# Exercises with solutions (Set C)

9. Given is a Bernoulli process $\mathbf{X} = \{X_n\}$ with the alphabet $\mathcal{A} = \{a, b\}$ and the pmf $p_X(a) = 1/4$ and $p_X(b) = 3/4$.

   (a) Consider Elias coding and derive the codeword for the symbol sequence "*abba*" using the iterative encoding procedure.

   *Solution:*

   The iterative encoding can be summarized as follows:

   - Derive the cumulative probability mass function $c_X$ excluding the current symbol

$$\begin{aligned} c_X(a) &= 0 \\ c_X(b) &= p_X(a) = \frac{1}{4} \end{aligned}$$

   - Initialize the interval width and the lower interval boundary

$$\begin{aligned} W_0 &= 1 \\ L_0 &= 0 \end{aligned}$$

   - Update the interval width and lower interval boundary for the first symbol "*a*"

$$\begin{aligned} W_1 &= W_0 \cdot p_X(a) = 1 \cdot \frac{1}{4} = \frac{1}{4} \\ L_1 &= L_0 + W_0 \cdot c_X(a) = 0 + 1 \cdot 0 = 0 \end{aligned}$$

   - Update the interval width and lower interval boundary for the second symbol "*b*"

$$\begin{aligned} W_2 &= W_1 \cdot p_X(b) = \frac{1}{4} \cdot \frac{3}{4} = \frac{3}{16} \\ L_2 &= L_1 + W_1 \cdot c_X(b) = 0 + \frac{1}{4} \cdot \frac{1}{4} = \frac{1}{16} \end{aligned}$$

   - Update the interval width and lower interval boundary for the third symbol "*b*"

$$\begin{aligned} W_3 &= W_2 \cdot p_X(b) = \frac{3}{16} \cdot \frac{3}{4} = \frac{9}{64} \\ L_3 &= L_2 + W_2 \cdot c_X(b) = \frac{1}{16} + \frac{3}{16} \cdot \frac{1}{4} = \frac{7}{64} \end{aligned}$$

   - Update the interval width and lower interval boundary for the fourth and last symbol "*a*"

$$\begin{aligned} W = W_4 &= W_3 \cdot p_X(a) = \frac{9}{64} \cdot \frac{1}{4} = \frac{9}{256} \\ L = L_4 &= L_3 + W_3 \cdot c_X(a) = \frac{7}{64} + \frac{9}{64} \cdot 0 = \frac{7}{64} \end{aligned}$$

- Determine the number of bits $K$ for the Elias codeword based on the derived interval width

$$K = \lceil -\log_2 W \rceil = \left\lceil \log_2 \frac{256}{9} \right\rceil = \log_2 \frac{256}{8} = \log_2 32 = 5$$

- Determine the value of $V = \lceil L \cdot 2^K \rceil$

$$V = \lceil L \cdot 2^K \rceil = \left\lceil \frac{7}{64} \cdot 2^5 \right\rceil = \left\lceil \frac{7}{2} \cdot 2^5 \right\rceil = 4$$

- Determine the value of $v = V \cdot 2^{-K} = \lceil L \cdot 2^K \rceil \cdot 2^{-K}$ and write it as binary fraction with $K$ digits after the binary point

$$v = \lceil L \cdot 2^K \rceil \, 2^{-K} = 4 \cdot 2^{-5} = 4 \cdot \frac{1}{32} = \frac{1}{8} = 0.00100_b$$

- Determine the Elias codeword by using the first $K$ bits after the binary point of the value $v$ (this is equivalent to using a binary representation of the value $V$ with $K$ bits)

$$\text{codeword}(\text{"}abba\text{"}) = \text{"00100"}$$

(b) Develop the complete code for an Elias coding of 3 symbols (i.e., determine the codewords for all symbol sequences that consist of 3 symbols).

Determine the average codeword length per symbol and compare it to the entropy rate and the average codeword length per symbol for a joint Huffman code for sequences of 3 symbols.

Is the Elias code more efficient than the Huffman code for the same number of jointly coded symbols?

*Solution:*

For developing the complete Elias code, we could apply the iterative encoding procedure as done in (9a) for each of the 8 symbol sequences. But since we want to develop the complete code, this can be simplified. We can sort the symbol sequences $\mathbf{s} = \{s_0, s_1, s_2\}$ in increasing order of the associated lower interval boundaries. The corresponding sorting index $i$ could be derived by

$$i(\mathbf{s}) = \sum_{k=0}^{2} z(s_k) \cdot 2^{2-k} \qquad \text{with} \qquad z(s) = \begin{cases} 0 & : \ s = a \\ 1 & : \ s = b \end{cases} .$$

Then, the interval width $W$ for a symbol sequence $\mathbf{s}_i = \{s_0, s_1, s_2\}$ is given by

$$W(\mathbf{s}_i) = \prod_{k=0}^{2} p_X(s_k),$$

and the lower interval boundaries are obtained by

$$L(\mathbf{s}_i) = \begin{cases} 0 & : \ i = 0 \\ L(\mathbf{s}_{i-1}) + W(\mathbf{s}_{i-1}) & : \ 0 < i < 8 \end{cases} .$$

The development of the Elias code is illustrated in the following table.

| $i$ | $\mathbf{s}_i$ | $W(\mathbf{s}_i)$ | $L(\mathbf{s}_i)$ | $-\log_2 W$ | $K$ | $L \cdot 2^K$ | $\lceil L \cdot 2^K \rceil$ | codeword |
|---|---|---|---|---|---|---|---|---|
| 0 | *aaa* | 1/64 | 0/64 | 6.00 | 6 | 0 | 0 | 000000 |
| 1 | *aab* | 3/64 | 1/64 | 4.42 | 5 | 1/2 | 1 | 00001 |
| 2 | *aba* | 3/64 | 4/64 | 4.42 | 5 | 2 | 2 | 00010 |
| 3 | *abb* | 9/64 | 7/64 | 2.83 | 3 | 7/8 | 1 | 001 |
| 4 | *baa* | 3/64 | 16/64 | 4.42 | 5 | 8 | 8 | 01000 |
| 5 | *bab* | 9/64 | 19/64 | 2.83 | 3 | 19/8 | 3 | 011 |
| 6 | *bba* | 9/64 | 28/64 | 2.83 | 3 | 7/2 | 4 | 100 |
| 7 | *bbb* | 27/64 | 37/64 | 1.25 | 2 | 37/16 | 3 | 11 |

