Problem Set 3 Solution

Solutions courtesy of Joy A. Thomas, with editing by Frank R. Kschischang.

5.3 *Slackness in the Kraft Inequality.* Let $\ell_{max} = \max\{\ell_1, \ell_2, \ldots, \ell_m\}$. There are $D^{\ell_{max}}$ sequences of length ℓ_{max}. Of these sequences, $D^{\ell_{max}-\ell_i}$ start with the ith codeword. Because of the prefix condition, any sequence of length $D^{\ell_{max}}$ whose prefix is the ith codeword must be distinct from a sequence of length $D^{\ell_{max}}$ whose prefix is the jth codeword, for any $j \neq i$. Hence the total number of sequences which start with some codeword is $\sum_{i=1}^{m} D^{\ell_{max}-\ell_i} = D^{\ell_{max}} \sum_{i=1}^{m} D^{-\ell_i} < D^{\ell_{max}}$. Hence there are is a set U of sequences of length $\ell_{\max}$ which do not start with any codeword. (This situation can be visualized with the aid of a tree.) These, and all longer sequences having the elements of U as prefixes, cannot be parsed into a concatenation of codewords. More generally, any sequence whose prefix consists of a finite concatenation of codewords followed by a sequence in U cannot be parsed into a concatenation of codewords.

Alternatively, we can map codewords onto D-adic intervals on the real line corresponding to real numbers whose D-ary expansions start with that codeword. Since the length of the interval for a codeword of length ℓ_i is $D^{-\ell_i}$, and $\sum D^{-\ell_i} < 1$, there exists some interval(s) not used by any codeword. The D-ary sequences in these intervals do not begin with any codeword and hence cannot be parsed into a concatenation of codewords.

5.5 *More Huffman codes.* One Huffman code for the source with probabilities $\left(\frac{1}{3}, \frac{1}{5}, \frac{1}{5}, \frac{2}{15}, \frac{2}{15}\right)$ has codewords $C = \{00, 10, 11, 010, 011\}$.

When applied to the uniform source U with probabilities $\left(\frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}\right)$, the code C has average codeword length 12/5. The average codeword length of *any* uniquely decodable code for the uniform source must have an average codeword length of the form $L = \frac{1}{5}\sum_{i=1}^{5} \ell_i = k/5$, for some integer k and must satisfy $L \geq H(U) = \log_2(5) = 2.32$. The smallest such k is $k = 12$, and thus C is optimal for U.

One could also prove the optimality of C by showing that a Huffman code for U has an average length 12/5 bits. (Since each Huffman code produced by the Huffman construction is optimal, they all have the same average length.)

5.9 *Optimal code lengths that require one bit above entropy.* Let X be a Bernoulli random variable with $P[X = 1] = p$. The optimal code has $L = 1$. However by choosing p very close to zero or one, we can have $H(X) \leq \epsilon$, and so $L \geq H(X) + 1 - \epsilon$.

5.12 *Shannon codes and Huffman codes.*

(a) In application of the Huffman procedure, there is no choice in the first step: probability masses 3/12 and 1/12 must be combined to become siblings in the code tree. In the second step, the three remaining masses are all equal, and so there is a choice about which two to combine. One choice leads to codeword lengths $(2, 2, 2, 2)$ and the other leads to codeword lengths $(1, 2, 3, 3)$.

(b) The Shannon length assignments are $(2, 2, 2, 4)$. The second Huffman code assigns a codeword of length 3 to the mass of probability $1/4$, which exceeds the Shannon length assignment. Thus the Huffman codeword for a particular symbol may be longer than the Shannon codeword for that symbol, but, on average, the Huffman code cannot have longer codeword lengths than the Shannon code. This example also shows that the Shannon length assignment is suboptimal in general.

5.24 *Optimal codes for uniform distributions.*

(a) After trying a few examples, you will be convinced that the Huffman procedure will always produce optimal codes in which the longest codewords are at most one bit longer than the shortest codewords.

