CSC 310 — Solutions to Mid-term Test

1. You would like to encode a sequence of symbols that come from an alphabet with $d + 3$ symbols. You want to encode symbols a_1, a_2, and a_3 using codewords that are three bits long. You want to encode symbols a_4, a_5, ..., a_{d+3} using codewords that are eight bits long. What is the maximum value of d for which this will be possible, if the code must be uniquely decodable?

   The codeword lengths are possible if and only if they satisfy the Kraft-McMillan inequality, which in this case is

$$\frac{1}{2^3} + \frac{1}{2^3} + \frac{1}{2^3} + \frac{d}{2^8} \le 1$$

   This is equivalent to

$$\frac{d}{256} \le \frac{5}{8}$$

   which is equivalent to $d \le 160$. *The maximum value possible value of* d *is therefore* 160.

2. Let C be a uniquely decodable binary code for an alphabet with I symbols. Assume that the probabilities of these symbols are all non-zero. Prove that if C is optimal (ie, has minimum expected codeword length), all codewords of C are at most I bits long.

   Since C *is uniquely decodable, its codeword lengths satisfy the Kraft-McMillan inequality. Since these codeword lengths satisfy the Kraft-McMillan inequality, there is an instantaneous code with these codeword lengths. Call this instantaneous code* C'. *If* C *is optimal,* C' *is also optimal, since its codeword lengths are the same. Since* C' *is instantaneous, it can be represented as a binary tree, in which the* I *codewords are the leaves. If this is a complete tree – ie, each non-terminal node has two children — then the maximum depth of a terminal node (corresponding to a codeword) is* $I - 1$. *This can be seen from the fact that as one descends each level from the root, one gets to a subtree with at least one fewer leaves than the previous subtree, and so one can descend only to level* $I - 1$, *at which point the subtree consists of just a single leaf.*

   It follows that if C *(and hence* C'*) has a codeword of length greater than* I, *the codeword tree for* C' *must not be complete. There must be some non-terminal node in this tree with only one child. But in that case, we can eliminate this non-terminal node, connecting its parent directly to its child. This reduces the length of all the codewords in the subtree headed by the child. Since these codewords are for symbols with non-zero probability, this change will produce an instantaneous (hence uniquely decodable) code with smaller expected codeword length than* C' *and* C, *contradicting the assumption that* C *is optimal. There must therefore be no codewords of length greater than* I *in* C.

   Notes: The statement could have been made stronger — the codewords in an optimal code must be at most $I - 1$ *bits long. An argument related to the above is that a Huffman code cannot have codewords longer than* I *bits (and in fact, they're no longer than* $I - 1$ *bits), as can be seen by considering the procedure for creating a Huffman code. However, this doesn't provide a full proof of the statement, since there are optimal codes that aren't Huffman codes.*

3. Find a binary Huffman code for the source alphabet $\{a_1, a_2, a_3, a_4, a_5\}$, with symbol probabilities $p_1 = 0.11$, $p_2 = 0.05$, $p_3 = 0.10$, $p_4 = 0.70$, and $p_5 = 0.04$. Show this code as a table giving the codeword for every source symbol. Show your work.

   In the Huffman algorithm, we will merge symbols a_2 and a_5, then a_3 and this merged symbol, then a_1 and the last merged symbol, and finally a_4 and the last merged symbol. Various codes can result from this, depending on how branches are labelled with 0 *and* 1. *One possible result is as follows:*

a_1 :	1 0
a_2 :	1 1 1 0
a_3 :	1 1 0
a_4 :	0
a_5 :	1 1 1 1

4. This question concerns arithmetic coding for a binary source alphabet in which the symbol probabilities are $p_0 = 1/3$ and $p_1 = 2/3$. Assume the arithmetic encoder operates as described in the lectures, with the coding interval (initially $[0, 1)$) shrinking as each symbol is encoded, bits then being output if they are determined, and the coding interval being expanded as bits are output, or when it is entirely in the middle of the coding range. Fill in the table below to show how this procedure operates as the symbols 1, 1, and 0 are encoded. Assume that symbol 0 is allocated the low part of the coding region (nearer 0).

Input symbol	Interval after shrinking for this symbol	Bits output at this point (if any)	Interval after expansion
1	[1/3, 1)	none	[1/3, 1)
1	[5/9, 1)	1	[1/9, 1)
0	[1/9, 11/27)	0	[2/9, 22/27)

5. Suppose we are compressing a source of symbols from a binary alphabet, $\{0, 1\}$, using arithmetic coding. We model the very first symbol by saying that 0 and 1 are equally likely. For the second and later symbols, we will use probabilities from a first-order Markov model.

   For the two questions below, consider the following sequence of symbols to be compressed:

$$1 \ 1 \ 1 \ 1 \ 0$$

   a) Suppose that both the encoder and the decoder fix the probabilities of transitions in the Markov chain as follows:

$$\begin{aligned} P(X_n = 1 \,|\, X_{n-1} = 0) &= 1/4 \\ P(X_n = 1 \,|\, X_{n-1} = 1) &= 7/8 \\ P(X_n = 0 \,|\, X_{n-1} = 0) &= 3/4 \\ P(X_n = 0 \,|\, X_{n-1} = 1) &= 1/8 \end{aligned}$$

      Approximately how many bits will the arithmetic coder output when compressing this sequence? You may write an arithmetic expression for this rather than an actual number.

      The Shannon information content of the sequence is minus the log of its probability, which is found by multiplying the probabilities of each symbol given the previous symbols:

$$-\log_2 \left[\frac{1}{2} \cdot \frac{7}{8} \cdot \frac{7}{8} \cdot \frac{7}{8} \cdot \frac{1}{8} \right] \quad = \quad 4.5779 \text{ bits}$$

      Arithmetic coding will produce approximately this many bits as output. More specifically, it will produce no more than this number of bits plus two, so it will output at most 6 bits.

   b) Suppose instead that the encoder and decoder don't know the probabilities for transitions, but estimate them adaptively using the Laplace scheme (adding one to the observed counts).

      Approximately how many bits will the arithmetic coder output when compressing this sequence? You may write an arithmetic expression for this rather than an actual number.

      The Shannon information content is now found using the adaptively estimate probabilities. The result is:

$$-\log_2 \left[\frac{1}{2} \cdot \frac{1}{2} \cdot \frac{2}{3} \cdot \frac{3}{4} \cdot \frac{1}{5} \right] \quad = \quad 5.322 \text{ bits}$$

      Arithmetic coding will produce approximately this many bits as output, and no more than this number of bits plus two. So it will output at most 7 bits.

      Note: Here I've assumed that we count only actual transitions, so that when the second '1' is encoded, there have been no transitions before. If you assumed that the first '1' follows an imaginary '0' or an imaginary '1', you would get a different answer.