# CS3236: Solutions to Tutorial 2
# (Symbol-Wise Source Coding)

## Part I – Decodability, Kraft Inequality, and Shannon-Fano Coding

1. **[Unique Decodability]**

   (a) Is the code $\{00, 11, 0101, 111, 1010, 100100, 0110\}$ uniquely decodeable?

   **Solution.** *No, the code is not uniquely decodeable. The sequence 11111 can be parsed as c(2)c(4) or c(4)c(2).*

   (b) Is the ternary code $\{00, 012, 0110, 0112, 100, 201, 212, 22\}$ uniquely decodeable?

   **Solution.** *Yes, the code is uniquely decodeable. The code is prefix-free, thus it is uniquely decodeable.*

| Symbol | Code $\mathcal{C}$ | Code $\mathcal{D}$ |
|---|---|---|
| $a_1$ | 1 | 10 |
| $a_2$ | 10 | 1 |
| $a_3$ | 100 | 01 |
| $a_4$ | 1000 | 001 |

   (c) Is the code $\mathcal{C}$ shown above uniquely decodeable?

   **Solution.** *Yes, the code is uniquely decodeable. Even though it is not prefix-free, 1 acts as a delimiter indicating the start of a codeword.*

   *(Alternatively, since the code is* **suffix-free***, we can read the output $C(x_1)\dots C(x_n)$ in reverse order and decode it like a prefix-free code. However, this requires receiving all symbols before decoding anything, which may be undesirable compared to "instantaneous" decoding.)*

   (d) Is the code $\mathcal{D}$ shown above uniquely decodeable?

   **Solution.** *No, the code is not uniquely decodeable. Since $\frac{1}{2^2} + \frac{1}{2^1} + \frac{1}{2^2} + \frac{1}{2^3} = \frac{9}{8} > 1$, it violates the Kraft inequality.*

2. **[Shannon-Fano Code Example]**

   Let $\mathcal{X} = \{1, 2, 3, 4\}$, with the symbols having associated probabilities $\{0.4, 0.3, 0.2, 0.1\}$.

   (a) Compute the lengths of the Shannon-Fano code.

   **Solution.** *Substituting $\ell(x) = \lceil \log_2 \frac{1}{P_X(x)} \rceil$ gives lengths $\{2, 2, 3, 4\}$.*

   (b) Construct a prefix-free code having those lengths.

   **Solution.** *There are many ways to do this; one example is* 00, 01, 100, 1010.

   (c) Is this the optimal prefix-free code in terms of average length?

   **Solution.** *Clearly not, because we can remove the last bit from the length-4 codeword (hence reducing the average length) and still have a prefix-free code.*

3. **[Optimal Prefix-Free Code]**

Let the probability distribution of random variable $X$ over an alphabet $\mathcal{X}$ be $P_X = \{p_1, p_2, ..., p_N\}$ where $N = |\mathcal{X}|$ and $p_1 < p_2 < .... < p_N$ . Let $C(X)$ be the optimal prefix-free code with codeword lengths $\{\ell_1, \ell_2, ..., \ell_N\}$.

(a) Show that the codeword lengths satisfy $\ell_1 \geq \ell_2 \geq .... \geq \ell_N$.

*(Hint: You might want to try a proof by contradiction.)*

**Solution.** *Suppose there exists an optimal prefix-free code $C'(X)$ with codeword lengths $\{\ell_1, \ell_2, ...., \ell_N\}$ that do not satisfy $\ell_1 \geq \ell_2 \geq .... \geq \ell_N$. Then there exists $\ell_i < \ell_j$ for some $i < j$ and $i, j \in \{1, 2, 3, .., N\}$. Let $x_i$ and $x_j$ be the corresponding symbols.*

*Consider exchanging the codewords of $x_i$ and $x_j$ in $C'(X)$ and call the new prefix-free code $C''(X)$. Since $i < j$ which implies $p_i < p_j$, and therefore $p_i(\ell_j - \ell_i) < p_j(\ell_j - \ell_i)$. Hence, $p_i\ell_j + p_j\ell_i < p_i\ell_i + p_j\ell_j$ which implies that $L(C'', X) < L(C', X)$ contradicting the optimality of the prefix-free code $C'(X)$.*

*This proof by contradiction means that the codeword lengths must satisfy $\ell_1 \geq \ell_2 \geq .... \geq \ell_N$ whenever $p_1 < p_2 < .... < p_N$.*

(b) Further show that $\ell_1 = \ell_2$.

