Exercise – Statistical coding

In this exercise, no programming is required. The goal is to create the Huffman code and the arithmetic code of a simple source.

Let S be a source that can deliver 8 symbols a_i with the following probability laws $Pr(a_i)$:

$\mathbf{a_i}$	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
Pr ($\mathbf{a_i}$)	$\frac{1}{64}$	$\frac{1}{32}$	$\frac{1}{16}$	$\frac{1}{4}$	$\frac{17}{32}$	$\frac{1}{16}$	$\frac{1}{32}$	$\frac{1}{64}$

1 – Entropy of the source
Calculate the entropy H(S) of the source S.

2 – Huffman code
Build the Huffman code that corresponds to this source.
Calculate the average length of the Huffman code and compare with the entropy of the source.

3 – Arithmetic code
Let us consider that the source sends the following sequence: $\{a_5, a_7, a_1, a_7\}$. Encode this sequence by using an arithmetic code.
Compare the number of bits needed for encoding this sequence (arithmetic code) with the number needed to encode this sequence with a Huffman code.

Solution to the exercise: Statistical coding

1 – The entropy H of the source S is defined by:

$$H(S) = -\sum_{i=1}^{8} p_i \log_2(p_i) = 1.9848 \text{ bits/symbol}$$

The entropy represents the minimum mean number of bits that are necessary to encode a symbol of S.

2 – To build the Huffman code that corresponds to the source, we sort the symbols by decreasing probabilities in the column $p^{(0)}$:

Message	$p^{(0)}$
a_5	17 /32
a_4	1 /4
a_3	1 /16
a_6	1 /16
a_2	1 /32
a_7	1 /32
a_1	1 /64
a_8	1 /64

The two messages with the smallest probabilities are associated by addition, then the probabilities are re-sorted in a new column $p^{(1)}$:

Message	$p^{(0)}$	$p^{(1)}$
a_5	17 /32	17 /32
a_4	1 /4	1 /4
a_3	1 /16	1 /16
a_6	1 /16	1 /16
a_2	1 /32	1 /32
a_7	1 /32	1 /32
a_1	1 /64	1 /32
a_8	1 /64	××××××××

We reiterate this operation in order to create the complete table:

Message	$p^{(0)}$	$p^{(1)}$	$p^{(2)}$	$p^{(3)}$	$p^{(4)}$	$p^{(5)}$	$p^{(6)}$
a_5	17 /32	17 /32	17 /32	17 /32	17 /32	17 /32	17 /32
a_4	1 /4	1 /4	1 /4	1 /4	1 /4	1 /4	15 /32
a_3	1 /16	1 /16	1 /16	3 /32	1 /8	7 /32	
a_6	1 /16	1 /16	1 /16	1 /16	3 /32		
a_2	1 /32	1 /32	1 /16	1 /16			
a_7	1 /32	1 /32	1 /32				
a_1	1 /64	1 /32					
a_8	1 /64						

We assign the bits '1' and '0' to the last two elements of each column:

Message	$p^{(0)}$	$p^{(1)}$	$p^{(2)}$	$p^{(3)}$	$p^{(4)}$	$p^{(5)}$	$p^{(6)}$
a_5	17 /32	17 /32	17 /32	17 /32	17 /32	17 /32	17 /32 1
a_4	1 /4	1 /4	1 /4	1 /4	1 /4	1 /4 1	15 /32 0
a_3	1 /16	1 /16	1 /16	3 /32	1 /8 1	7 /32 0	
a_6	1 /16	1 /16	1 /16	1 /16 1	3 /32 0		
a_2	1 /32	1 /32	1 /16 1	1 /16 0			
a_7	1 /32	1 /32 1	1 /32 0				
a_1	1 /64 1	1 /32 0					
a_8	1 /64 0						

For each symbol a_i, we go through the table from left to right and in each column we can see the associated probability $p^{(i)}$. For example, the table below shows the probabilities of the symbol a_8 (blue path) and the associated bits (red markers):

Message	$p^{(0)}$	$p^{(1)}$	$p^{(2)}$	$p^{(3)}$	$p^{(4)}$	$p^{(5)}$	$p^{(6)}$
a_5	17/32	17/32	17/32	17/32	17/32	17/32	17/32
a_4	1/4	1/4	1/4	1/4	1/4	1/4	**15/32** 0
a_3	1/16	1/16	1/16	**3/32**	1 /8	**7/32** 0	
a_6	1/16	1/16	1/16	1/16	**3/32** 0		
a_2	1/32	1/32	**1/16** 1	1/16			
a_7	1/32	1/32	1/32				
a_1	1/64	**1/32** 0					
a_8	**1/64** 0						

The code-word is thus obtained by simply reading the marked bits from right to left. For the symbol a_8 the code-word is thus: 0-0-0-1-0-0

By following the same procedure for each symbol, we obtain:

Message	Mot-code
a_5	1
a_4	01
a_3	0011
a_6	0010
a_2	0000
a_7	00011
a_1	000101
a_8	000100

We can thus calculate the average length E(n) of the code-words:

$E(n) = \sum_{i=1}^{8} p_i n_i$, where n_i stands for the number of bits that are necessary to encode the symbol a_i i.e. the length of the code-word associated with a_i.

