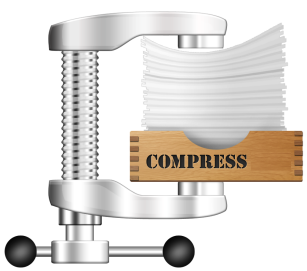
Week 2: Source Coding (Compression)

Prof. Michael Gastpar

Slides by Prof. M. Gastpar and Prof. em. B. Rimoldi

EPFL

Spring Semester 2025



Outline

LAST WEEK

1) REVIEW: BASIC PROBABILITY.

2) DEFINITION : **ENTROPY**

$$H_2(S) = \sum_i p(s) \log_2 \frac{1}{p(s)}$$ "surprise"

$$= \frac{7}{8} \log_2 \frac{8}{7} + \frac{1}{8} \log_2 8$$

$$\approx \frac{7}{8} \cdot 0.2 + \frac{1}{8} \cdot 3$$

average 'surprise'

(average) randomness.

$$\approx 0.55$$

$$H_2(S) = \sum_i p(s) \log_2 \frac{1}{p(s)}$$

$$\log_2\left(\frac{8}{7}\right) \approx 0.2$$

$$= \frac{1}{8}\log_2 8 + \frac{7}{8}\log_2 \frac{8}{7}$$

$$\approx \frac{3}{8} + \frac{7}{8} \cdot 0.2 \approx 0.55$$

DEFINITION (ENTROPY, UNCERTAINTY)

$$H_b(S) := -\sum_{s \in \mathrm{supp}(p_S)} p_S(s) \log_b p_S(s),$$

where $\mathrm{supp}(p_S) = \{s : p_S(s) > 0\}$.

But what does this definition mean?

- We will see how entropy is a fundamental "physical" converse bound to algorithms — it leads to **impossibility results.**
- At the same time, it gives **guidance on how to design algorithms that attain or approach** the fundamental bounds.
- Our first concrete test case is *source coding / data compression.*

ENTROPY BOUNDS

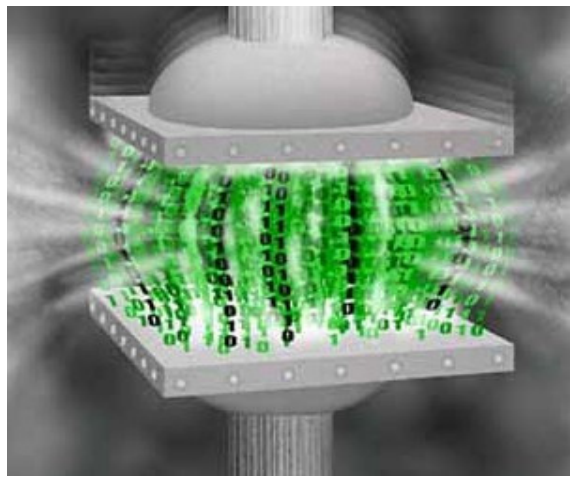
$$0 \leq H_b(S) \leq \log_b |\mathcal{A}|$$

Eq. iff S is not random

Eq. iff uniform

SOURCE CODING PURPOSE

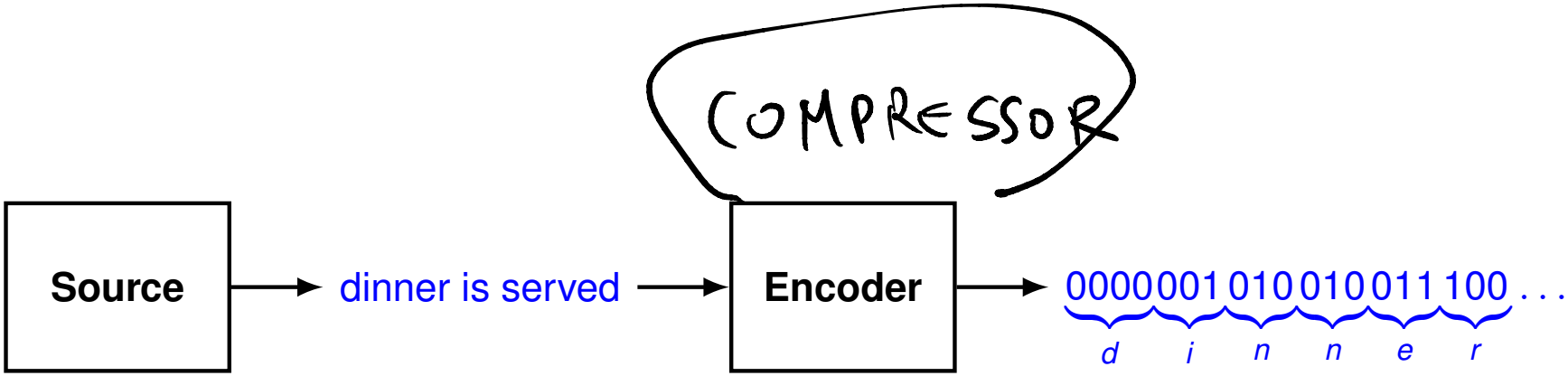
Source coding is often seen as a way to compress the source.



More generally, the goal of source coding is to efficiently describe the source output.

For a fixed description alphabet (often binary), we want to minimize the average description length.

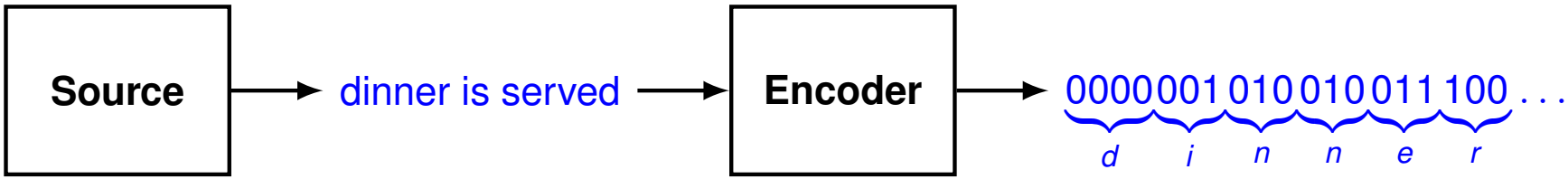
SETUP



The **source** is specified by the source alphabet $\mathcal{A}$ and by the source statistic.

EXAMPLE

The source alphabet is $\mathcal{A} = \{a, \ldots, z, 0, \ldots, 9\}$, and source symbols are independent and identically distributed (iid) over $\mathcal{A}$.



The **encoder** is specified by:

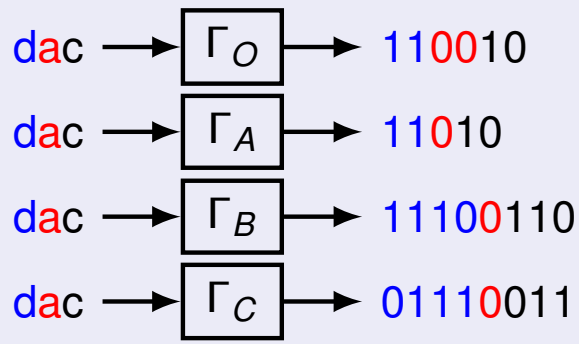
- the input alphabet $\mathcal{A}$ (the same as the source alphabet);
- the output alphabet $\mathcal{D}$ (typically $\mathcal{D} = \{0, 1\}$);
- the codebook $\mathcal{C}$ which consists of finite sequences over $\mathcal{D}$;
- by the one-to-one encoding map $\Gamma : \mathcal{A}^k \to \mathcal{C}$, where k is a positive integer.

For now, $k = 1$.

Example (Four little codes)

For each code, the encoding map Γ is specified in the following table:

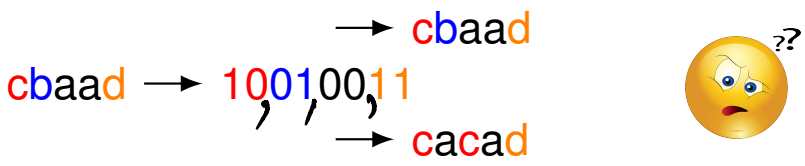
$\mathcal{A}$	Γ_O	Γ_A	Γ_B	Γ_C
a	00	0	0	0
b	01	01	10	01
c	10	10	110	011
d	11	11	1110	0111



The source alphabet $\mathcal{A}$, the k, the code alphabet $\mathcal{D}$ and the codebook $\mathcal{C}$ are implicit from the encoding map.

$C_O = \{00, 01, 10, 11\}$

We want to avoid the following problem (encoding map Γ_A):



DEFINITION

The code is *uniquely decodable* if every concatenation of codewords has a unique parsing into a sequence of codewords.

Recall that the encoding function Γ is one-to-one by assumption.

If we can identify codeword boundaries, we can decode sequences of codewords.

Uniquely decodable codes allow us to store (or transmit) sequences of codewords without storing (or sending) separators between codewords.

Example

Code A is **not** uniquely decodable:

$bc \mapsto 0110$

$ada \mapsto 0110$.

Try also to decode 0100.

$\mathcal{A}$	Γ_O	Γ_A	Γ_B	Γ_C
a	00	0	0	0
b	01	01	10	01
c	10	10	110	011
d	11	11	1110	0111

Example

Code B is uniquely decodable.

$\mathcal{A}$	Γ_O	Γ_A	Γ_B	Γ_C
a	00	0	0	0
b	01	01	10	01
c	10	10	110	011
d	11	11	1110	0111

Example: the codeword sequence 10111010110 can only be parsed as $10, 1110, 10, 110$.

It is uniquely decodable, because every 0 marks the end of a codeword. (The 0 plays the role of a separator.)

EXAMPLE

Code C is uniquely decodable.

$\mathcal{A}$	Γ_O	Γ_A	Γ_B	Γ_C
a	00	0	0	0
b	01	01	10	01
c	10	10	110	011
d	11	11	1110	0111

It is uniquely decodable, because every 0 marks the beginning of a codeword.

EXAMPLE

Code O is uniquely decodable.

$\mathcal{A}$	Γ_O	Γ_A	Γ_B	Γ_C
a	00	0	0	0
b	01	01	10	01
c	10	10	110	011
d	11	11	1110	0111

A fixed-length code is always uniquely decodable.

Prefix-free codes

Definition

If no codeword is a prefix of another codeword, the code is said to be prefix-free.

Example

The codeword 01 is a prefix of 011.
The codeword 10 is not a prefix of 110.

$\mathcal{A}$	Γ_O	Γ_A	Γ_B	Γ_C
a	00	0	0	0
b	01	01	10	01
c	10	10	110	011
d	11	11	1110	0111

Code O is prefix-free.
Code B is prefix-free (because of the 0 that marks the codeword end).
Codes A and C are not prefix-free.

- A prefix-free code is always uniquely decodable.
- A uniquely decodable code is **not necessarily** prefix-free.

Example

Code C is **not** prefix-free, yet it is uniquely decodable. (Its reverse — read every codeword from right to left — is prefix-free.)

$\mathcal{A}$	Γ_O	Γ_A	Γ_B	Γ_C
a	00	0	0	0
b	01	01	10	01
c	10	10	110	011
d	11	11	1110	0111

Note: A code is uniquely decodable iff its reverse is uniquely decodable.

A prefix-free code is also called **instantaneous** code.

- Think of phone numbers;
- Think about streaming: instantaneous codes minimize the decoding delay (for given codeword lengths).

Example

If we are using code C and the decoder sees 0, it might or might not be looking at a codeword.

The decoder needs to look past the end of a codeword.

$\mathcal{A}$	Γ_O	Γ_A	Γ_B	Γ_C
a	00	0	0	0
b	01	01	10	01
c	10	10	110	011
d	11	11	1110	0111

It is possible to construct uniquely decodable codes for which the decoder has to wait until the end of the transmission before it can parse, hence before it can decode.

This can lead to unbounded delays.

Example (curiosity: analogy with natural languages)

Suppose that we are describing numbers between 0 and 100:

- 83 $\longrightarrow$ quatre-vingt trois (not instantaneous)
- 83 $\longrightarrow$ Dreiundachtzig (not instantaneous)
- 83 $\longrightarrow$ ottanta tre (almost instantaneous)
- 83 $\longrightarrow$ otgonta treis (almost instantaneous)

Instantaneity is not the only thing that matters.

