20SK – Signals and Codes

Lecture 08 – Coding for discrete data sources, prefix-free codes (2018/11/19)

Topics discussed:

- Formal definition of a code
- Fixed-length codes for discrete sources
- Variable-length codes for discrete sources
- Unique decodability
- The Kraft inequality for prefix-free codes
- Optimal prefix-free codes, Huffmann coding

The relevant literature is [1, chapter 2], [2, chapter 2] and [3, section 1.2]. Huffmann coding and arithmetic coding are also described in Wikipedia.

Resources

[1] Gallager, R.: Course materials for 6.450 *Principles of Digital Communications I*, Fall 2006. MIT OpenCourseWare (http://ocw.mit.edu/), Massachusetts Institute of Technology.

[2] Adámek, J: *Foundations of Coding: Theory and Applications of Error-Correcting Codes with an Introduction to Cryptography and Information Theory*. Wiley Interscience, 1991, 352 pp.

[3] Seibt, P.: Algorithmic Information Theory – Mathematics of Digital Information Processing. Springer, 2006, 447 pp.

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

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

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

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

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

Fixed-length codes

block codes
→ length of all code words is constant.

$\mathcal{A} = \{a_1, a_2, \dots, a_n\}$ .... 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: $\mathcal{A} = \{a, b, c\}$

$K(a) = 0$
$K(b) = 10$
$K(c) = 11$

- 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

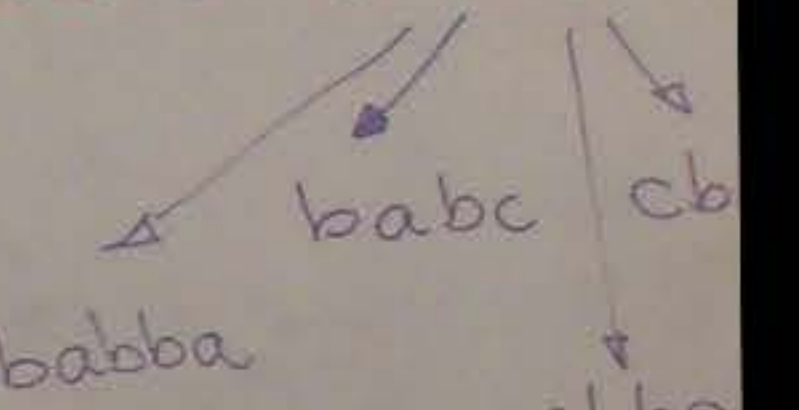
Ex: Morse

E → .
T → −
A → .−
Y → −.−−

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

a → 0
b → 1
c → 10

babc → 10110



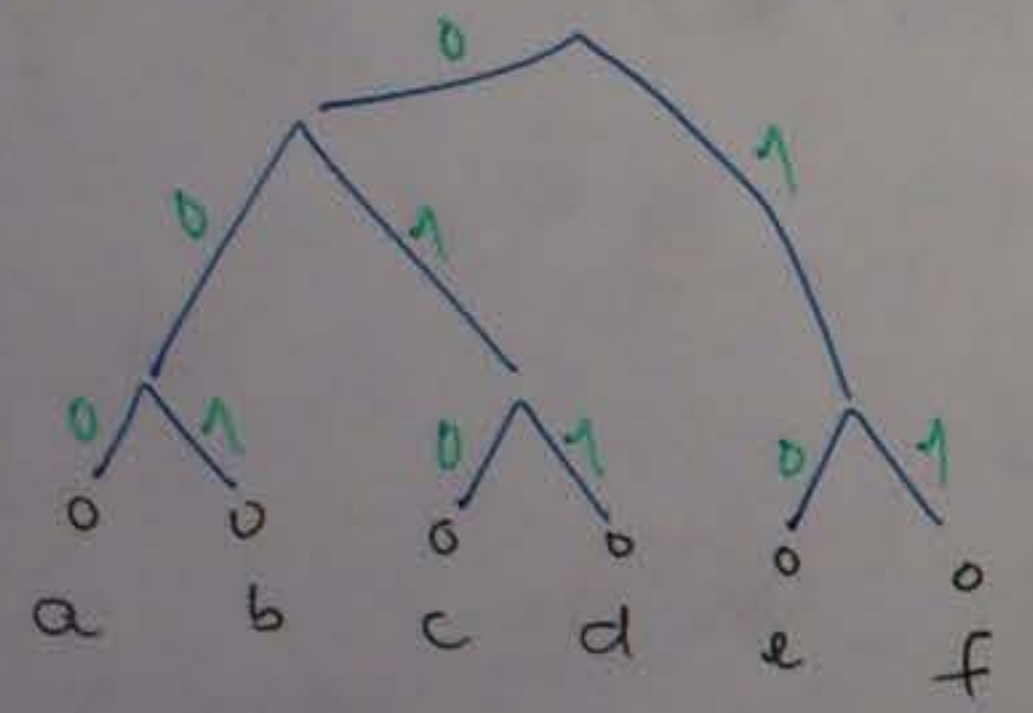
Ex:

a → 1
b → 10
c → 100

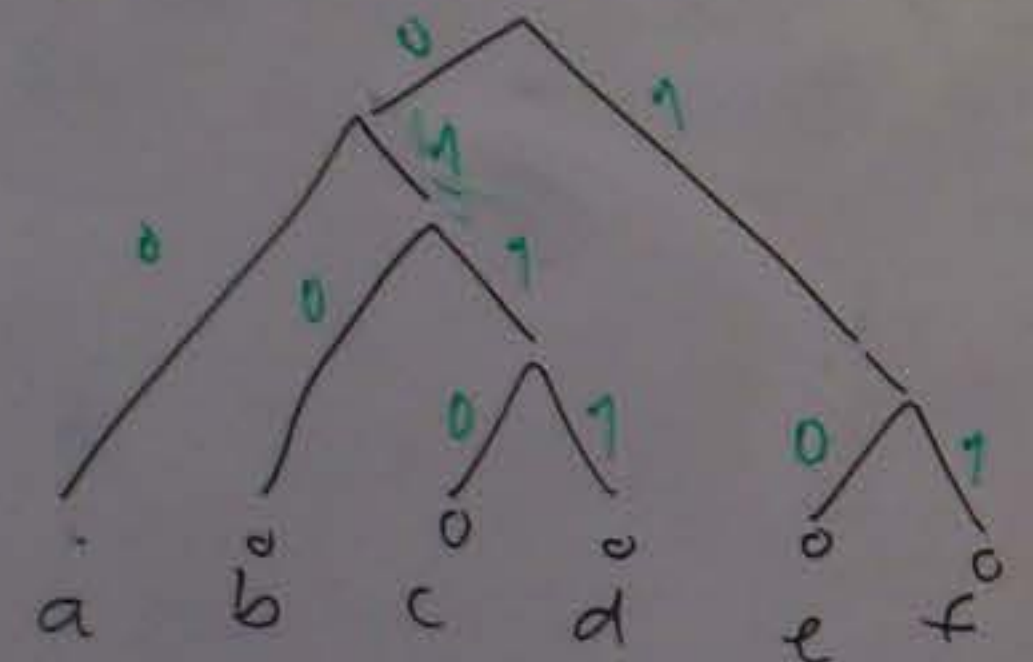
a → 1
b → 01
c → 001

Construction of a prefix-free code

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



$a \to 000$
$b \to 001$
$c \to 010$
$d \to 011$
$e \to 10$
$f \to 11$



$a \to 00$
$b \to 010$
$c \to 0110$
$d \to 0111$
$e \to 10$
$f \to 11$

→ all source symbols form a leaf of a binary tree