Here, we obtain: E(n) = 2.
The efficiency   of the Huffman code is thus:

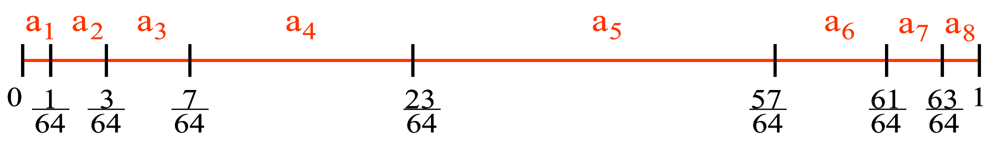
$$= H(S) / E(n) = 1.9848 / 2 = 99.24\%$$

The code is really close to the optimal code (entropic coding).

3 – Now we want to encode the sequence of symbols $\{a_5, a_7, a_1, a_7\}$ with an arithmetic code.
First we initialize a first interval with two bounds: the lower bound $L_c = 0$ and the upper bound $H_c = 1$. This interval [0, 1[is subdivided into 8 subintervals $[La_i, Ha_i[$ according to the probabilities of the symbols a_i of the source:

$$La_i = \sum_{k=1}^{i-1} p_i \quad \text{and} \quad Ha_i = \sum_{k=1}^{i} p_i$$

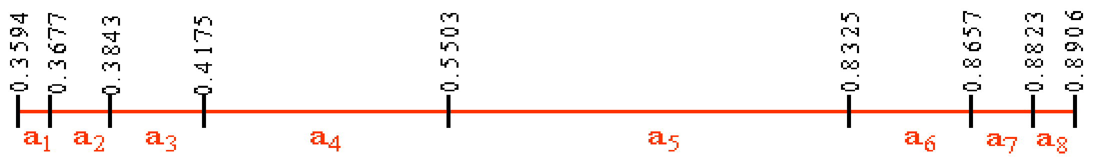
The length of the subinterval $[La_i, Ha_i[$ is thus equal to: $La_i - Ha_i = p_i$.
We obtain thus the following initial subdivision:



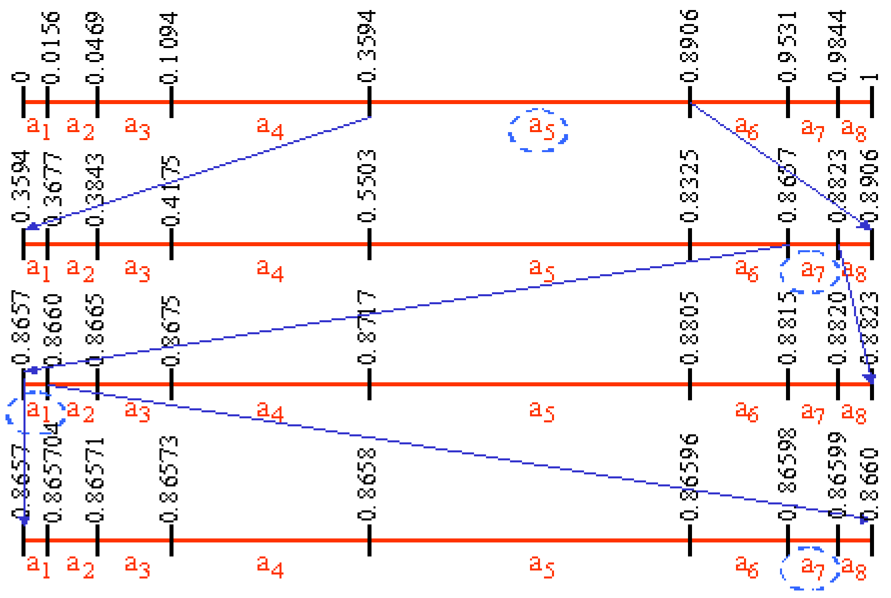
The first symbol of the sequence to encode is the symbol a_5. We subdivide thus the half-open interval $[La_5, Ha_5[= [23/64, 57/64[$ into 8 new subintervals $[La_i, Ha_i[$ defined by:

$$La_i = La_5 + p_5 \times \sum_{k=1}^{i-1} p_i \quad \text{et} \quad Ha_i = La_5 + p_5 \times \sum_{k=1}^{i} p_i$$

We obtain the following subdivision:



By repeating this procedure for the three next symbols in the sequence, we obtain the following subdivisions (the procedure has been implemented here in Matlab):



Consequently, we can encode the sequence {a_5, a_7, a_1, a_7} by any value in the half-open range [0.86598 ; 0.86599[.

We code this sequence with a binary code-word $m^{(k)}$ of k bits that is written as: $m^{(k)} = b_1 2^{-1} + b_2 2^{-2} + \dots + b_k 2^{-k}$ (with b_i = 0 or 1) and so that $m^{(k)}$ and $m^{(k+1)}$ belong to the half-open range [0.86598 ; 0.86599[and $m^{(k-1)} < 0.86598$.

The value 0.865982 encodes the sequence, and its binary representation is: 1101110110110001, we need 16 bits to encode this sequence.

To encode the same sequence with the Huffman code, we need 1+5+6+5 = 17 bits. On average, the arithmetic coding allows you to represent a sequence of symbols more efficiently than the Huffman coding. For a given alphabet, the longer the sequence is, the greater the efficiency is.