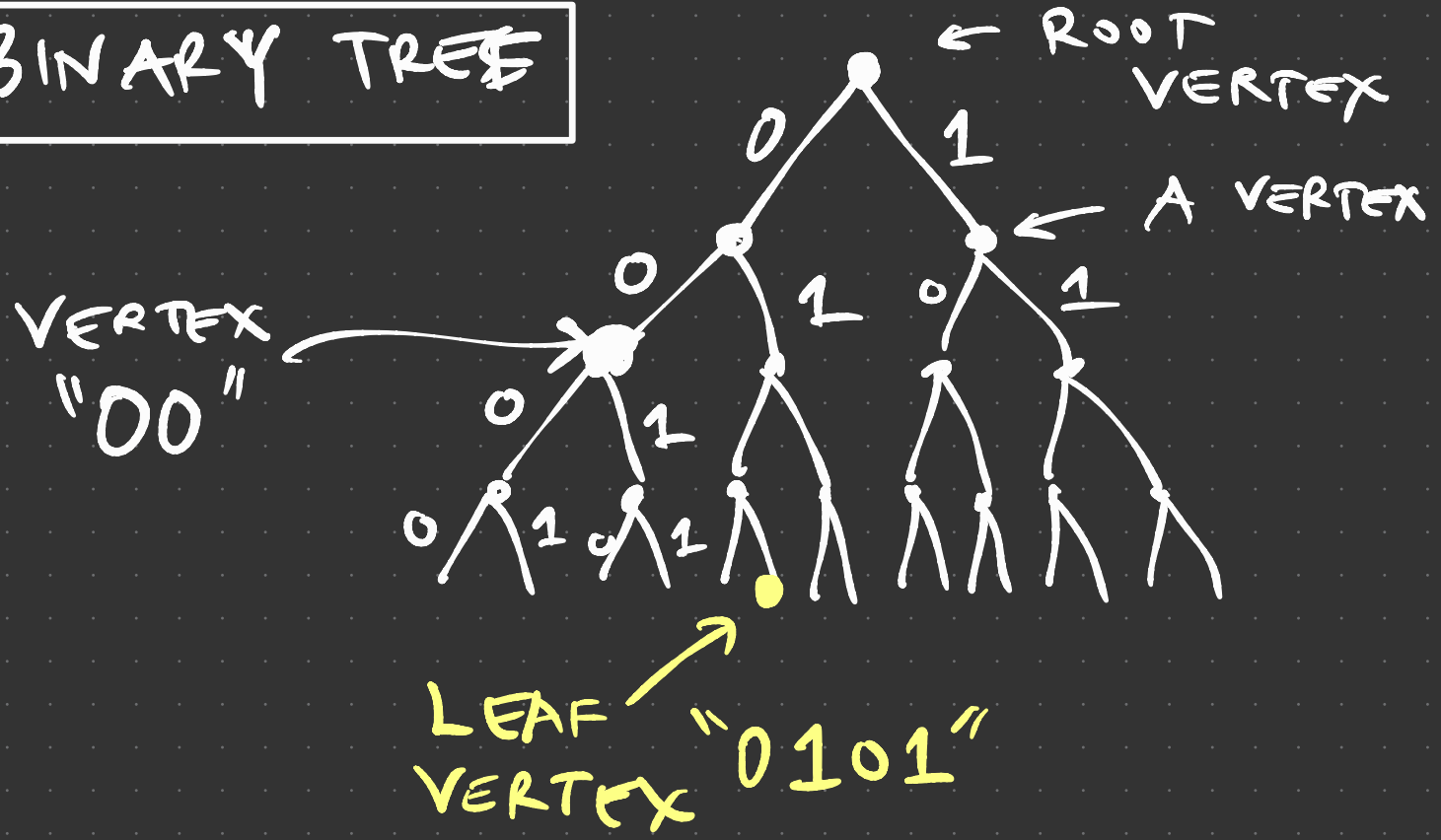
The length of the description matters as well! We'll come back to this shortly.

CODES FOR ONE RANDOM VARIABLE

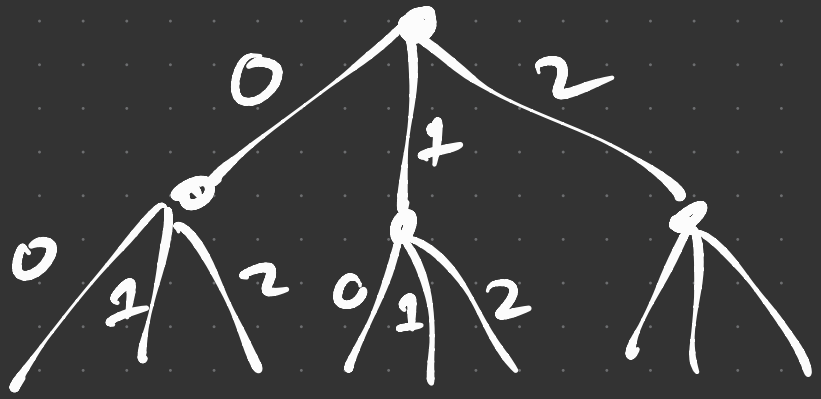
We start by considering codes that encode one single random variable $S \in \mathcal{A}$.

To encode a sequence $S_1, S_2, \ldots$ of random variables, we encode one random variable at a time.

BINARY TREE

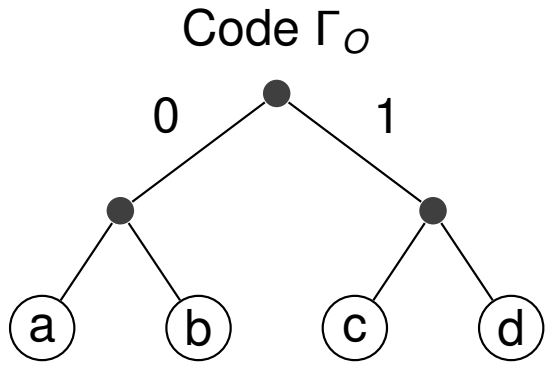


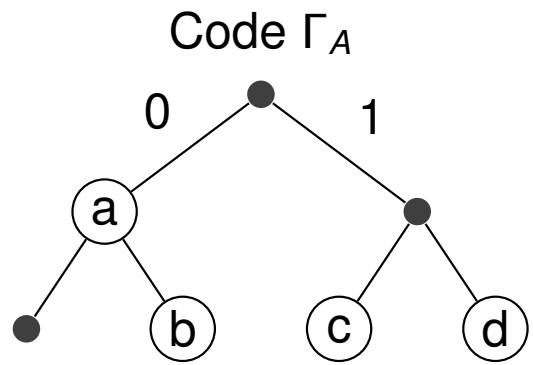
TERNARY TREE



Complete tree of a code

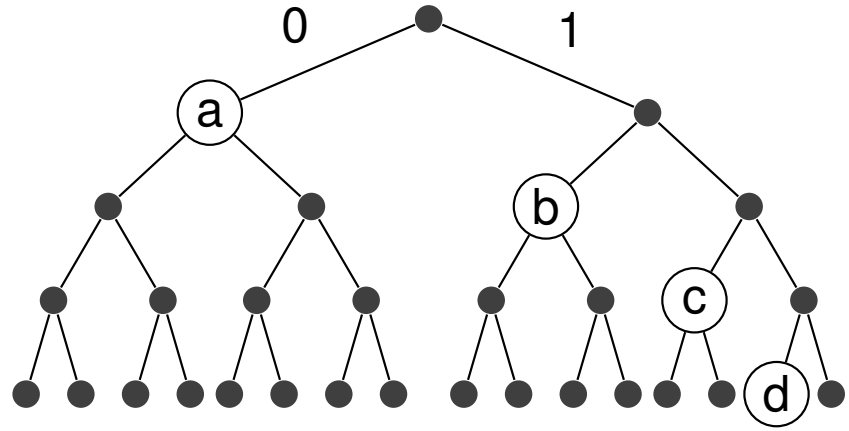
$\mathcal{A}$	Γ_O	Γ_A
a	00	0
b	01	01
c	10	10
d	11	11



