Information Theory and Coding

Eeshaan Jain

October 3, 2022

Contents

Chapter 1

Lecture 1

1.1 The Flow of Information

1.2 Source Coding

Definition 1. Given an alphabet $\mathcal{U}$ with $|\mathcal{U}| < \infty$. A source code c is a mapping from the alphabet to the set of all finite binary strings represented as $\{0,1\}^* = \{null, 0, , 1, 00, 01, 10, 11, 000, \dots\}$ *i.e.* $c : \mathcal{U} \to \{0,1\}^*$.

Example. Consider $\mathcal{U} = \{a, b, c, d, e\}$. Then we can have a $c : \mathcal{U} \to \{0,1\}^*$ as $c(a) = 0$, $c(b) = 0$, $c(c) = null$, $c(d) = 1$, $c(e) = 010100$.

Definition 2. A code c is called singular if $\exists u, v \in \mathcal{U}$ s.t. $u \neq v$ but $c(u) = c(v)$.

Definition 3. A code c is called non-singular or injective if it is not singular.

Example. For the alphabet in the previous example, we can have an injective code as $c(a) = 0$, $c(b) = 00$, $c(c) = null$, $c(d) = 1$ and $c(e) = 010100$.

Definition 4. Given $c : \mathcal{U} \to \{0,1\}^*$, we write the concatenation of the codes of two letters $u_1, u_2 \in \mathcal{U}$ as $c(u_1)c(u_2)$. We further define

$$c^2 : \mathcal{U}^2 \to \{0,1\}^* \text{ to be } c^2(u_1, u_2) = c(u_1)c(u_2)$$

$$c^n : \mathcal{U}^n \to \{0,1\}^* \text{ to be } c^n(u_1, \dots, u_n) = c(u_1)c(u_2)\dots c(u_n)$$

$$c^* : \mathcal{U}^* \to \{0,1\}^* \text{ to be } c^*(u_1, \dots, u_n) = c(u_1)c(u_2)\dots c(u_n)$$

Definition 5. A code $c : \mathcal{U} \to \{0,1\}^*$ is called uniquely decodable if c^* is injective, *i.e.* if $u_1, \dots, u_n \neq v_1 \dots, v_k$, then $c(u_1)\dots c(u_n) \neq c(v_1)\dots c(v_k)$.

Example. With the same alphabet as before, we can have $c(a) = 0$, $c(b) = 10$, $c(c) = 110$, $c(d) = 1110$ and $c(e) = 1111$. This code is uniquely decodable and is called a prefix free code.

Definition 6. A code $c : \mathcal{U} \to \{0,1\}^*$ is called prefix free if $\forall u, v \in \mathcal{U}$ s.t $u \neq v$, $c(u)$ is not a prefix of $c(v)$.

Theorem 1. If a code c is prefix free (p.f.), then it is uniquely decodable (u.d.).

Proof. We prove the above theorem by contradiction. Consider there exists a code c which is p.f. but not u.d and we have $u_1, \dots, u_n, v_1, \dots, v_k \in \mathcal{U}$ such that $u_1, \dots, u_n \neq v_1, \dots, v_k$ and $c(u_1) \dots c(u_n) = c(v_1) \dots c(v_k)$. Traversing the code of the sequence of alphabets from the left, we immediately notice that if $\ell(u_1) > \ell(v_1)$ then $c(v_1)$ is a prefix of $c(u_1)$. Similarly if $\ell(u_1) < \ell(v_1)$, $c(u_1)$ is a prefix of $c(v_1)$. But since the code is prefix free, we arrive at a contradiction. □

Definition 7. Given a code $c : \mathcal{U} \to \{0,1\}^*$, we define the Kraft Sum

$$\mathcal{KS}(c) = \sum_{u \in \mathcal{U}} 2^{-\ell(u)}$$

where $\ell(u) = \text{length}(c(u))$. In case of confusion, we subscript the length by the code, i.e $\ell_c(u) \equiv \ell(u) = \text{length}(c(u))$