*(Hint: Again, proof by contradiction is useful.)*

**Solution.** *From part (a) we know that codeword lengths must satisfy $\ell_1 \geq \ell_2 \geq .... \geq \ell_N$.*

*Suppose there exists an optimal prefix-free code $C(X)$ satisfying $\ell_1 > \ell_2 \geq .... \geq \ell_N$. Now, consider the codeword with largest length $\ell_1$, and let $\ell_2$ be the second-largest length. Since $C$ is prefix-free, if we remove the final $\ell_2 - \ell_1$ bits from the longest codeword, we must still have a prefix-free code (since those removed bits have no impact on the answer to the question "is [some other codeword] a prefix of [this codeword]?"). Let $C'(X)$ be the resulting modified code.*

*Now notice the codeword lengths of $C'(X)$ are $\ell_2 = \ell_2 \geq .... \geq \ell_N$. Also, notice that new code $C'(X)$ is prefix-free. It is easy to see that $L(C', X) < L(C, X)$ contradicting the optimality of the prefix-free code $C(X)$.*

*Thus $\ell_1 = \ell_2$.*

4. **[The Poisoned Glass]**

'Mathematicians are curious birds', the police commissioner said to his wife. 'You see, we had all those partly filled glasses lined up in rows on a table in the hotel kitchen. Only one contained poison, and we wanted to know which one before searching that glass for fingerprints. Our lab could test the liquid in each glass, but the tests take time and money, so we wanted to make as few of them as possible by simultaneously testing mixtures of small samples from groups of glasses. The university sent over a mathematics professor to help us. He counted the glasses, and thought of doing a binary search strategy,[1] smiled and said: "Pick any glass you want, Commissioner. We'll test it first."
"But won't that waste a test?" I asked.
"No," he said, "There are 65 glasses, and 64 is a much nicer number to work with. Let's test a one glass first – it doesn't matter which one – and if it doesn't turn out to be the poisoned one, then we can test the rest using a binary search." '

Create a strategy suggested by the mathematics professor which includes picking the first glass and testing it on its own, followed by a binary search on the remaining 64 if needed. Also, consider an alternative strategy that does binary search right from the start with rounding, splitting 65 glasses into groups of size 32 and 33, then either 33 into 16 and 17 **or** 32 into 16 and 16, and so on.

[1] Binary search works as follows: Split the $N$ glasses into 2 groups of size $N/2$ (or sizes $\lfloor N/2 \rfloor$ and $\lceil N/2 \rceil$ if $N$ is an odd number). Mix all the liquids in one group together and test it – if it's poisonous, we know that group has the poison glass, and otherwise, it's the other group that has it. Recursively apply this procedure on the group with the poison until only one glass remains. It is easy to show that this requires $\lceil \log_2 N \rceil$ tests.

Calculate the average number of tests of each strategy, assuming the poison was placed uniformly at random into one of the 65 glasses. What does this have to do with source coding?

**Solution.** *The mathematics professor thinks that testing one glass does not matter, and he thought of binary-search-like strategy: Testing of a mixture of liquid from half of the glasses and discarding half each time – until only one remains. Such a strategy works nicely with powers of* 2 *because then there are no rounding issues from trying to halve odd numbers.*

*Based on the professor's strategy above, there is a* $\frac{1}{65}$ *probability it takes 1 test, and a* $\frac{64}{65}$ *probability it takes 1 + 6 = 7 tests. Thus the average number of tests is* $\frac{1}{65} \cdot 1 + \frac{64}{65} \cdot 7 = \frac{449}{65} \approx 6.9076$.