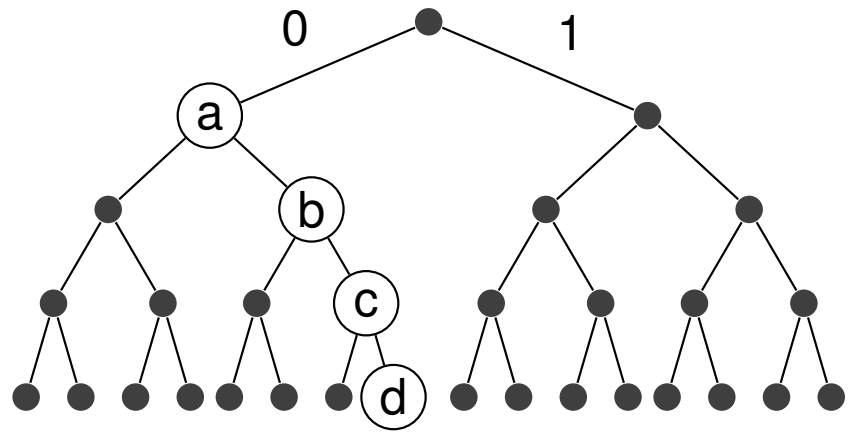


$\mathcal{A}$	Γ_B	Γ_C
a	0	0
b	10	01
c	110	011
d	1110	0111

Code Γ_B

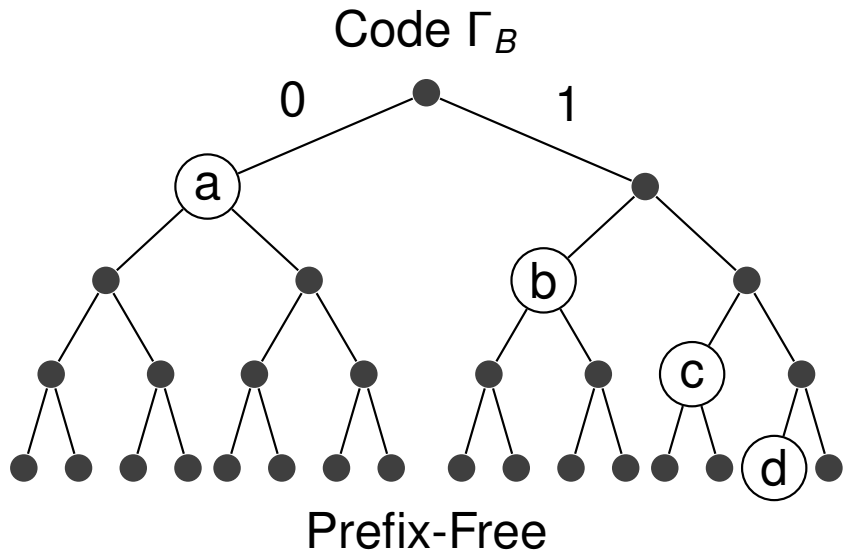


Code Γ_C

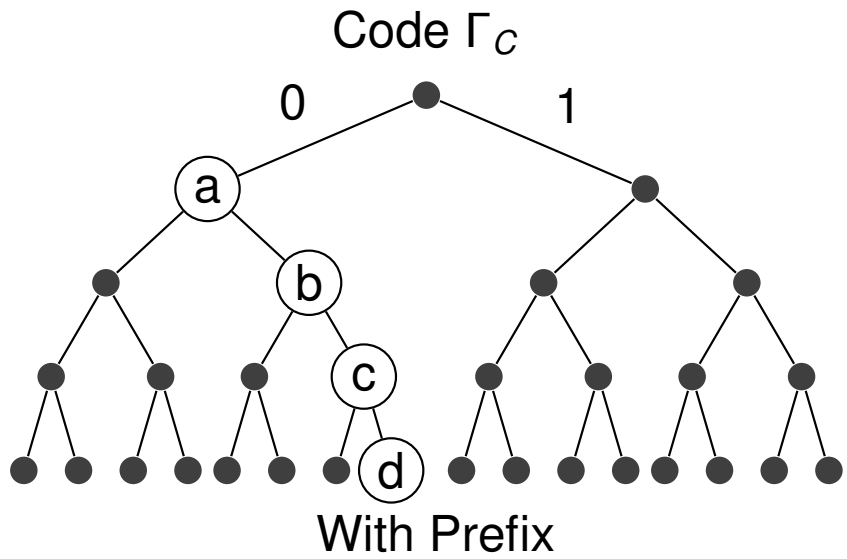


WITH/WITHOUT PREFIX

$\mathcal{A}$	Γ_B	Γ_C
a	0	0
b	10	01
c	110	011
d	1110	0111



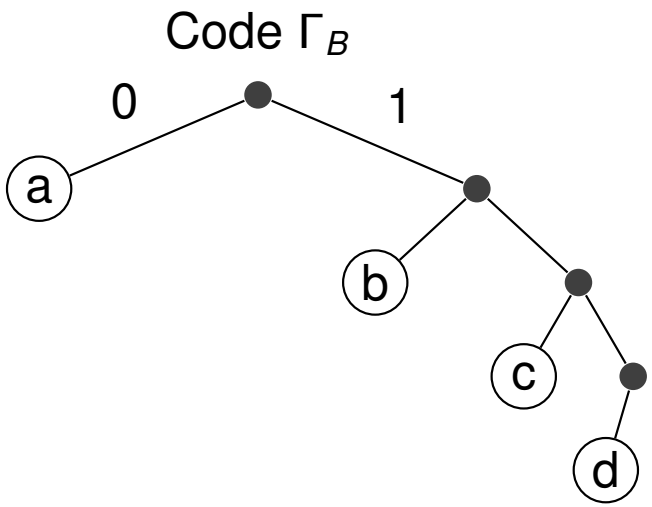
Prefix-Free



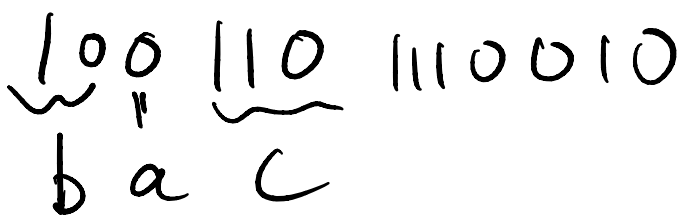
With Prefix

Decoding tree

- Obtained from the complete tree by keeping only branches that form a codeword.
- Useful to visualize the decoding process.



$\mathcal{A}$	Γ_O	Γ_A	Γ_B	Γ_C
a	00	0	0	0
b	01	01	10	01
c	10	10	110	011
d	11	11	1110	0111



Codeword Length

- The codeword length is defined the obvious way.
- Example:

$\mathcal{A}$	Γ_B	codeword lengths
a	0	1
b	10	2
c	110	3
d	1110	4

- We would like the average codeword-length to be as small as possible.

Kraft-McMillan

Part 1: Necessary condition for the code to be uniquely decodable

Theorem (Kraft-McMillan, Textbook Thm. 2.2)

If a D-ary code is uniquely decodable then its codeword lengths $l_1, \ldots, l_M$ satisfy

$$D^{-l_1} + \cdots + D^{-l_M} \leq 1 \quad \text{(Kraft's inequality)}.$$

Example

For Code O we have

$$2^{-2} + 2^{-2} + 2^{-2} + 2^{-2} = 1$$

Hence Kraft's inequality is fulfilled with equality.

$\mathcal{A}$	Γ_O	codeword lengths
a	00	2
b	01	2
c	10	2
d	11	2

Example

For Codes B and C we have $2^{-1} + 2^{-2} + 2^{-3} + 2^{-4} = 0.9375 < 1$.

Kraft's inequality is fulfilled.

Code B is prefix-free.

Code C is not prefix-free (but there is a prefix-free code that has the same codeword lengths).

$\mathcal{A}$	Γ_A	Γ_B	Γ_C
a	0	0	0
b	01	10	01
c	10	110	011
d	11	1110	0111

Recall Kraft-McMillan, Part 1:

Theorem (Kraft-McMillan, Textbook Thm. 2.2)

If a D-ary code is uniquely decodable then its codeword lengths $l_1, \ldots, l_M$ satisfy

$$D^{-l_1} + \cdots + D^{-l_M} \leq 1 \quad \text{(Kraft's inequality)}.$$

Exercise

What is the **contrapositive** of Kraft-McMillan part 1?

See next example.

Example

For Code A we have $2^{-1} + 2^{-2} + 2^{-2} + 2^{-2} = 1.25 > 1$.

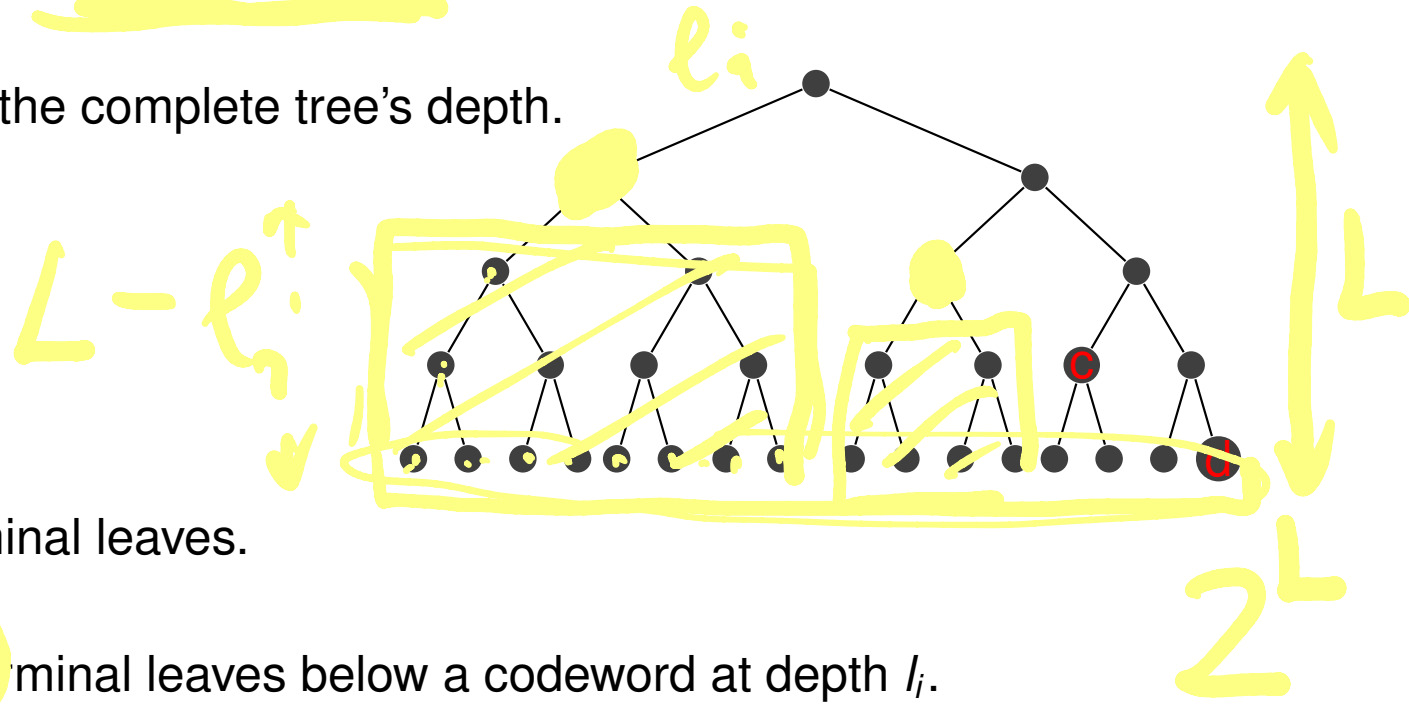
Kraft-McMillan's inequality is not fulfilled.

There exists no uniquely decodable code with those codeword lengths.

$\mathcal{A}$	Γ_A	Γ_B	Γ_C
a	0	0	0
b	01	10	01
c	10	110	011
d	11	1110	0111

Proof of K-MM Part I: We prove a slightly weaker result, namely that the codeword lengths of prefix-free codes satisfy K-MM's inequality. [1]

Let $L = \max_i l_i$ be the complete tree's depth.



There are D^L terminal leaves.

There are D^{L-l_i} terminal leaves below a codeword at depth l_i.

No two codewords share a terminal leaf. (The code is prefix-free.)

Hence $D^{L-l_1} + D^{L-l_2} + \cdots + D^{L-l_M} \leq D^L$.

After dividing both sides by D^L we obtain Kraft's inequality

$D^{-l_1} + D^{-l_2} + \cdots + D^{-l_M} \leq 1.$ □

[1] The full proof appears in the LTU book

Recall Kraft-McMillan, Part 1:

Theorem (Kraft-McMillan, Textbook Thm. 2.2)

If a D-ary code is uniquely decodable then its codeword lengths $l_1, \ldots, l_M$ satisfy

$$D^{-l_1} + \cdots + D^{-l_M} \leq 1 \quad \text{(Kraft's inequality)}$$

Exercise

What is the **converse** of Kraft-McMillan part 1?

The **converse** of Kraft-McMillan part 1 is not true. (Consider e.g. two codewords: 01 and 0101.)

However, the following statement is almost as good.

THEOREM (KRAFT-MCMILLAN, TEXTBOOK THM. 2.2)

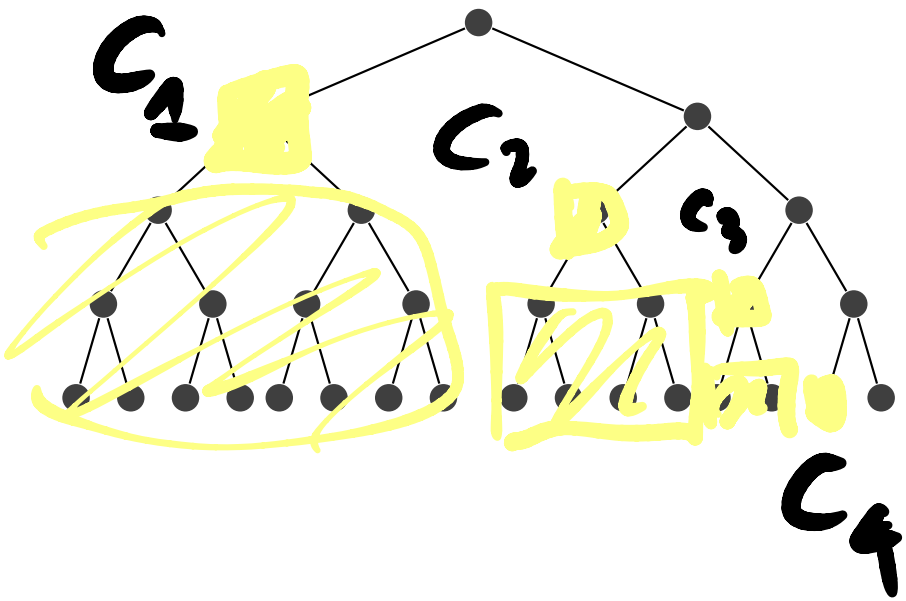
If the positive integers $l_1, \ldots, l_M$ satisfy Kraft's inequality for some positive integer D, then there exists a D-ary prefix-free code (hence uniquely decodable) that has codeword lengths $l_1, \ldots, l_M$.

EXERCISE

Let $l_1 = 1, l_2 = 2, l_3 = 3, l_4 = 4$. Because $\sum_{i=1}^{4} 2^{-l_i} = \frac{15}{16} < 1$, there exists a binary prefix-free code with the given codeword lengths.
Construct such a code.

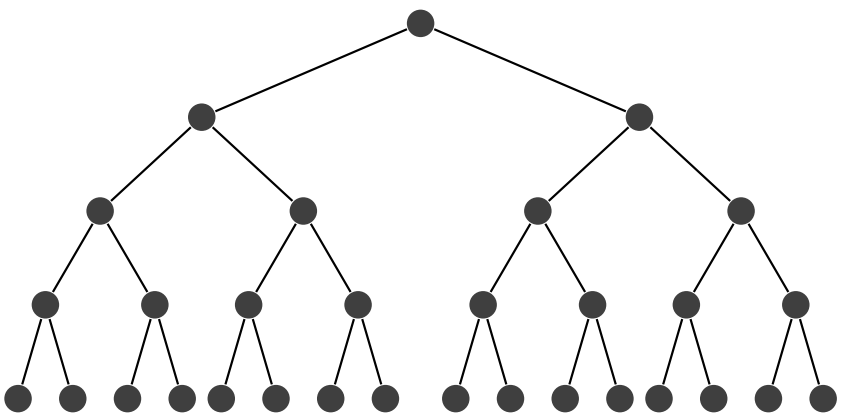
(Use the following tree as a starting point.)

