Math 5251: Error Correcting Codes & Finite Fields: Final Exam

*Due by **noon on Thursday, 12/21/2006** in my mailbox in Vincent 107, in the "Visiting Faculty" section. My mailbox was moved during the semester (don't ask...) so you might have to search for it.*

Directions. This is an open book, open notes, open library, open Internet exam, but you may *not* collaborate with others; I am the only person you are allowed to consult. Standard academic rules apply; if you use something from a source other than our textbook or lectures, you must cite it or risk losing credit. You are expected to write your own solutions, so under no circumstances would it be acceptable, say, to copy a large chunk of text from a web page as (any part of) your solution. Having gotten all of that legelese out, you're all trustworthy and none of this should be an issue.

Also remember my warning in class: there are a number of excellent coding theory websites, but some of them (including Wikipedia) include things that are only true with different definitions than ours, or even downright false. In the case of competing definitions, go with our textbook. Note that nothing on this exam requires any external source, so you can avoid the Internet and this issue entirely if you wish.

The guidelines for homework problems apply on this exam. Answers to problems should include any computations necessary to get the final answer. To receive full credit, you must also explain what you've done and why you did it. You should write in complete sentences with (reasonably) correct grammar.

(1) (15 Points) Determine whether or not there exists an instantanous code with the following alphabets and codeword lengths. For any code that exists, exhibit one. For each code that cannot exist, explain why.
   (a) $\Sigma = \{0, 1\}$, lengths 1, 2, 3, 4, 4, 5
   (b) $\Sigma = \{0, 1, 2\}$, lengths 1, 1, 2, 2, 3, 3, 3
   (c) $\Sigma = \{0, 1, 2, 3\}$, lengths 1, 1, 2, 2, 4

(2) (10 Points) Construct a binary code with at least 8 codewords which is uniquely decipherable but *not* instantaneous. Explain why it satisfies these conditions.[1]

(3) (15 Points) Consider a memoryless source which emits the letters $\{A, B, C, D, E, F\}$ with probabilities $\{0.1, 0.2, 0.3, 0.1, 0.2, 0.1\}$, respectively.
   (a) Compute the entropy of this source.
   (b) Construct a binary Huffman code for this source; make sure that your code satisfies the bound in the Noiseless Coding Theorem.

(4) (20 Points) Consider the following matrix, whose rowspace is a linear $[n, k]$ code C over $\mathbb{F}_2$.

$$G = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

   (a) Determine n and k.
   (b) Find a generating matrix in standard form for C, and determine the encoding of a sourceword $abc \in \mathbb{F}_2^3$.
   (c) Find a check matrix for C and determine the minimum distance of the code. Find e, the number of errors the code can correct.
   (d) Compute a syndrome lookup table for use in decoding, under the assumption that up to e errors might occur during transmission.

[1] In practice we're not interested in a code like this, but after the confusion on the first exam I promised this issue would return on the Final. See the solutions to Exam 1 for more.

(5) (15 Points) Given two codes C and D, there are many ways to combine them into a third code E. One fairly simple method is to define a set of codewords

$$\{cd \,|\, c \in C,\, d \in D\}$$

where cd just means concatenation: write down a codeword from C followed by a codeword from d. In this problem we'll use the notation $C + D$ for this construction.

(a) If C_1 and C_2 have minimum distances d_1 and d_2, respectively, find the minimum distance of the code $C_1 + C_2$.

(b) If C_1 and C_2 are $[n_1, k_1]$ and $[n_2, k_2]$ linear codes over $\mathbb{F}_q$, respectively, find the parameters $[n, k]$ for the code $C_1 + C_2$. Given generating and check matrices for C_1 and C_2, how would you construct a generating matrix and check matrix for $C_1 + C_2$?

(6) (5 Points) Prove that any binary Hamming Code (as described in Chapter 17) is perfect.

(7) (20 Points) Suppose NASA sends a spacecraft to Saturn to take high-quality pictures of its rings in full 24-bit color. (That means each pixel in the picture could be any of $2^{24} = 16,777,216$ colors.) They anticipate a lot of transmission errors between Saturn and the Earth. Construct a Reed Solomon code with at least 2^{24} codewords which can correct 5 errors in a codeword.

To "construct" the code you should find a polynomial $g(x)$ **in factored form** which generates the appropriate code, find the parameters $[n, k]$ of your code, and write down the generating matrix. Your matrix might be large, but should still fit on a piece of paper! I'll post a list of primitive roots on the course website if it helps you. You may use Mathematica or Maple to multiply out your generating polynomial if you wish. In Mathematica, for example, you could compute $(x - a) \cdots (x - z) \bmod n$ with the command

`PolynomialMod[`$(x - a) \cdots (x - z)$`, `n`]`

Stop by my office or email me if you need help.

An aside, not worth any points: I mentioned once in class that there are ways to deal with large burst errors. Here is one method. Suppose the spacecraft takes a digital photograph with 1000 rows and 1000 columns for a total of $1000^2 = 1$ million pixels. The easiest way to transmit the picture back to earth is to use your codewords to send it pixel by pixel: the first thousand transmitted codewords represent the first row, the second thousand represent the second row, and so on, until 1 million words have been sent. However, if an asteroid gets in the way, we could lose thousands of pixels.

As an alternative, suppose we send the first digits of all 1 million pixels; then send the second digits of all 1 million pixels, and so on. None of our codewords are complete until the last digits are sent, at which point we can reassemble everything. (As a short example, suppose I want to send the codewords 147, 258 and 369 to you; with this new method I would send 1, 2, 3, 4, 5, 6, 7, 8, 9; until you get the 7, 8 and 9, you don't know what any of your received codewords will be.)

This "shuffling" of the codeword digits might seem bizarre, but with this method we can now theoretically **correct 5 million errors**! If we have a 5 million digit burst error, this transmission method will spread them out so that each of the million pixels has 5 errors, which your code can correct. Cool! Food for thought: why is this method more appropriate for things like satellite transmissions than for cell phones or computer networks?