→ travelling the the tree from root produces the code word

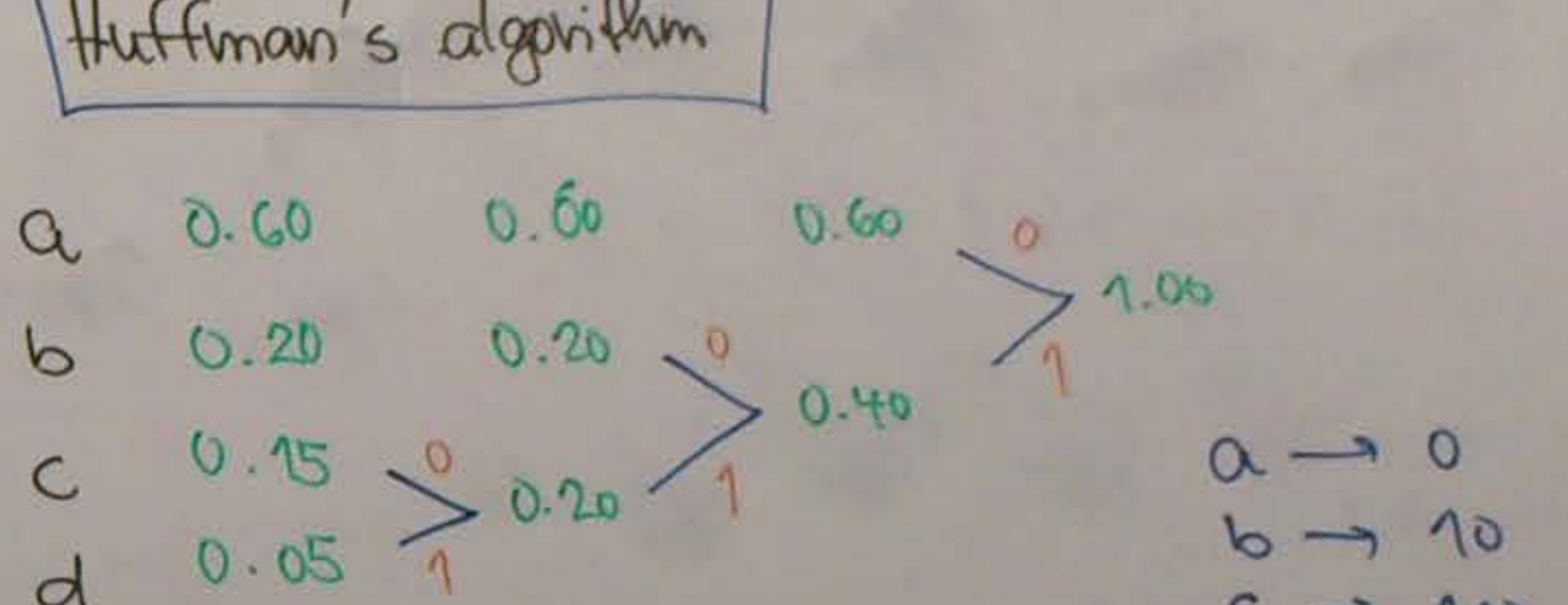
Optimal code word length

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

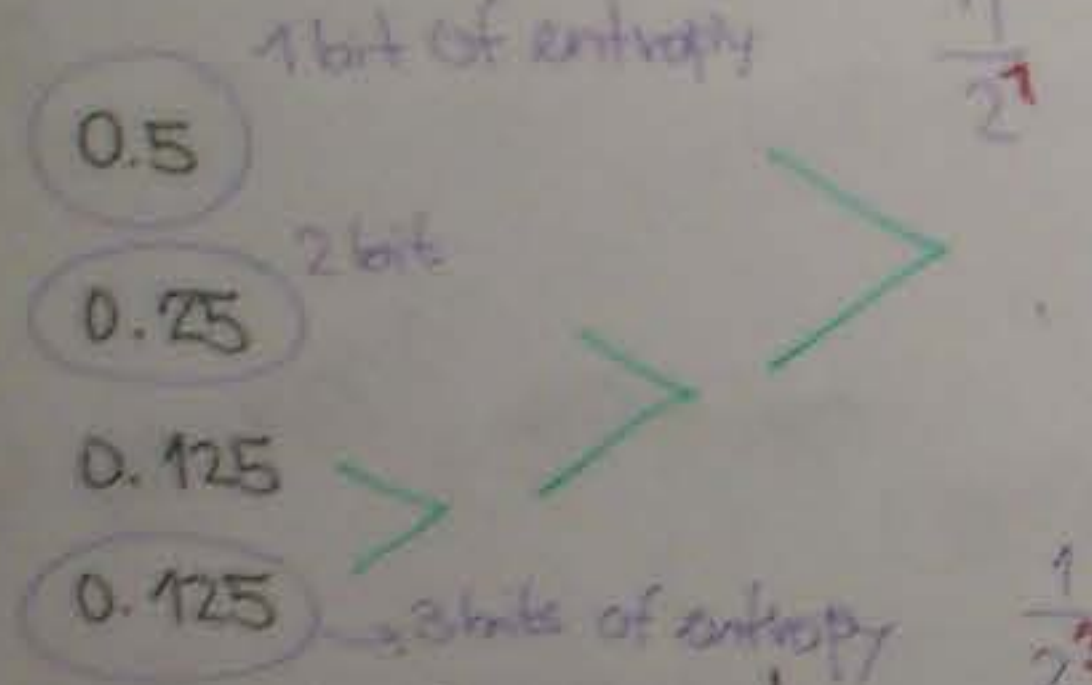
$p(a_1), p(a_2), \dots, p(a_m)$

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

Huffman's algorithm



$a \to 0$
$b \to 10$
$c \to 110$
$d \to 111$



result is the same!