With $p_{\mathbf{X}}(\mathbf{s}) = p_X(s_0)p_X(s_1)p_X(s_2)$ being the joint pmf for symbol

sequences of 3 symbols, the average codeword length per symbol is

$$\begin{aligned}
\bar{\ell}_E &= \frac{1}{3}\sum_{i=0}^{7} p_{\mathbf{X}}(\mathbf{s}_i)\,\ell_E(\mathbf{s}_i) = \frac{1}{3}\sum_{i=0}^{7} W(\mathbf{s}_i)\,\ell_E(\mathbf{s}_i) \\
&= \frac{1}{3}\left(\frac{1}{64}\cdot 6 + \frac{3}{64}\cdot(5+5+5) + \frac{9}{64}\cdot(3+3+3) + \frac{27}{64}\cdot 2\right) \\
&= \frac{1}{192}(6+45+81+54) = \frac{186}{192} \\
&= \frac{31}{32} = 0.96875
\end{aligned}$$

An example for a joint Huffman code is given in the following table

| $i$ | $\mathbf{s}_i$ | $p_{\mathbf{X}}(\mathbf{s}_i)$ | codeword |
|---|---|---|---|
| 0 | *aaa* | 1/64 | 00000 |
| 1 | *aab* | 3/64 | 00001 |
| 2 | *aba* | 3/64 | 00010 |
| 3 | *abb* | 9/64 | 001 |
| 4 | *baa* | 3/64 | 00011 |
| 5 | *bab* | 9/64 | 010 |
| 6 | *bba* | 9/64 | 011 |
| 7 | *bbb* | 27/64 | 1 |

The average codeword length for the joint Huffman code is

$$\begin{aligned}
\bar{\ell}_H &= \frac{1}{3}\sum_{i=0}^{7} p_{\mathbf{X}}(\mathbf{s}_i)\,\ell_H(\mathbf{s}_i) \\
&= \frac{1}{3}\left(\frac{1}{64}\cdot 5 + \frac{3}{64}\cdot(5+5+5) + \frac{9}{64}\cdot(3+3+3) + \frac{27}{64}\cdot 1\right) \\
&= \frac{1}{192}(5+45+81+27) = \frac{158}{192} \\
&= \frac{79}{96} \approx 0.8229
\end{aligned}$$

The entropy rate of the given source is

$$\begin{aligned}
\bar{H}(\mathbf{X}) &= H(X) = -p_X(a)\log_2 p_X(a) - p_X(b)\log_2 p_X(b) \\
&= -\frac{1}{4}\log_2\frac{1}{4} - \frac{3}{4}\log_2\frac{3}{4} \\
&= \frac{1}{2} + \frac{3}{4}\,(2 - \log_2 3) \\
&= \frac{1}{4}(8 - 3\log_2 3) \approx 0.81128
\end{aligned}$$

The absolute redundancy for the Elias and Huffman code are

$$\begin{aligned}\rho_E &= \bar{\ell}_E - \bar{H}(\mathbf{X}) = \frac{31}{32} - \frac{1}{4}(8 - 3\log_2 3) \\ &= \frac{1}{32}(24\log_2 3 - 33) \approx 0.15747\end{aligned}$$

$$\begin{aligned}\rho_H &= \bar{\ell}_H - \bar{H}(\mathbf{X}) = \frac{79}{96} - \frac{1}{4}(8 - 3\log_2 3) \\ &= \frac{1}{96}(72\log_2 3 - 113) \approx 0.01164\end{aligned}$$

For the relative redundancies, we obtain

$$\begin{aligned}\rho_E^* &= \frac{\rho_E}{\bar{H}(\mathbf{X})} = \frac{24\log_2 3 - 33}{8 \cdot (8 - 3\log_2 3)} \approx 19.41\% \\ \rho_H^* &= \frac{\rho_H}{\bar{H}(\mathbf{X})} = \frac{72\log_2 3 - 113}{24 \cdot (8 - 3\log_2 3)} \approx 1.43\%\end{aligned}$$

The Huffman code is more efficient than the Elias code.

(c) Decode the 3-symbol sequence $\{s_0, s_1, s_2\}$ represented by the bit string "100" using the iterative Elias decoding algorithm.

*Solution:*

For decoding a symbol of an $M$-symbol alphabet, we have to calculate $M-1$ thresholds (lower interval boundaries) and compare the value $v$ given by the bit string with these threshold. Here, we consider a binary source and, hence, we only have to calculate a single threshold, which is the lower interval boundary of the second (upper) interval.

The iterative decoding can be summarized as follows:

- Determine the value $v$ that is represented by the codeword "100"

$$v = 0.100_b = 2^{-1} = \frac{1}{2}$$

- Initialize the interval width and the lower interval boundary

$$W_0 = 1 \qquad \text{and} \qquad L_0 = 0$$

- Calculate the threshold $T_0$ (i.e., the lower interval boundary for the symbol $b$) for the first symbol

$$T_0 = L_1(b) = L_0 + W_0 \cdot c_X(b) = 0 + 1 \cdot \frac{1}{4} = \frac{1}{4}$$

  Since $v \geq T_0$, the first symbol $s_0$ is equal to "$b$".
- Update the interval with and lower interval boundary for the decoded symbol

$$\begin{aligned} W_1 &= W_0 \cdot p_X(s_0) = 1 \cdot \frac{3}{4} = \frac{3}{4} \\ L_1 &= L_0 + W_0 \cdot c_X(s_0) = 0 + 1 \cdot \frac{1}{4} = \frac{1}{4} \end{aligned}$$

- Calculate the threshold $T_1$ (i.e., the lower interval boundary for the symbol $b$) for the second symbol

$$T_1 = L_2(b) = L_1 + W_1 \cdot c_X(b) = \frac{1}{4} + \frac{3}{4} \cdot \frac{1}{4} = \frac{7}{16}$$

  Since $v \geq T_1$, the second symbol $s_1$ is equal to "$b$".
- Update the interval with and lower interval boundary for the decoded symbol

$$\begin{aligned} W_2 &= W_1 \cdot p_X(s_1) = \frac{3}{4} \cdot \frac{3}{4} = \frac{9}{16} \\ L_2 &= L_1 + W_1 \cdot c_X(s_1) = \frac{1}{4} + \frac{3}{4} \cdot \frac{1}{4} = \frac{7}{16} \end{aligned}$$

- Calculate the threshold $T_2$ (i.e., the lower interval boundary for the symbol $b$) for the third symbol

$$T_2 = L_3(b) = L_2 + W_2 \cdot c_X(b) = \frac{7}{16} + \frac{9}{16} \cdot \frac{1}{4} = \frac{37}{64}$$

  Since $v < T_2$, the third symbol $s_2$ is equal to "$a$".

