Homework no. 6

1. **Shannon code.** Consider the following method for generating a code for a random variable X which takes on m values $\{1, 2, \ldots, m\}$ with probabilities $p_1, p_2, \ldots, p_m$. Assume that the probabilities are ordered so that $p_1 \geq p_2 \geq \cdots \geq p_m$. Define

$$F_i = \sum_{k=1}^{i-1} p_i, \tag{1}$$

the sum of the probabilities of all symbols less than i. Then the codeword for i is the number $F_i \in [0, 1]$ rounded off to l_i bits, where $l_i = \lceil \log \frac{1}{p_i} \rceil$.

   (a) Show that the code constructed by this process is prefix-free and the average length satisfies

$$H(X) \leq L < H(X) + 1. \tag{2}$$

   (b) Construct the code for the probability distribution $(0.5, 0.25, 0.125, 0.125)$.

2. **Ternary Lempel-Ziv 77** Consider a source with ternary alphabet $\{0, 1, 2\}$. The source sequence is

120212021202120212021202120212021202120212021202

is to be encoded with LZ1 (LZ77) (the sequence consist of ten repetitions of the pattern 1202).

   (a) Partition the sequence into distinct LZ1 phrases - how many phrases are there?

   (b) What is the bit sequence generated by the encoder ?

   (c) For the number of phrases in (a), try to construct the longest sequence you can over the alphabet $\{0, 1, 2\}$ that has exactly this many LZ1 phrases in it.

3. **Lempel-Ziv LZSS with perfect knowledge** Consider a windowless LZSS algorithm, and compress a string s of length n- another way of looking at this is assuming that a pointer is $\lceil \log_2 n \rceil$ bits long. Compress the string and obtain p_1 phrases: ex. $AACABAABAA$ parses as A, A, C, A, B, AA, BAA.

   Now consider a variation of the LZSS algorithm that works its way back from the end: find the biggest t such that $x_{n-t+1}, \ldots, x_n$ is the prefix of a substring $x_j, \ldots, x_{j+t-1}$ with $j < n - t + 1$, and work your way back.
   For example, this variation would start parsing $AACABAABAA$ by noting that $ABAA$ (the last 4 characters) also occurs starting from position 3, hence after the first comma the parsing would be $AACABA, ABAA$. Continuing this way, the algorithm parses the string as $A, A, C, A, B, A, ABAA$ - hence, it produces a parsing that is in general different from the standards LZSS.

   Let p_2 denote the number of phrases in which the string s is parsed

   • Show that $p_2 \geq p_1 - 1$, that is, that by working our way back from the end we do not reduce by much the number of phrases.

4. **Look-ahead Lempel-Ziv - LZSS** In class we discussed a variation of LZSS (see above), where we compare the result of parsing the next phrase with the standard LZSS to the result of emitting the next symbol and parsing from the following symbol using the standard LZSS.
   • Show that the number of phrases produced by the LZSS variant is not smaller than the number of phrases produced by the standard LZSS.
   Note: the reason why the variant of LZSS (or LZ77) can outperform LZSS is therefore not due to the fact that it produces fewer phrases, but to the fact that encoding one symbol is in general less expensive than encoding a pointer, and therefore encoding a symbol + a long phrase is in general less expensive than two shorter phrases.