Homework February 3, 2015 (to be returned on February 10)

1) Consider the random variable X that takes seven possible values with probabilities $\vec{p} = (0.49, 0.26, 0.12, 0.04, 0.04, 0.03, 0.02)$.

(a) Find a binary Huffman code for X; (b) Find the expected code length for this encoding; (c) Find a ternary Huffman code, i.e. using symbols $0, 1, 2$, for X; (d) Find a ternary code for the case where the seventh event is impossible, i.e. the possible symbols are just six, and the probabilities $\vec{p} = (0.49, 0.26, 0.12, 0.05, 0.05, 0.03)$. You will run short of symbols and will need a dummy one appropriately placed...

What is the number k of merges for a D-ary code with n (non-dummy) symbols? How many dummy symbols must be included given n and D?

2) Find a probability distribution (p_1, p_2, p_3, p_4) such that the Huffman construction can lead to two optimal codes that assign different lengths $\{\ell_i\}$ to the four symbols.

3) Show that the procedure defined by the Shannon-Fano-Elias coding has expected length $< H(X) + 2$ bits. Show that the code is prefix-free, namely that the intervals $[0.z_1z_2 \dots z_l, 0.z_1z_2 \dots z_l + 1/2^l)$ defined by the various codewords $z_1z_2 \dots z_l$ (where l denotes their length) are disjoint.

4) Lempel-Ziv coding.

The basic idea for this method of compression is to replace a substring with a pointer to an earlier occurrence of the same substring. This idea is widely used for data compression, e.g. for the compress and gzip commands. We shall discuss here just a few examples. If you are interested in proofs of optimality and performance you can find a detailed discussion in Chap. 13 of Cover and Thomas book.

A string 1011010100010 is parsed into an ordered dictionary of substrings that have not appeared before as follows: $\lambda, 1, 0, 11, 01, 010, 00, 10$, where we include the empty substring λ and the substrings are ordered by the order in which they emerged from the source. After every comma we look ahead until we have found a substring that has not been marked off before. This new substring will be one of those previously marked plus one bit (this is why the λ substring is included). We can then encode the new substring by giving a pointer to the existing substring shorter by 1 bit and by the extra bit by which the new and the old substring differ. If at the n-th bit of the string we have enumerated $s(n)$ substrings we can encode the pointer by a maximum of $\lceil \log_2 s(n) \rceil$ bits.

The code for the above sequence is: $\lambda \mapsto (,)$; $1 \mapsto (,1)$; $0 \mapsto (0,0)$; $11 \mapsto (01,1)$; $01 \mapsto (10,1)$; $010 \mapsto (100,0)$; $00 \mapsto (010,0)$; $10 \mapsto (001,0)$. The empty symbol does not need any encoding and the first pointer (of the symbol 1 for the string above) is empty because there is just λ as substring in the dictionary. The symbol 00 is for instance encoded as above because the existing substring prefix is 0 (and its pointer is 2, i.e. 010 in binary) and the extra bit is 0. The encoded string is then 10001110110000100010, which is actually longer than the original one as no major redundancy was present.

(a) Encode the string 000000000000100000000000 using the basic algorithm described above.