Mathematics 392 – Seminar in Coding Theory
Final Projects
February 27, 2002

General Information

As announced in the course information distributed at the start of the semester, one of the assignments for the class will be a group final project, worth 20% of the course grade. You may work on this project in groups of three or four people (four groups in all). I will leave the composition of the groups up to you; do not feel constrained by the composition of the in-class discussion groups. If you have any difficulty getting a good group together, I would be happy to help, though.

There are five suggested topics outlined in detail below, and several other ideas to consider if none of those look interesting. It would also be possible to design your own topic. If you do want to pursue one of those other ideas or to design a topic of your own, *please see me as soon as possible – certainly well before March 22! – to get my approval and to get some suggestions for how to start the research phase of your work.* The ultimate outcomes of the projects will be:

- A paper of about 10-15 pages on your subject, or for the third topic below, one or more `C++` or Circuit Maker programs and a shorter paper giving documentation for the design and implementation of your encoder and decoder.
- An oral presentation of 30-40 minutes to the class that will cover some or of the same ground as your paper.

To keep the projects moving along smoothly, here are some deadlines:

1) **By Friday, March 22**: Inform me who you will be working with and what topic you have chosen.
2) **During week of April 8**: I would like to meet with all groups outside of class time to discuss your progress. At that time, I would be happy to look over drafts or interim results, etc. Of course, I'm also available during Office Hours whenever you have questions about the projects.
3) **April 22,24,26**: Class days set aside for project presentations. We will schedule who will present which day when we get a bit closer to those dates.
4) **April 29**: (last day of class) Project papers due. This is an absolutely firm deadline. *No extensions will be possible, so please plan your work accordingly.*

Suggested Topics

1. Statistics and Coding: Shannon's Theorem.

This topic would introduce you to an interesting connection between topics we have discussed in this course and material from the Probability and Statistics course (recommended only for students taking both).

The beginnings of both information theory as a part of mathematics and coding theory date from Claude Shannon's 1948 paper "A Mathematical Theory of Communication." For this topic, you would research what Shannon did, and present a proof of his major theoretical result:

Shannon's Theorem. *Consider a binary symmetric channel with bit error probability p, and let $C(p) = 1 + p\log_2(p) + (1-p)\log_2(1-p)$ (called the channel capacity). For all $\epsilon > 0$, no matter how small, and all R with $0 < R < C(p)$ there exist codes C with rate R, for which the probability of incorrectly decoding a single codeword is $< \epsilon$.*

The proof of Shannon's Theorem I know (same as the one in van Lint's book mentioned below) relies on a bit of probability and statistics (the standard deviation of a random variable, and a result called Chebyshev's Inequality), and a considerable amount of cleverness. The main goal of this project would be to work through the details of some particular proof carefully and present them to the class and in your paper. As with many mathematical results, you will find slightly different presentations of the proof and different notations in different sources. *So my advice for the proof itself would be to choose one that you find readable, and work from that rather than trying to compare several different versions.*

Shannon's paper is reprinted in his Collected Papers. A concise but complete proof can also be found in van Lint's text book *An Introduction to Coding Theory*. Both of these are on reserve for our class in the Science Library. Many other coding theory texts also include proofs, and you can probably find a lot of material related to this subject on the Web.

2. An Applied Topic: Reed Solomon Codes and The Compact Disc Audio System.

One of the major successes of coding theory in commercial electronics was the Compact Disc Digital Audio system that effectively replaced the LP as the mainstream medium for recorded music in the 1980's. Because of the sensitivity of the laser pickup used to read information off a CD to dust particles, small scratches, fingerprints, etc. it was quickly realized that some form of error-control coding would be necessary to replay music with the degree of fidelity demanded by serious listeners. So engineers at Philips, the Dutch electronics giant, developed a coding scheme based on interleaving two different Reed-Solomon codes. The goal was to protect against both random and "burst" errors.