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

Problems with Huffman coding

- optimality
- 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'

Kraft's inequality

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

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

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 $p(a_i) = \frac{1}{2^k}$

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'

Kraft's inequality

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

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

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

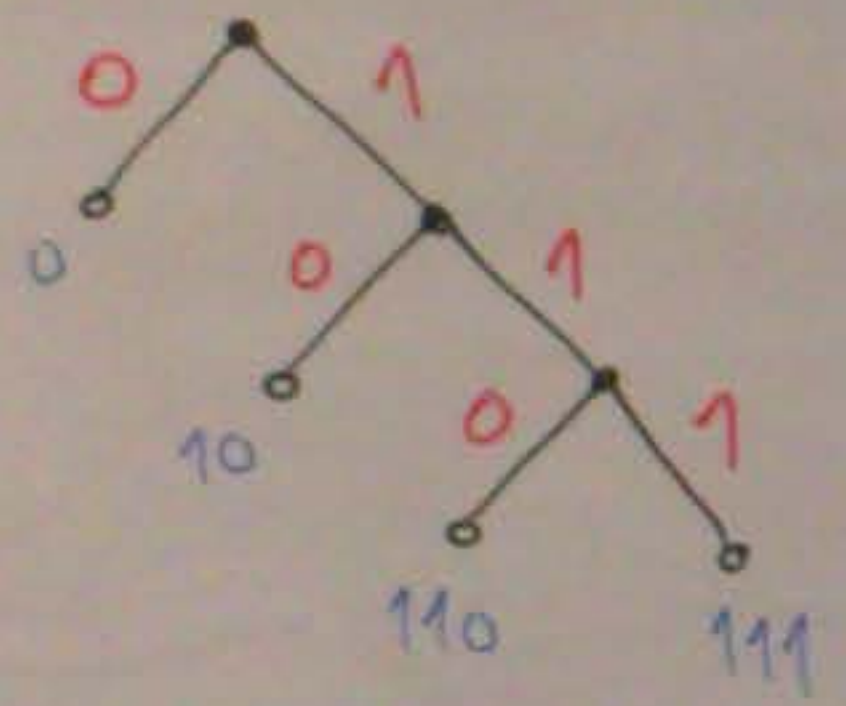
Ex: $\{1,2,3,3,4\}$ ... it cannot be constructed

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

Huffmann coding

- prefix-free code
  symbol length is given a-priori
  (see Kraft ineq.)

