

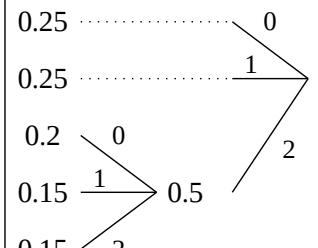
Definition: Ternary Huffman Codes

We saw how to construct *binary* Huffman codes. We can also generate Huffman codes for larger alphabets, resulting in ternary, quaternary, or, more generally, *d-ary Huffman codes*.

Example: Ternary Huffman code

We build a Huffman code with the alphabet $\{0, 1, 2\}$ for the same distribution as in the previous example.

x	$P_X(x)$		code
a	0.25	0	0
b	0.25	1	1
c	0.2	0	20
d	0.15	1	21
e	0.15	2	22



Now, use the above procedure to construct a ternary code for the source P_X with $\mathcal{X} = \{a, b, c, d, e, f\}$ and $P_X(a) = P_X(b) = 0.25$, $P_X(c) = 0.2$, $P_X(d) = P_X(e) = P_X(f) = 0.1$. Can you find another code with a smaller average codeword length?

We have to be careful, because with an alphabet size of greater than 2, the above procedure does not always give an optimal code! In fact, a d -ary code is only optimal if $|\mathcal{X}|$ is of the form $k(d - 1) + 1$ for some $k \in \mathbb{N}$. This ensures that at every step, we can combine exactly d symbols to use the alphabet at full capacity. The ternary code in the example above is optimal because $|\mathcal{X}| = 5 = 2(3 - 1) + 1$, but the code you constructed in the exercise is not. To remedy this, one can add one or more 'dummy' symbols to the source (each with probability zero) until an appropriate size of \mathcal{X} of the form $|\mathcal{X}| = k(d - 1) + 1$ for some $k \in \mathbb{N}$ is reached. The codewords for those dummy symbols are discarded at the end. It turns out that Huffman codes indeed have optimal code length (see Cover/Thomas, Section 5.8)

Ternary code applet