Example. Consider the code for the same alphabet as $c(a) = 0$, $c(b) = 1$, $c(c) = null$, $c(d) = 00$ and $c(e) = 010$. Then we have

$$\mathcal{KS}(c) = 2^{-1} + 2^{-1} + 2^0 + 2^{-2} + 2^{-2} = 2.375$$

Lemma 1. Suppose $c : \mathcal{U} \to \{0,1\}^*$ and $d : \mathcal{V} \to \{0,1\}^*$ are two codes, and denote $(c \times d) : \mathcal{U} \times \mathcal{V} \to \{0,1\}^*$ as $(c \times d)(u, v) = c(u)d(v)$. Then

$$\mathcal{KS}(c \times d) = \mathcal{KS}(c)\mathcal{KS}(d)$$

Proof. By definition, we have

$$\begin{aligned}\mathcal{KS}(c \times d) &= \sum_{u,v} 2^{-\ell_{c\times d}((u,v))} = \sum_{u,v} 2^{-[\ell_c(u)+\ell_d(v)]} \\ &= \sum_{u,v} 2^{-\ell_c(u)} \times 2^{-\ell_d(v)} = \sum_u 2^{-\ell_c(u)} \sum_v 2^{-\ell_d(v)} = \mathcal{KS}(c)\mathcal{KS}(d)\end{aligned}$$

□

Corollary. $\mathcal{KS}(c^n) = [\mathcal{KS}(c)]^n$

Theorem 2 (Kraft's Inequalities). Suppose $c : \mathcal{U} \to \{0,1\}^*$ is injective, then

$$\mathcal{KS}(c) \leq \log_2(1 + |\mathcal{U}|)$$

Suppose c is uniquely decodable, then

$$\mathcal{KS}(c) \leq 1$$

Suppose c is prefix free, then

$$\mathcal{KS}(c) \leq 1$$

Proof. Suppose that c is injective, and let $|\mathcal{U}| = k$ and $\mathcal{U} = \{1, 2, \cdots, k\}$. To maximize the Kraft Sum, we put $c(1) = null$, $c(2) = 0$, $c(3) = 1$ and so on. Also, let $k = 1+2+4+8+\cdots+2^{m-1}+r$ where $0 \leq r < 2^m$. Then, we can write

$$\mathcal{KS}(c) = 2 + 2 \times 2^{-1} + 2^2 \times 2^{-2} + \cdots + 2^{m-1} \times 2^{-(m-1)} + r \times 2^{-m} = m + r \cdot 2^{-m}$$

Now, consider $\log_2(1 + k) = \log_2(1 + 2^m - 1 + r) = \log_2(2^m + r)$. We can then write

$$\log_2(1 + k) \leq \log_2(2^m(1 + r \cdot 2^{-m})) = m + \log_2(1 + r \cdot 2^{-m})$$

Notice that $r \cdot 2^{-m} \in [0, 1)$ and using the fact that $\log_2(1 + x) \geq x$ for $x \in [0, 1]$, we get

$$m + \log_2(1 + r2^{-m}) \geq m + r \cdot 2^{-m} = \mathcal{KS}(c)$$

Having proved for c being injective, we now show it for c being uniquely decodable. First observe that c is u.d. $\Leftrightarrow$ c^* is injective $\Rightarrow \forall n$, c^n is injective. Hence, we can write

$$[\mathcal{KS}(c)]^n = \mathcal{KS}(c^n) \leq \log_2(1 + k^n)$$

Now, notice that the LHS is exponentially growing in $\mathcal{KS}(c)$ while the RHS is linearly growing in $\mathcal{KS}(c)$. Since exponential growth cannot be dominated by linear growth, we notice that the term growing exponentially must be less than or equal to 1. Thus, we have $\mathcal{KS}(c) \leq 1$.
Finally, since every prefix free code is uniquely decodable, the last inequality follows immediately. □

Remark. Although the Kraft Sum for uniquely decodable codes is less than or equal to 1, the reverse doesn't hold true. Take the code $\mathcal{U} = \{a, b, c\}$ and the code c to be $c(a) = 0$, $c(b) = 00$ and $c(c) = 00$. It is clear that c is not uniquely decodable, but –

$$\mathcal{KS}(c) = 2^{-1} + 2^{-2} + 2^{-2} = 1 \leq 1$$

Chapter 2

Lecture 2

2.1 Prefix Free Codes

Example. Consider $\mathcal{U} = \{a, b, c\}$ and consider two codes

$$c: \quad c(a) = 0,\ c(b) = 10,\ c(c) = 11$$
$$\widetilde{c}: \quad \widetilde{c}(a) = 0,\ \widetilde{c}(b) = 01,\ \widetilde{c}(c) = 11$$

The first code is prefix free, and hence is uniquely decodable. The second is not prefix free, but we can still uniquely decode it! To do so, when we receive the entire sequence of bits, we reverse the sequence and apply decode it as if it was encoded using code c (note that $\widetilde{c}$ is the reverse of c).