(It is convenient to order the codeword lengths in increasing order.)

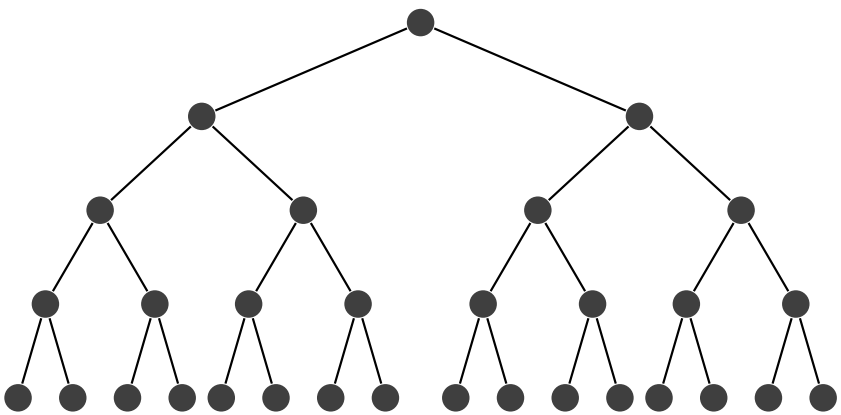


Proof Outline of K-MM part 2

- suppose that $l_1 \leq l_2 \leq \cdots \leq l_M$ are the desired codeword lengths
- which fulfill the Kraft-McMillan's inequality
- start with a full D-ary tree of depth $L = l_M$
- choose a node at depth l_1. Declare the path from the root to that node as codeword c_1
- delete the subtree to the chosen node. This guarantees that subsequent codewords are prefix-free



- proceed similarly to find prefix-free codewords of lengths $l_2, \ldots, l_i$ for $i < M$
- because the lengths $l_1, \ldots, l_i$ satisfy Kraft-McMillan with strict inequality, there are unused terminal leaves
- hence we can choose a codeword of length l_{i+1} that does not extend any of the already chosen codewords. Etc.



□

Important Consequence of Kraft-McMillan

Part I

If a D-ary code is uniquely decodable, then its codeword lengths $l_1, \ldots, l_M$ satisfy Kraft's inequality

$$D^{-l_1} + \cdots + D^{-l_M} \leq 1.$$

Part II

If the positive integers $l_1, \ldots, l_M$ satisfy Kraft's inequality for some positive integer D, then there exists a D-ary prefix-free code that has those codeword lengths.

The Kraft-McMillan theorem implies that any uniquely decodable code can be substituted by a prefix-free code of the same codeword lengths.

PREFIX-FREE CODES

Our focus will be on prefix-free codes. Reasons:

- no loss of optimality: codewords can be as short as for any uniquely decodable code;
- a prefix-free codeword is recognized as soon as its last digit is seen:
  - important for, e.g., a phone number;
  - advantageous to limit the decoding delay in, say, streaming;

Average Codeword Length

- The typical use of a code is to encode a sequence of random variables into the corresponding codeword sequence.
- We are interested in minimizing the average codeword-length.

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

$D = 2$

$s \in \mathcal{A}$	$\Gamma(s)$	$\ell(s)$
a	0	1
b	10	2
c	110	3
d	1111	4

Ex: $\mathcal{A} = \{a, b, c, d\}$ $D = 2$

$s \in \mathcal{A}$	$\Gamma(s)$	$\ell(s)$	$p(s)$
a	0	1	0.05
b	10	2	0.05
c	110	3	0.1
d	1111	4	0.8

$$L(S, \Gamma) = \mathbb{E}[\text{Length}] = 0.05 \cdot 1 + 0.05 \cdot 2 + 0.1 \cdot 3 + 0.8 \cdot 4 = 3.65 \ \frac{\text{bits}}{\text{symbol}}$$

Definition (Average codeword length)

Let $l(\Gamma(s))$ be the length of the codeword associated to $s \in \mathcal{A}$.

The average codeword-length is

$$L(S, \Gamma) \stackrel{\text{def}}{=} \sum_{s \in \mathcal{A}} p_S(s) l(\Gamma(s)).$$

$L(p(s), \Gamma)$

Units

The units of $L(S, \Gamma)$ are code symbols.

When $D = 2$, the units of $L(S, \Gamma)$ are bits.

Example

Can we do better than Γ_O with a binary code?

$\mathcal{A}$	$p_S(s)$	Γ_O	Γ_B	$\Gamma_{B'}$	Γ_C
a	0.05	00	0	1110	0
b	0.05	01	10	110	01
c	0.1	10	110	10	011
d	0.8	11	1110	0	0111

$$L(S, \Gamma_B) = 0.05 \times 1 + 0.05 \times 2 + 0.1 \times 3 + 0.8 \times 4 = 3.65$$

$$L(S, \Gamma_{B'}) = 0.05 \times 4 + 0.05 \times 3 + 0.1 \times 2 + 0.8 \times 1 = 1.35$$

$$L(\Gamma_{B'}) < L(\Gamma_O) < L(\Gamma_B) = L(\Gamma_C).$$

Is there a lower bound to the average codeword-length?

AVERAGE CODEWORD LENGTH: LOWER BOUND

THEOREM (TEXTBOOK THM 3.1)

Let $\Gamma : \mathcal{A} \to \mathcal{C}$ be the encoding map of a D-ary code for the random variable $S \in \mathcal{A}$.

If the code is uniquely decodable, then the average codeword-length is lower bounded by the entropy of S, namely

$$H_D(S) \leq L(S, \Gamma),$$

with equality iff, for all $s \in \mathcal{A}$, $p_S(s) = D^{-l(\Gamma(s))}$. An equivalent condition is $l(\Gamma(s)) = \log_D \frac{1}{p_S(s)}$.

$$
\begin{aligned}
H(S) &= \sum_s p(s)\,\ell(s) \\
&= -\sum_s p(s)\log p(s) - \sum_s p(s)\,\ell(s) \\
&= -\sum_s p(s)\log p(s) - \sum_s p(s)\log 2^{\ell(s)} \\
&= -\sum_s p(s)\log\left(p(s)\,2^{\ell(s)}\right) \\
&= \sum_s p(s)\log\left(\frac{1}{p(s)}\,2^{-\ell(s)}\right) \\
&\le \sum_s p(s)\left(\frac{1}{p(s)}\,2^{-\ell(s)} - 1\right) \quad C
\end{aligned}
$$

$$= \left(\sum_s 2^{-\ell(s)} - \underbrace{\sum_s p(s)}_{=1} \right) C$$

$$\leq 0$$

Proof:

$$
\begin{aligned}
H_D(S) - L(S,\Gamma) &= \sum_i p_i \log_D \frac{1}{p_i} - \sum_i p_i \log_D D^{l_i} \\
&= \sum_i p_i \log_D \frac{1}{p_i D^{l_i}} \\
&\overset{(IT-Ineq.)}{\leq} \log_D(e) \sum_i p_i \left(\frac{1}{p_i D^{l_i}} - 1 \right) \\
&= \log_D(e) \left(\sum_i D^{-l_i} - \sum_i p_i \right) \\
&= \log_D(e) \left(\sum_i D^{-l_i} - 1 \right) \\
&\overset{(K-MM)}{\leq} 0.
\end{aligned}
$$

The first inequalities hold with equality iff, for all i, $p_i = D^{-l_i}$. When this is the case, also the second inequality holds with equality. □

YESTERDAY

$s \in \mathcal{A}$	$\Gamma(s)$	$\ell(\Gamma(s))$	$p(s)$
hello	0 0 1 0 2 1	6	$1/100$
good	2 2 1 2 2	5	$2/100$
morning	0 0 0	3	$1/1000$
⋮	⋮	⋮	↓
			average length

YESTERDAY

- TREE REPRESENTATION
- DEF: UNIQUELY DECODABLE CODE
- DEF: PREFIX-FREE CODE.
- PREFIX-FREE $\Rightarrow$ UNIQUELY DECODABLE

YESTERDAY

- CODE WITH CODEWORD LENGTHS
$$\{ \ell_1, \ell_2, \ldots, \ell_M \}$$
- IF UNIQUELY DECODABLE, THEN MUST HAVE
$$\sum_i D^{-\ell_i} \leq 1.$$

YESTERDAY

- IF $\sum_i D^{-\ell_i} \leq 1$,

  THEN THERE EXISTS

  A D-ARY PREFIX-FREE CODE WITH CODEWORD LENGTHS

  $\{\ell_1, \ell_2, \ldots, \ell_M\}$

YESTERDAY

- AVERAGE CODEWORD LENGTH:

$$L(S, \Gamma) = \sum_s p(s)\, \ell(\Gamma(s))$$

- MUST SATISFY:

$$H_D(S) \leq L(S, \Gamma)$$

Example (cont.)

- $H(S) = -2 \times 0.05 \times \log_2 0.05 - 0.1 \times \log_2 0.1 - 0.8 \times \log_2 0.8 = 1.022$
- Recall that $L(S, \Gamma_{B'}) = 1.35$
- We verify $H(S) < L(\Gamma_{B'}) < L(\Gamma_O) < L(\Gamma_B) = L(\Gamma_C)$

$\mathcal{A}$	$p_S(s)$	Γ_O	Γ_B	$\Gamma_{B'}$	Γ_C
a	0.05	00	0	1110	0
b	0.05	01	10	110	01
c	0.1	10	110	10	011
d	0.8	11	1110	0	0111

A Key Observation

The right-hand side of

$$L(S,\Gamma) \stackrel{\text{def}}{=} \sum_{s\in\mathcal{A}} p(s) l(\Gamma(s))$$

and

$$H_D(S) \stackrel{\text{def}}{=} \sum_{s\in\mathcal{A}} p(s) \log_D \frac{1}{p_S(s)}$$

are identical if $l(\Gamma(s)) = \log_D \frac{1}{p_S(s)}$.

- Unfortunately $l(\Gamma(s)) = \log_D \frac{1}{p_S(s)}$ is often not possible (not an integer).
- How about choosing $l(\Gamma(s)) = \lceil \log_D \frac{1}{p_S(s)} \rceil$?
- Is it a valid choice for a prefix-free code? (Is Kraft's inequality satisfied?)

Good Code: Shannon-Fano Codes

Theorem (Textbook Thm 3.2)

- For every random variable $S \in \mathcal{A}$ and every integer $D \geq 2$, there exists a prefix-free D-ary code for S such that, for all $s \in \mathcal{A}$,

$$l(\Gamma(s)) = \lceil -\log_D p_S(s) \rceil$$

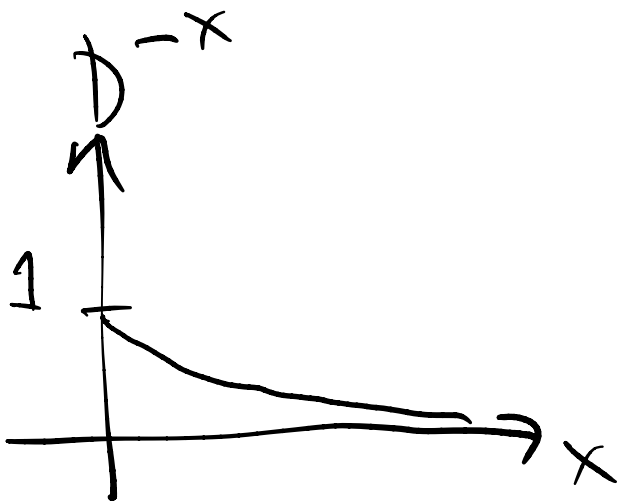
- Such codes are called *D-ary Shannon-Fano codes*.

Proof: Using a simplified notation, we need to check that the choice

$$l_i = \lceil -\log_D p_i \rceil, \quad i = 1, \ldots, |\mathcal{A}|$$

fulfills Kraft's inequality.

We use the fact that D^{-x} is a monotonically decreasing function of x for $D > 1$.



$$\begin{aligned}
\sum_i D^{-l_i} &= \sum D^{-\lceil -\log_D p_i \rceil} \\
&\leq \sum_i D^{\log_D p_i} \\
&= \sum_i p_i \\
&= 1
\end{aligned}$$

□

EXERCISE

Construct a binary Shannon-Fano code for the following random variable.

