CS222: Homework 2. Due December 1.

Please work on problems on this assignment completely on your own. (You may of course ask the lecturer or the TF for assistance, but avoid working with other students.) You may use standard mathematical software (Maple, Matlab, etc.) and you may write programs to help you determine answers. You should do 5 of the 7 problems; if you do 6, we'll use your top 5 scores.

1. A fair coin is flipped until the first head occurs. Let X denote the number of flips required. Find the entropy $H(X)$ in bits. Suppose your friend flips a fair coin until the first head is flipped to generate a value for X, and now you want to ask a series of yes-no questions (of the form "is X contained in the following set?") to determine the value of X generated. Describe what questions you ask, determine the expected number of questions you ask, and compare your result with $H(X)$.

2. Consider an n-sided die, where the i-th face comes up with probability p_i. Show that the entropy of a die roll is maximized when each face comes up with equal probability $1/n$. (Hint: you may wish to show more generally that if two faces have probability p_i and p_j with $p_i < p_j$, the entropy increases if you change the probabilities of these two faces to $p_i + \varepsilon$ and $p_j - \varepsilon$ for some suitable ε.)

3. Explain in reasonably pedantic detail the steps of the Burrows-Wheeler algorithm (both compression and decompression) on the Sesame Street phrase "wabbawabbawoo". (Go up through the Move-To-Front stage; you don't have to do a final coding after that.)

4. Consider arithmetic coding for the string *aacbca* when the probability of an a is 0.2, a b is 0.3, and a c is 0.5. Show each step for idealized arithmetic coding, with real number arithmetic. Now suppose someone gave you the real number 0.63215699. Decode a sequence of length 10 corresponding to the above model. For both problems, you should take the real interval $[0,1]$ as broken up into subintervals $[0,0.2]$,$[0.2,0.5]$ and $[0.5,1]$, and similarly recursively.

5. Consider the following eight by eight table, corresponding to pixel values for an image.

124	125	122	120	122	119	117	118
120	120	120	119	119	120	120	120
125	124	123	122	121	120	119	118
125	124	123	122	121	120	119	118
130	131	132	133	134	130	126	122
140	137	137	133	133	137	135	130
150	147	150	150	150	150	150	150
160	160	162	164	168	170	172	175

Go through the steps of compressing and decompressing this eight by eight block with JPEG. (We will skip the entropy coding step, and just do the quantization.)

Recall the appropriate compression steps:

- Translate so that the original values 0 to 255 instead vary between -128 and 127.
- Apply the discrete cosine transform.
- Round according to the quantization table, given below.

And to decompress:

- Take the compressed form, and reverse the quantization.
- Apply the inverse of discrete cosine transform.
- Round to integers and necessary.

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

In one sentence, how well do you think JPEG does?

6. Compress the string "abracadabraarbadacarba" using the LZ77 approach (sliding windows), with a window size of six and a lookahead buffer of size six. Similarly, compress it using the LZ78 and LZW approaches. For the LZW approach, assume the letters a, b, c, d, and r start in the dictionary in alphabetical order.

7. Delta encoding has been suggested as a means of compressing sorted lists of URLs and in other aspects of Web compression. We'll examine a variation here for sorted lists of numbers. You will need to find a good random number generator for your platform. You may also use any sorting routine available on your platform.

Generate a list of 10,000 random numbers on the range $[0, 2^{30})$. If we were not using any compression at all, we would need to use 300,000 bits to send all the numbers.

Now suppose we sort the numbers, so they form a sorted sequence $a_0, a_1, a_2, \ldots, a_{9999}$. Now we calculate all the differences $a_1 - a_0$, $a_2 - a_1$, $a_3 - a_2$, and so on. Take the largest difference D, and calculate $k = \lceil \log_2 D \rceil$. We know that representing each of the differences requires only k bits.

To transmit the numbers, we first transmit a_0 using 30 bits and k using 5 bits, just to be safe (since $k < 32$, we only need five bits to represent it). We then transmit all the differences $a_1 - a_0$, $a_2 - a_1$, $a_3 - a_2$, and so on, using only k bits for each. Argue that this is enough information to reconstruct the numbers. How many bits does it take to send this information? Repeat the experiment 10 times, and give the results for each trial.

Now let's look at a slightly different approach, similar to delta coding. We will split each 30-bit number in our sorted sequence into two parts, which gives us two ordered sequences. The first part of each string will consist of the "leftmost" (high order) m bits, and the second part of each string will consist of the "rightmost" (low order) $30 - m$ bits. You should decide how to pick m; I suggest changing it so the sequence of m-bit parts is easy to compress. Suppose we compress the m-bit parts – use whatever method you think best – and then separately send the 10,000 strings consisting of the $30 - m$ remaining bits for each each string uncompressed. Argue that this is enough information to reconstruct the numbers. How many bits does it take to send this information? Repeat the experiment 10 times, and give the results for each trial.