The decoded symbol sequence $\{s_0, s_1, s_2\}$ is equal to "$bba$".

(d) Consider the case in which the developed Elias code is used for coding multiple 3-symbol sequences. The codewords for the 3-symbol sequences are concatenated. Given is a bit string "10011100".

Decode the symbol sequence using the developed code table.

Decode the first three symbols (i.e., the first 3-symbol sequence) using the iterative decoding algorithm.

What do you observe?

*Solution:*

The Elias code represents a prefix code and can be decoded by comparing the bits inside the bit string with the codewords starting from the beginning of the bit string.

It is obvious that the bit string "10011100" consists of the codewords "100", "11", and "100". Hence, it represents the symbol sequence "bbabbbbba" ("*bba*", "*bbb*", "*bba*").

Using the iterative decoding procedure, the following sequence for the first 3 symbols is decoded.

- Determine the value $v$ that is represented by the given bit string "10011100"

  $$v = 0.10011100_b = 2^{-1}+2^{-4}+2^{-5}+2^{-6} = \frac{1}{2}+\frac{1}{16}+\frac{1}{32}+\frac{1}{64} = \frac{39}{64}$$

  Note that the decoder cannot know the border between the codewords, so that the complete bit string has to used for determining the value $v$.
- Initialize the interval width and the lower interval boundary

  $$\begin{aligned} W_0 &= 1 \\ L_0 &= 0 \end{aligned}$$

- Calculate the threshold $T_0$ (i.e., the lower interval boundary for the symbol $b$) for the first symbol

  $$T_0 = L_1(b) = L_0 + W_0 \cdot c_X(b) = 0 + 1 \cdot \frac{1}{4} = \frac{1}{4}$$

  Since $v \geq T_0$, the first symbol $s_0$ is equal to "*b*".
- Update the interval with and lower interval boundary for the decoded symbol

  $$\begin{aligned} W_1 &= W_0 \cdot p_X(s_0) = 1 \cdot \frac{3}{4} = \frac{3}{4} \\ L_1 &= L_0 + W_0 \cdot c_X(s_0) = 0 + 1 \cdot \frac{1}{4} = \frac{1}{4} \end{aligned}$$

- Calculate the threshold $T_1$ (i.e., the lower interval boundary for the symbol $b$) for the second symbol

  $$T_1 = L_2(b) = L_1 + W_1 \cdot c_X(b) = \frac{1}{4} + \frac{3}{4} \cdot \frac{1}{4} = \frac{7}{16}$$

  Since $v \geq T_1$, the second symbol $s_1$ is equal to "*b*".

- Update the interval with and lower interval boundary for the decoded symbol

$$\begin{aligned} W_2 &= W_1 \cdot p_X(s_1) = \frac{3}{4} \cdot \frac{3}{4} = \frac{9}{16} \\ L_2 &= L_1 + W_1 \cdot c_X(s_1) = \frac{1}{4} + \frac{3}{4} \cdot \frac{1}{4} = \frac{7}{16} \end{aligned}$$

- Calculate the threshold $T_2$ (i.e., the lower interval boundary for the symbol $b$) for the third symbol

$$T_2 = L_3(b) = L_2 + W_2 \cdot c_X(b) = \frac{7}{16} + \frac{9}{16} \cdot \frac{1}{4} = \frac{37}{64}$$

  Since $v \geq T_2$, the third symbol $s_2$ is equal to "*b*".

The first 3 symbols of the decoded symbol sequence $\{s_0, s_1, s_2, \cdots\}$ are equal to "*bbb*". A correct decoding of the symbol sequence is not possible using the iterative decoding procedure.

If multiple symbol sequences have to be decoded, an Elias code with $K = \lceil -\log_2 W \rceil$ bits for the codewords is not uniquely decodable using the iterative decoding procedure.

(e) How many bits have to be used for a codeword in order to make an Elias code uniquely decodable using the iterative decoding algorithm for a sequence of codewords.

Derive a lower and upper bound for the average codeword length per symbol for the corresponding Elias code if a codeword is generated for a sequences of $N$ symbols.

*Solution:*

For guaranteeing unique decodability for a sequence of codewords (using the iterative algorithm), we have to ensure than each possible value of $v$ that may be obtained by using a concatenation of codewords "$c_n c_{n+1} \cdots$" lies inside the interval $[L_n, L_n + W_n)$.

By using the assignment $v_n = \lceil L_n \cdot 2^K \rceil \cdot 2^{-K}$, the value of $v$ obtained by any concatenation of codewords cannot become smaller than the lower interval, independently of the selected number $K$ of bits. However, for certain values of $K$, the value $v$ may become larger than or equal to the upper interval boundary $L_n + W_n$.

The worst case is obtained if all bits following the codeword for the first symbol sequence are equal to 1,

$$v^* = v_n + \sum_{i=K+1}^{\infty} 2^{-i}$$

To ensure unique decodability, we have to ensure that $v^*$ is less than the upper interval boundary

$$v^* = v_n + \sum_{i=K+1}^{\infty} 2^{-i} < L_n + W_n$$

The sum in the above inequality is always less than $2^{-K}$ and, hence, the inequality is always fulfilled if

$$v_n + 2^{-K} \leq L_n + W_n$$

Inserting the expression for $v_n$ gives

$$\lceil L_n \cdot 2^K \rceil \cdot 2^{-K} + 2^{-K} \leq L_n + W_n$$

Using $\lceil x \rceil < x + 1$, we can state that the above inequality is always fulfilled if

$$\begin{aligned} (L_n \cdot 2^K + 1) \cdot 2^{-K} + 2^{-K} &\leq L_n + W_n \\ L_n + 2^{-K} + 2^{-K} &\leq L_n + W_n \\ 2^{1-K} &\leq W_n \\ 1 - K &\leq \log_2 W_n \\ K &\geq 1 - \log_2 W_n \end{aligned}$$

