, 2^2 of length 2, etc. Thus, there are $2^1 + 2^2 + \ldots + 2^n = 2^{n+1} - 1$.

C. All finite bit strings
Countably infinite. We can list these strings as follows: {0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, 110, 111, 0000, …}. This gives us a bijection between the (countable) natural numbers, so these are countably infinite.

D. All infinite bit strings
Uncountably infinite. We can construct a bijection between this set and the set of real numbers between 0 and 1. We can represent these real numbers using binary—they are of the form 0.0110001010110…. The bijection between
By diagonalization, the set of real numbers between 0 and 1 is uncountably infinite; therefore, so is this set.

E. All finite or infinite bit strings.
Uncountably infinite. This is the union of a countably infinite set (part 3) and an uncountably infinite set (part 4), so it is uncountably infinite.

2. Find a bijection between N and the set of all integers congruent to 1 mod n, for a fixed n.
The set of integers congruent to 1 mod n is $A = \{1 + kn \mid k \in Z\}$. Define $g : Z \rightarrow A$ by $g(x) = 1 + x*n$; this is a bijection because it is clearly one-to-one, and is onto by the definition of A. We can combine this with the bijective mapping $f: N \rightarrow Z$ from the notes, defined by

$$f(x) = \begin{cases} \frac{x}{2} & \text{if } x \text{ is even} \\ \frac{-(x+1)}{2} & \text{if } x \text{ is odd} \end{cases}$$

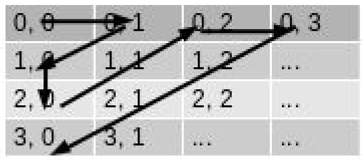
Then $f \circ g$ is a function from N to A, which is a bijection.

3. Every infinite subset of a countable set is countable.

True. Define E as the subset. Define function f where f(1) = min(E), f(k) = kth smallest element of E. We see a bijective mapping clearly exists between f and E. and since the x-values in f are just the natural numbers, there is a mapping between E and N.

4. If A and B are both countable, then AxB is countable.

True. Can draw a bijection where first elem of A maps to 1, first elem of B maps to 2, second elem of A maps to 3, etc. This will include AxB at the end, and because there is a bijection from A to N and B to N, there is a bijection here from AxB to N. There is clearly mapping from AxB to NxN, and NxN to N, so there is a bijective mapping from AxB to N



0, 0	0, 1	0, 2	0, 3
1, 0	1, 1	1, 2	...
2, 0	2, 1	2, 2	...
3, 0	3, 1	...	...

5. Every infinite set that contains an uncountable set is uncountable.

Let A be an uncountable subset of B.
Assume that B is countable.
f: N → B is a bijection.
There must be a subset of N such that f: M → A is a bijection
Then A is countable.
This is a contradiction.
So A must be uncountable.

IIIB. Self Reference

The Halting Problem: Does a given program ever halt when executed on a given input?

$$\texttt{TestHalt(P,x)} = \begin{cases} \text{“yes"}, & \text{if program } P \text{ halts on input } x \\ \text{“no"}, & \text{if program } P \text{ loops on input } x \end{cases}$$

How do we prove that TestHalt doesn't exist? Let's assume that it does, and hope we reach a contradiction.

Define another program:

```
Turing(P):
     if TestHalt(P, P) = "yes" then loop
     else halt
```

What happens when we call Turing(Turing)?

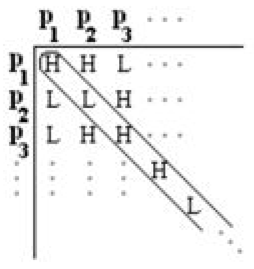
Case 1: It halts

If Turing(Turing) halts then TestHalt(Turing, Turing) must have returned no. But TestHalt(Turing Turing) calls Turing(Turing) and calling Turing(Turing) must loop. But we assumed that Turing(Turing) halted. Contradiction.

Case 2: It loops

This implies that TestHalt(Turing, Turing) returned yes, which by the way that TestHalt is defined implies that Turing halted. But we assumed that Turing(Turing) looped. Contradiction.

How is this just a reformulation of proof by diagonalization?



List all possible programs as rows and columns. The rows are the programs and the columns are the inputs. Turing must be one of the rows, say row n. If entry (n, n) is H then Turing will loop by definition. If entry (n, n) is Ll then Turing will halt by definition. Therefore Turing cannot be on the list of all programs and therefore it does not exist.