*On the other hand, consider the alternative strategy described above. A moment's thought reveals that if we ever split an odd remaining number* $N$ *into* $\frac{N-1}{2}$ *and* $\frac{N+1}{2}$ *and proceed with the former (the smaller one), then we find the glass after* 6 *tests. For instance, the number of glasses remaining might change as follows:*

- $\mathbf{65} \to \mathbf{32} \to 16 \to 8 \to 4 \to 2$
- $65 \to 33 \to \mathbf{17} \to \mathbf{8} \to 4 \to 2$

*There are also two glasses that, if poisoned, would lead to the pattern* $65 \to 33 \to 17 \to 9 \to 5 \to 3 \to 2$, *for a total of* 7 *tests. But the other 63 glasses, if poisoned, only require* 6 *tests. This means that the average number of tests is* $\frac{2}{65} \cdot 7 + \frac{63}{65} \cdot 6 = \frac{392}{65} \approx 6.0307$, *which is a fair bit smaller than the professor's strategy.*

*This problem is in fact source coding in disguise: Consider a random variable* $X$ *taking values on* $\{1, \ldots, 65\}$ *uniformly at random. A source coding strategy maps each* $X$ *to a binary sequence, like* $37 \to 10000$ *or* $32 \to 1100$. *Now consider assigning these codewords to glasses in the poison glass problem, and then sequentially using the following test strategy:*

*(i) Test the mixture of the glasses whose codeword has '1' in the next bit*

*(ii) If poison is detected, keep only those glasses just tested and discard the rest; otherwise, discard them and keep only the other glasses (the ones with '0' in that bit). Then return to Step (i) (unless only 1 glass remains, in which case we are done).*

*Thus, the tests give us the answers to the questions "Is the first codeword bit a* 1*?", 'Is the second codeword bit a* 1*?", and so on, and if the code is prefix-free, these questions will lead to uniquely identifying the glass (or codeword). The average number of tests will exactly be the average code length.*

5. **[Proof of Existence Theorem for Kraft's Inequality]**

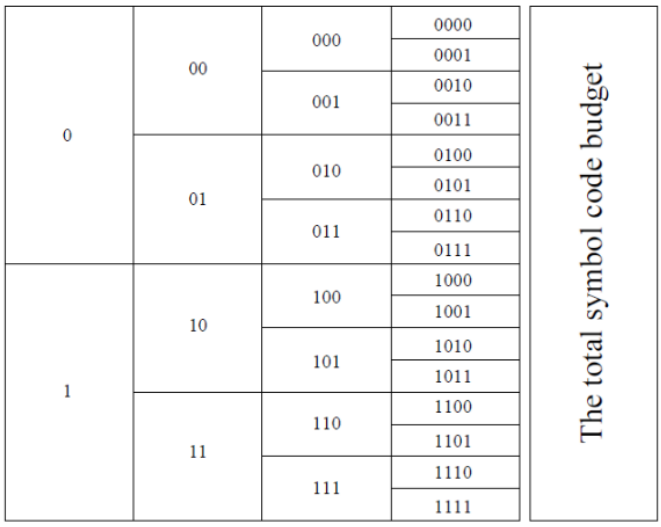
In the Lecture, we have discussed that for any uniquely decodable code $C(X)$ for a random variable $X$ over the alphabet $\mathcal{X}$, the codeword lengths must satisfy: $\sum_{x \in \mathcal{X}} 2^{-\ell(x)} \leq 1$.

For notational convenience, let $\mathcal{X} = \{1, \ldots, N\}$, and denote the corresponding lengths by $\{\ell_1, \ldots, \ell_N\}$. Prove that if these lengths satisfy $\sum_{i=1}^{N} 2^{-\ell_i} \leq 1$, then there exists a prefix free code with these codeword lengths $\{\ell_1, \ldots, \ell_N\}$.

*(Hint: Try to use Figure 1 as a guide and show that suitable codeword allocations can be done without running out of space. Start with the shortest codewords. You could try a few simple examples before trying the general case.*

**Solution.** *We wish to prove that for any set of codeword lengths* $\{\ell_i\}$ *satisfying the Kraft's inequality, there is a prefix code having those lengths. This is proved by thinking of the codewords illustrated in Figure 1 as being in a 'codeword supermarket', with size indicating cost.*

*We imagine purchasing codewords one at a time, starting from the shortest codewords (i.e., the biggest purchases), using the budget shown at the right of Figure 1. We start at one side of the codeword supermarket, say the top, and purchase the first codeword of the required length* $\ell$. *We advance down*




The codeword supermarket and the symbol coding budget. The 'cost' $2^{-l}$ of each codeword (with length $l$) is indicated by the size of the box it is written in. The total budget available when making a uniquely decodeable code is 1.

Figure 1: Illustration of the "supermarket budget".

