S-72.340 INFORMATION THEORY

Exercise 4, October 11, 2000 Solutions

1. a) The number of sequences with at most 3 **1**'s is

$$N = \binom{100}{0} + \binom{100}{1} + \binom{100}{2} + \binom{100}{3} = 166751\,.$$

If the length of each codeword is n bits, there are 2^n different code words. Thus, the minimum length n_{min} is the smallest integer value of n satisfying the condition

$$2^n \geq N\,.$$

By taking logarithm to the base 2, we get

$$n \geq \log_2 N \approx 17.3473 \;\Rightarrow\; n_{min} = 18\,.$$

b) There was an error here in the original problem sheet. We assume here that $P(\mathbf{1}) = 0.005$ and $P(\mathbf{0}) = 0.995$. The probability that there is no code word for the sequence is equal to the probability that the sequence contains more than 3 **1**'s. This is given by

$$\begin{aligned} &P(\text{`more than 3 }\mathbf{1}\text{'s'}) = 1 - P(\text{`at most 3 }\mathbf{1}\text{'s'}) \\ &= 1 - \sum_{k=0}^{3} \binom{100}{k} \cdot 0.005^k \cdot 0.995^{100-k} \approx 0.0017\,. \end{aligned}$$

2. The self information of a sequence $\mathbf{u}_K = [u_1, \ldots, u_K]$ is the sum of K i.i.d. random variables $I(u_k)$, $k = 1, \ldots, K$ with a common mean $H(U)$. Thus,

$$E\left[\frac{I(\mathbf{u}_K)}{K}\right] = E[I(u_k)] = -\frac{3}{4}\log\frac{3}{4} - \frac{1}{4}\log\frac{1}{4} \approx 0.8113\,.$$

To apply the Chebyshev inequality, we also need to evaluate the variance of $I(\mathbf{u}_K)/K$:

$$\begin{aligned}
E[I^2(u_k)] &= \frac{3}{4}\left(\log\frac{3}{4}\right)^2 + \frac{1}{4}\left(\log\frac{1}{4}\right)^2 \approx 1.1292 \\
\sigma^2 &= \mathrm{Var}\left[I(u_k)\right] = E[I^2(u_k)] - (E[I(u_k)])^2 \approx 0.4710 \\
\sigma_K^2 &= \mathrm{Var}\left[\frac{I(\mathbf{u}_K)}{K}\right] = \mathrm{Var}\left[\frac{I(u_1)+\cdots+I(u_K)}{K}\right] = \frac{\sigma^2}{K} = \frac{0.4710}{K}\,.
\end{aligned}$$

We use the Chebyshev inequality in the following form: if X is a random variable with mean μ and variance σ^2, then

$$P(|X-\mu| \geq \epsilon) \leq \frac{\sigma^2}{\epsilon^2}\,.$$

In this case, we have $\epsilon = 0.1$, and we should make K large enough, so that the desired inequality holds. The value K_0 is the smallest integer value of K that satisfies:

$$\frac{\sigma_K^2}{\epsilon^2} \leq \delta(K,\epsilon) = 0.05\,.$$

$$\begin{aligned}
\frac{\sigma^2}{K\epsilon^2} &= \delta(K,\epsilon) \;\Rightarrow\; K = \frac{\sigma^2}{\delta(K,\epsilon)\cdot\epsilon^2} \\
&= \frac{0.4710}{0.05\cdot 0.01} \approx 942.0398 \;\Rightarrow\; K_0 = 943\,.
\end{aligned}$$

3. In the binary Huffman algorithm, the source symbols are first arranged in descending order so that a_1 has the largest probability, and the last symbol has the smallest probability. In the first step, the last bits of the code words corresponding to the two least probable source symbols are set to **0** and **1**. Their probabilities are added together, and the sum is compared to the other probabilities in the next step, when again the two smallest probabilities are chosen and the corresponding branches of the tree are labelled with **0** and **1**. This process continues until there is only one probability $(=1)$ left. The code words corresponding to the source symbols can be read from the tree from right to left. Figure 1 illustrates how the algorithm works in the case of this problem.

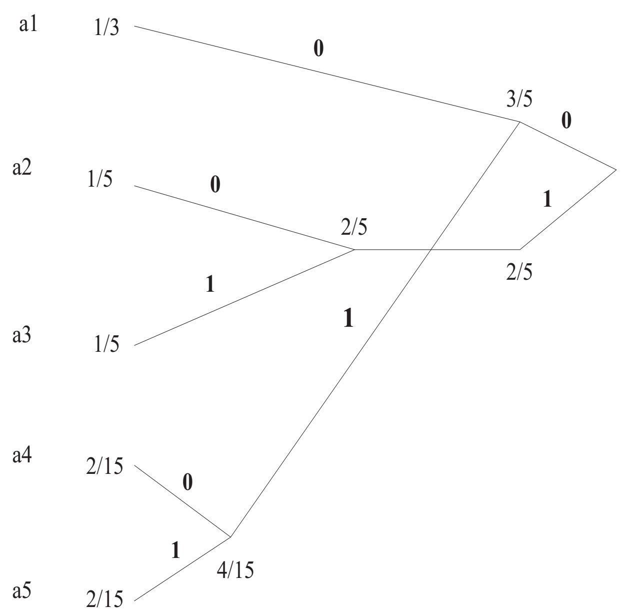

Figure 1: The binary Huffman algorithm

From the figure we see that the code words corresponding to $a_1, \ldots, a_5$ are **00**, **10**, **11**, **010** and **011**. Note that this solution is not unique.