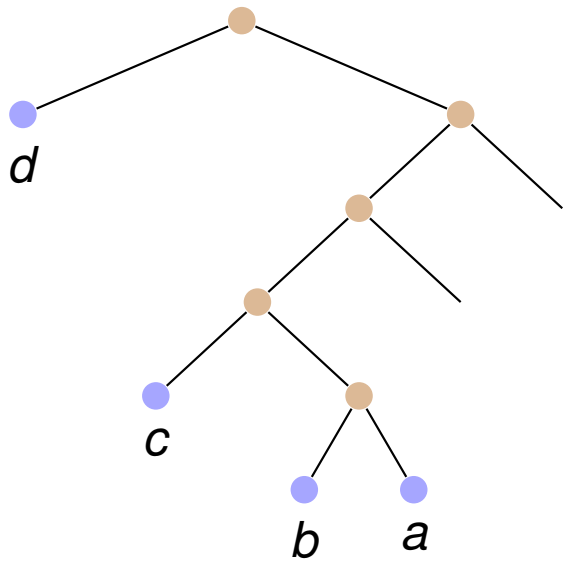
$s \in \mathcal{A}$	$p_S(s)$	$\lceil -\log_2 p_S(s) \rceil$
a	0.05	
b	0.05	
c	0.1	
d	0.8	

$s \in \mathcal{A}$	$p_S(s)$	$\lceil -\log_2 p_S(s) \rceil$
a	0.05	
b	0.05	
c	0.1	
d	0.8	

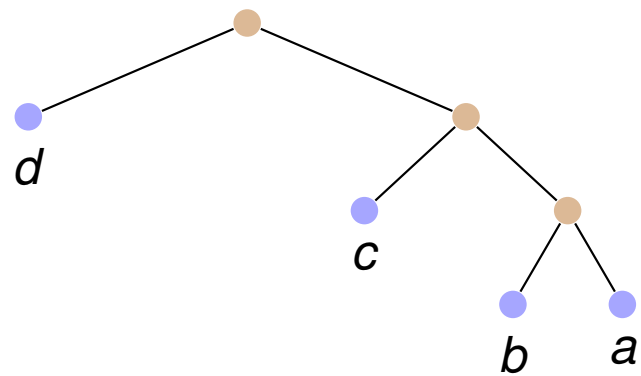
x	0.05	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
$-\log_2(x)$	4.3219	3.3219	2.3219	1.7370	1.3219	1.0000	0.7370	0.5146	0.3219	0.1520

$s \in \mathcal{A}$	$p_S(s)$	$\lceil -\log_2 p_S(s) \rceil$
a	0.05	5
b	0.05	5
c	0.1	4
d	0.8	1

$s \in \mathcal{A}$	$p_S(s)$	$\lceil -\log_2 p_S(s) \rceil$
a	0.05	5
b	0.05	5
c	0.1	4
d	0.8	1



Shannon-Fano code



shorter code

Theorem (Textbook Thm 3.3)

The average codeword-length of a D-ary Shannon-Fano code for the random variable S fulfills

$$H_D(S) \leq L(S, \Gamma_{SF}) < H_D(S) + 1.$$

Proof: It suffices to prove the upper bound (we have already proved the lower bound).

First suppose that we could use $l_i = -\log p_i$. The average length would be

$$L(S, \Gamma) = \sum_i p_i l_i = \sum_i p_i(-\log_D p_i) = H_D(S).$$

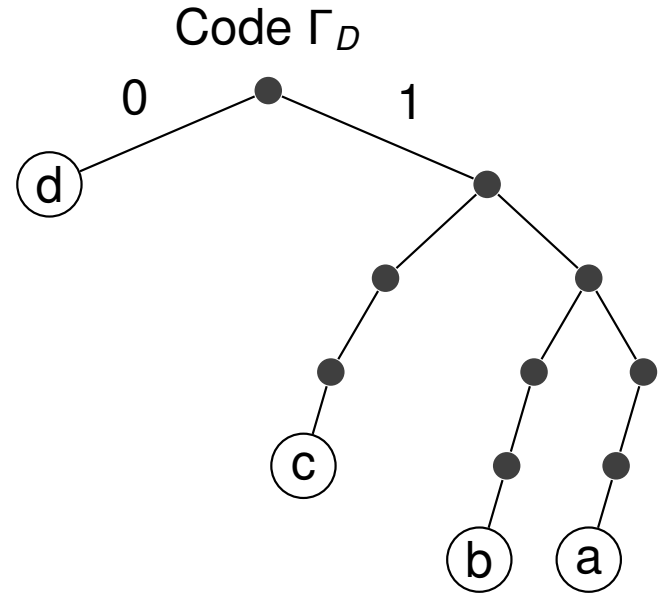
Instead we use $l_i = \lceil -\log p_i \rceil < -\log p_i + 1$.

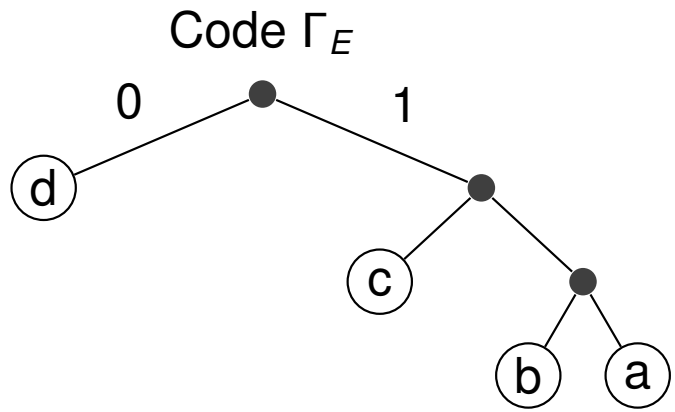
Since each term of an average increases by less than 1, the average itself increases by less than 1. □

Example

Does there exist a binary code Γ_E having a shorter average length than the binary Shannon-Fano code Γ_D?

$\mathcal{A}$	$p_S(s)$	Γ_D	Γ_E
a	0.05	11100	111
b	0.05	11000	110
c	0.1	1000	10
d	0.8	0	0
$L(S, \Gamma)$		1.7	1.3

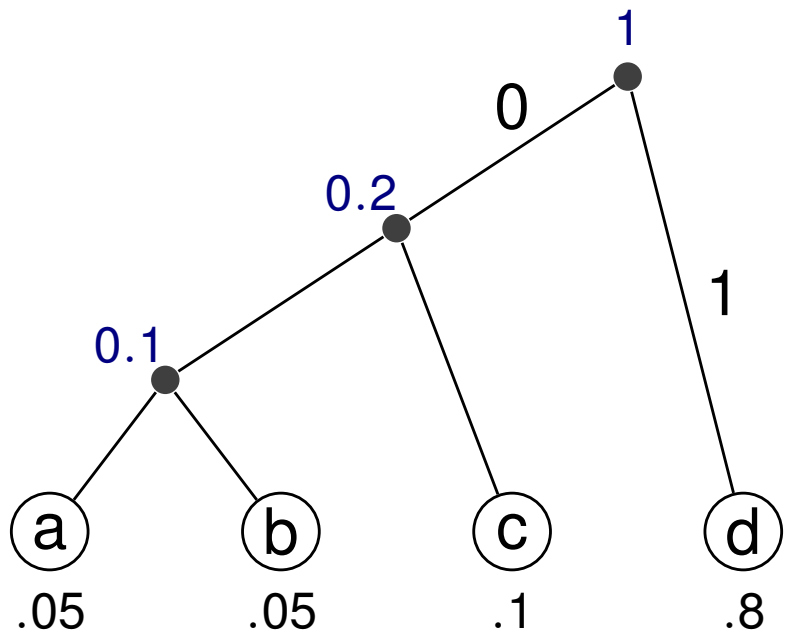




So, Shannon-Fano codes are good, but not optimal.

Optimal Code: Huffman Code

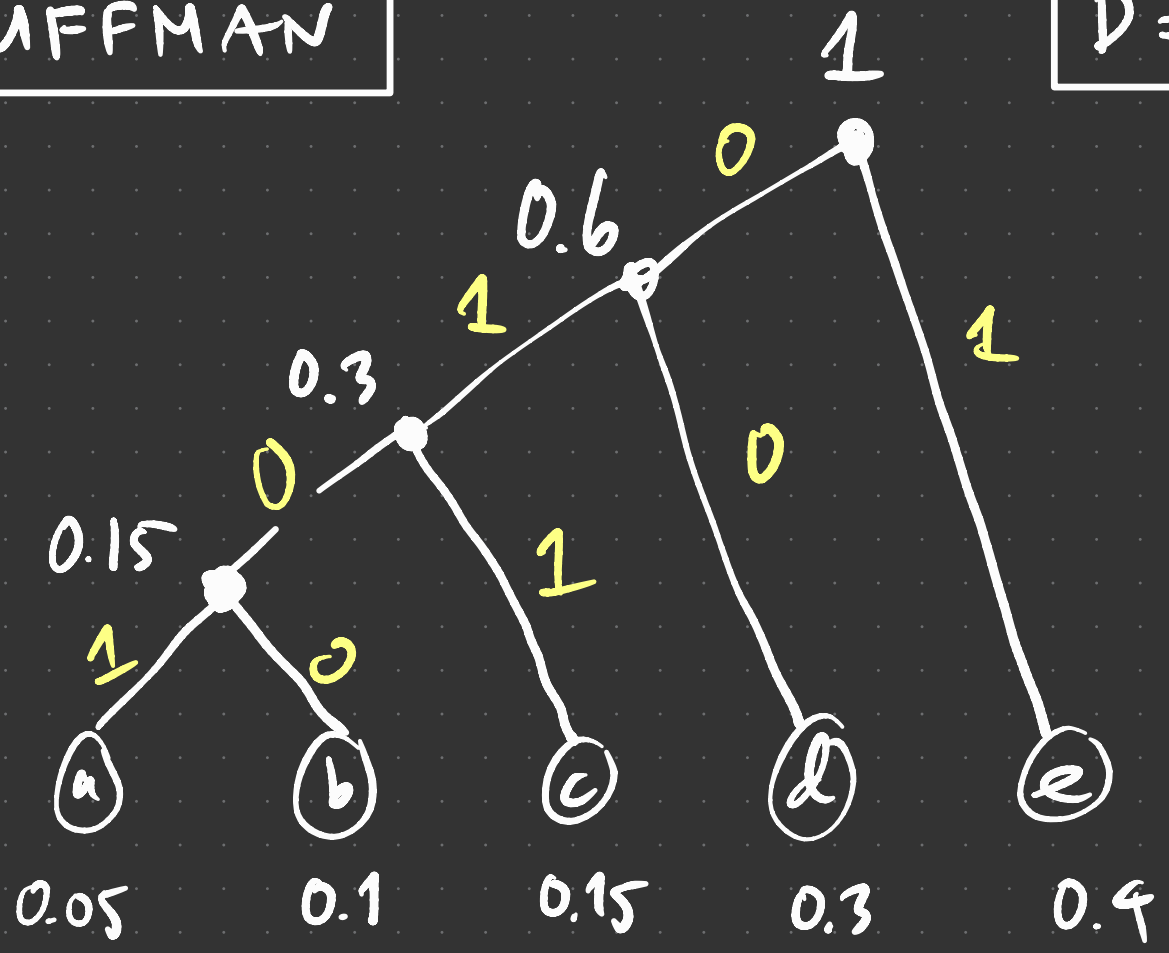
Unlike the Shannon-Fano code, the construction of the Huffman code starts from the leaves.



A Huffman code for a random variable is prefix-free and optimal, in the sense that no code can achieve a smaller average codeword-length. (To be proved.)