*the supermarket a distance* $2^{-\ell}$*, and purchase the next codeword of the next required length, and so forth.*

*Because the codeword lengths are getting longer, and the corresponding intervals are getting shorter, we can always buy an adjacent codeword to the latest purchase, so there is no wasting of the budget. Thus at the I'th codeword we have advanced a distance* $\sum_{i=1}^{N} 2^{-\ell_i}$ *down the supermarket; if* $\sum_{i=1}^{I} 2^{-\ell_i} \leq 1$*, we will have purchased all the codewords without running out of budget.*

6. **[Shannon-Fano Code Length Bound]** It is known that the Shanon-Fano code achieves an average length $L(C)$ satisfying

$$H(X) \leq L(C) < H(X) + 1,$$

for *any* source distribution $P_X$. Explain why when making such a claim for arbitrary $P_X$, we cannot get a better (i.e., smaller) constant than 1 on the right-hand side, *even if we use an optimal symbol-wise code.*

**Solution.** *Note that any prefix-free code must have an average length of at least one, because there are no length-zero codewords (except perhaps for symbols that have zero probability and hence don't exist for all intents and purposes). However,* $H(X)$ *can be arbitrarily close to zero, e.g., via a binary source with probabilities* $(1-\epsilon, \epsilon)$ *and an extremely small value of* $\epsilon$*. So in the worst case, even the optimal code may have an average length only very slightly less than* 1 *bit higher than the entropy.*

# Part II – Huffman Coding

7. **[Optimal Codes with Different Lengths]**

Find a probability distribution $\{p_1, p_2, p_3, p_4\}$ such that there are two optimal prefix-free codes whose *unordered lists of lengths* assigned are different (e.g., lengths $1, 2, 3, 4, 4$ and $4, 1, 4, 2, 3$ would be considered the same "list of lengths" because they are the same numbers in a different order, whereas lengths $1, 2, 3, 4, 4$ and $2, 2, 2, 3, 3$ would be considered two different lists of lengths).

*(Hint: Try the Huffman algorithm and see if two solutions arise.)*

**Solution.** *The set of probabilities* $\{p_1, p_2, p_3, p_4\} = \{\frac{1}{3} - p, p, \frac{1}{3}, \frac{1}{3}\}$ *where* $0 \leq p \leq \frac{1}{3}$ *gives rise to two different optimal sets of codelengths, because at the second step of the Huffman coding algorithm we can choose any of the three possible pairings. We may either put them in a constant length code* $\{00, 01, 10, 11\}$ *or the code* $\{000, 001, 01, 1\}$*. Both codes have expected length* 2*.*

8. [**(Im)possible Huffman Codes**]: Which of these codes cannot be Huffman codes for any probability assignment?

   (a) $\{0, 10, 11\}$.
   **Solution.** $\{0, 10, 11\}$ *is a Huffman code for the distribution* $(1/2, 1/4, 1/4)$*.*

   (b) $\{00, 01, 10, 110\}$.
   **Solution.** $\{00, 01, 10, 110\}$ *is not a Huffman code because the average length could be reduced by changing the last codeword to* 11*.*

   (c) $\{01, 10\}$.
   **Solution.** *The code* $\{01, 10\}$ *can be shortened to* $\{0, 1\}$ *without losing its instantaneous property, and therefore is not optimal and not a Huffman code.*

9. [**Huffman Code Length**]

   Consider a source $X$ with four symbols $\mathcal{X} = \{a_1, a_2, a_3, a_4\}$ and probability distribution $P_X = \{p_1, p_2, p_3, p_4\}$. Let $\ell_i$ be the length of the codeword associated with symbol $a_i$, generated by a Huffman code applied to the source $X$.

   Suppose $p_1 \geq p_2 = p_3 = p_4$. Find the smallest value of $q$ such that $p_1 > q$ implies $\ell_1 = 1$.

   **Solution.** *Note that* $p_2 = p_3 = p_4 = \frac{1-p_1}{3}$*. Without loss of generality, the Huffman algorithm will first combine symbols* $a_4$ *and* $a_3$*, creating a super-symbol with probability* $\frac{2}{3}(1 - p_1)$*, which is larger than* $\frac{1-p_1}{3}$*.*

   *For symbol* $a_1$ *to be assigned a codeword of length 1, we need the Huffman algorithm to combine the super-symbol and* $a_2$*. Therefore, we must have* $p_1 > \frac{2}{3}(1 - p_1)$*, or equivalently,*

$$p_1 > q = \frac{2}{5}\,.$$

10. [**Huffman Code with Ideal Code Lengths**]

    Consider a source $X$ with $n$ symbols $\mathcal{X} = \{a_1, a_2, a_3..., a_n\}$ and probability distribution $P_X(a_i) = 2^{-i}$ for $0 < i < n$ and $P_X(a_n) = 2^{-n+1}$. Let $l_i$ be the length of the codeword associated with symbol $a_i$, generated by a Huffman code applied to the source $X$.