A point to note is that in the second case, we might need to store an arbitrary length binary representation before we can decode the first letter and hence we require a Turing Machine! But in the first case, we need only a finite memory since we can decode the first letter when the required number of bits come in. For this reason, prefix free codes are also called **instantaneously decodable** codes. The above example also showed that although all prefix free codes are uniquely decodable, the reverse is not true. There exist uniquely decodable codes which are not prefix free.

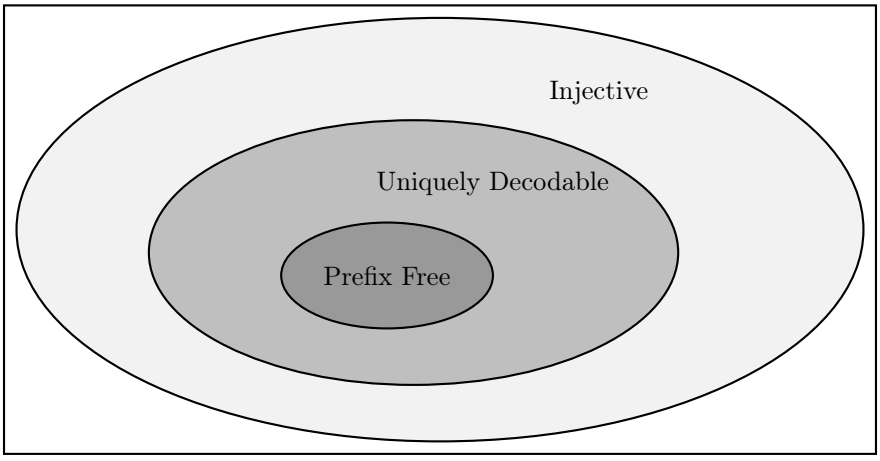


Figure 2.1: The box represents all codes possible, prefix free codes are a small category of it.

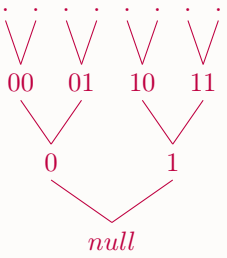
Remark. Although prefix free codes are a subset of uniquely decodable codes, there is no inherent advantage of using non prefix free codes which are uniquely decodable.

Theorem 3. Suppose $c : \mathcal{U} \to \{0,1\}^*$ for which $\mathcal{KS}(c) \leq 1$. Then $\exists$ a code $\widetilde{c} : \mathcal{U} \to \{0,1\}^*$ s.t.

- $\widetilde{c}$ is prefix free

- $\ell_{\widetilde{c}}(u) = \ell_c(u)\ \forall\ u$

Proof. Let $\mathcal{U} = \{1, 2, \ldots, k\}$ and let $\ell_1 = \ell_c(u_1), \ldots \ell_k = \ell_c(u_k)$ s.t. $\ell_1 \leq \ell_2 \leq \cdots \leq \ell_k$. It is also known that $\sum_i 2^{-\ell_i} \leq 1$.
Consider the complete binary tree on the right. It is constructed starting from *null* and each time you grow each level one bit at a time. Each parent is a prefix of its children. Now we consider the following algorithm:



1. Start with the complete binary tree of depth ℓ_k as shown in the figure. In the beginning, mark every node as available
2. For $i = 1, \ldots, k$:
   - Find a node available at depth ℓ_i (say it is n_i)
   - Set $\widetilde{c}(i) = n_i$
   - Mark the node n_i and all if its ascendants as unavailable