The main goal of this project would be to look at the coding methods they used, demonstrate how the interleaving works for both encoding and decoding, and show how both random error and burst error correction is achieved.

Good first sources for this topic are the paper "Reed Solomon Codes and The Compact Disc" by Kees A. Schouhamer Immink from the collection *Reed Solomon Codes and Their Applications*, Steven Wicker and Vijay Bhargava, editors and Chapter 7 of our textbook. See me for copying rights to the article (I want to keep my personal copy of this book available for my use too!) The reference list for this paper contains several other sources you may want to consult, but which are not available in the Holy Cross library. See Mr. Stankus or his assistant in the Science Library *soon* if you want to try to get copies via interlibrary loan.

3. An Exploratory and Computational Project: Greedy Codes

Suppose we want to construct binary codes, or more generally codes over any other field $\mathbf{F}$, of some given length n and some given minimum distance d. One possible approach is to list all the words in $\mathbf{F}^n$ starting from the all-zero word in some order, then to proceed through the list putting the next word from the list into the code whenever the distance between that word and all the words chosen for the code already is $\geq d$. This approach is an example of what is often called a "greedy" algorithm (because we are automatically putting that new word into the code without considering the other later words that might be excluded by that choice). The resulting codes are called "greedy codes", and they have surprisingly good and interesting properties.

Small greedy codes can be found by hand, but to do more extensive examples, you will want to code a GreedyCode program in a high-level language like `C++` and use it to collect some data, look for patterns, etc. So this project will involve an initial programming/data gathering phase, then proceed into a more theoretical phase where you will try to understand your results, prove that things are true, etc.

1) Lexicographic binary greedy codes. One way to order the list of all words of length n is the *lexicographic order*. For binary words, for instance, this corresponds to the usual numerical order on the integers if we think of the binary word as the base-2 expansion of an integer. Binary lexicographic codes are very surprising and interesting. Try examples and see if you can *prove* what you are seeing.
2) Lexicographic greedy codes over other fields. Lex greedy codes over the fields $GF(2^a)$ also have corresponding interesting properties. You can define a lexicographic order on $\mathbf{F}^n$ for other fields $\mathbf{F}$ as long as you have some definite way to associate an integer to each element of $\mathbf{F}$ (for example by way of base-p expansions).
3) Greedy codes for other orderings. In addition to the lexicographic ordering, many other orderings are possible. In particular, there is an ordering called the *graded reverse lexicographic ordering* What happens if you list your n-tuples (possible codewords) in this order, and then do the same greedy algorithm as before? Do these greedy codes have the same properties? Are they better or worse? What about other orders besides the graded reverse lex order? How close to the sphere-packing bound can you get by using greedy codes?

The following paper contains a wealth of information and examples, plus another way to think about greedy codes using ideas from game theory. Because of the nature of this topic though, I will insist that you *not* look at it until after you have spent some time thinking about point 1) above, worked out examples, etc.

Reference

1) Conway, J. and Sloane, N. *Lexicographic Codes: Error-Correcting Codes from Game Theory*, *IEEE Transactions on Information Theory*, **32** (1986), 337-348.

4. A Computer Topic: Implementing Encoders and Decoders

The encoding and decoding functions for linear block codes that we have described mathematically must actually be implemented (in either hardware or software) if coding is to be used in practice. Usually, the decision whether to use a hardware or a software implementation would depend on the need for *speed of processing* – a good hardware implementation would, of course, almost always be faster! For audio disc playback, for instance, speed is vital, so hardware implementation would be the way to go. On the other hand, for processing of data from deep space exploration craft, time would not be so pressing, and a software implementation of decoding might be perfectly acceptable.

For this topic, the goal would be to implement:

1) polynomial division encoding, and
2) Berlekamp-Massey decoding (with Chien search and the Forney error value formula) as in our textbook

for single words using the 2-error correcting Reed-Solomon code $RS(11, 16)$ over $\mathbf{F}_{16}$ ($n = 15, k = 11, d = 5$ over $\mathbf{F}_{16}$). Of course, this will involve computing sums and products of elements of the field, so you will want to start with implementing the field arithmetic – the addition and multiplication operations in $\mathbf{F}_{16}$.