    (a) Show that the entropy of the source $X$ is given by $2(1 - 2^{-n+1})$.
    *(Hint: The identity* $\sum_{i=1}^{i=n-1} 2^{-i} i = 2^{-n+1} \cdot (2^n - n - 1)$ *is useful.)*
    **Solution.** *We have*

$$\begin{aligned}
H(X) &= \sum_{a_i \in \mathcal{X}} P_X(a_i) \log\left(\frac{1}{P_X(a_i)}\right) \\
&= \sum_{i=1}^{i=n-1} P_X(a_i) \log\left(\frac{1}{P_X(a_i)}\right) + P_X(a_n) \log\left(\frac{1}{P_X(a_n)}\right) \\
&= \sum_{i=1}^{i=n-1} 2^{-i} \log\left(\frac{1}{2^{-i}}\right) + 2^{-n+1} \log\left(\frac{1}{2^{-n+1}}\right) \\
&= \sum_{i=1}^{i=n-1} 2^{-i} i + 2^{-n+1}(n-1) \\
&= 2(1 - 2^{-n+1}),
\end{aligned}$$

*where the last step uses the hint.*

(b) Give one explicit form for the codeword associated with symbol $a_i$, generated by a Huffman code. In other words, what is the length of the codeword associated with symbol $a_i$?

**Solution.** *Notice that probabilities are arranged in descending order and also,* $P_X(a_i) \geq P_X(a_{i+1}) + P_X(a_{i+2})$. *Hence, the Huffman code contruction is the following as shown in Figure* (1) *:*

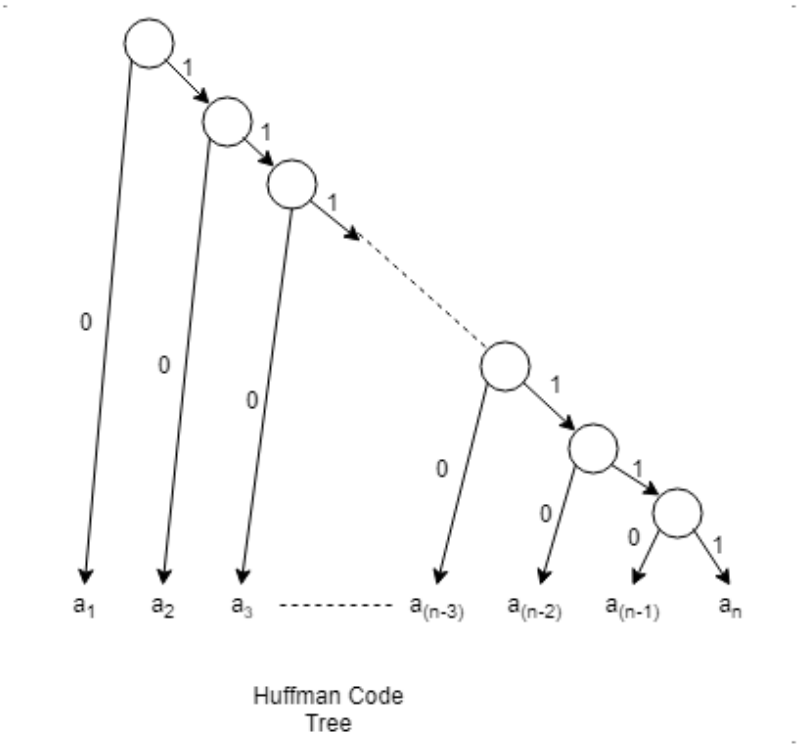



Figure 2:

*Hence, one explicit form for the codeword associated with symbol* $a_i$ *is*

$$\underbrace{1\ldots1}_{i-1 \text{ times}}\ 0$$

*for* $0 < i < n$, *with the final symbol* $a_n$ *being*

$$\underbrace{1\ldots1}_{n-1 \text{ times}}\ .$$

*In this case, the length of the codeword associated with symbol* $a_i$ *is* $i$ *for* $0 < i < n$ *and for* $a_n$ *is* $n-1$.