Therefore the Halting Problem is unsolvable. We can use this to prove that other problems are also unsolvable. Say we are asked if program M is solvable. To prove it is not, we just need to prove the following claim:

If we can compute program M, **then** we could also compute the halting problem.

This would then prove that M can not exist, since the halting problem is not computable. This amounts to proof by contradiction.

Exercise

Say that we have a program M that decides whether any input program halts as long as it prints out the string "ABC" as the first operation that it carries out. Can such a program exist?

Answer: **No**. Such a program M can **not** exist. We proceed as follows: we show that if such a program existed, the halting problem would be computable.

Consider any program P. If we wanted to decide if P halted, we could simply create a new program P' where P' first prints out "ABC", then proceed to do exactly what P would. However, if M existed, we could determine whether **any** program P halted by converting it to a P' and feeding it into M. This would solve the halting problem- but this is a contradiction, since by diagonalization we can prove that the halting problem is not solvable.

Therefor, M **can not exist!**

IV. **Counting**

How many ways are there to choose 3 cards from a deck? (Order matters)

52 * 51 * 50

How many distinct hands of 3 cards are there? (Order does not matter)

Use the bin analogy: We know that there are 52!/49! ways to choose 3 cards. But this would count (K, J, Q) and (Q, K, J) separately. We want to count them together. So lets create bins to place each set that contains the same cards together. So (K, J, Q) and (Q, K, J) would go to the same bins. How many elements are in each of the bins? Each of the bins is just all of the different ways to permute a set of 3 cards. So the number of elements in each bin is 3!. If we know that there are 52!/49! "sets" in total and that each bin has 3! "sets", then there must be 52!/49!3! bins. Each bin is a distinct permutation.

Rules of Counting:

1) If we make a sequence of choices, and there are n_1 ways to make the first choice, n_2 ways to make the second choice for every way of making the first choice, …, n_k ways to make the kth choice for every way of making all of the choices before it, then the total number of distinct objects we get will be $n_1 * n_2 * \dots * n_k$
2) Assume that the order of the choices does not matter. Let A be the ordered objects and B be the unordered objects. If there is a mapping from A to B (k to 1) then we can count the number of ordered objects and divide by k to get the order of B.

Exercises

1) How many ways are there to order the following words?

MENTORS

7!

there are 7 places to put the M, then 6 remaining places to put the E, then 5 places for the N... down to only 1 remaining place to put the S, so multiply 7 * 6 * 5 * … * 1 = 8!

CORRINA

7!/2!

There are 7! ways to rearrange all the letters. But 7! counts RCORINA AND RCORINA as different permutations, when they are actually the same permutation.

How many permutations are counted doubly? 2! Because there are 2! ways to switch the R's places with each other i.e. If we wanted to place R's in these blanks _CO_INA , there are 2! ways to do it

MISSISSIPPI

11! / (4!4!2!)

2) When Order Matters
You have 15 chairs in a room and there are 9 people. How many different ways can everyone sit down?
15! / 6!
There are 15 places to put the first person, then 14 places to put the second person, 13 places to put the third person, etc… all the way to the last person who has 7 places to sit
Another way to think about this is like the anagram example above. We have 9 unique "letters" and 6 repeats (our empty spaces). We divide by the repeats giving us:
15*14*13*12*11*10*9*8*7 = 15! / 6!

How many ways are there to fill 9 of the 15 chairs? (We don't care who sits in them)
15 choose 9 = 15! / 9!(15-9)!
In this example, we don't care about the uniqueness of each person, so we can just count each person as a "repeat". So like the anagram example we'll divide for every repeat. We have 9 human repeats, and 6 empty space repeats. Hence 15! / (9!6!)

3)
The numbers 1447, 1005, and 1231 have something in common. Each of them is a four digit number that begins with 1 and has two identical digits. How many numbers like this are there?

Case 1: the identical digits are 1
11xy   1x1y   1xy1
Since there can only be two numbers that are identical, x and y cannot be 1 and x != y.
[Possible formats] * [Possible x values] * [Possible y values] = 3 * 9 * 8 = 216
Case 2: identical digits are not 1
1xxy   1xyx   1yxx
[Possible formats] * [Possible x values] * [Possible y values] = 3 * 9 * 8 = 216
216 + 216 = 432