To prove this formally, suppose an optimal prefix code C for the uniform source contains two codewords $C(x_i)$ and $C(x_j)$ such that $l(C(x_i)) - l(C(x_j)) \geq 2$. Thus $C(x_j)$ is at least two bits shorter than $C(x_i)$. Create a new code C' such that $C'(x_i) = C(x_j)0$ and $C'(x_j) = C(x_j)1$. In other words C' is obtained by splitting the shorter codeword of C into two codewords, each being one bit longer. The code C' is still a prefix code, in which the length of the shorter codeword from C has increased by one and the length of the longer codeword from C has decreased by at least one. Since the source symbols are equally likely, it follows that $L' \leq L$. By this method, we can transform any optimal code into a code in which the length of the shortest and longest codewords differ by at most one bit.

Let ℓ be the length of the shortest codewords in such an optimal code, and suppose that the code contains t codewords of length $\ell + 1$, where $0 \leq t < m$. Then the code contains $m - t$ codewords of length ℓ. Since the Huffman procedure never requires dummy symbols in the binary case, every optimal binary code satisfies Kraft's inequality with equality, so we must have $(m-t)2^{-\ell} + t2^{-(\ell+1)} = 1$, which implies that $2(m-t) + t = 2^{\ell+1}$, i.e., $t = 2(m - 2^{\ell})$. Since $t \geq 0$, we have $m \geq 2^{\ell}$, and since $t < m$, we have $m < 2^{\ell+1}$. Thus $2^{\ell} \leq m < 2^{\ell+1}$, from which we deduce that $\ell = \lfloor \log_2 m \rfloor$. The average codeword length is then

$$\begin{aligned} L_m &= \frac{1}{m}\left((m-t)\ell + t(\ell+1)\right) \\ &= \frac{m\ell + t}{m} \\ &= \ell + \frac{t}{m} \\ &= \lfloor \log_2 m \rfloor + \frac{2}{m}\left(m - 2^{\lfloor \log_2 m \rfloor}\right). \end{aligned}$$

(b) To have $L_m = \log_2 m$ requires that $m - 2^{\lfloor \log_2 m \rfloor} = 0$ which occurs if and only if $m = 2^k$ for some integer k.

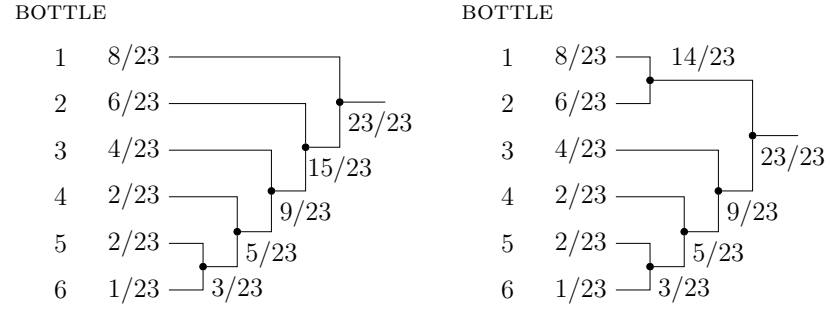
(c) Write $m = 2^\ell + r$ where $0 \le r < 2^\ell$, so that $t = 2r$. Then $L_m = \ell + \frac{2r}{m}$ and

$$\begin{aligned}\rho &= L_m - \log_2 m \\ &= \ell + \frac{2r}{m} - \log_2(m) \\ &= \ell + \frac{2r}{2^\ell + r} - \log_2(2^\ell(1 + r/2^\ell)) \\ &= 2\frac{r/2^\ell}{1 + r/2^\ell} - \log_2(1 + r/2^\ell)\end{aligned}$$

Consider the function $f(x) = \frac{2x}{1+x} - \ln(1+x)/\ln(2)$ over the range $x \in [0, 1)$. We have $f'(x) = \frac{\ln(4)-1-x}{(1+x)^2 \ln(2)}$ and $f''(x) = \frac{1+x-4\ln(2)}{(1+x)^3 \ln(2)}$. Since $f''(x) < 0$ for $x \in [0, 1)$, we see that $f(x)$ is concave, reaching a maximum when $f'(x) = 0$ at $x = \ln(4) - 1 \approx 0.386294$. Thus the worst case redundancy is achieved when $r/2^\ell$ is close to $\ln(4) - 1$, i.e., when $m = 2^\ell(1 + r/2^\ell)$ is close to $2^\ell \ln(4) \approx (1.38629)2^\ell$.