(c) Calculate the expected length of the Huffman code for the source $X$ that you mention in part (b). Compare with the entropy of the source $X$.

**Solution.** *We have*

$$\begin{aligned}\mathbb{E}[\ell(X)] &= \sum_{i=1}^{i=n-1} P_X(a_i)(l_i) + P_X(a_n)(l_n) \\ &= \sum_{i=1}^{i=n-1} 2^{-i} i + 2^{-n+1}(n-1) \\ &= 2(1-2^{-n+1})\,.\end{aligned}$$

*The entropy of the source is exactly equal to the expected length of the Huffman code.*

11. **[Yes/No Question Game]**

Let the random variables $D_1$ and $D_2$ be independent and identically distributed uniformly in $\{1,2,3,4,5,6\}$. Let $D = |D_1 - D_2|$. Alice knows $D$. Your task is to ask Alice a sequence of questions with yes/no answers to find out $D$. For instance, you may ask questions like "Is $D$ equal to 0?" and "Is $D$ one of the values in $\{0,-2,3\}$?".

Devise a strategy that achieves the minimum possible average number of questions. Calculate the average number of questions for your strategy.

*(Hint: This is source coding in disguise.)*

**Solution.** *By direct calculations we see that* $\Pr(D=0) = 6/36, \Pr(D=1) = 10/36, \Pr(D=2) = 8/36, \Pr(D=3) = 6/36, \Pr(D=4) = 4/36, \Pr(D=5) = 2/36$.

*Since we know the distribution of random variable $D$, we can use Huffman tree to construct series of questions to be asked to Alice. A prefix free binary code for random variable $D$ obtained from the Huffman tree is* $\{001, 01, 11, 10, 0001, 0000\}$ *corresponding to symbols* $\{0,1,2,3,4,5\}$ *respectively. The average number of questions that need to be asked is expected length of this Huffman code which is:*

$$\frac{6}{36}\times 3 + \frac{10}{36}\times 2 + \frac{8}{36}\times 2 + \frac{6}{36}\times 2 + \frac{4}{36}\times 4 + \frac{2}{36}\times 4 \;=\; 2.5\;.$$

*Phrasing the questions in plain English can be a bit tricky; one easy solution is to consider the values of $D$ on the Huffman tree, and repeatedly ask Alice "Does the codeword corresponding to $D$ lie down this branch of the tree?". Then again, once you have built that tree, any such question can be rephrased in the form "Does $D$ lie in the set $\{\ldots\}$?".*

12. **[Huffman Code Lengths]**

Let $p_1 > p_2 > p_3 > p_4$ be the symbol probabilities for a source alphabet size $N = |\mathcal{X}| = 4$, and let the correspond codeword lengths be $\ell_1, \ell_2, \ell_3, \ell_4$.

(a) What are the possible sets of codeword lengths $\{\ell_1, \ell_2, \ell_3, \ell_4\}$ for a Huffman code for the given type of source?

**Solution.** *Only* $(1,2,3,3)$ *or* $(2,2,2,2)$ *are possible. This can be argued as follows:*

- *The smallest length clearly must be either* 1 *or* 2 *(if all codewords had length* 3 *or more, the code clearly wouldn't be optimal)*
- *If the smallest length is* 1*, the second-smallest should be* 2 *(again, if all were* 3 *or more it wouldn't be optimal). According to Kraft's inequality, we can't have another length-*2 *codeword after that, so the final two must have length* 3*.*
- *If the smallest length is* 2*, it would be suboptimal to do anything else except let all of the lengths be* 2*.*

*Another way to answer to this question is to imagine running the Huffman algorithm with 3 branching steps, and observing that it is the second step alone that determines the final codeword lengths.*

(b) Suppose that $p_1 > p_3 + p_4$. What are the possible sets of codeword lengths now?

**Solution.** *Note that $p_1 > p_3 + p_4$ means that at the second stage of the Huffman algorithm, $p_2$ will merge with the node $p_3 + p_4$ (combined at the first stage). So $\ell_1 = 1$ and the codeword lengths are $\{1, 2, 3, 3\}$.*

(c) What are the possible sets of codeword lengths if $p_1 < p_3 + p_4$?