Hence, the smallest number of bits for guaranteeing unique decodability using the iterative decoding algorithm is

$$K = \lceil -\log_2 W_n \rceil + 1,$$

which is 1 bit more than we require for uniquely decoding a single symbol sequence using the Elias code.
The average codeword length per symbol for the considered Elias code is

$$\bar{\ell}_{E1} = \frac{1}{N} E\left\{\lceil \log_2 W_n \rceil + 1\right\} = \frac{1}{N} E\left\{\lceil \log_2 p_{\mathbf{S}^N}(\mathbf{S}^N) \rceil + 1\right\},$$

where $p_{\mathbf{S}^N}$ is the $N$-th order joint pmf.
Using $\lceil x \rceil \geq x$ and $\lceil x \rceil < x + 1$, we obtain

$$\begin{aligned} \frac{1}{N} E\left\{\log_2 p_{\mathbf{S}^N}(\mathbf{S}^N) + 1\right\} &\leq \bar{\ell}_{E1} < \frac{1}{N} E\left\{\log_2 p_{\mathbf{S}^N}(\mathbf{S}^N) + 2\right\} \\ \frac{H_N(\mathbf{S}^N)}{N} + \frac{1}{N} &\leq \bar{\ell}_{E1} < \frac{H_N(\mathbf{S}^N)}{N} + \frac{2}{N} \end{aligned}$$

where $H_N(\mathbf{S}^N)$ denotes the block entropy for $N$ symbols.

10. Given is a stationary discrete iid process $\mathbf{X} = \{X_n\}$ with the symbol alphabet $\mathcal{A} = \{\text{'M'}, \text{'I'}, \text{'S'}, \text{'P'}\}$ and the pmf

$$p_X(x) = \begin{cases} 0.1 & : \quad x = \text{'M'} \\ 0.3 & : \quad x = \text{'I'} \\ 0.4 & : \quad x = \text{'S'} \\ 0.2 & : \quad x = \text{'P'} \end{cases}$$

Consider binary arithmetic coding of the given source.

(a) Use a fixed-length code for binarizing the given source $\mathbf{X}$.
Verify on the given example that binarization does not have any impact on the coding efficiency (assuming a successive coding that achieves the entropy rate).
What binarization schemes can be used in the context of binary arithmetic coding?
Show that binarization does not change the lower bound for the average codeword length per symbol.

*Solution:*
The following table lists a possible fixed-length binarization scheme and the associated pmfs.

| $x$ | $p_X(x)$ | $b_0$ | $b_1$ | $p_{B_0B_1}(b_0, b_1)$ | $p_{B_0}(b_0)$ | $p_{B_1\|B_0}(b_1\|b_0)$ |
|---|---|---|---|---|---|---|
| 'M' | 1/10 | 0 | 0 | 1/10 | 4/10 | 1/4 |
| 'I' | 3/10 | 0 | 1 | 3/10 | | 3/4 |
| 'S' | 4/10 | 1 | 0 | 4/10 | 6/10 | 2/3 |
| 'P' | 2/10 | 1 | 1 | 2/10 | | 1/3 |

The entropy rate of the given iid process is

$$\begin{aligned} \bar{H}(\mathbf{X}) &= H(X_n) = -\frac{1}{10}\log_2\frac{1}{10} - \frac{2}{10}\log_2\frac{2}{10} - \frac{3}{10}\log_2\frac{3}{10} - \frac{4}{10}\log_2\frac{4}{10} \\ &= \log_2 10 - \left(\frac{1}{10}\cdot 0 + \frac{2}{10}\cdot 1 + \frac{3}{10}\cdot\log_2 3 + \frac{4}{10}\cdot 2\right) \\ &= \log_2 10 - 1 - \frac{3}{10}\log_2 3 \approx 1.84644 \end{aligned}$$

For the binary process, the even random variables $B_{2n}$ are independent of the other random variables; the odd random variables $B_{2n+1}$ depends only on the directly preceding random variables $B_{2n}$. Hence, the entropy rate is

$$\bar{H}(\mathbf{B}) = \frac{1}{2}H(B_{2n}, B_{2n+1}) = \frac{1}{2}\left(H(B_{2n}) + H(B_{2n+1}|B_{2n})\right)$$

For the entropy $H(B_{2n})$, we obtain

$$\begin{aligned} H(B_{2n}) &= -\frac{4}{10}\log_2\frac{4}{10} - \frac{6}{10}\log_2\frac{6}{10} \\ &= \log_2 10 - \left(\frac{4}{10}\cdot 2 + \frac{6}{10}\cdot(1+\log_2 3)\right) \\ &= \log_2 10 - \frac{7}{5} - \frac{3}{5}\log_2 3 \approx 1.13401 \end{aligned}$$

For the conditional entropy, we obtain

$$\begin{aligned} H(B_{2n+1}|B_{2n}) &= -\frac{1}{10}\log_2\frac{1}{4} - \frac{3}{10}\log_2\frac{3}{4} - \frac{4}{10}\log_2\frac{2}{3} - \frac{2}{10}\log_2\frac{1}{3} \\ &= \frac{2}{10} + \frac{6}{10} - \frac{3}{10}\log_2 3 + \frac{4}{10}\log_2 3 - \frac{4}{10} + \frac{2}{10}\log_2 3 \\ &= \frac{2}{5} + \frac{3}{10}\log_2 3 \approx 0.79396 \end{aligned}$$

Hence, the entropy rate $\bar{H}(\mathbf{B})$ is

$$\begin{aligned} \bar{H}(\mathbf{B}) &= \frac{1}{2}(H(B_{2n}) + H(B_{2n+1}|B_{2n})) \\ &= \frac{1}{2}\left(\log_2 10 + \frac{-7+2}{5} + \frac{3-6}{10}\log_2 3\right) \\ &= \frac{1}{2}\left(\log_2 10 - 1 - \frac{3}{10}\log_2 3\right) \\ &= \frac{1}{2}\bar{H}(\mathbf{X}) \end{aligned}$$

Since each symbol $x_n$ is mapped onto two binary symbols $b_{2n}$ and $b_{2n+1}$ with the chosen binarization scheme ($x_n \mapsto \{b_{2n}, b_{2n+1}\}$), this verifies that the chosen binarization does not have any impact on the coding efficiency.

When using binary arithmetic coding, the input sequence of symbols is first mapped to a sequence of binary symbols and then the sequence of binary symbols is arithmetically coded. For obtaining a uniquely decodable code, both steps have to be uniquely decodable. Hence, the binarization scheme has to represent a uniquely decodable code.