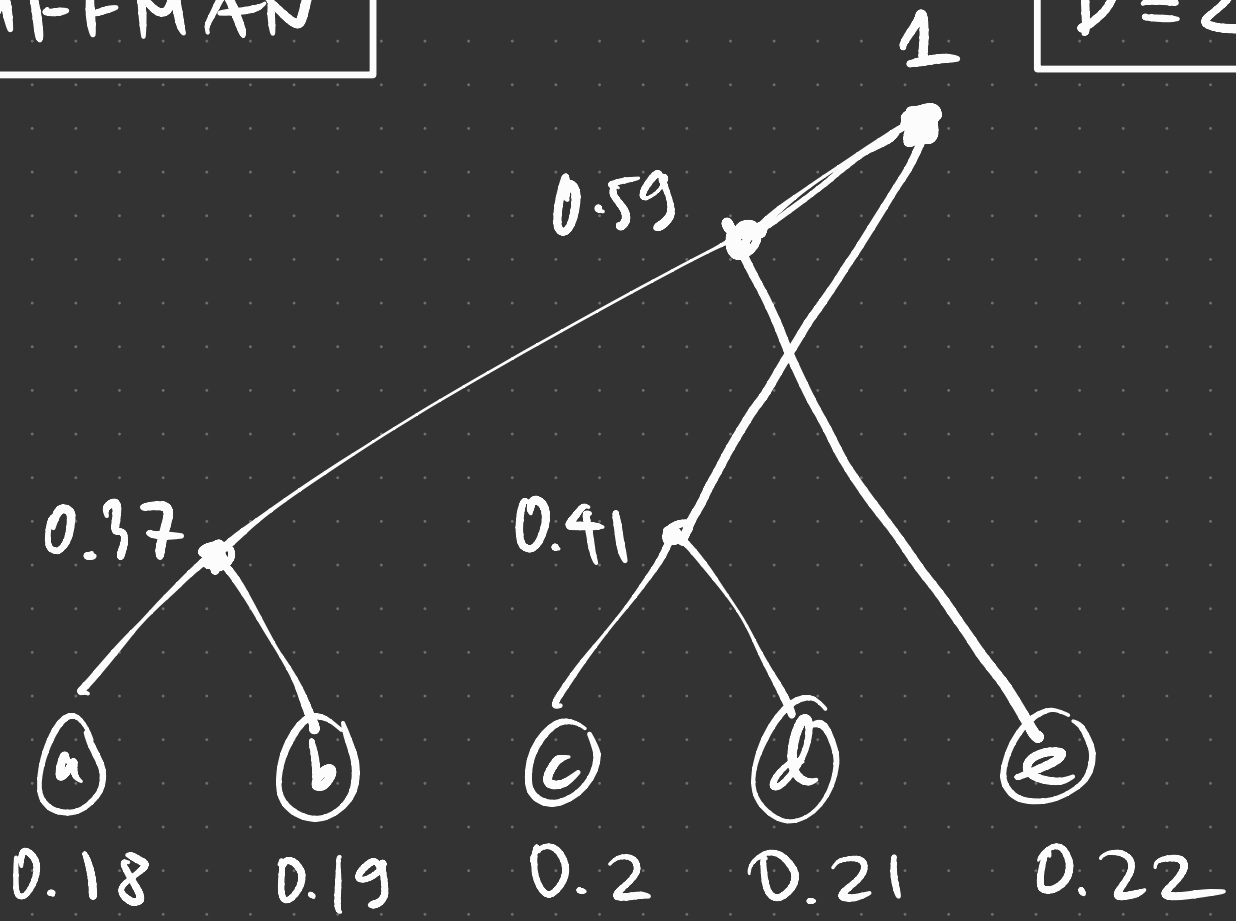
HUFFMAN

D = 2



HUFFMAN

D = 2



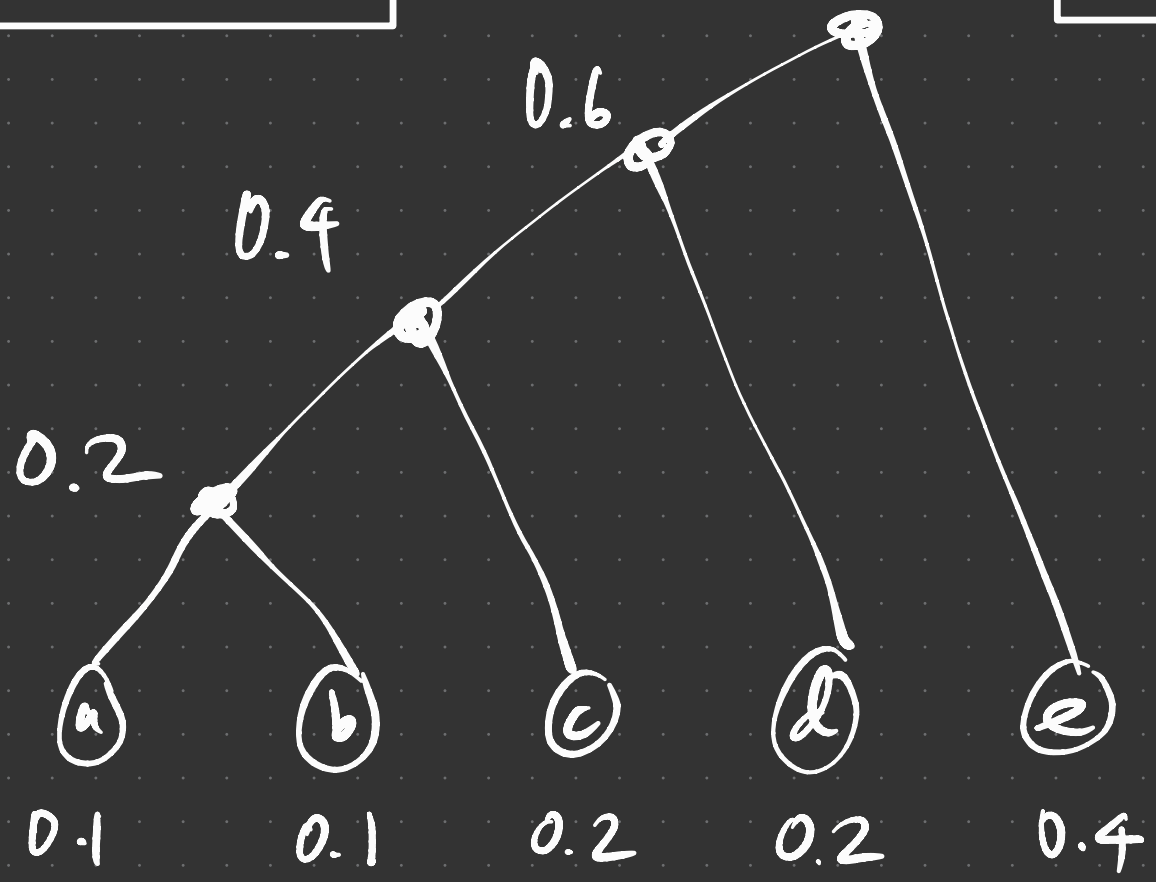
HUFFMAN

D = 2

a	b	c	d	e
0.18	0.19	0.2	0.21	0.22

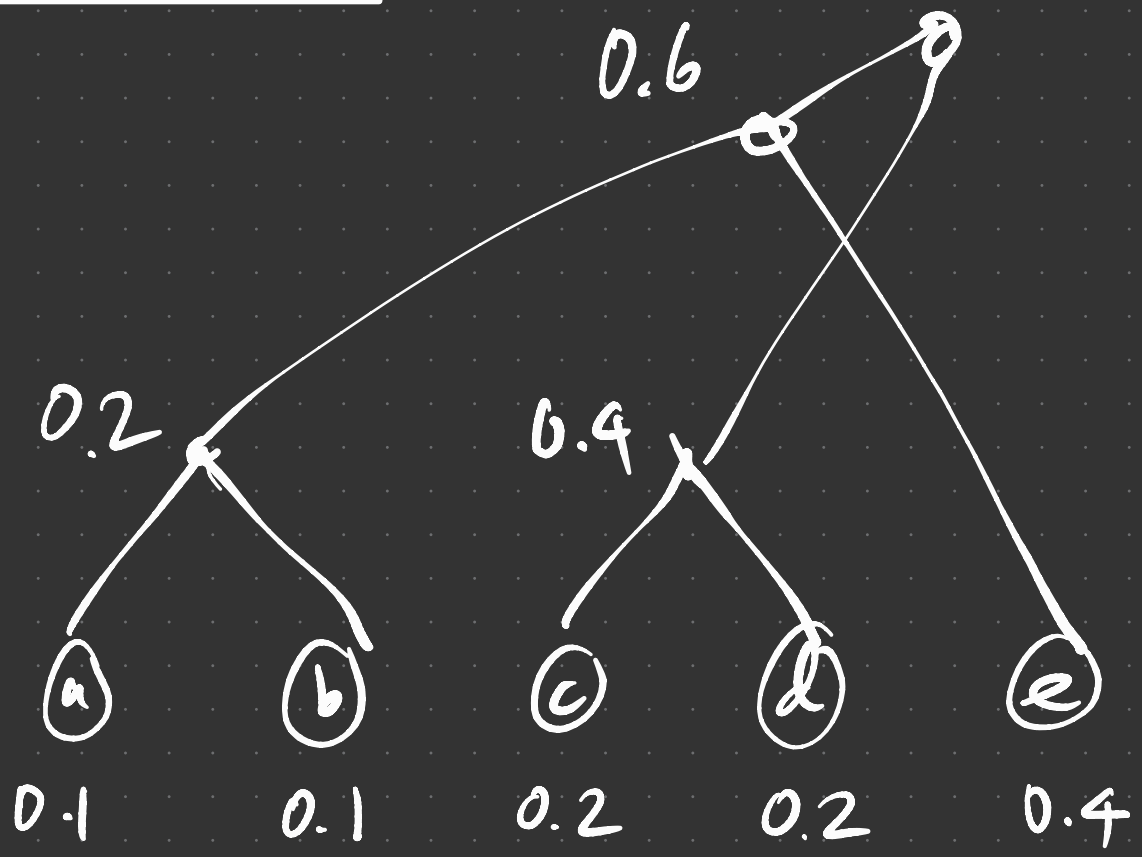
HUFFMAN

D = 2



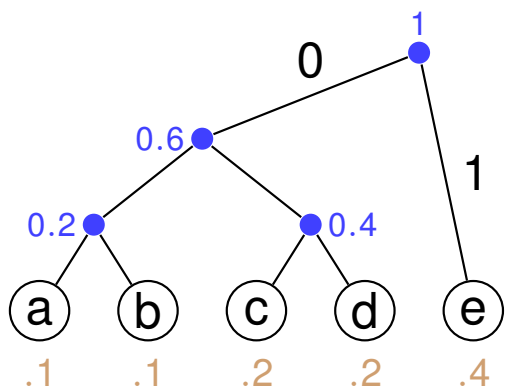
HUFFMAN

D = 2



TREE WITH PROBABILITIES: A HANDY TOOL

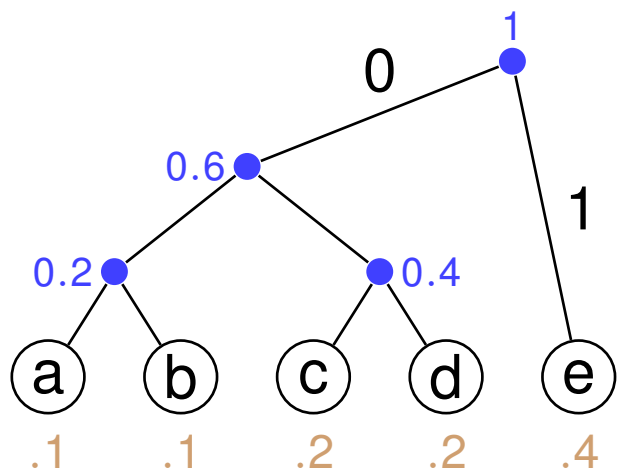
- Consider a tree with probabilities assigned to leaf nodes, like the decoding tree of a prefix-free code
- The probabilities of the leaf nodes induce probabilities to the intermediate notes (like in Huffman's construction).
- The result is called a **tree with probabilities**.