**Solution.** *Similar argument for $p_1 < p_3 + p_4$. At the second stage $p_1$ will merge with $p_2$ (since they are now the two lowest probabilities). In this case, the lengths will be $\{2, 2, 2, 2\}$.*

(d) **(Harder)** What is the smallest value of $\rho$ such that $p_1 > \rho$ implies that $\ell_1 = 1$? *(Hint: What are the largest values of $p_1, p_3, p_4$ for which part (b) holds?)*

**Solution.** *Keeping in mind the assumptions $\{p_1 > p_2 > p_3 > p_4\}$, and $\sum_i p_i = 1$ and following the hint, we examine a few relevant cases.*

*We start with some intuition: Consider ways of keeping $p_1$ as small as possible (meaning the sum of $(p_2, p_3, p_4)$ is as large as possible), while still getting $p_1 > p_3 + p_4$ in accordance with part (b). There are two extreme cases:*

- *If $(p_2, p_3, p_4)$ are (roughly) uniformly distributed, then the four probabilities are of the form $\left(p_1, \frac{1-p_1}{3}, \frac{1-p_1}{3}, \frac{1-p_1}{3}\right)$, and the condition $p_1 > p_3 + p_4$ reduces to $p_1 > \frac{2}{3} - \frac{2p_1}{3}$, which simplifies to $p_1 > 2/5$.*
- *If $p_2$ is (roughly) equal to $p_1$, then the probabilities are of the form $(p_1, p_1, p_3, p_4)$ with $p_3 + p_4 = 1 - 2p_1$. So the condition $p_1 > p_3 + p_4$ reduces to $p_1 > 1 - 2p_1$, which simplifies to $p_1 > 1/3$.*

*We may therefore expect the answer to be the more stringent of these, giving $\rho = 2/5$. We proceed by making this intuition precise.*

*First, notice that if $p_1 > 2/5$ then $p_2 + p_3 + p_4 < 3/5$. Since $p_3$ and $p_4$ are the two smallest out of the last three values, they make up for at most $2/3$ of the total, i.e., $p_3 + p_4 < (2/3) * (3/5) = 2/5$. Combining these give $p_1 > p_3 + p_4$, so that part (b) above indeed applies. [Alternative approach mentioned in tutorial: Repeat this argument starting with $p_1 > \rho$ instead of of 2/5, and get $p_3 + p_4 < f(\rho)$, and then solve the equation $\rho = f(\rho)$ to get $\rho = 2/5$.]*

*To establish that $2/5$ is the smallest possible value, we consider the distribution $(2/5 - 6\epsilon, 1/5 + 3\epsilon, 1/5 + 2\epsilon, 1/5 + \epsilon)$ for very small $\epsilon > 0$ (we cannot set $\epsilon = 0$, because the question assumed $p_1 > p_2 > p_3 > p_4$). Then $p_1 < p_3 + p_4$, so we in fact end up in part (c) above, not part (b).*

(e) **(Harder)** What is the largest value of $\lambda$ such that $p_1 < \lambda$ implies $\ell_1 = 2$?

**Solution.**

*The same intuition as the previous solution suggests that we should consider the most stringent of $p_1 < 2/5$ and $p_1 < 1/3$, which is $p_1 < 1/3$. We proceed by verifying that $\lambda = 1/3$ is correct.*

*First, if $p_1 < 1/3$ then $p_2 < p_1 < 1/3$, meaning that $p_3 + p_4 > 1 - 1/3 - 1/3 = 1/3$. This means that $p_1 < p_3 + p_4$, so that part (c) above applies. [Alternative approach mentioned in tutorial: Repeat this argument starting with $p_1 < \lambda$ instead of of 1/3, and get $p_3 + p_4 > g(\lambda)$, and then solve the equation $\lambda = g(\lambda)$ to get $\lambda = 1/3$.]*

*On the other hand, if we consider probabilities $(1/3 + 2\epsilon, 1/3 + \epsilon, 1/6 - \epsilon, 1/6 - 2\epsilon)$ for very small $\epsilon > 0$, then we get $p_1 > p_3 + p_4$, so that part (b) above applies. Therefore, $\lambda = 1/3$ is indeed the highest possible value.*