For investigating the impact of binarization we consider the entropy of a particular random variable $X$ given any condition $\mathcal{C}$. Without loss of generality we assume that a symbol $x$ is mapped to at maximum $N$ binary symbols. The conditional entropy for a random variable $X$ given the condition $\mathcal{C}$ is

$$\begin{aligned} H(X|\mathcal{C}) &= H(B_0, B_1, \cdots, B_{N-1}|\mathcal{C}) \\ &= H(B_0|\mathcal{C}) + H(B_1|B_0, \mathcal{C}) + H(B_2|B_0, B_1, \mathcal{C}) + \\ &\quad \cdots + H(B_{N-1}|B_0, B_1, \cdots, B_{N-2}, \mathcal{C}) \end{aligned}$$

Hence, mapping the input symbols to binary symbols does not have any impact on the coding efficiency presuming that the correct conditional pmfs are used for the following coding process.

Note that depending on the actually chosen binarization scheme, not all sequences of binary symbols are possible. The probability masses for the impossible sequences of binary symbols are equal to 0 and do not contribute to the conditional entropies.

(b) For arithmetic coding, the probability masses have to be represented with finite precision. Round the pmfs for the binary symbols to a precision of $V = 4$ bit.

What conditions need to be fulfilled for the rounded version of a pmf? Are these conditions fulfilled for the example?

Investigate the impact on the average codeword length per symbol of the given source $\mathbf{X}$ (assuming that the following arithmetic coding process does not have any negative impact on the coding efficiency).

*Solution:*

Let $p^*$ denote the rounded version of a pmf $p$. The rounded version of a pmf, with a precision of $V$ bit, can be obtained by

$$p^*(x) = \lfloor p(x) \cdot 2^V + 0.5 \rfloor \cdot 2^{-V}$$

The pmfs for the binary symbols and their rounded versions are summarized in the following table.

| $b$ | $p_{B_0}(b)$ | $p^*_{B_0}(b)$ |
|---|---|---|
| 0 | 2/5 | 6/16=3/8 |
| 1 | 3/5 | 10/16=5/8 |

| $b$ | $p_{B_1\|B_0=0}(b)$ | $p^*_{B_1\|B_0=0}(b)$ | $p_{B_1\|B_0=1}(b)$ | $p^*_{B_1\|B_0=1}(b)$ |
|---|---|---|---|---|
| 0 | 1/4 | 4/16=1/4 | 2/3 | 11/16 |
| 1 | 3/4 | 12/16=3/4 | 1/3 | 5/16 |

The following two conditions have to be fulfilled:

- A non-zero probability mass must not be rounded to zero

$$\forall_{x:p(x)>0} \quad p^*(x) > 0$$

- The sum over all rounded probability masses must not be greater than 1

$$\sum_x p^*(x) \leq 1$$

Both conditions are fulfilled for the rounded pmfs of the example.

The increase in average codeword length per binary symbol for each of the binary pmfs $p$ is given by

$$\begin{aligned} \Delta\bar{\ell} &= -\sum_{k=0}^{1} p(k) \left( \log_2 p(k) - \log_2 p^*(k) \right) \\ &= \sum_{k=0}^{1} p(k) \log_2 \frac{p^*(k)}{p(k)} \end{aligned}$$

Hence, we obtain

$$\begin{aligned}
\Delta\bar{\ell}_{B_0} &= \frac{2}{5}\log_2\frac{16}{15} + \frac{3}{5}\log_2\frac{24}{25} \\
&= \frac{1}{5}(17 + \log_2 3 - 8\log_2 5) \approx 0.0019075 \\
\Delta\bar{\ell}_{B_1|B_0=0} &= \frac{1}{4}\log_2 1 + \frac{3}{4}\log_2 1 = 0 \\
\Delta\bar{\ell}_{B_1|B_0=1} &= \frac{2}{3}\log_2\frac{32}{33} + \frac{1}{3}\log_2\frac{16}{15} \\
&= \frac{1}{3}(14 - 3\log_2 3 - \log_2 5 - 2\log_2 11) \approx 0.0014404
\end{aligned}$$

The increase in average codeword length for a random variable $X$ is given by

$$\begin{aligned}
\Delta\bar{\ell}_X &= \Delta\bar{\ell}_{B_0} + p_{B_0}(0)\cdot\Delta\bar{\ell}_{B_1|B_0=0} + p_{B_0}(1)\cdot\Delta\bar{\ell}_{B_1|B_0=1} \\
&= \frac{1}{5}(31 - 2\log_2 3 - 9\log_2 5 - 2\log_2 11) \approx 0.0027718
\end{aligned}$$

The increase in average codeword length increases by about 0.0028 bit per symbol $X$ due to the rounding of the pmfs to 4 bit. This corresponds to an increase of approximately 0.15% relative to the entropy rate.

(c) For arithmetic coding, the interval width has to be represented with finite precision.

Show that, for a coding of $N$ symbols, the corresponding increase in average codeword length per arithmetically coded symbol is less than $1/N + \log_2(1 + 2^{1-U})$ bits, if $U$ is the number of bits used for representing the interval width.

Determine the minimum precision $U$ that is required to guarantee that the coding efficiency loss due to rounding the interval width is less than 0.1% when more than 10000 symbols $X$ are transmitted.

*Solution:*

For analyzing the impact of representing the interval boundaries with finite precision, we compare the codeword length $\ell_A$ of arithmetic coding with that of Elias coding $\ell_E$. The average increase in codeword length per symbol $\Delta\ell$ is given by

$$\Delta\ell = \ell_A - \ell_E = \lceil -\log_2 W_N \rceil - \lceil -\log_2 p_{\mathbf{S}^{(N)}}(\mathbf{s}^{(N)}) \rceil$$

By applying the inequalities $\lceil x \rceil \geq x$ and $\lceil x \rceil < x + 1$, we obtain

$$\begin{aligned}
\Delta\ell &< 1 - \log_2 W_N - \lceil -\log_2 p_{\mathbf{S}^{(N)}}(\mathbf{s}^{(N)}) \rceil \\
&< 1 - \log_2 W_N + \log_2 p_{\mathbf{S}^{(N)}}(\mathbf{s}^{(N)}) \\
&= 1 + \log_2 \frac{p_{\mathbf{S}^{(N)}}(\mathbf{s}^{(N)})}{W_N} \\
&= 1 + \log_2 \frac{\prod_{k=0}^{N-1} p(s_k)}{\prod_{k=0}^{N-1} \frac{W_{k+1}}{W_k}} \\
&= 1 + \sum_{k=0}^{N-1} \log_2 \frac{W_k \cdot p(s_k)}{W_{k+1}}
\end{aligned}$$

It should be noted that the same expression is obtained if we compare the codeword length for the arithmetic code with the optimal codeword length "$-\log_2 p_{\mathbf{S}^{(N)}}(\mathbf{s}^{(N)})$".

