CODING FOR DISCRETE SOURCES

Three major types of data sources:

a) analog : continuos wave

b) analog sequence: continuos in value
discrete samples

c) discrete : symbols from finite alphabet X

Ex: $X = \{A, B, C, D, \dots, Z, 0, 1, \dots 9, \dots\}$

$X = \{\cdot, -, \llcorner\}$

$X = \{0x00 \dots 0xff\}$

Coding

- Source alphabet $\mathcal{A}$
- code alphabet $\mathcal{B}$
- coding: symbols from $\mathcal{A} \longrightarrow$ words in $\mathcal{B}$
- word: nonempty finite sequence of symbols

Ex: words 00000, 00001, ..., 11111
10 of them contain two symbols 1
$\Rightarrow$ 2-out-of-5 code

0 ⟶ 00011
1 → 11000
2 → 10100
3
⋮ ⋮

211 → 10100|11000|11000

Unique decodability

a) $a_1 \neq a_2 \in \mathcal{A} \longrightarrow K(a_1) \neq K(a_2) \in \mathcal{B}$

b) no two combinations of source symbols produce the same sequence of code words

Ex: $\mathcal{A} = \{0, 1, 2, \dots, A, B, \dots, F\}$
$\mathcal{B} = \{0, 1\}$

$K(\mathcal{A})$:
0 ⟶ 0000
1 ⟶ 0001
⋮
A ⟶ 1010
⋮
F → 1111

Fixed-length codes

block codes

→ length of all code words is const.

$\mathcal{A} = \{a_1, a_2, \dots, a_n\}$ .... n symbols

$\mathcal{B} = \{0, 1\}$

? length of a code word M

$$M = \lceil \log_2 n \rceil \qquad n \leq 2^M$$

Variable-length codes

Ex: $\mathcal{A} = \{a, b, c\}$

$K(a) = 0$

$K(b) = 10$

$K(c) = 11$

Ex: Morse

E → •

T → —

A → • —

Y → — • — —

Ex: $\mathcal{A}\{a, b, c\}$

a → 0

b → 1

c → 10

babc → 10110

babc

cb

babba

cbba

Ex:

a → 1

b → 10

c → 100

a → 1

b → 01

c → 001

- codewords of different lengths
- buffering

→ **Prefix-free codes**

a) source codeword can be decoded as soon as its last bit arrives

b) if a uniquely-decodable code with the given code lengths exists it can be made prefix

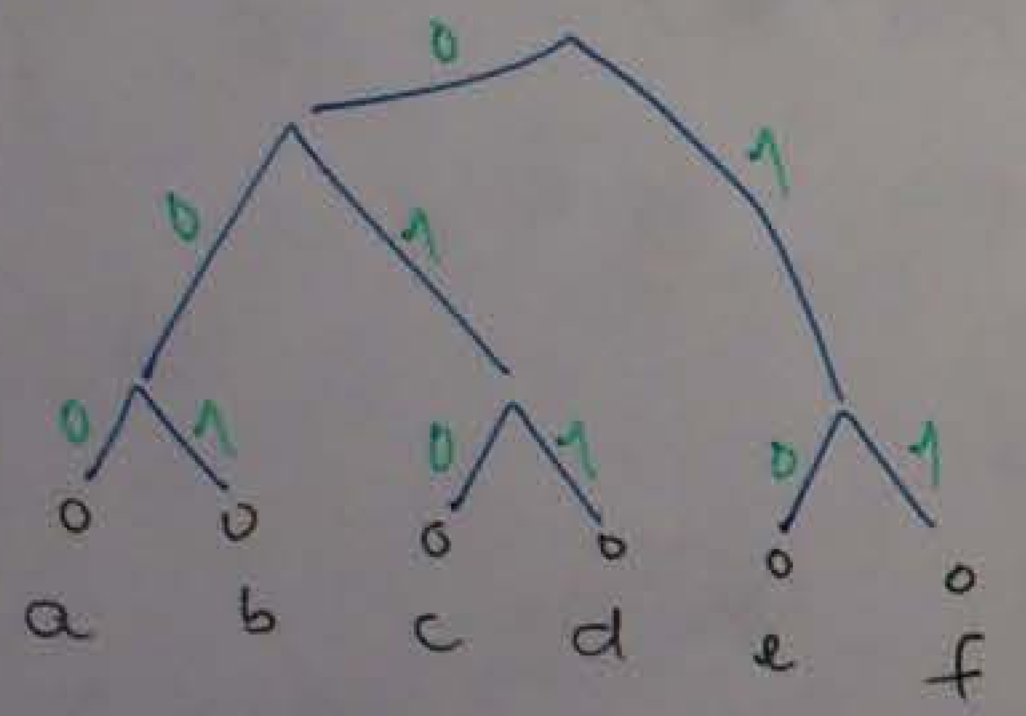
c) given the probability distribution of the source symbols, it is possible to design a prefix-free code with optimum code-word lengths