Ex: $l_i \in \{1,2,3,3\}$

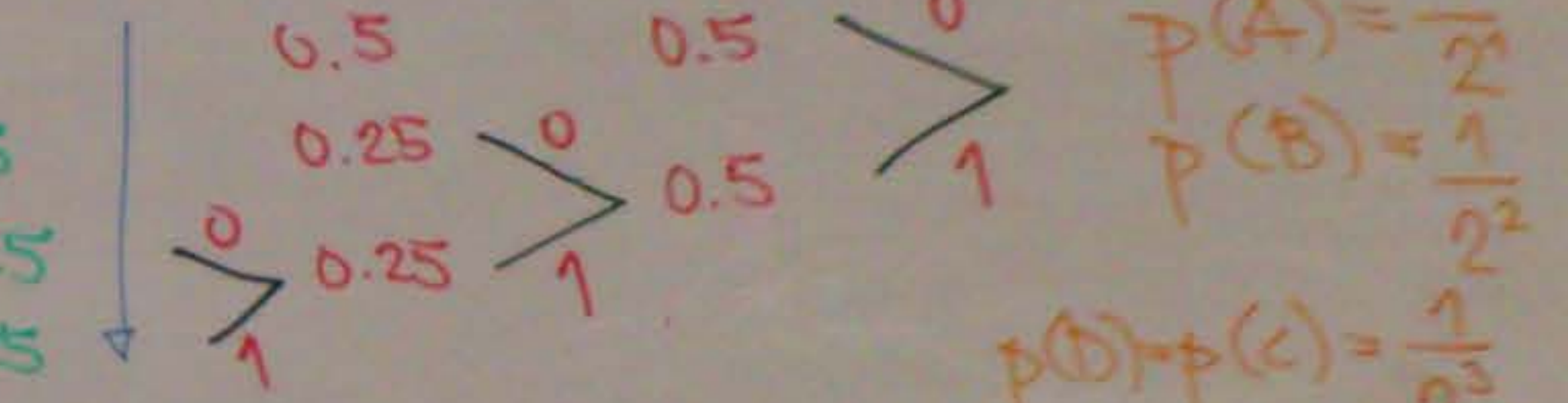


Q: why $\{1,2,3,3\}$ and not $\{2,2,2,2\}$?

Q: which source symbol shall be assigned when ?

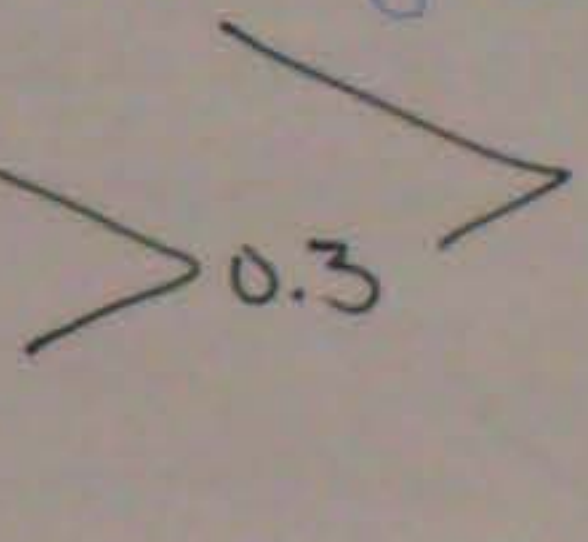
↳ this should reflect the entropy (σ, p_i)

Ex: A → p(A) = 0.5
B → p(B) = 0.25
C → p(C) = 0.125
D → p(D) = 0.125



descending order

Ex: p(A) → 0.7
p(B) → 0.2
p(C) → 0.09
p(D) → 0.01



↳ optimal!
Huff. c. works well esp. for these!!

Huffmann coding

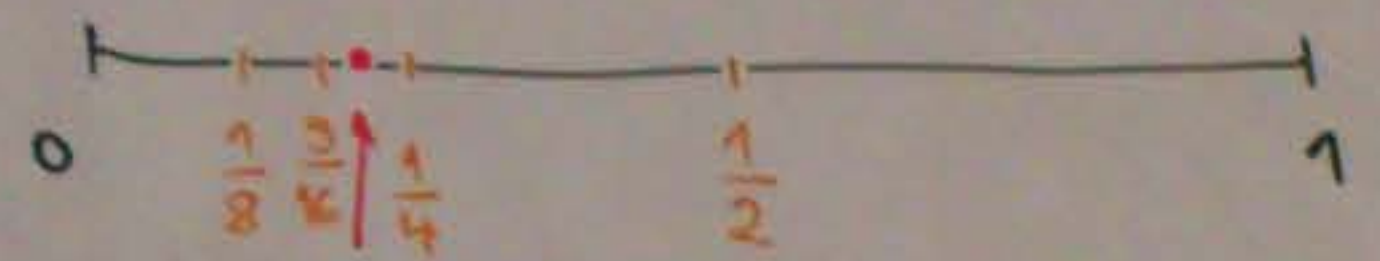
- pbdeus:
  a) optimality
  b) 2-pass approach
     pass #1: determine p_i (source symbol probabilities)
     pass #2: encode
     ⇒ adaptive Huffman coding
     start with a-priori distribution
     and upate it on-line
     → re-construction of Huffman tree
  c) it is symbol-based
     in English: 'the', 'a', 'is', 'qu'
     are very frequent

Arithmetic coding

Arithmetic coding will be a topic of lab #5...