Although the algorithm does find such a $\widetilde{c}$, we need to rigorously show that we can always find a node at depth ℓ_i. To show this, notice that at the beginning of the algorithm, there were 2^{ℓ_i} nodes at depth ℓ_i. At every iteration until $i-1$, some nodes at depth ℓ_i get marked unavailable. Hence, at the i^{th} iteration, we have the number of available nodes as

$$\#\text{ available nodes} = 2^{\ell_i} - \sum_{j=1}^{i-1} 2^{\ell_i - \ell_j} = 2^{\ell_i}\left[1 - \sum_{j=1}^{i-1} 2^{-\ell_j}\right]$$

Since the Kraft Sum is at most 1, we have $\sum_{j=1}^{i-1} 2^{-\ell_j} < 1$ strictly. Hence the RHS of the above equation is always positive. But since it is just a difference of integers, the RHS must be at least 1. Hence, the algorithm executes successfully and we can construct such a $\widetilde{c}$. □

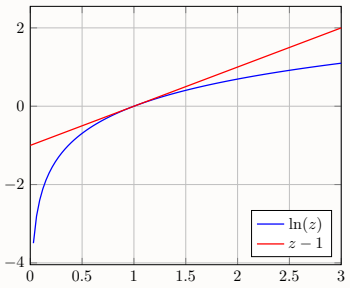
2.2 The Logarithm

Lemma 2. $\ln(z) \leq z - 1$ and equality only occurs at $z = 1$

Proof. We consider the finite Taylor's expansion of $\ln(z)$ around $z = 1$.
Recall that, the Taylor series expansion of $f(z)$ around z_0, for α between z and z_0 is given as

$$f(z) = f(z_0) + (z - z_0)f'(z_0) + \frac{1}{2}(z - z_0)^2 f''(\alpha)$$

$$\Rightarrow \ln(z) = \ln(1) + (z-1)\cdot 1 + \frac{1}{2}(z-1)^2 \cdot \left(-\frac{1}{2\alpha^2}\right)$$
$$= (z-1) - \frac{1}{2}\left(\frac{z-1}{\alpha}\right)^2 \leq z - 1$$



□

Corollary. Suppose $p_1, \ldots, p_k \geq 0$ are such that $\sum_i p_i = 1$. Suppose $z_1, \ldots, z_k > 0$, then we have

$$\sum_i^k p_i \ln(z_i) \leq \ln\left(\sum_{i=1}^k p_i z_i\right)$$

with equality only when $z_1 = z_2 = \cdots = z_k$.

Proof. Let $\mu = \sum_i p_i z_i$ and we need to show that

$$\sum_i p_i \ln(z_i) \leq \ln \mu \Rightarrow \sum_i p_i \left[\ln \left(\frac{z_i}{\mu} \right) \right] \leq 0$$

Using the previous lemma, we have

$$\sum_i p_i \left[\ln \left(\frac{z_i}{\mu} \right) \right] \leq \sum_i p_i \left[\frac{z_i}{\mu} - 1 \right] = \frac{1}{\mu} \sum_i p_i z_i - \sum p_i = 0$$

□

Definition 8. Suppose p is a probability distribution on $\mathcal{U}$ and suppose $q : \mathcal{U} \to [0, \infty)$. Then we define the Kullbeck-Liebler Divergence as

$$D(p \parallel q) \triangleq \sum_u p(u) \log \left(\frac{p(u)}{q(u)} \right)$$

Remark. If for any u, $p(u) = 0$, we consider the value of the term to be 0, and if for any u, $q(u) = 0$, then we say that $D(p \parallel q) \to \infty$.

Lemma 3. $D(p \parallel q) \geq -\log \left(\sum_u q(u) \right)$

Proof.

$$\begin{aligned} -D(p \parallel q) &= \sum_u p(u) \log \left(\frac{p(u)}{q(u)} \right) \\ &\leq \log \left[\sum_u p(u) \frac{q(u)}{p(u)} \right] = \log \left(\sum_u q(u) \right) \\ \Rightarrow D(p \parallel q) &\geq -\log \left(\sum_u q(u) \right) \end{aligned}$$

□

Remark. In particular when q is also a probability distribution on $\mathcal{U}$, we have $D(p \parallel q) \geq 0$.

Suppose we have an alphabet $\mathcal{U}$, a distribution p defined on $\mathcal{U}$ and a code $c : \mathcal{U} \to \{0,1\}^*$. If we choose a letter U randomly from the alphabet using the probability distribution, we can say that $\mathbb{E}[\ell(U)]$ denotes the expected number of bits representing a letter.

