Algorithms in the Real World (296.3), Spring 2011
Assignment #2 Due: March 16, 2011

Complete all problems. You are not permitted to look at solutions of previous year assignments. You should do the programming assignment individually. Please submit the following as a single .zip file: 1) your code 2) readme containing instructions on compiliation and 3) *importantly* a PDF containing screenshots of your code working for at least 3 reasonable inputs.

Building Your Own Error Correcting Code

In this assignment you will build your own error-correcting code based on the discrete Fourier transform (DFT). Your coding scheme will work as follows. You will code blocks consisting of k message bytes. Each byte represents a symbol from the finite field $GF(2^8)$. The k symbols in a message can be thought of as the coefficients in a degree $k-1$ polynomial. Your schemes will produce n-symbol codewords. To keep things simple, we will only correct *erasures*, i.e., missing symbols, rather than corrections.

Problem 1: Implementing $GF(2^8)$ operations [10pt]

Your first task is to implement addition, multiplication, and inverses over $GF(2^8)$. Your addition and multiplication functions should take as inputs two bytes representing symbols a and b from $GF(2^8)$ and produce an output byte $c = a + b$ or $c = a * b$. Your inverse functions should take as input a single byte a and produce an output byte $-a$ or a^{-1}. Your implementation should be much faster, performing all operations using table look-up.

Problem 2: Evaluating a degree $k-1$ polynomial at n locations [10pt]

Your second task is to evaluate the degree $k-1$ polynomial $m_{k-1}x^{k-1} + \cdots + m_1 x + m_0$ (where $m_{k-1}, \ldots, m_0$ represent the message symbols) at n locations using the discrete Fourier transform. These n locations should be the n distinct powers of a generator α for $GF(2^8)$. Your output will be an n-symbol codeword $c_{n-1}, ..., c_0$. You should use the simple matrix-vector multiplication algorithm that takes $O(nk)$ time to implement this step, which is called the Discrete Fourier Transform (DFT). Your implementation should work for any $k \leq n < 2^8$. Remember that all of your operations must be performed over $GF(2^8)$.

Problem 3: Fast Fourier Transform [10pt]

The fast Fourier transform (FFT) is a faster algorithm for performing the same matrix-vector multiplication, optimized for the particular matrix used in the DFT. The FFT runs in $O(n \log n)$ time. Reimplement the second task using the FFT for $n = 256$. Unfortunately, for this part of the assignment you will not be able to use $GF(2^8)$ because the DFT requires that n be a power of 2, and $GF(2^8)$ has only 255 non-zero members. So instead, you will

use $GF(257)$ which is isomorphic to Z_{257} because 257 is prime. Sadly, this means that you won't be able to store a symbol in a single byte. You will probably have to read about the FFT algorithm in a text book. It is covered well in the algorithms book by Cormen, Leiserson, Rivest, and Stein. You can check that your FFT is working correctly by comparing your results with those of your simpler matrix-vector multiplication implementation (using $GF(257)$ instead of $GF(2^8)$).

Problem 4: Decoding [20 pt]

Finally, you will have to write a program to decode a codeword that is missing up to $2s = n-k$ elements. You may assume that you are given a codeword long with an indication of which elements are missing, and your task is to recover the original message. If there were no missing elements in the codeword, you could apply the inverse DFT to recover the message, but that won't quite work here. Instead, you should observe that as long as you have received at least k codeword elements, you can write a set of k equations over $GF(2^8)$ to solve for the k unknown message variables. You should be able to apply Gaussian elimination to solve this set of equations. A description of Gaussian elimination can also be found in many textbooks. Show that your decoder works whether you use $GF(2^8)$ or $GF(257)$.