It should also be noted that we considered the initial interval width $W_0$ to be equal to 1. Often the initial interval width is actually set equal to $W_0 = (2^U - 1) \cdot 2^{-U} = 1 - 2^{-U}$, in order not to have a special case for the first symbol ($A_0$ would otherwise exceed the range of $U$ bits). Then the right side of the above inequality would additionally include the term $\log_2(1 - 2^{-U})$. We will ignore this aspect.

In the following, we will derive an upper bound for $\frac{W_k \cdot p(s_k)}{W_{k+1}}$, and thus find an upper bound for $\Delta\ell$ and the average codeword length per symbol.

Since we want to analyze the effect of the quantization of the interval width, we assume that the probability masses $p(x)$ are accurately represented by $V$ bit integers $p_V(x)$ according to

$$p(x) = p_V(x) \cdot 2^{-V}$$

In each iteration step, the interval width $W_n$ is represented by an $U$ bit integer value $A_n$,

$$W_n = A_n \cdot 2^{-z_n},$$

where $z_n$ is an integer shift parameter. For maximizing the precision of this representation, $z_n$ is chosen in a way that

$$2^{U-1} \leq A_n < 2^U,$$

i.e., so that the most significant bit of the binary representation of $A_n$ is equal to 1.

In order to guarantee unique decodability, we have to ensure that the intervals do not overlap. This condition is always fulfilled if the rounded intervals are less than or equal to the non-rounded intervals, i.e, we have to ensure that

$$\begin{aligned} W_{n+1} &\leq W_n \cdot p(s_n) \\ A_{n+1} \cdot 2^{-z_{n+1}} &\leq A_n \cdot 2^{-z_n} \cdot p_V(s_n) \cdot 2^{-V} \end{aligned}$$

The terms $2^{-z_n}$ and $2^{-V}$ represent bit shift parameters. The value of $A_{n+1}$ has to be derived from the product $A_n \cdot p_V(s_n)$.

Let $x_n$ denote the number of leading zeros in the $(U+V)$-bit representation of the product $A_n \cdot p_V(s_n)$. The value of $x_n$ is in the interval $[0; V]$. In practice, the value of $x_n$ can be determined by comparing the product with the sequence of thresholds $2^{U+V-1}$, $2^{U+V-2}$, etc. If the product is less than $2^{U+V-x}$, but greater than or equal to $2^{U+V-x-1}$, the number of leading zeros in the $(U+V)$-bit representation of $A_n \cdot p_V(s_n)$ is equal to $x_n = x$. The value of $x_n$ can also be expressed as

$$x_n = U + V - \lfloor 1 + \log_2(A_n \cdot p_V(s_n)) \rfloor$$

For guaranteeing that $2^{U-1} \leq A_{n+1} < 2^U$, we have to increase the bits shift $z$ by the number of leading zeros in the binary representation of $A_n \cdot p_V(s_n)$. Hence, we have

$$z_{n+1} = z_n + x_n,$$

which yields

$$\begin{aligned} A_{n+1} \cdot 2^{-z_{n+1}} &\leq A_n \cdot 2^{-z_n} \cdot p_V(s_n) \cdot 2^{-V} \\ A_{n+1} &\leq A_n \cdot p_V(s_n) \cdot 2^{x_n - V} \end{aligned}$$

And for guaranteeing that $W_{n+1}$ is less than $W_n \cdot p(s_n)$, i.e., for guaranteeing that the condition above is fulfilled, we choose

$$A_{n+1} = \lfloor A_n \cdot p_V(s_n) \cdot 2^{x_n - V} \rfloor$$

The rounding towards zero can in practice be replaced by a bit shift to the right, i.e., $A_n = (A_n \cdot p_V(s_n)) \gg (V - x_n)$. Note that is equivalent to using the first $U$ significant bits in the representation of the product $A_n \cdot p_V(s_n)$ for $A_{n+1}$.

As a consequence, the interval width $W_{n+1}$ is

$$W_{n+1} = A_{n+1} \cdot 2^{z_{n+1}} = \lfloor A_n \cdot p_V(s_n) \cdot 2^{x_n - V} \rfloor \cdot 2^{-z_n - x_n}$$

By applying the inequality $\lfloor x \rfloor > x - 1$, we can write

$$\begin{aligned} W_{n+1} &= \left\lfloor A_n \cdot p_V(s_n) \cdot 2^{x_n - V} \right\rfloor \cdot 2^{-z_n - x_n} \\ &> A_n \cdot p_V(s_n) \cdot 2^{-z_n - V} - 2^{-z_n - x_n} \\ &= A_n \cdot 2^{-z_n} \cdot p(s_n) - 2^{-z_{n+1}} \\ &= W_n \cdot p(s_n) - \frac{W_{n+1}}{A_{n+1}} \end{aligned}$$

Dividing the inequality by $W_{n+1}$ and using the property $A_n \geq 2^{U-1}$ yields

$$\frac{W_n \cdot p(s_n)}{W_{n+1}} < 1 + \frac{1}{A_{n+1}} \leq 1 + 2^{1-U}$$

Inserting this expression into the inequality that we derived at the beginning yields

$$\Delta\ell < 1 + \sum_{k=0}^{N-1} \log_2 \frac{W_k \cdot p(s_k)}{W_{k+1}} < 1 + N \cdot \log_2(1 + 2^{1-U})$$

For the increase in codeword length per symbol, we obtain

$$\frac{\Delta\ell}{N} < \frac{1}{N} + \log_2(1 + 2^{1-U})$$

Now, we consider the given source $\mathbf{X}$ and determine the precision $U$ that is required for guaranteeing that the coding efficiency loss is less than 0.1% when more than $N = 10000$ symbols are transmitted.

Hence, for the increase in codeword length per symbol $x$, we have to obey

$$\frac{\Delta\ell}{N} < \frac{H(X_n)}{1000},$$

where $N$ is the number of coded symbols $x$.