Example. Consider $\mathcal{U} = \{a, b, c, d, e\}$, let $p = (0.3, 0.1, 0.15, 0.2, 0.25)$ and $c = (00, 01, 10, 110, 111)$. Then we have $\mathbb{E}[\ell(u)] = 0.3 \times 2 + 0.1 \times 2 + 0.15 \times 2 + 0.2 \times 3 + 0.15 \times 3 = 2.45$

Theorem 4. For any code $c : \mathcal{U} \to \{0,1\}^*$ and a probability distribution p defined on $\mathcal{U}$, we have

$$\mathbb{E}[\ell(u)] \geq \sum_u p(u) \log_2 \frac{1}{p(u)} - \log_2[\mathcal{KS}(c)]$$

In particular, when c is uniquely decodable, we have

$$\mathbb{E}[\ell(u)] \geq \sum_u p(u) \log_2 \frac{1}{p(u)}$$

Proof. We define q on $\mathcal{U}$ as $q(u) = 2^{-\ell(u)} \Rightarrow \ell(u) = \log_2 \frac{1}{q(u)}$. Noticing the correspondence, we

can write $\mathbb{E}[\ell(u)] = \sum_u p(u) \log_2 \frac{1}{q(u)} = D(p \parallel q) + \sum_u p(u) \log \frac{1}{p(u)}$. Now, we have

$$\begin{aligned}
\mathbb{E}[\ell(u)] &= \sum_u p(u) \log \frac{1}{p(u)} + D(p \parallel q) \\
&\geq \sum_u p(u) \log \frac{1}{p(u)} - \log(\sum q(u)) \\
&= \sum_u p(u) \log \frac{1}{p(u)} - \log \mathcal{KS}(c)
\end{aligned}$$

□

Theorem 5. Given $\mathcal{U}, p$, $\exists$ a prefix free code c s.t.

$$\mathbb{E}[\ell(u)] \leq \sum p(u) \log \frac{1}{p(u)} + 1$$

Proof. We set $\ell(u) \triangleq \lceil \log_2 \frac{1}{p(u)} \rceil$. Due to the ceil function, it is clear that $\sum_u 2^{-\ell(u)} \leq 1$ (it was an equality without the ceil). We have

$$\begin{aligned}
\ell(u) &\leq \log_2 \frac{1}{p(u)} + 1 \\
\Rightarrow \mathbb{E}[\ell(u)] &\leq \sum_u p(u) \log \frac{1}{p(u)} + 1
\end{aligned}$$

□

Chapter 3

Lecture 3

3.1 Entropy

It is quite possible that the information source produces more than 1 letter at a time, i.e the source output is a sequence $u_1, u_2, \ldots$ of letters. We can also consider the code $c_n : \mathcal{U}^n \to \{0,1\}^*$.

Example. We can take $\mathcal{U} = \{a, b, c\}$ and have c_2 as $c(aa) = 0$, $c(ab) = 10$, $c(ac) = 110$ etc.

We can extend the notion from previous lecture for code c_n as follows – if $c_n : \mathcal{U}^n \to \{0,1\}^*$ is uniquely decodable, then

$$\mathbb{E}[\ell(u_1 \ldots u_n)] \geq \sum_{u_1,\ldots,u_n} p(u_1, \ldots, u_n) \log_2 \frac{1}{p(u_1, \ldots, u_n)}$$

Moreover, there exists a prefix free code $\widetilde{c}_n$ also s.t.

$$\mathbb{E}[\ell_{\widetilde{c}}(u_1 \ldots u_n)] \leq \sum_{u_1,\ldots,u_n} p(u_1, \ldots, u_n) \log_2 \frac{1}{p(u_1, \ldots, u_n)} + 1$$

A more interesting quantity to look at is the number of bits per letter. Hence, we have

$$\frac{1}{n}\mathbb{E}[\ell(u_1 \ldots u_n)] \geq \frac{1}{n} \sum_{u_1,\ldots,u_n} p(u_1, \ldots, u_n) \log_2 \frac{1}{p(u_1, \ldots, u_n)}$$

$$\frac{1}{n}\mathbb{E}[\ell_{\widetilde{c}}(u_1 \ldots u_n)] \leq \frac{1}{n} \sum_{u_1,\ldots,u_n} p(u_1, \ldots, u_n) \log_2 \frac{1}{p(u_1, \ldots, u_n)} + \frac{1}{n}$$

Notice that as n gets large, the two bounds are extremely close.

