Huffman coding

Want to compress a source (e.g. speech, video, music etc.) which is modelled as an iid sequence $X_0, X_1, X_2, \ldots$ iid according to RV X with pmf $p_X(x)$.

(so $X_0 \sim X$ with pmf $p_X(x)$
$X_1 \sim X$ with pmf $p_X(x)$
$\vdots$)

We want to map this sequence into a new sequence according to a source code C which is a mapping from

$X_0 \to C(X_0)$
$X_1 \to C(X_1)$
$\vdots$
$X_n \to C(X_n)$
$\vdots$

$C: \mathcal{S}_X \longmapsto D^*$, the set of finite length strings of symbols from a D-ary alphabet.
(we will consider $D=2$, meaning binary alphabets).

Let $C(x)$ denote the codeword corresponding to x and let $\ell(x)$ denote its length. Then define the expected length of source code C for RV X with pmf $p_X(x)$ as:

$$L(C) \triangleq E_X[\ell(X)] = \sum_{x \in \mathcal{S}_X} p_X(x)\,\ell(x)$$

E.g. Let X be a RV which takes on 4 values with probabilities $1/2, 1/4, 1/8, 1/8$. We are given the following code C for the RV X:

Source symbol	Probability of source symbol	Code word
A	1/2	$C(A) = 0$
B	1/4	$C(B) = 10$
C	1/8	$C(C) = 110$
D.	1/8	$C(D) = 111$

- What is S_X? $\{A, B, C, D\}$.
- What is D^*? $D=2$ $\{0,1\}$. D^* is the set of finite length binary strings.

$D^* = \{0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, \ldots\}$.

- What is $H(X)$? $H(X) = \frac{1}{2}\log 2 + \frac{1}{4}\log(4) + \frac{1}{8}\log 8 + \frac{1}{8}\log 8 = 1.75$ bits.

$\log_2$ — wow!

- What is $L(C)$? $L(C) = \frac{1}{2}\cdot 1 + \frac{1}{4}\cdot 2 + \frac{1}{8}\cdot 3 + \frac{1}{8}\cdot 3 = 1.75$ bit.

- In general $H(X) \leq \underbrace{L(C)}_{\text{optimal}} \leq H(X) + 1$.

- In general, $0 \leq H(X) \leq \log |S_X|$

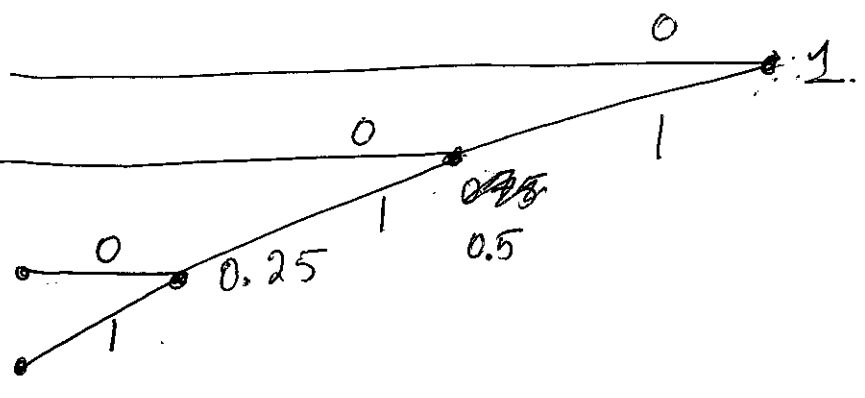
Huffman coding

- assume we have a source X with pmf $p_X(x)$ that wants to be compressed.
- Huffman coding produces a code C that is optimal (minimal) in terms of expected codeword length (minimizes $L(C)$).

E.g: (same as before)

Huffman coding tree.

Symbol	Prob
1	0.5
2	0.25
3	0.125
4	0.125



combine the 2 smallest (for a binary code with $D=2$) + repeat, labelling branches! End when reach 1.

