This is an open book, take home exam. The exam is due at 11:59pm Sunday, April 3. You are free to use any readings, but cannot talk to other people (within and outside the class) about questions on the exam. Also, if you use a source beyond what was given in class you must cite the source. Please direct any questions to Harish (harish at cs.duke.edu) before contacting me.

Problem 1: bzip

Decode the following bit string that has been encoded using transforms 1 and 2 of the bzip algorithm described in the lecture slides. First the Burrows Wheeler transform has been applied, and then the move-to-front transform has been applied using the gamma code (as defined on page 25 of the lecture 1 and 2 slides from the compression lectures) to encode the move-to-front numbers. Assume that the alphabet only has three characters a, b, and c, and that the original order in move-to-front is $[a, b, c]$—that is a is at location 1 (the front), b at location 2, and c at location 3 in the order. The last integer in the sequence of gamma codes specifies the position of the start character of the string, where the position number starts at 1.

10110101010110011000

(A) What is the integer sequence obtained after decoding the gamma codes?

(B) Give the character sequence after decoding using move-to-front.

(C) What is the output string?

Problem 2: Compressing Error Correction

In some sense error correction and compression do opposite things—error correction adds redundancy and compression removes it. To see how this works out, lets assume we apply byte-based PPM compression (with arithmetic coding) to a random string that is encoded with a $(6, 3, 4)_{256}$ systematic Reed-Solomon code. For (A) and (B) assume no errors.

(A) Assuming a very long string (e.g. 10^9), what is the compression rate (approximately)?

(B) About how long does the string have to be so that it gets any significant compression?

(C) Assuming the compression algorithm knows that bytes are encoded in blocks of six, assuming a very long string again, and assuming about 1% of the bytes are corrupted randomly and independently before compression (and not corrected by the compression algorithm), what is the compression rate?

Problem 3: LDPC-like codes with Reed-Solomon

Consider the following variant on LDPC codes. The code is given by a bipartite graph such that all nodes on the left have degree 3 and all nodes on the right have degree 9. Each node on the left represents an element from $\mathrm{GF}(2^8)$ and each node on the right will enforce a $(9, 7, 3)_{256}$ Reed-Solomon code. A codeword on the left is valid if it satisfies all codes on the right.

1. For a codeword of length n (a multiple of 3) what are the sizes of the Generator and Parity check matrices for the code.
2. What is the rate of the code?
3. Assuming the bipartite graph has expansion (α, β), what is the minimum β that is needed so that the distance is at least αn?

Problem 4: Cryptography: Rijndael

What is the inverse of the polynomial $3x^3 + x^2 + x + 2$ modulo $x^4 + 1$ used by Rijndael?