Definition 9. When $U \in \mathcal{U}$ is a random variable, we define its entropy as

$$H(U) = \sum_u p(u) \log \frac{1}{p(u)}$$

where $p(u) = \Pr[U = u]$.

Thus, in terms of entropy, we can state that for any uniquely decodable code $c : \mathcal{U} \to \{0,1\}^*$ and for any random variable $U \in \mathcal{U}$,

$$\mathbb{E}[\ell(U)] \geq H(U) \text{ and } \exists \text{ prefix free } \widetilde{c} \text{ s.t. } \mathbb{E}[\ell_{\widetilde{c}}(U)] \leq H(U) + 1$$

Again, replacing U $U_1, \ldots U_n$, we get for any uniquely decodable $c_n : \mathcal{U}^n \to \{0,1\}^*$ such that

$$\frac{1}{n}\mathbb{E}[\ell(u_1 \ldots u_n)] \geq \frac{1}{n} H(u_1, \ldots, u_n)$$

Moreover there exists a prefix free code $\widetilde{c}_n$ s.t.

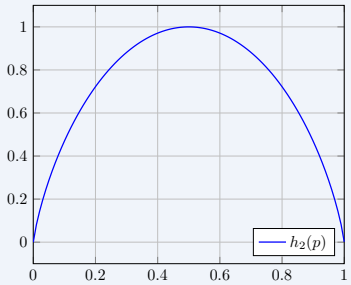
$$\frac{1}{n}\mathbb{E}[\ell_{\widetilde{c}}(u_1 \ldots u_n)] \leq \frac{1}{n} H(u_1, \ldots, u_n) + \frac{1}{n}$$

Example. Consider $\mathcal{U} = \{a, b\}$ with $p(a) = p$ and $p(b) = 1 - p$.

The entropy for the above is given as

$$H(U) = p \log \frac{1}{p} + (1-p) \log \frac{1}{p}$$

The above quantity is defined as the binary entropy function $h_2(p)$. It has a maxima at $p = \frac{1}{2}$ and $h_2(0.5) = 1$.



3.2 The Optimization Problem and Optimal Codes

Consider the following - say we have an alphabet $\mathcal{U}$ and a probability distribution p defined on $\mathcal{U}$ as input. At the output, we want the expected length of the codes to be minimum. Thus the output is $c : \mathcal{U} \to \{0,1\}^*$ s.t. c is prefix free and s.t. $\sum_u p(u)\ell(u)$ is as small as possible. We can frame this problem as

$$\min \left\{ \sum_{u \in \mathcal{U}} p(u)\ell(u) \; ; \; \sum_u 2^{-\ell(u)} \leq 1 \right\} \text{ and } \ell(u) \in \mathbb{N}$$

The integer constraint makes the problem much tougher to solve. If that constraint is removed, the trivial solution is $\ell(u) = -\log p(u)$.

Example. We can see another problem where integer constraint increases complexity. Consider the problem - given $w_1, \ldots, w_k \geq 0$, and let $w = \sum w_i$. Find $S \subset \{1, \ldots, k\}$ s.t.

$$\min_S \left| \sum_{i \in S} w_i - \sum_{i \notin S} w_i \right| \equiv \min_S \left| \sum_{i \in S} w_i - \frac{w}{2} \right| \equiv \min_{x_1, \ldots, x_k \in \{0,1\}} \left| \sum_{w_i x_i} - \frac{w}{2} \right|$$

If we remove the integer constraint, we can easily see that $x_i = \frac{1}{2}$ solves the problem.

We have the following properties of optimal codes:

1. If $p(u) < p(v)$ then $\ell(u) \geq \ell(v)$. The proof is trivial, because if that's not the case, we can swap $\ell(v)$ and $\ell(u)$ and improve the expected length of the code by $[p(v) - p(u)][\ell(v) - \ell(u)]$.

   Corollary. One of the longest codewords must belong to one of the least probable letters.