(d) At $x = \ln(4) - 1$, we have $f(x) = \frac{\ln(4)-\ln(\ln(4))-1}{\ln 2} \approx 0.0860713$. This is achieved with arbitrary accuracy as $\ell \to \infty$, by setting $m \approx \ln(4)2^\ell$. (The quantity $\frac{\ln(4)-\ln(\ln(4))-1}{\ln 2}$ is one of the lesser fundamental constants of the universe. See R. G. Gallager, "Variations on a theme by Huffman," *IEEE Trans. Info. Theory*, vol. IT-24, pp. 668–674, 1978.)

5.32 *Bad wine.* A tasting is a binary test: either the bad wine is in the glass, or it is not, in which case all wines in the glass are eliminated. A sequence of tastings that never re-tests wines that have been eliminated in a previous tasting can be represented by a binary tree in which the wines are the leaf nodes. Each interior vertex of the tree represents a tasting of a mixture of the leaf nodes that descend from the left (or, equivalently, from the right) of that node. For example, a linear search (as in the first part of the question) would be represented by a tree in which each left descendant of an interior vertex is a leaf, as shown below. Tastings begin from the root of the tree. If the bad wine is detected in a given tasting, the left branch is descended; otherwise, the right branch is descended. The tastings terminate when a leaf node is reached, at which point one concludes that the corresponding bottle contains bad wine. The expected number of tastings required to reach a conclusion is equal to the expected depth of the leaf nodes in the tree, which is equal to the sum of the interior vertex weights.



(a) linear search (b) optimal search

(a) The first bottle to be tasted should be the one with highest probability of being bad (namely, the one with probability 8/23).

(b) Taste the bottles in order of decreasing probability. The expected number of tastings required is the sum of the interior node weights, namely

$$\frac{1}{23}(3+5+9+15+23) = \frac{55}{23} \approx 2.39.$$

This tasting order clearly minimizes the sum of the inter node weights under the constraint that each left descendant of an interior node must be a leaf.

(c) The Huffman algorithm minimizes the sum of the interior node weights. Using the Huffman tree shown above results in an expected number of tastings equal to

$$\frac{1}{23}(3+5+9+14+23) = \frac{54}{23} \approx 2.35.$$

(d) According to the tree, we should taste a mixture of bottles 1 and 2 first (or, equivalently, a mixture of bottles 3, 4, 5, and 6, which might be preferable since it is less likely to taste bad).

4.1 *Doubly stochastic matrices.*

(a) Since the entries of $\mathbf{a}$ and P are non-negative, so are the entries of $\mathbf{b}$. Furthermore,

$$\sum_{i=1}^{n} b_i = \mathbf{b}(1,1,\ldots,1)^T = \mathbf{a}P(1,1,\ldots,1)^T = \mathbf{a}(1,1,\ldots,1)^T = 1,$$

where the second-last equality follows from the property that all rows of P sum to one, and the last equality follows from the property that $\mathbf{a}$ is itself a probability vector. Thus $\mathbf{b}$ is a probability vector. Next observe that

$$\begin{aligned}
H(b_1,\ldots,b_n) - H(a_1,\ldots,a_n) &= \sum_{i=1}^{n} a_i \log a_i - \sum_{j=1}^{n} b_j \log b_j \\
&= \sum_{i=1}^{n} a_i \log a_i - \sum_{j=1}^{n}\sum_{i=1}^{n} a_i P_{ij} \log b_j \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} a_i P_{ij} \log a_i - \sum_{i=1}^{n}\sum_{j=1}^{n} a_i P_{ij} \log b_j \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} a_i P_{ij} \log \frac{a_i P_{ij}}{b_j P_{ij}} \\
&\geq \left(\sum_{i=1}^{n}\sum_{j=1}^{n} a_i P_{ij}\right) \log \frac{\left(\sum_{i=1}^{n}\sum_{j=1}^{n} a_i P_{ij}\right)}{\left(\sum_{i=1}^{n}\sum_{j=1}^{n} b_j P_{ij}\right)} \\
&= 1 \log \frac{1}{1} \\
&= 0,
\end{aligned}$$

where the inequality follows from the log sum inequality.