Read off the codewords, "backwards". (start at root)

$1 \leftrightarrow C(1) = 0$

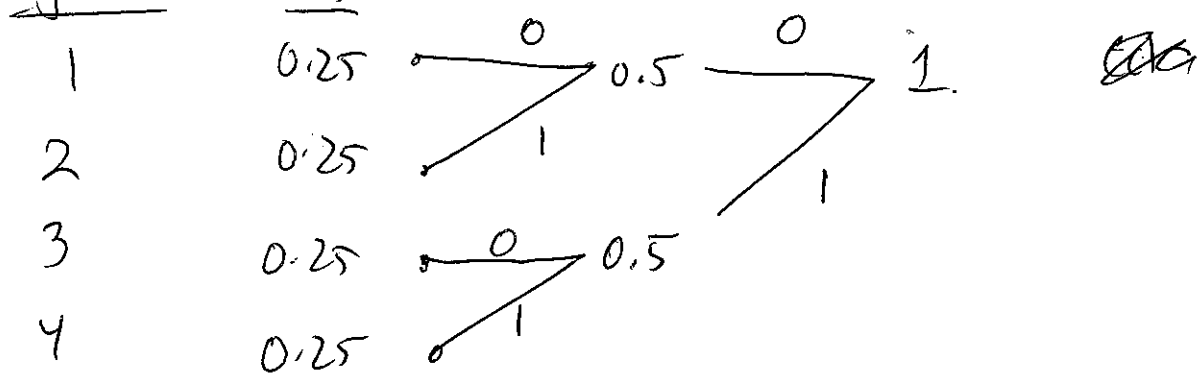
$2 \leftrightarrow C(2) = 10$

$3 \leftrightarrow C(3) = 110$

$4 \leftrightarrow C(4) = 111$

What if have 2 of equal prob somewhere? Which to combine?

Doesn't matter in terms of $L(C)$.



Here we see $C(1) = 00$, $C(2) = 01$, $C(3) = 10$, $C(4) = 11$.

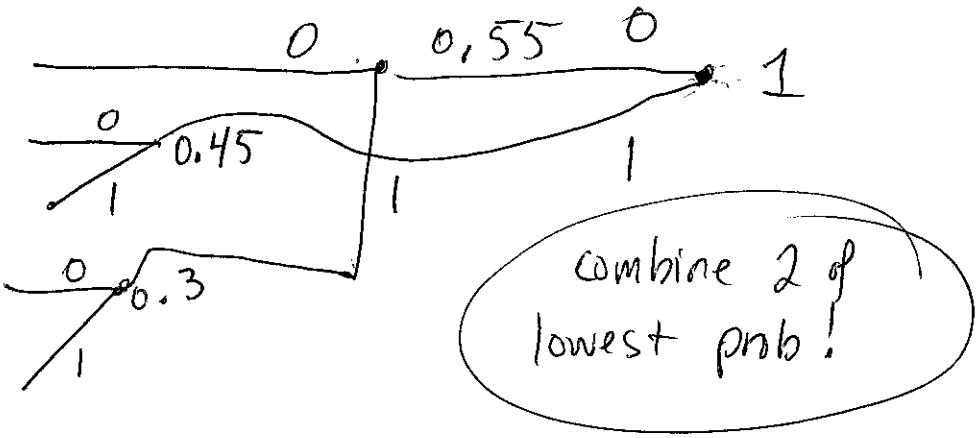
Eg: Given a source X with distribution 0.25, 0.25, 0.2, 0.15, 0.15

1) Find $H(X)$

2) Find a Huffman code C.

3) Find $L(C)$.

Codeword	Symbol	Prob
00	1	0.25
10	2	0.25
11	3	0.2
010	4	0.15
011	5	0.15



1) $H(X) = \frac{1}{4}\log 4 + \frac{1}{4}\log 4 + \frac{1}{5}\log 5 + 2(0.15)\log\left(\frac{1}{0.15}\right) = 2.285$ bits.

ALWAYS LOG BASE 2 FOR THIS COURSE

3) $L(C) = 0.25(2) + 0.25(2) + 0.2(2) + 0.15(3)(2)$

$= 2.3$ bits / source symbol

In general, for a symbol-by-symbol source code, if you take ECE 534, we show that for any "good" code:

$$H(X) \leq \underbrace{L(C)^*}_{\text{optimal}} \leq H(X) + 1.$$

expected codeword length

Note: 1 bit penalty is due to symbol-by-symbol compression.

In general, when we group symbols in chunks of length n.

$$H(X) \leq L(X) \leq H(X) + \frac{1}{n}$$

Efficiency of a code C is defined as

$$\frac{H(X)}{L(C)} \times 100 = \text{efficiency in \%}$$