2. There have to be at least two longest codewords (assuming $|\mathcal{U}| > 1$). Further, the two longest codewords must be siblings. This is also immediate - if there is a single longest codeword, it can be moved down the binary tree, and the codeword will get shorter.

   Corollary. There is an optimal code in which the two least probable letters are assigned sibling codewords

   Proof. Consider a code c in which the two least probable letters are not siblings. Just swap one of the codewords to make them siblings. □

In an optimal code for $(\mathcal{U}, p)$, we know that the two least likely letters say k and $k-1$ must have codewords $[\ldots]0$ and $[\ldots]1$. Let the common prefix $[\ldots]$ have length $\widetilde{\ell}$. Then we can write

$$\begin{aligned} \mathbb{E}[\ell(U)] &= p(1)\ell(1) + p(2)\ell(2) + \cdots + p(k-2)\ell(k-2) + [p(k) + p(k-1)](1 + \widetilde{\ell}) \\ &= p(1)\ell(1) + \cdots + [p(k) + p(k-1)]\widetilde{\ell} + [p(k) + p(k-1)] \end{aligned}$$

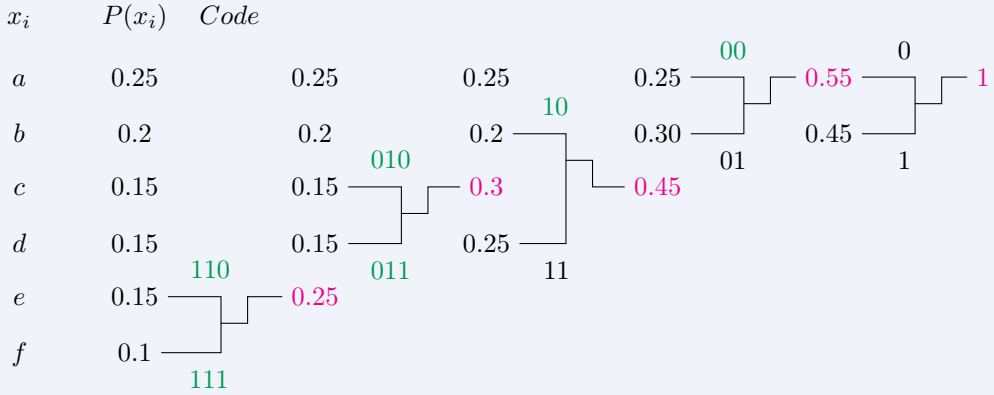
Thus our optimization problem is as follows -

$$\min_{\ell(1),\ldots,\ell(k)} \sum_{i=1}^{k} p(i)\ell(i) \text{ s.t. } \sum_{i} 2^{-\ell(i)} \leq 1$$

$$\equiv \min_{\ell(1),\cdots,\ell(k-2),\widetilde{\ell}} \sum_{i=1}^{k-2} p(i)\ell(i) + [p(k-1)+p(k)]\widetilde{\ell} + [p(k)+p(k-1)] \text{ s.t. } \left(\sum_{i=1}^{k-2} 2^{-\ell(i)}\right) + +2^{-\widetilde{\ell}-1} + 2^{-\widetilde{\ell}-1} \leq 1$$

The constraint is same as $2^{-\ell(1)}+2^{-\ell(2)}+\cdots+2^{-\ell(k-2)}+2^{-\widetilde{\ell}} \leq 1$. Thus, we reduced the optimization problem to $k-1$ variables, i.e. $(\mathcal{U} = \{1,\ldots,k\}, p = (p(1),\ldots,p(k)) \to (\mathcal{V} = \{1,\ldots,k-1\}, q = (p(1),\cdots,p(k-2), p(k-1)+p(k))$. We can keep on repeating this procedure and bring the problem down even from the $(\mathcal{V}, q)$ code.

Example. We show the Huffman procedure on the following code: $\mathcal{U} = \{a, b, \ldots, f\}$ and $p = (0.25, 0.2, 0.15, 0.15, 0.15, 0.1)$.



Following the procedure, we get the codes (marked in green for the corresponding letter in the alphabet). The nodes marked in magenta are the fake nodes, and can be used to calculate the expected length of the code. The expected length of the code is given by

$$\mathbb{E}[\ell(u)] = \sum (\text{fake node values}) = 0.25 + 0.3 + 0.45 + 0.55 + 1 = 2.55$$