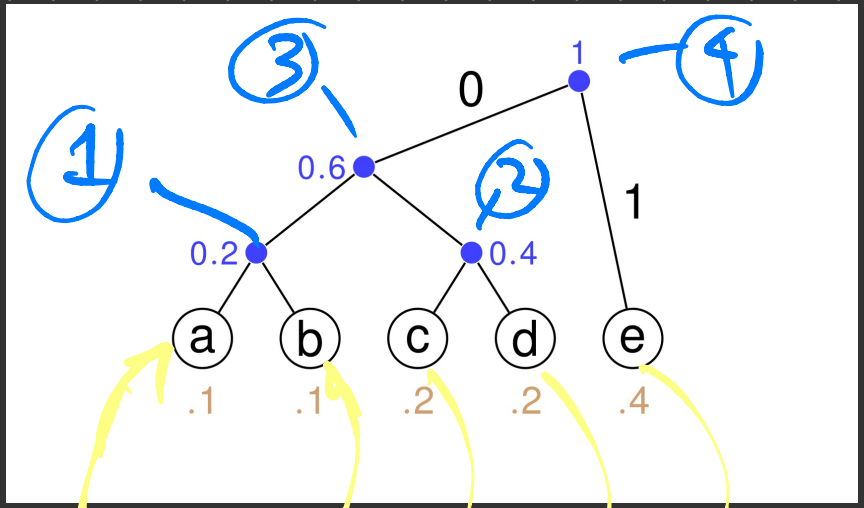


Lemma (Path-Length Lemma)

The average path length of a tree with probabilities is the sum of the probabilities of the intermediate nodes (root included):

$$\sum_i p_i l_i = \sum_j q_j$$





1 2 3 4 5

$\Pi_{3,3} = 1$

$\Pi_{1,5} = 0.$

$\Pi_{j,i}$

leaf vertex

intermediate vertex

Proof: Define the indicator function

$$\mathbb{I}_{j,i} = \begin{cases} 1, & \text{if node } j \text{ is on the path to leaf } i \\ 0, & \text{otherwise.} \end{cases}$$

Notice that

$$q_j = \sum_i p_i \mathbb{I}_{j,i}.$$

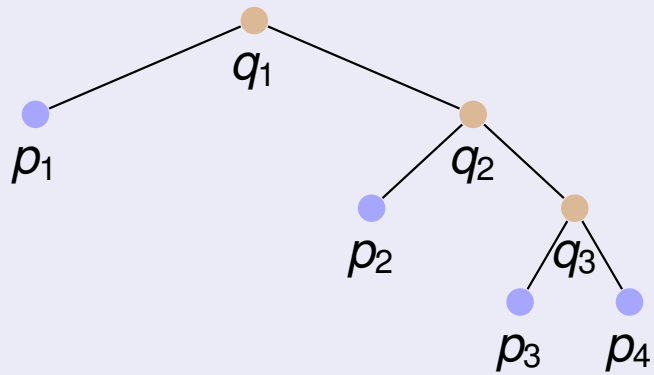
Hence

$$\begin{aligned} \sum_j q_j &= \sum_j \sum_i p_i \mathbb{I}_{j,i} \\ &= \sum_i p_i \sum_j \mathbb{I}_{j,i} \\ &= \sum_i p_i l_i. \end{aligned}$$

□

Example

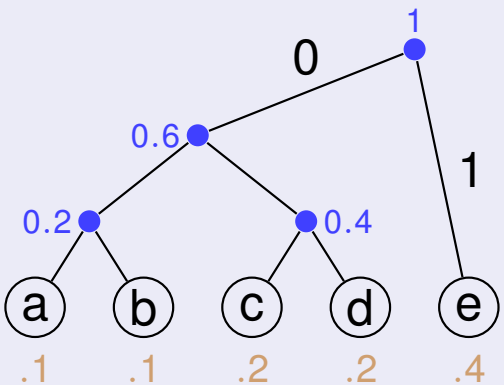
The following example mirrors the proof.



$$
\begin{array}{rcccccccc}
q_1 & = & p_1 & + & p_2 & + & p_3 & + & p_4 \\
q_2 & = & & + & p_2 & + & p_3 & + & p_4 \\
q_3 & = & & & & + & p_3 & + & p_4 \\
\hline
q_1 + q_2 + q_3 & = & p_1 \times 1 & + & p_2 \times 2 & + & p_3 \times 3 & + & p_4 \times 3 \\
 & = & p_1 \times l_1 & + & p_2 \times l_2 & + & p_3 \times l_3 & + & p_4 \times l_4
\end{array}
$$

The average codeword-length of a prefix-free code can be computed efficiently using the Path-Length Lemma.

Example



$$L(S, \Gamma) = 0.2 + 0.4 + 0.6 + 1 = 2.2.$$

Theorem (Huffman's construction is optimal)

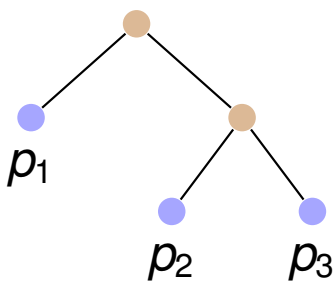
If Γ_H is a Huffman code (prefix-free by construction) and Γ is another uniquely decodable code for the same source S, then it is guaranteed that

$$L(S, \Gamma_H) \leq L(S, \Gamma).$$

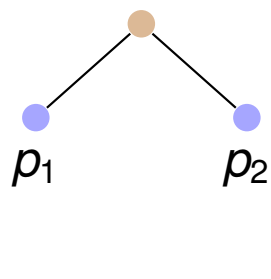
Proof: For simplicity, we consider only binary codes. Let $L = \max_i l_i$. The proof is based on the following three facts.

Fact 1: In the decoding tree of an optimal binary code, each intermediate node has exactly two offsprings.

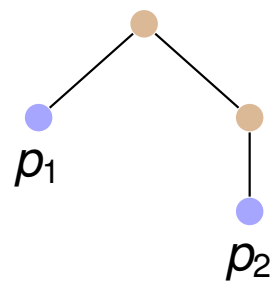
Examples:



This is OK



This is OK

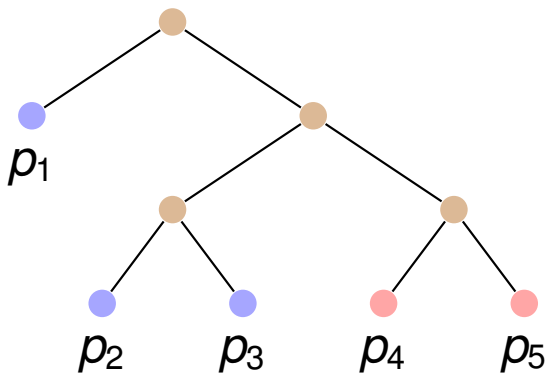


This is NOT OK

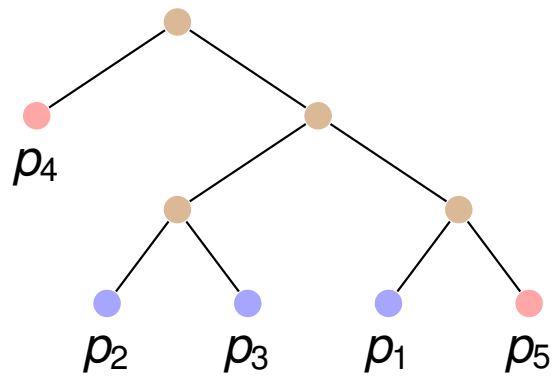
In particular, any leaf at depth L has a sibling.

Fact 2: An optimal encoder assigns shorter codewords to higher-probability letters.

Examples: Suppose that $p_5 \leq p_4 \leq p_3 \leq p_2 \leq p_1$.



This is OK

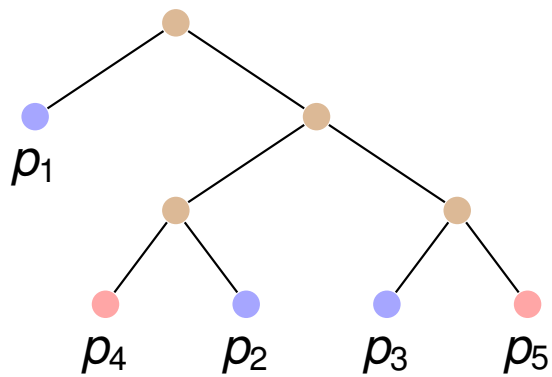


This is NOT OK

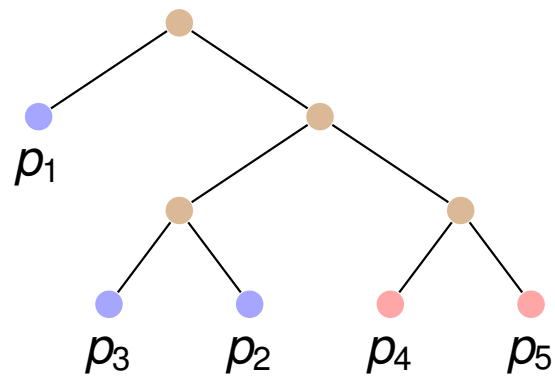
This and Fact 1 imply that two of the least likely codewords have length L.

Fact 3: Based on Fact 2, without loss of optimality, we may require that two of the least-likely leaves be siblings at depth L.

Example: Suppose that $p_5 \leq p_4 \leq p_3 \leq p_2 \leq p_1$.



If this is optimal . . .

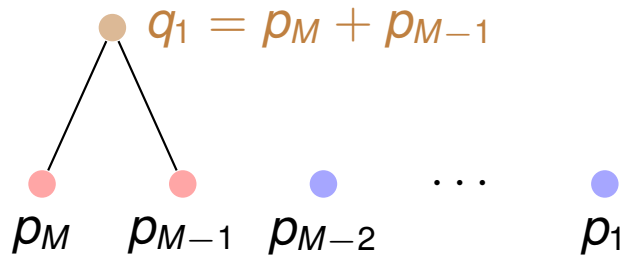


. . . then this is also optimal

Code Construction:

We seek an optimal (minimum average-length) code for the given probabilities (in increasing order from left to right).

We do the first step as in the figure.



Suppose that we construct a code $\tilde{\Gamma}$ for $q_1, p_{M-2}, \ldots, p_1$.

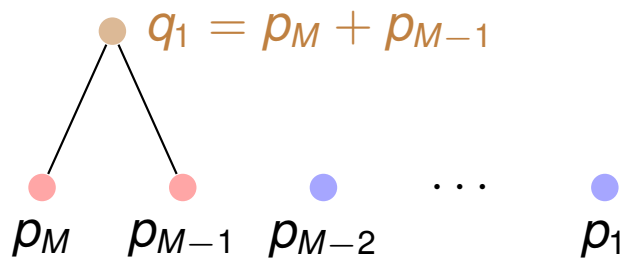
In $\tilde{\Gamma}$, q_1 is a leaf node.

By extending q_1 as in the above figure, the code $\tilde{\Gamma}$ becomes a code Γ for $p_M, p_{M-1}, p_{M-2}, \ldots, p_1$.

Fact 3 above guarantees the existence of a code $\tilde{\Gamma}$ such that Γ is optimal. The question is how to find $\tilde{\Gamma}$.

Let L and $\tilde{L}$ be the average length of Γ and $\tilde{\Gamma}$, respectively.

Except for q_1, codes Γ and $\tilde{\Gamma}$ have the same intermediate nodes.



By the path-length lemma, $L = \tilde{L} + q_1$.

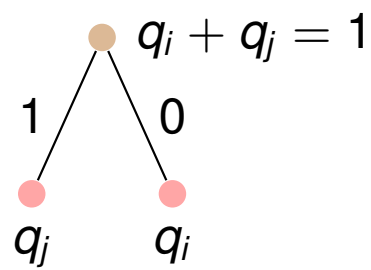
Hence, L is as small as possible iff $\tilde{L}$ is as small as possible.

We are done if we find an optimal code $\tilde{\Gamma}$ for $q_1, p_{M-2}, \ldots, p_1$.

This is progress: we have reduced the size of the alphabet from M to $M - 1$.

Continuing the same way, after $M - 2$ steps we are left with the problem of constructing an optimal code for an alphabet of two letters.