Construction of a prefix-free code

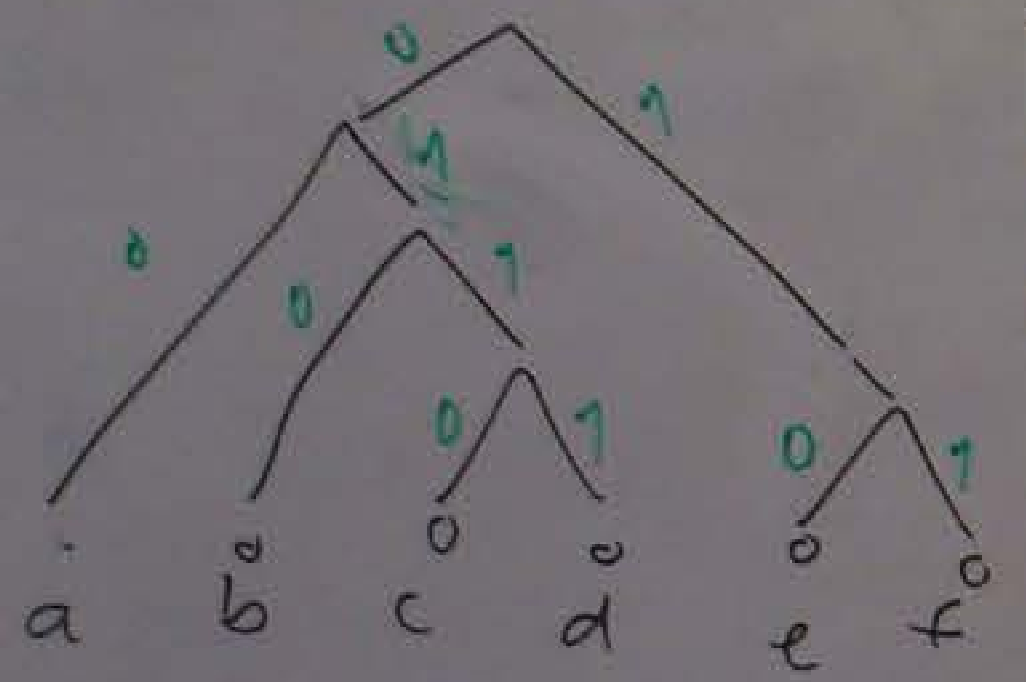
→ all source symbols form a leaf of a binary tree

