Alternative Homework 1: Information Compression

Note. This is derived from MIT OCW content. See https://ocw.mit.edu/terms/. The original assignments and related materials can be retrieved from two MIT courses: https://bit.ly/37Aywtf and https://bit.ly/2BdK8GA

Question 1. Run-length encoding is a popular variable-length lossless compressor used in fax machines, image compression, etc. In this problem consider compression of X^n – an i.i.d. (independent, identically distributed) sequences of n bits. Here one message X is a Bernoulli random variable with very small probability to get value 1 (namely, $P(X = 1) = \frac{1}{100}$ and the probability to get value 0 is $P(X = 0) = \frac{99}{100}$).
We encode sequences of n such messages X^n using run-length encoding function $f(X^n)$ that is defined on X^n (any sequence of n bits) and follows these rules:

1. A chunk of consecutive $r \leq 128$ zeros (resp. ones) is encoded as one bit equal to `0` (resp. 1) followed by an 7-bit binary encoding of r. (Therefore, the length of the encoding of a single chunk is exactly 8 bits.)

2. If the input contains a run $r > 128$ (more than 128 consecutive zeros or ones), then split this into multiple chunks where the last chunk can be shorter than 128 bits. Encode every chunk as described in the previous step.

Compute the average achieved compression rate

$$\lim_{n\to\infty} \frac{1}{n} E(\ell(f(X^n)).$$

This formula shows the expected value E of the random variable $\ell(f(X^n))$, where ℓ denotes the length of the encoded (compressed) message. How does it compare with the optimal lossless compressor? Compute both code lengths in bits (and round them up to the precision 10^{-5}).
(This is a parody of Exercise 4, see https://bit.ly/2Y6mUKa.)

Question 2. A source with an alphabet size of $M = |X| = 4$ has symbol probabilities $\{1/3, 1/3, 2/9, 1/9\}$.

1. Use the Huffman algorithm to find an optimal prefix-free code for this source.

2. Use the Huffman algorithm to find another optimal prefix-free code with a different set of lengths.

3. Find another prefix-free code that is optimal but cannot result from using the Huffman algorithm.

This is an exact copy of 2.12. See https://bit.ly/2Y4PKMe, page 55.

Question 3.
Assume that the Markov process (Figure 1) has already reached a steady state, (this means that we have done a sufficiently long random walk in this probabilistic graph and the starting state does not affect the probability distributions).

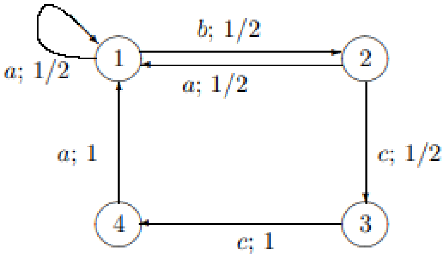

Figure 1. Markov chain.

The corresponding Markov source X generates messages from the alphabet $\{a, b, c\}$ with certain probabilities. Sequences of messages from X^n (sequences of n letters where you can assume than n is an even number) are compressed by two different people in two different ways:

1. Alice uses Arithmetic encoding to compress individual messages from the alphabet $\{a, b, c\}$. Find the average number of bits that Alice spends per **one** message.

2. Bob first splits the sequence X^n into pairs and then uses Arithmetic encoding to compress pairs of messages $\{aa, ab, ac, ba, bb, bc, ca, cb, cc\}$ (some pairs may be impossible). Find the average number of bits that Bob spends per **one** message.

This is a parody of problem 2.30. See https://bit.ly/2Y4PKMe, page 60. (Arithmetic coding and its code length is defined in the textbook by G.Blelloch, see https://bit.ly/2N3ywsf.)