arbitrary precision arithmetic

Idea: Represent the message as a rational number and represent this rational number in binary arithmetic on $[0,1]$



$$\frac{11}{16} = \frac{1}{2} + \frac{0}{4} + \frac{1}{8} + \frac{1}{16}$$

0.1011

a) $\frac{a}{b}$ ... a fraction representing the message

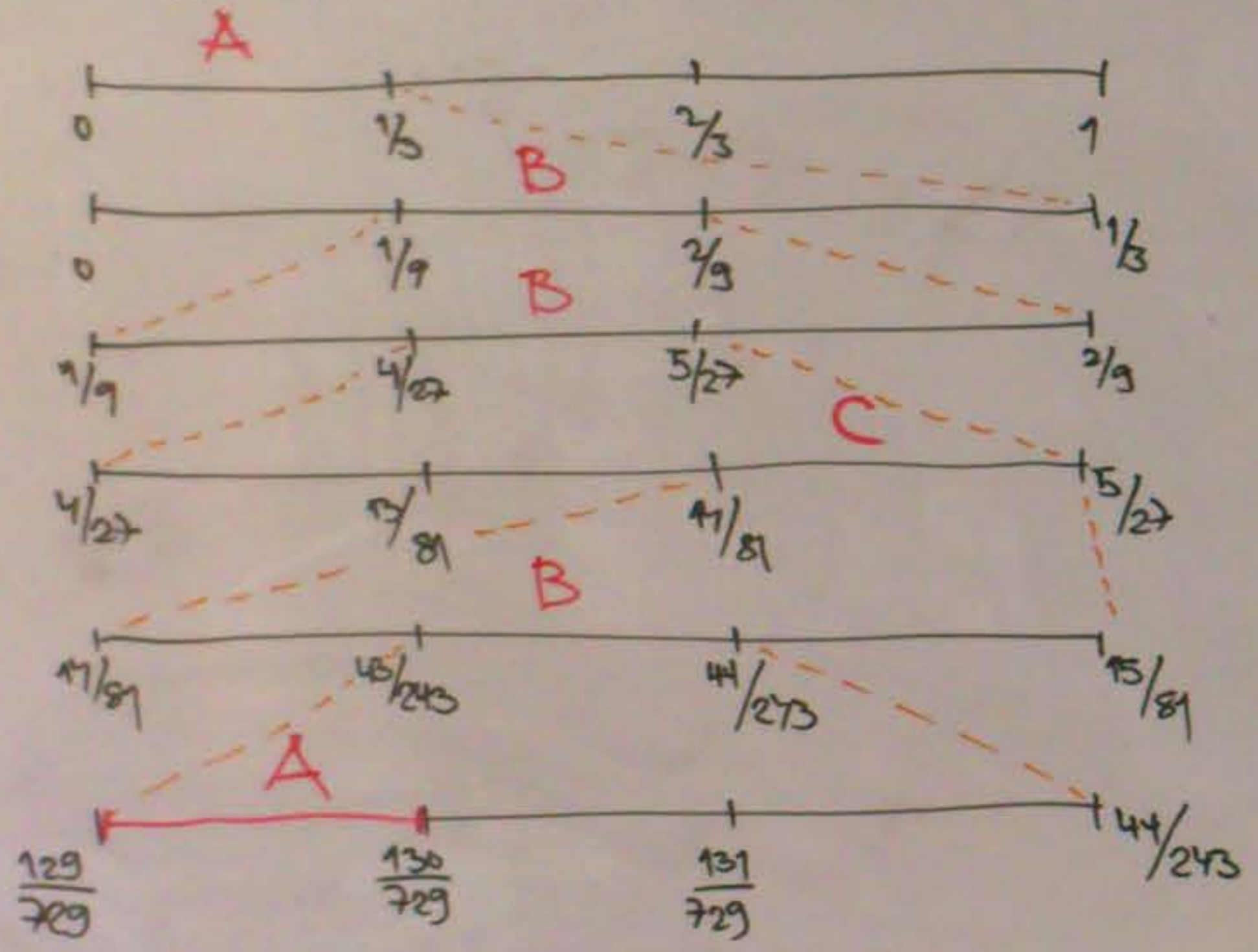
$0.1_2 = 2^{-1} = \frac{1}{2}$, $0.01_2 = 2^{-2} = \frac{1}{4}$

