CIS 575. Introduction to Algorithm Analysis
Material for April 15, 2024

Optimal Huffman Codes

The topic of this note is presented in *Cormen*'s Section 15.3.

1 Huffman Encoding

Setting We want to transmit sequences of symbols, with each symbol α encoded as a bit string, $B(\alpha)$. We assume that in such sequences, each symbol α occurs with a certain frequency, $F(\alpha)$ (the frequencies must add up to 1). Our **goal** is to come up with an encoding that requires the **smallest** number of bits.

Motivating Example We shall consider a language with 4 symbols: a, b, c and d, with frequencies given by

α	a	b	c	d
$F(\alpha)$	0.4	0.1	0.3	0.2

We may first consider a **fixed-length** encoding, for example:

α	a	b	c	d
$B(\alpha)$	00	01	10	11

in which case it obviously takes $2n$ bits to transmit n symbols.
But a **variable-length** encoding will be better; with for example

a	b	c	d
0	110	10	111

the number of bits needed to transmit n symbols is given by

$$1 \cdot 0.4 \cdot n + 3 \cdot 0.1 \cdot n + 2 \cdot 0.3 \cdot n + 3 \cdot 0.2 \cdot n = \mathbf{1.9 \cdot n}$$

which turns out to be optimal.

One may wonder: why not choose even shorter encodings, such as

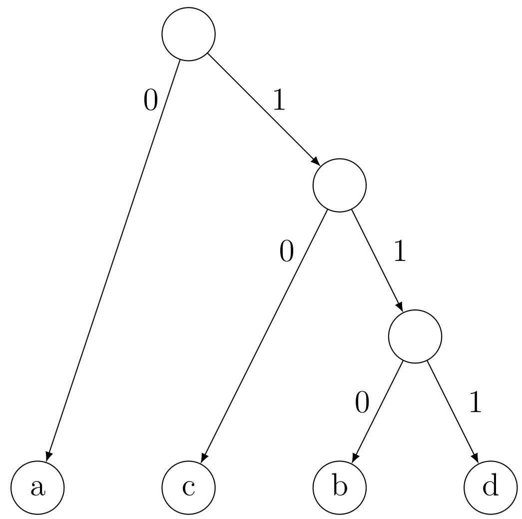
a	b	c	d
0	10	1	11

But then we cannot distinguish say the encoding of b and the encoding of ca.

To make decoding unambiguous, we must require that no code is a prefix of another code, and hence an encoding can be represented as a *binary tree* where the symbols are on the *leaves*. For example, the encoding

a	b	c	d
0	110	10	111

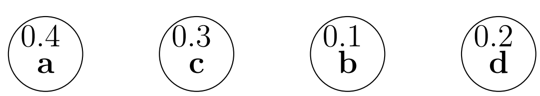
can be represented as the binary tree



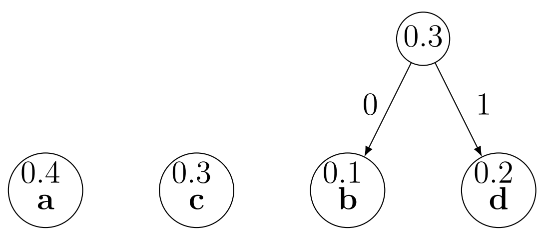
Huffman's Algorithm for Finding Optimal Encoding We shall now show how to construct a binary tree representing an optimal encoding. Along the way, we shall maintain a *forest* of binary trees, where each tree t has a frequency $F(t)$. Initially, for each symbol α there is a singleton tree t_α such that $F(t_\alpha) = F(\alpha)$. We then repeatedly

1. find trees t_1 and t_2 such that $F(t_1)$ and $F(t_2)$ are minimal
2. form a tree t, by creating a new node which has t_1 and t_2 as children (it doesn't really matter which is the 0-child and which is the 1-child); and let $F(t) = F(t_1) + F(t_2)$.

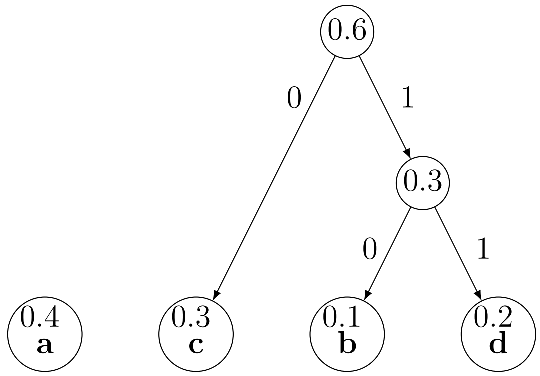
For our example, we start with



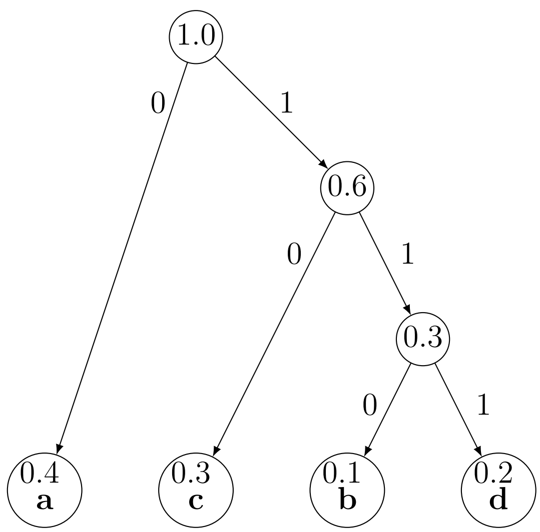
and then b and d are combined (as 0.1 and 0.2 are minimal among the frequencies) into one tree:



which is combined with c (since $0.3 < 0.4$) into one tree:



which is finally combined with a:



We see this is isomorphic (even identical) to the encoding we presented in the beginning of this section.

One can prove that Huffman's algoritm *always* produces an optimal encoding. We shall not present this proof in detail, but just mention a key idea:

> if T is an optimal encoding
> then there exists (another) optimal encoding T'
>   where the two least frequent symbols are siblings.