Since we code 2 binary symbols per symbol $x$, the total number of arithmetically coded symbols $N_B = 2N$ is equal to 20000, and the maximum increase in codeword length per arithmetically coded symbol has to fulfill the inequality,

$$\frac{\Delta\ell}{N_B} < \frac{H(X_n)}{2000}.$$

This inequality is fulfilled if we set $U$ in a way that the inequality

$$\frac{1}{N_B} + \log_2(1 + 2^{1-U}) < \frac{H(X_n)}{2000}$$

is fulfilled. By reformulating this inequality (and using $N_B = 2N$),

we obtain

$$\begin{aligned}
\log_2(1+2^{1-U}) &< \frac{H(X_n)}{2000} - \frac{1}{2N} \\
1+2^{1-U} &< 2^{(\frac{H(X_n)}{2000}-\frac{1}{2N})} \\
2^{1-U} &< 2^{(\frac{H(X_n)}{2000}-\frac{1}{2N})} - 1 \\
1-U &< \log_2\left(2^{(\frac{H(X_n)}{2000}-\frac{1}{2N})} - 1\right) \\
U &> 1 - \log_2\left(2^{(\frac{H(X_n)}{2000}-\frac{1}{2N})} - 1\right)
\end{aligned}$$

By inserting the values for $N$ and $H(X_n)$, we obtain

$$\begin{aligned}
U &> 11.689697\ldots \\
U &\geq 12
\end{aligned}$$

If we use a precision of 12 bits for representing the interval width, it is guaranteed that the increase in codeword length per symbol $x$ is less than 0.1% if we code 10000 or more symbols.

(d) Consider arithmetic coding that uses $U$ bits for representing the interval width and $V$ bits for representing the probability masses.

Show that the lower interval boundaries can be represented by a counter and an integer value of $U + V$ bits.

*Solution:*

We start with investigating the binary representation of $W_n = A_n \cdot 2^{-z_n}$, which is shown in the following:

$$W_n = 0.\overbrace{\underbrace{00000\cdots0}_{z_n - U \text{ bits}}\underbrace{1xx\cdots x}_{U \text{ bits}}}^{z_n \text{ bits}}000\cdots$$

The first $z_n - U$ bits after the binary point are equal to 0. These bits are followed by $U$ bits that represent the integer value $A_n$. All following bits are equal to 0.

The binary representation of $c(s_n) = c_V(s_n) \cdot 2^{-V}$ consist of $V$ bits after the binary point that represent the value of the integer $c_V(s_n)$. All remaining bits are equal to 0.

$$c(s_n) = 0.\underbrace{xxx\cdots x}_{V \text{ bits}}000\cdots$$

The binary representation of the product $W_n \cdot c(s_n)$ is given as follows:

$$W_n \cdot c(s_n) = 0.\overbrace{\underbrace{00000\cdots0}_{z_n - U \text{ bits}}\underbrace{xxx\cdots x}_{U+V \text{ bits}}}^{z_n + V \text{ bits}}000\cdots$$

The first $z_n - U$ bits after the binary point are equal to 0. The following $U + V$ bits represent the product $A_n \cdot c_V(s_n)$ and depend on the current symbol $s_n$ and the previously coded symbols. All following bits are equal to 0.

Given the binary representation of $W_n \cdot c(s_n)$, we can conclude which bits of the lower interval boundary $L_n$ can be changed by the interval update $L_{n+1} = L_n + W_n \cdot c(s_n)$:

$$L_n = 0.\overbrace{\underbrace{ssssssss\cdots s}_{z_n - U - c_n \text{ bits}}\underbrace{0111\cdots1}_{c_n \text{ bits}}}^{z_n - U \text{ bits}}\underbrace{xxx\cdots x}_{U+V \text{ bits}}000\cdots$$

The first $z_n - U$ bits after the binary point are not directly changed (the corresponding bits in $W_n \cdot c(s_n)$ are equal to 0), but may be modified by a carry from the following bits. The following $U + V$ bits, which are called *active bits* are directly modified, since the corresponding bits in $W_n \cdot c(s_n)$ represent the integer $A_n \cdot c_V(s_n)$. All bits that follow these bits are equal to zero and are not modified in the update $L_n \mapsto L_{n+1}$, but may be modified in future updates.

The only possibility to modify the first $z_n - U$ bits after the decimal point is a carry from the following $U + V$ bits. We can partition

the first $z_n - U$ bits after the decimal point into two sections. The first section, which is called *outstanding bits*, consists of all bits equal to 1 (if any) that directly precede the $U + V$ active bits and the zero (if any) that precedes these bits equal to 1. All other bits are referred to as *settled bits*. These settled bits cannot be modified by the current or any following interval update (all $L_{n+k}$ are less than $L_n + W_n$, since the intervals are nested). Hence, the settled bits can be output as soon as they become settled, and the remaining bits can be represented by an $(U + V)$-bit integer $B_n$ representing the active bits and a counter $c_n$ representing the number of outstanding bits. The total number of bits to output is

$$K = \lceil -\log_2 W_N \rceil = z_N - \lfloor \log_2 A_N \rfloor = z_N - U + 1$$

Hence, at the end of the coding process, all outstanding bits and the most significant bit of the $(U + V)$-bit integer $B_N$ have to be output.

The arithmetic encoding process can be stated as follows:

(1) *Initialization*: $A_0 = 2^U - 1$, $B_0 = 0$, $c_0 = 0$

(2) *Iterative Coding*: For $n = 0$ to $n = N - 1$, do the following:

[Find the bit shift $x_n = z_{n+1} - z_n$]

- Determine the value $A^*_{n+1} = A_n \cdot p_V(s_n)$
- Determine the value $B^*_{n+1} = B_n + A_n \cdot c_V(s_n)$
- Determine the number $x_n$ of leading zeros in the $(U + V)$-bit representation of $A^*_{n+1}$

[Set the parameters $A_{n+1}$ and $B_{n+1}$]

- Set $m = (1 \ll (U + V - x_n)) - 1$ [“$\ll$” – bit shift to the left]
- Set $A_{n+1} = A^*_{n+1} \gg (V - x_n)$ [“$\gg$” – bit shift to the right]
- Set $B_{n+1} = (B^*_{n+1} \,\&\, m) \ll x_n$ [“&” – bit-wise “and”]

[Output the settled bits and set the counter $c_{n+1}$]