(b) Let $\mu = \frac{1}{n}(1, 1, \ldots, 1)$. Then

$$\mu P = \frac{1}{n}(1, 1, \ldots, 1)P = \frac{1}{n}(1, 1, \ldots, 1) = \mu,$$

which shows that μ is a stationary distribution.

(c) Let μ be defined as above and let P be a stochastic matrix. As such, the rows of P are probability vectors. If $\mu = \mu P$, then

$$\frac{1}{n} = \mu_j = \sum_{i=1}^{n} \mu_i P_{ij} = \frac{1}{n} \sum_{i=1}^{n} P_{ij}$$

or $\sum_{i=1}^{n} P_{ij} = 1$, which implies that the columns of P are also probability vectors and hence P is doubly stochastic.

4.7 *Entropy rates of Markov chains.*

(a) The stationary distribution $\mu = (\mu_0, \mu_1)$ must satisfy $\mu P = \mu$ subject to the constraint that $\mu_0 + \mu_1 = 1$. We solve to find that

$$\mu_0 = \frac{p_{10}}{p_{01} + p_{10}}, \quad \mu_1 = \frac{p_{01}}{p_{01} + p_{10}}.$$

Therefore the entropy rate is

$$\begin{aligned} H(X_2 \mid X_1) &= \mu_0 H(p_{01}, 1 - p_{01}) + \mu_1 H(p_{10}, 1 - p_{10}) \\ &= \frac{p_{10}\mathcal{H}(p_{01}) + p_{01}\mathcal{H}(p_{10})}{p_{01} + p_{10}}, \end{aligned}$$

where $\mathcal{H}$ denotes the binary entropy function.

(b) The entropy rate is at most 1 bit because the process has only two states. This rate can be achieved if (and only if) $p_{01} = p_{10} = 1/2$, in which case the process is actually i.i.d. with $\Pr(X_i = 0) = \Pr(X_i = 1) = 1/2$.

(c) Setting $p_{01} = p$ and $p_{10} = 1$ we get

$$H(X_2 \mid X_1) = \frac{\mathcal{H}(p)}{1 + p}.$$

(d) Note that

$$\frac{\mathrm{d}}{\mathrm{d}p}\mathcal{H}(p) = \log \frac{1 - p}{p},$$

thus

$$\frac{\mathrm{d}}{\mathrm{d}p}\frac{\mathcal{H}(p)}{1 + p} = \frac{2\log(1 - p) - \log(p)}{(1 + p)^2}.$$

The derivative is zero when $(1 - p)^2 = p$, i.e., when $p = \frac{3-\sqrt{5}}{2} \approx 0.381966$. The maximum entropy rate is then

$$H(X_2 \mid X_1) = \frac{2\mathcal{H}((3 - \sqrt{5})/2)}{5 - \sqrt{5}} \approx 0.694242 \text{ bit}$$

(e) The Markov chain of part (c) forbids consecutive ones. Let $N(t)$ be the number of allowable sequences of length t. Then $N(1) = 2$ since $\{0, 1\}$ are allowable sequences of length 1 and $N(2) = 3$ since $\{00, 01, 10\}$ are the only allowable sequences of length two. Now consider the set of allowable sequences of length $t > 2$, of which there are $N(t)$. This set can be partitioned into two disjoint subsets: sequences that start with a 1 and sequences that start with a 0. If the first symbol of a sequence is 1, then the next symbol must be 0, and the following $t - 2$ symbols can be any of the $N(t - 2)$ allowable sequences. Thus there are $N(t - 2)$ sequences in the first subset. If the first symbol is 0, then the following $t - 1$ symbols can be any of the $N(t - 1)$ allowable sequences. Thus there are $N(t - 1)$ sequences in the second subset. It follows that the number $N(t)$ of allowable sequences of length t satisfies the recurrence

$$N(t) = N(t - 1) + N(t - 2).$$

From the initial conditions $N(1) = 2$ and $N(2) = 3$ we find that

$$N(3) = 5,\ N(4) = 8,\ N(5) = 13,\ N(6) = 21,\ N(7) = 34, \ldots$$

(These are the Fibonacci numbers!)
The sequence $N(t)$ grows exponentially, that is, $N(t) \approx c\lambda^t$, where λ is the maximum magnitude solution of the characteristic equation

$$1 = z^{-1} + z^{-2}\,.$$

Solving the characteristic equation yields $\lambda = (1+\sqrt{5})/2$, the Golden Ratio. (The sequence $\{N(t)\}$ is the sequence of Fibonacci numbers.) Therefore

$$H_0 = \lim_{n\to\infty} \frac{1}{t} \log N(t) = \log(1 + \sqrt{5})/2 = 0.694 \text{ bit}\,.$$