The average length of a code word is given by

$$\bar{n} = \left(\frac{1}{3} + \frac{2}{5}\right) \cdot 2 + \frac{4}{15} \cdot 3 \approx 2.2667\,.$$

4. In the Shannon-Fano algorithm, the probabilities are again arranged in decreasing order. Then the symbols are divided into two groups with approximately equal probability. The first bits of the 'higher' and the 'lower' group are **0** and **1**, respectively (could also be the other way around). This procedure is repeated for the two new groups, and so on until every symbol is represented. Figure 2 shows how code words are determined in this case.

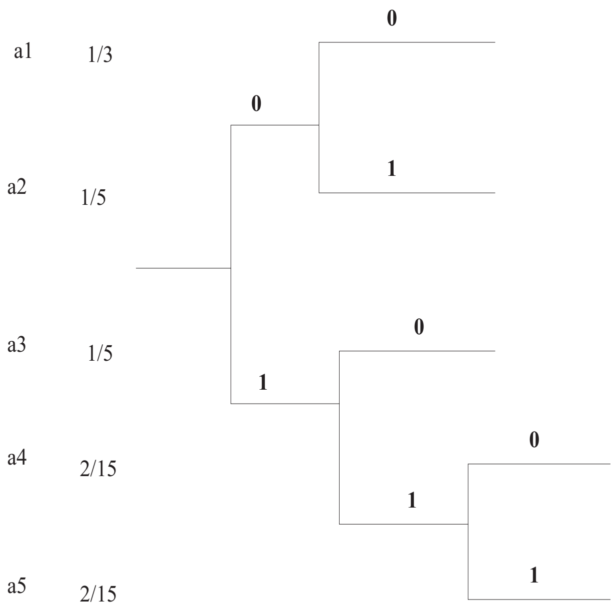

Figure 2: The Shannon-Fano algorithm

The resulting code words for $a_1, \ldots, a_5$ are read from the tree from left to right, and they are **00**, **01**, **10**, **110** and **111**. It is easy to see that $\bar{n} \approx 2.2667$ also in this case. In general, however, the Shannon-Fano algorithm is sub-optimal, whereas the Huffman algorithm is optimal in the sense that it minimizes $\bar{n}$.

5. Figure 3 is a schematic picture of the binary erasure channel.

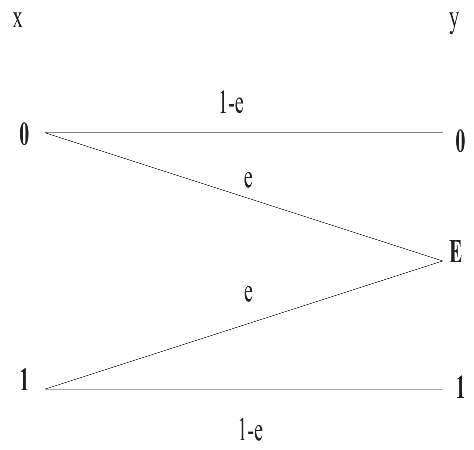

Figure 3: The binary erasure channel

The average mutual information is given by

$$
\begin{aligned}
I(X;Y) &= \sum_{x,y} P(x,y) \log \frac{P(x|y)}{P(x)} \\
&= P_{XY}(0,0) \log \frac{P_{X|Y}(0|0)}{P_X(0)} + P_{XY}(0,E) \log \frac{P_{X|Y}(0|E)}{P_X(0)} \\
&+ P_{XY}(0,1) \log \frac{P_{X|Y}(0|1)}{P_X(0)} + P_{XY}(1,0) \log \frac{P_{X|Y}(1|0)}{P_X(1)} \\
&+ P_{XY}(1,E) \log \frac{P_{X|Y}(1|E)}{P_X(1)} + P_{XY}(1,1) \log \frac{P_{X|Y}(1|1)}{P_X(1)} \\
&= p(1-\epsilon) \log \frac{1}{p} + p\epsilon \log \frac{p}{p} + 0 \cdot \log \frac{0}{p} \\
&+ 0 \cdot \log \frac{0}{1-p} + (1-p)\epsilon \log \frac{1-p}{1-p} + (1-p)(1-\epsilon) \log \frac{1}{1-p} \\
&= (1-\epsilon) \left[p \log \frac{1}{p} + (1-p) \log \frac{1}{1-p} \right].
\end{aligned}
$$

During the differentiation of $I(X;Y)$ we assume that the logarithms are natural:

$$
\begin{aligned}
&\frac{d}{dp} \left\{ \frac{I(X;Y)}{1-\epsilon} \right\} = -\log p - 1 + \log(1-p) + 1 = 0 \\
&\Rightarrow \log \frac{1-p}{p} = 1 \Rightarrow p = \frac{1}{2}.
\end{aligned}
$$

If $p = 1/2$, then

$$\begin{aligned}
I(0;0) &= \log \frac{P_{X|Y}(0|0)}{P_X(0)} = \log 2 = 1 \\
I(0;E) &= \log \frac{P_{X|Y}(0|E)}{P_X(0)} = \log \frac{p}{p} = 0 \\
I(0;1) &= \log \frac{P_{X|Y}(0|1)}{P_X(0)} = \log \frac{0}{p} = 0
\end{aligned}$$

In the same way,

$$I(1;1) = 1 \quad \text{and} \quad I(1;0) = I(1;E) = 0\,.$$