- [Carry] If $B^*_{n+1} \geq 2^{U+V}$, do the following:
  - Set $r = \max(c_n - 2, 0)$
  - Output a bit equal to “1” and $r$ bits equal to “0”
  - Modify $c_n = c_n - r - 1$
  - Modify $B^*_{n+1} = B^*_{n+1} - 2^{U+V}$
- Modify $B^*_{n+1} = B^*_{n+1} \gg (U + V - x_n)$
- Determine the number $y$ of trailing bits equal to 1 in the binary representation of $B^*_{n+1}$
- [Case 1] If $y = x_n$ and $c_n = 0$, do the following:
  - Output $y$ bits equal to “1”
  - Set $c_{n+1} = 0$
- [Case 2] If $y = x_n$ and $c_n > 0$, set $c_{n+1} = c_n + y$
- [Case 3] If $y < x_n$, do the following:

- If $c_n > 0$,
  - Output a bit equal to "0"
  - Output $c_n - 1$ bits equal to "1"
- Set $k = x_n - 1$
- While $k > y$, do
  - Output a bit equal to $(B^*_{n+1} \gg k) \,\&\, 1$
  - Modify $k = k - 1$

(3) *Termination*: Do the following:

- If $c_n > 0$,
  - Output a bit equal to "0"
  - Output $c_n - 1$ bits equal to "1"
- Output a bit equal to $(B_N \gg (U + V - 1)) \,\&\, 1$

(e) Consider binary arithmetic coding for the given source $\mathbf{X}$ with fixed-length binarization, $V = 4$ bits for representing the probability masses, and $U = 4$ bits for representing the interval width.

Generate the arithmetic codeword for the symbol sequence "MISS".

Compare the length of the arithmetic codeword with the length of the codeword that would be generated by Elias coding.

*Solution:*

The encoding procedure can be done as described above. The main steps for the example are summarized in the following table.

| $s$ | $b$ | $p_V$ | $c_V$ | parameter updates & output |
|---|---|---|---|---|
| *initialization* | | | | $A_0 = 15 =$ '1111'<br>$c_0 = 0$<br>$B_0 = 0 =$ '0000 0000' |
| "M" | 0 | 6 | 0 | $A_0 \cdot p_V = 15 \cdot 6 = 90 =$ '0101 1010'<br>$B_0 + A_0 \cdot c_V = 0 + 15 \cdot 0 = 0 =$ '0 0000 0000'<br>$x_0 = 1$ |
| | | | | $A_1 =$ '1011' $= 11$<br>$c_1 = 1$<br>$B_1 =$ '0000 0000' $= 0$<br>**output = "**" |
| | 0 | 4 | 0 | $A_1 \cdot p_V = 11 \cdot 4 = 44 =$ '0010 1100'<br>$B_1 + A_1 \cdot c_V = 0 + 11 \cdot 0 = 0 =$ '0 0000 0000'<br>$x_1 = 2$ |
| | | | | $A_2 =$ '1011' $= 11$<br>$c_2 = 1$<br>$B_2 =$ '0000 0000' $= 0$<br>**output = "00"** |
| "I" | 0 | 6 | 0 | $A_2 \cdot p_V = 11 \cdot 6 = 66 =$ '0100 0010'<br>$B_2 + A_2 \cdot c_V = 0 + 11 \cdot 0 = 0 =$ '0 0000 0000'<br>$x_2 = 1$ |
| | | | | $A_3 =$ '1000' $= 8$<br>$c_3 = 1$<br>$B_3 =$ '0000 0000' $= 0$<br>**output = "0"** |
| | 1 | 12 | 4 | $A_3 \cdot p_V = 8 \cdot 12 = 96 =$ '0110 0000'<br>$B_3 + A_3 \cdot c_V = 0 + 8 \cdot 4 = 32 =$ '0 0010 0000'<br>$x_3 = 1$ |
| | | | | $A_4 =$ '1100' $= 12$<br>$c_4 = 1$<br>$B_4 =$ '0100 0000' $= 64$<br>**output = "0"** |
| "S" | 1 | 10 | 6 | $A_4 \cdot p_V = 12 \cdot 10 = 120 =$ '0111 1000'<br>$B_4 + A_4 \cdot c_V = 64 + 12 \cdot 6 = 136 =$ '0 1000 1000'<br>$x_4 = 1$ |
| | | | | $A_5 =$ '1111' $= 15$<br>$c_5 = 2$<br>$B_5 =$ '0001 0000' $= 16$<br>**output = "**" |
| | 0 | 11 | 0 | $A_5 \cdot p_V = 15 \cdot 11 = 165 =$ '1010 0101'<br>$B_5 + A_5 \cdot c_V = 16 + 15 \cdot 0 = 16 =$ '0 0001 0000'<br>$x_5 = 0$ |
| | | | | $A_6 =$ '1010' $= 10$<br>$c_6 = 2$<br>$B_6 =$ '0001 0000' $= 16$<br>**output = "**" |
| "S" | 1 | 10 | 6 | $A_6 \cdot p_V = 10 \cdot 10 = 100 =$ '0110 0100'<br>$B_6 + A_6 \cdot c_V = 16 + 10 \cdot 6 = 76 =$ '0 0100 1100'<br>$x_6 = 1$ |
| | | | | $A_7 =$ '1100' $= 12$<br>$c_7 = 1$<br>$B_7 =$ '1001 1000' $= 152$<br>**output = "01"** |
| | 0 | 11 | 0 | $A_7 \cdot p_V = 12 \cdot 11 = 132 =$ '1000 0100'<br>$B_7 + A_7 \cdot c_V = 152 + 12 \cdot 0 = 152 =$ '0 1001 1000'<br>$x_7 = 0$ |
| | | | | $A_8 =$ '1000' $= 8$<br>$c_8 = 1$<br>$B_8 =$ '1001 1000' $= 152$<br>**output = "**" |
| *termination* | | | | **output = "01"** [outstanding & first bit of $B_8$] |

The codeword generated by the arithmetic encoding procedure is "0000 0101". It consists of 8 bits.

The length of the codeword that would be generated by Elias coding (i.e., without rounding the probabilities and interval widths) is given by

$$\begin{aligned} K_E &= \left\lceil -\log_2 p(s_0, s_1, s_2, s_3) \right\rceil = \left\lceil -\log_2 \big(p(s_0)p(s_1)p(s_2)p(s_3)\big) \right\rceil \\ &= \left\lceil -\log_2 \left(\frac{1}{10} \cdot \frac{3}{10} \cdot \frac{4}{10} \cdot \frac{4}{10}\right) \right\rceil = \left\lceil -\log_2 \left(\frac{48}{10000}\right) \right\rceil \\ &= \left\lceil 7.7\cdots \right\rceil = 8 \end{aligned}$$

It has the same size as the codeword generated by arithmetic coding with $V = 4$ and $U = 4$.