You can either do this in "hardware," i.e. in digital logic circuits designed–and tested! (via Circuit Maker, for instance), or as software (in `C++`, for instance) as you like. *Note: I recommend this topic only for really strong programmers; I do not know a lot of C++ or CircuitMaker, so I will not be able to give much technical assistance there*

The (pre-1980) "bible" of coding theory, *The Theory of Error-Correcting Codes* by MacWilliams and Sloane (on reserve for our class in the Science Library) has a lot of relevant information for either approach. There are also a couple of articles in the collection *Reed Solomon Codes and Their Applications*, Steven Wicker and Vijay Bhargava, editors. that you may want to consult. See me for copying rights (I want to keep my personal copy of this book available for my use too!)

5. Coding theory and Cryptography

Although we have been concentrating on *reliability* issues in this course, the theory of error-correcting codes also has significant connections with *cryptography* – the study of methods for *secret, or secure* communications. For this topic, you would explore some of these connections.

You should start out by presenting the idea of a *public-key* cryptosystem, and include a brief but reasonably detailed discussion of the *RSA public-key system* that is commonly used today for secure communications over the Internet. (That portion of your project will supply some background ideas, but it is not directly connected with error-correcting codes.)

One of the most-studied proposed cryptographic methods that uses error-control coding theory is known as the *McEliece cryptosystem.* The basic idea is this:

- To set up a McEliece system, you first find a large family $\mathcal{F}$ of linear codes with the same blocklength n and dimension k.

- The *private key* (known only to the person or persons owning the system) consists of three items:
  + First, the generator matrix G for one of the codes in the family $\mathcal{F}$. To make the system practical, the code described by G should be known (by the owner, at least) to have an efficient decoding algorithm. (McEliece originally proposed a family of codes called "classical Goppa codes" – generalizations of Reed-Solomon codes, that can all be decoded by methods related to the Berlekamp-Massey algorithm we will study)
  + Second, a random invertible $k \times k$ "scrambler matrix" S and
  + Third, a random $n \times n$ permutation matrix P.
- The *public key* used to send messages to the owners of the system consists of the $k \times n$ "scrambled up" generator matrix $G' = SGP$, plus the number $t = \lfloor \frac{d-1}{2} \rfloor$ giving the error-correction capacity of the private key code.
- *Encryption* of a message word m consists of mapping m to $x = mG' + e$, where e is a randomly chosen error of weight $\leq t$. (Note: the basic form of the McEliece system in effect ignores the possibility of additional errors in transmission of the encrypted form of the message, but that could easily be taken care of too; can you see how?)
- *Decryption* of x by the owner consists of computing $xP^{-1} = mG'P^{-1} + eP^{-1} = mSG + eP^{-1}$, followed by application of the efficient decoding algorithm to recover mS. Then the owner can use S^{-1} to recover m.

The security of this system is supposed to lie in the choice of G out of a big family of possible generator matrices, plus the choices of S and P, out of big collections of such matrices. Moreover, the fast decoding method for the secret code should depend in some way on which G is used. Without knowledge of the private key, the idea is that finding the right decoding algorithm, plus the right S and P should be prohibitively expensive in time and computational effort.

Your paper and presentation would go into this in more detail, and also discuss some of the attacks that have been developed against cryptosystems of this form. How secure is this method, really??

Important Note: If you want to try this topic, you will need to let me know *well in advance* of March 22. You will need to leave plenty of time for gathering sources via interlibrary loan. Many of the articles you will want to look at are in proceedings of the EUROCRYPT conferences, which are not in the Holy Cross library collection.

Additional Ideas

Other possible ideas for topics:

6. More of the mathematics behind the binary Golay codes
7. Convolutional codes and symbolic dynamics
8. Coding theory and biology (DNA)