Since there are only $N(t)$ possible outcomes for $X_1, \ldots, X_t$, an upper bound on $H(X_1, \ldots, X_t)$ is $\log N(t)$, and so the entropy rate of the Markov chain of part (c) is at most H_0. In fact, we saw in part (d) that this upper bound can be achieved.

4.12 *Entropy rate of a dog looking for a bone.*

(a) By the chain rule,

$$\begin{aligned} H(X_0, X_1, \ldots, X_n) &= H(X_0) + H(X_1 \mid X_0) + H(X_2 \mid X_0, X_1) + \cdots \\ &\quad + H(X_n \mid X_0, X_1, \ldots, X_{n-1}) \\ &= H(X_0) + H(X_1 \mid X_0) + \sum_{i=2}^{n} H(X_i \mid X_{i-1}, X_{i-2}), \end{aligned}$$

since, for $i \geq 2$, the next position depends only on the previous two. We have $H(X_0) = 0$ (the initial position is deterministic) and $H(X_1 \mid X_0) = 1$ (the first step is equally likely to be positive or negative). Furthermore, for $i \geq 2$,

$$H(X_i \mid X_{i-1}, X_{i-2}) = H\left(\frac{1}{10}, \frac{9}{10}\right).$$

Therefore

$$H(X_0, X_1, \ldots, X_n) = 1 + (n-1)\mathcal{H}\left(\frac{1}{10}\right),$$

where $\mathcal{H}$ denotes the binary entropy function.

(b) From (a)

$$\begin{aligned}\frac{H(X_0, X_1, \ldots X_n)}{n+1} &= \frac{1 + (n-1)\mathcal{H}(1/10)}{n+1} \\ &\to \mathcal{H}(1/10) \text{ as } n \to \infty.\end{aligned}$$

(c) The dog must take at least one step to establish the direction of travel from which it ultimately reverses. Letting S be the number of steps taken until the first reversal. Then $S = i$ with probability $p(1-p)^{i-1}$ for $i = 1, 2, \ldots$, i.e., S is geometrically distributed with parameter $p = 1/10$. From a probability course we know that $E(S) = 1/p = 10$. More explicitly, note that

$$E(S) = \sum_{i=1}^{\infty} ip(1-p)^{i-1} = p + \sum_{i=1}^{\infty}(i+1)p(1-p)^i$$

and

$$(1-p)E(S) = \sum_{i=1}^{\infty} ip(1-p)^i$$

thus, subtracting, we find

$$pE(S) = p + \sum_{i=1}^{\infty} p(1-p)^i$$

or

$$E(S) = 1 + \sum_{i=1}^{\infty}(1-p)^i = \frac{1}{p} = 10.$$

Starting at time 0, the expected number of steps to the first reversal is 11.

4.33 *Chain inequality.* First, Markovity implies that

$$I(X_2; X_3|X_4) \geq I(X_1; X_3|X_4). \tag{1}$$

In fact, the preceding can be derived by expanding $I(X_1, X_2; X_3|X_4)$ in two different ways, as in the data-processing inequality.

Now,

$$I(X_2; X_3|X_4) = H(X_2|X_4) - H(X_2|X_3, X_4) = H(X_2|X_4) - H(X_2|X_3),$$

where the second equality follows by Markovity, and similarly

$$I(X_1; X_3|X_4) = H(X_1|X_4) - H(X_1|X_3).$$

Finally,

$$I(X_2; X_3|X_4) = H(X_2|X_4) - H(X_2|X_3) = I(X_2; X_3) - I(X_2; X_4)$$

and likewise

$$I(X_1; X_3|X_4) = H(X_1|X_4) - H(X_1|X_3) = I(X_1; X_3) - I(X_1; X_4).$$

Finally, substituting these into (1), we have

$$I(X_2; X_3) + I(X_1; X_4) \geq I(X_1; X_3) + I(X_2; X_4)$$

as desired.