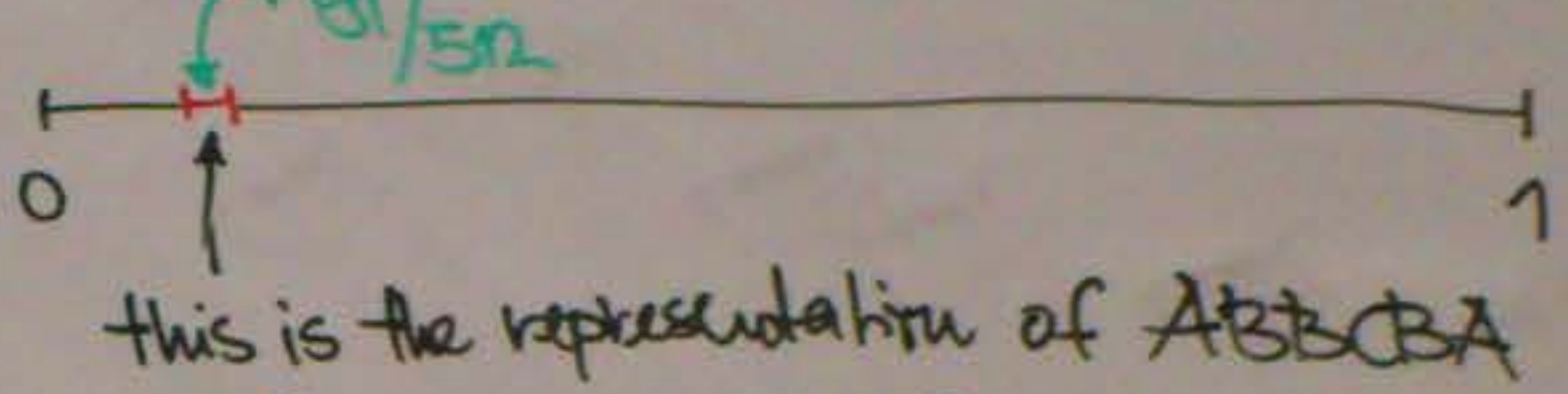
b) look for an interval in $[0,1]$ where $\frac{a}{b}$ lies by binary search (bisection)
→ 'bracket'

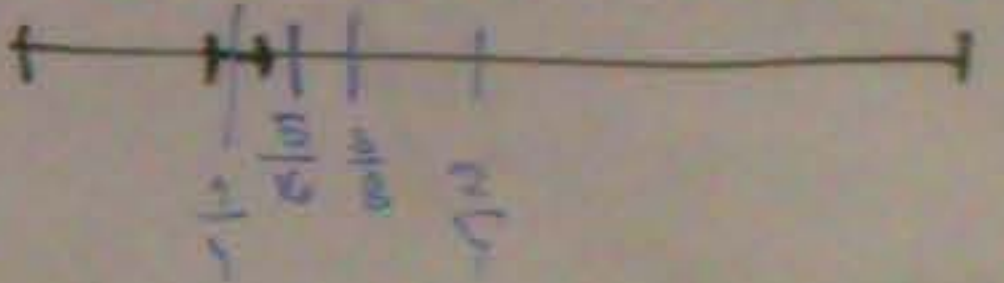
Ex: $(0.1)_{10} \equiv (0.0001)_2 \in [\frac{1}{16}, \frac{2}{16}]$
and $(0.2)_{10} \in [\frac{2}{16}, \frac{3}{16}]$

Ex: $p(A) = p(B) = p(C) = 1/3$

Arithmetic code for ABBCBA







now: find a binary fraction $\frac{x}{2^k}$ such that

$$\frac{x}{2^k} \in \left(\frac{129}{729}, \frac{130}{729}\right)$$

! the 'shortest' one !

the binary string representing the fraction is the encoded message!

$$\frac{129}{729} = 0.17695 < \frac{x}{2^k} < 0.17832 = \frac{130}{729}$$

$k=1: \frac{1}{2} > 0.17$ ×

$k=2: \frac{1}{4} > 0.17$ ×

⋮

$k=7: \frac{22}{128} < 0.176, \frac{23}{128} > 0.178$ ×

$k=8: \frac{45}{256} < 0.176, \frac{46}{256} > 0.178$ ×

$k=9: \frac{91}{512} > 0.176, \frac{91}{512} < 0.178$ ✓

binary rep. of $\frac{91}{512} = \frac{1}{8} + \frac{1}{32} + \frac{1}{64} + \frac{1}{256} + \frac{1}{512} \Rightarrow 0.001011011$