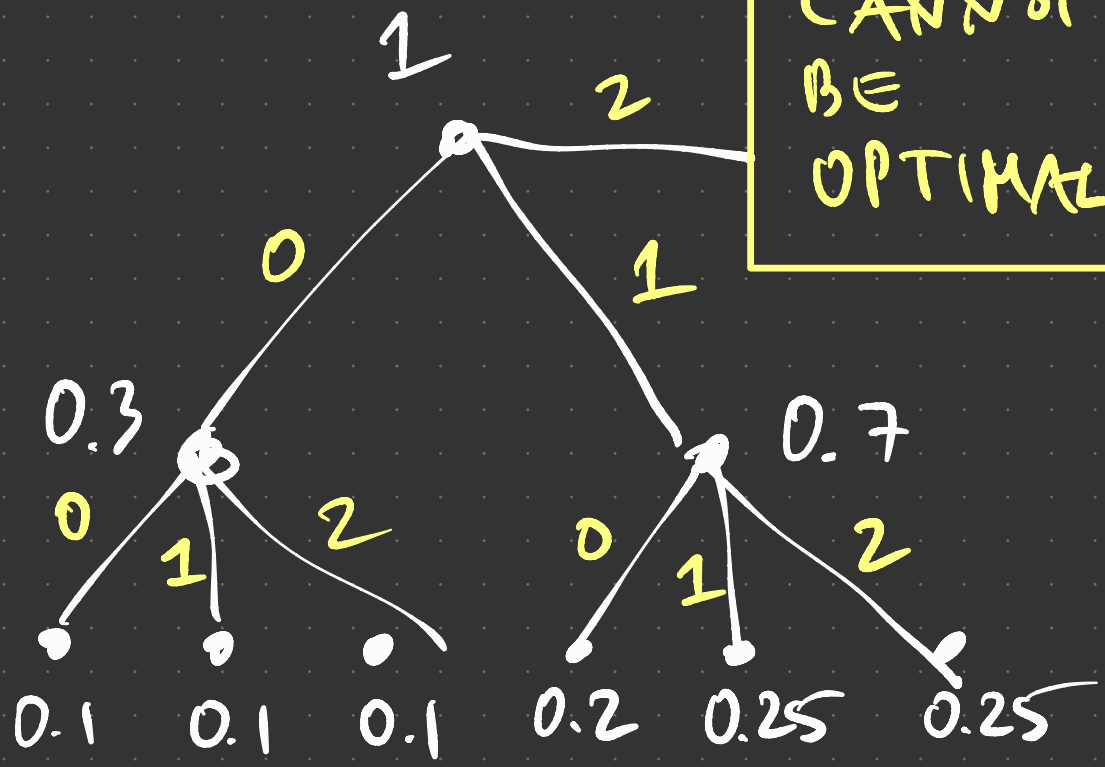
An optimal code is to assign codeword 0 to one letter, and codeword 1 to the other letter.



We have described Huffman's code construction. No binary code Γ has a smaller average codeword-length. □

HUFFMAN: BEYOND BINARY

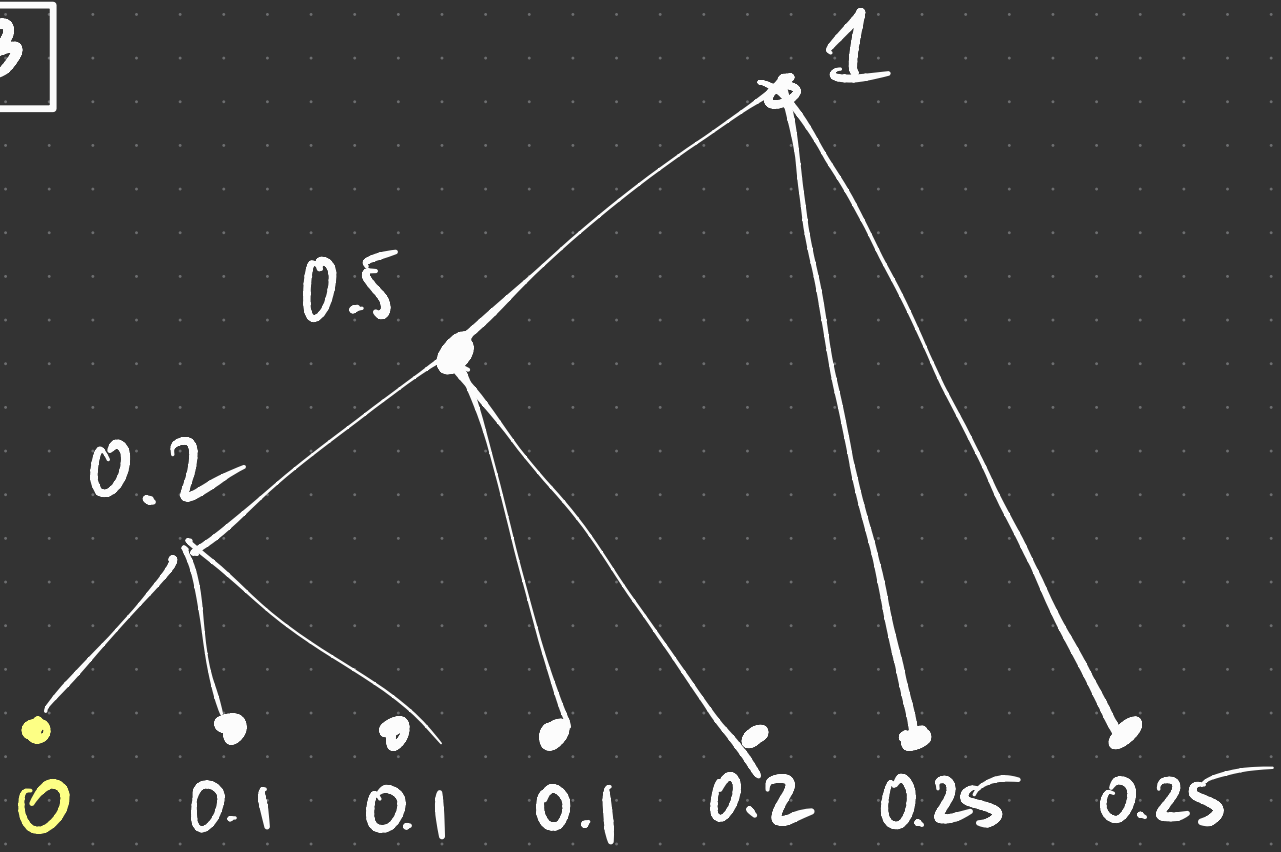
D=3



CANNOT BE OPTIMAL

HUFFMAN: BEYOND BINARY

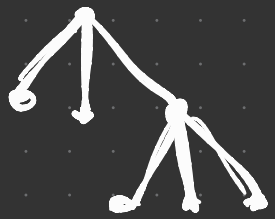
D=3



HUFFMAN: BEYOND BINARY

D=3

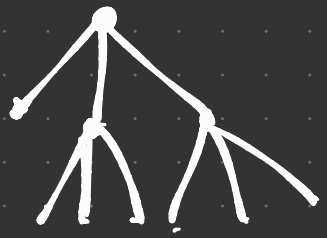
OBS: A FULL CODE TREE HAS HOW MANY LEAVES ?



3

5

7



HUFFMAN: BEYOND BINARY

GENERAL D

OBS: A FULL CODE TREE HAS HOW MANY LEAVES ?

↳ $D + k(D-1)$

IDEA: ADD PROBABILITY-ZERO SYMBOLS TO MATCH !

HUFFMAN: BEYOND BINARY

D=3

Main Result

The **expected codeword length** of any useful source code satisfies the following bounds:

Theorem (Textbook Thm 3.3)

The average codeword-length of a uniquely decodable code Γ for S must satisfy

$$H_D(S) \leq L(S, \Gamma)$$

and there exists a uniquely decodable code Γ_{SF} satisfying

$$L(S, \Gamma_{SF}) < H_D(S) + 1.$$

Key Idea

- **Pack multiple symbols into "supersymbols"!**
- $(S_1, S_2, S_3, \ldots, S_n)$
- Now, apply our Main Result to such supersymbols:

Theorem (Textbook Thm 3.3)

The average codeword-length of a uniquely decodable code Γ for S must satisfy

$$H_D(S_1, S_2, \ldots, S_n) \leq L((S_1, S_2, \ldots, S_n), \Gamma)$$

and there exists a uniquely decodable code Γ_{SF} satisfying

$$L((S_1, S_2, \ldots, S_n), \Gamma_{SF}) < H_D(S_1, S_2, \ldots, S_n) + 1.$$

- Why is this clever?
- Let us study the entropy of the supersymbol $H_D(S_1, S_2, \ldots, S_n)$ next.

Recall: Joint Entropy

Recall from the first week: The formula for the entropy of a random variable S extends to any number of random variables. If X and Y are two discrete random variables, with (joint) probability distribution $p_{X,Y}$ then

$$H_D(X, Y) = \mathbb{E}[-\log_D p_{X,Y}(X, Y)],$$

which means

$$H_D(X, Y) = -\sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} p_{X,Y}(x, y) \log_D p_{X,Y}(x, y).$$

Recall: Joint Entropy

Now suppose that X and Y are **independent.**

This means that $p_{X,Y}(x, y) = p_X(x)p_Y(y)$.

$$
\begin{aligned}
H_D(X, Y) &= -\sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} p_X(x)p_Y(y)\log_D p_X(x)p_Y(y) \\
&= -\sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} p_X(x)p_Y(y)\log_D p_X(x) - \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} p_X(x)p_Y(y)\log_D p_Y(y) \\
&= -\sum_{x\in\mathcal{X}} p_X(x)\log_D p_X(x) - \sum_{y\in\mathcal{Y}} p_Y(y)\log_D p_Y(y) \\
&= H_D(X) + H_D(Y).
\end{aligned}
$$

- **Pack multiple symbols into "supersymbols"!**
- Consider the case where S_i are all *independent* and follow the same distribution.
- Then, $H_D(S_1, S_2, S_3, \ldots, S_n) = nH_D(S)$.

THEOREM (TEXTBOOK THM 3.3)

Suppose that $S_1, S_2, \ldots, S_n$ are independent and follow the same distribution. The average codeword-length of a uniquely decodable code Γ for $(S_1, S_2, \ldots, S_n)$ must satisfy

$$nH_D(S) \leq L((S_1, S_2, \ldots, S_n), \Gamma)$$

and there exists a uniquely decodable code Γ_{SF} satisfying

$$L((S_1, S_2, \ldots, S_n), \Gamma_{SF}) < nH_D(S) + 1.$$

Main Result

- **Pack multiple symbols into "supersymbols"!**
- Consider the case where S_i are all *independent* and follow the same distribution.
- Then, $H_D(S_1, S_2, S_3, \ldots, S_n) = nH_D(S)$.

Theorem (Textbook Thm 3.3)

Suppose that $S_1, S_2, \ldots, S_n$ are independent and follow the same distribution. The average codeword-length of a uniquely decodable code Γ for $(S_1, S_2, \ldots, S_n)$ must satisfy

$$H_D(S) \leq \frac{L((S_1, S_2, \ldots, S_n), \Gamma)}{n}$$

and there exists a uniquely decodable code Γ_{SF} satisfying

$$\frac{L((S_1, S_2, \ldots, S_n), \Gamma_{SF})}{n} < H_D(S) + \frac{1}{n}.$$

GENERAL KRAFT INEQUALITY

THM A UNIQUELY DECODABLE CODE MUST SATISFY

$$\sum_{s} D^{-\ell(\Gamma(s))} \leq 1.$$

$\mathcal{A}$	codewords	length
a	$\Gamma(a)$	$\ell(a)$
b	$\Gamma(b)$	$\ell(b)$
c	$\Gamma(c)$	$\ell(c)$
d	$\Gamma(d)$	$\ell(d)$

$$L = \max_{x \in \mathcal{A}} \ell(x)$$

$\mathcal{A}$	codewords	length
a	$\Gamma(a)$	$\ell(a)$
b	$\Gamma(b)$	$\ell(b)$
c	$\Gamma(c)$	$\ell(c)$
d	$\Gamma(d)$	$\ell(d)$

$$L = \max_{x \in \mathcal{A}} \ell(x)$$

$\mathcal{A} \times \mathcal{A}$	codewords	length
aa	$\Gamma(a)\,\Gamma(a)$	$\ell(a)+\ell(a)$
ab	$\Gamma(a)\,\Gamma(b)$	$\ell(a)+\ell(b)$
$\vdots$	$\vdots$	$\vdots$
dd	$\Gamma(d)\,\Gamma(d)$	$\ell(d)+\ell(d)$

- max length $2L$
- uniquely decodable!

KRAFT SUM

$$2^{-(\ell(a)+\ell(a))} + 2^{-(\ell(a)+\ell(b))} + \dots$$

$$= \left(2^{-\ell(a)} + 2^{-\ell(b)} + 2^{-\ell(c)} + 2^{-\ell(d)} \right)^2$$

$$= \sum_{m=1}^{2L} N(m)\, 2^{-m}$$

CLAIM : $N(m) \le 2^m$

$\leq 2L$

NOW, DO THE EXTENSION TO k:

$$\left(2^{-\ell(a)} + 2^{-\ell(b)} + 2^{-\ell(c)} + 2^{-\ell(d)}\right)^{k}$$

$\leq k L$

HENCE, FOR EVERY $k \in \mathbb{Z}_+$:

$$2^{-\ell(a)} + 2^{-\ell(b)} + 2^{-\ell(c)} + 2^{-\ell(d)} \leq \sqrt[k]{kL}$$

as $k \to \infty$, this goes to 1.