→ travelling the the tree from root produces the code word

$\mathcal{A} = \{a, b, c, d, e, f\}$



a → 000
b → 001
c → 010
d → 011
e → 10
f → 11



a → 00
b → 010
c → 0110
d → 0111
e → 10
f → 11

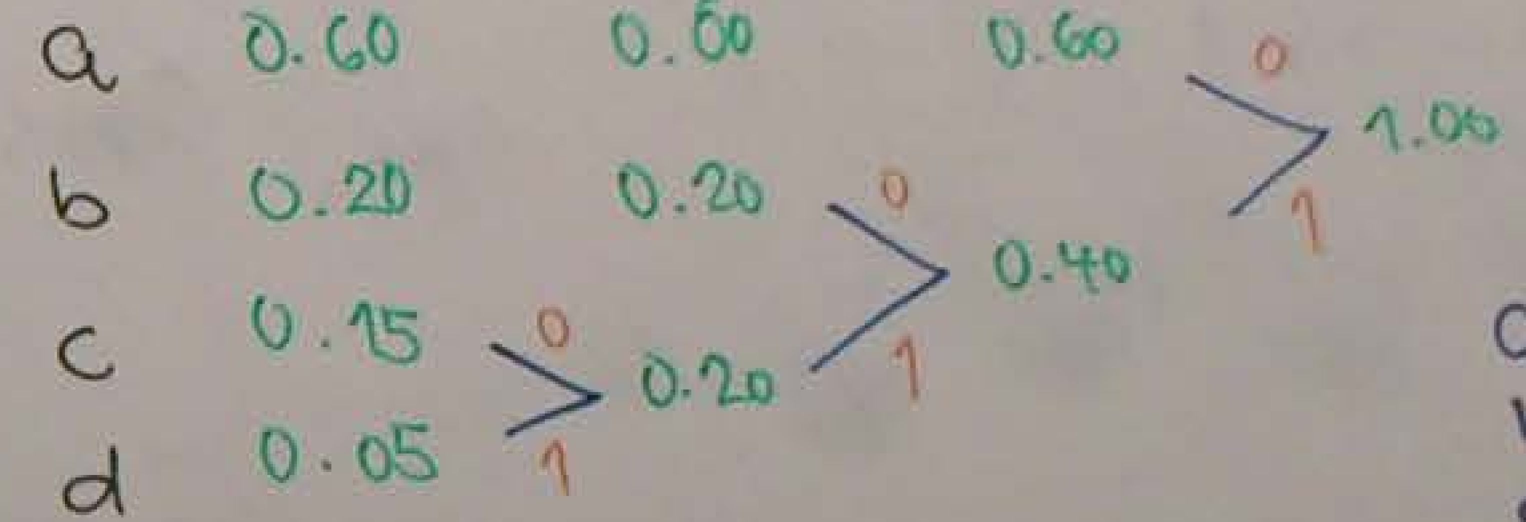
Optimal code word length

$\mathcal{A} = \{a_1, a_2, \dots, a_m\}$ + probabilities

$p(a_1), p(a_2), \dots, p(a_m)$

→ code word lengths approximating $-\log_2 p(a_i)$

Huffman's algorithm



a → 0
b → 10
c → 110
d → 111

1 bit of entropy $\frac{1}{2^1}$

0.5

2 bit

0.25

0.125

0.125 → 3 bits of entropy $\frac{1}{2^3}$

→ result is the same!

Optimal only for integer entropies of symbols, ie for $\boxed{p(a_i) = \frac{1}{2^k}}$

Problems with Huffman coding

- opimality
- requires 2 passes over input data (or buffer)
  - 1st pass: determine $p(a_i)$
  - 2nd pass: encode

  ⇒ adaptive Huffman coding (assume a-priori distribution of a_i; update it)
- it is symbol based

  Ex: English: 'qu'

Kaft's inequality

Prefix-free code with code-word lengths $l(a_1), l(a_2), \dots, l(a_n)$ exists iff

$$\sum_{i=1}^{m} 2^{-l(a_i)} \leq 1$$

Ex: $\frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \frac{1}{2^3} = \frac{1}{2} + \frac{1}{4} + \frac{2}{8} = 1$

→ Full prefix-free code

Ex: $\{1, 2, 3, 3, 4\}$ ... it cannot be constructed

$\{1, 2, 3, 4, 4\}$ ... full again

a	0.5	1 bit of entropy	$\frac{1}{2^1}$
b	0.25	2 bit	
c	0.125		
d	0.125	→ 3 bits of entropy	$\frac{1}{2^3}$

→ result is the same !

→ Optimal only for integer entropies of symbols, ie for $\boxed{p(a_i)=\frac{1}{2^k}}$

Problems with Huffman coding

- opimality
- requires 2 passes over input data (or buffer)
  - 1st pass: determine $p(a_i)$
  - 2nd pass: encode

  ⇒ adaptive Huffman coding (assume a-priori distribution of a_i; update it)
- it is symbol based

  Ex: English: 'qu'

Kaft's inequality

Prefix-free code with code-word lengths $l(a_1), l(a_2), \dots, l(a_n)$ exists iff

$$\sum_{i=1}^{n} 2^{-l(a_i)} \le 1$$

Ex: $\frac{1}{2}+\frac{1}{2^2}+\frac{1}{2^3}+\frac{1}{2^3} = \frac{1}{2}+\frac{1}{4}+\frac{2}{8} = 1$

→ Full prefix-free code

Ex: $\{1,2,3,3,4\}$ ... it cannot be constructed

$\{1,2,3,4,4\}$ ... full again