Advanced Information, Computation, Communication II

Prof. M. Gastpar

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

EPFL

Spring Semester 2025— *Slides Version 1.0*

Outline

AICC-I

- **Computation**
- Algorithms
- Discrete Structures

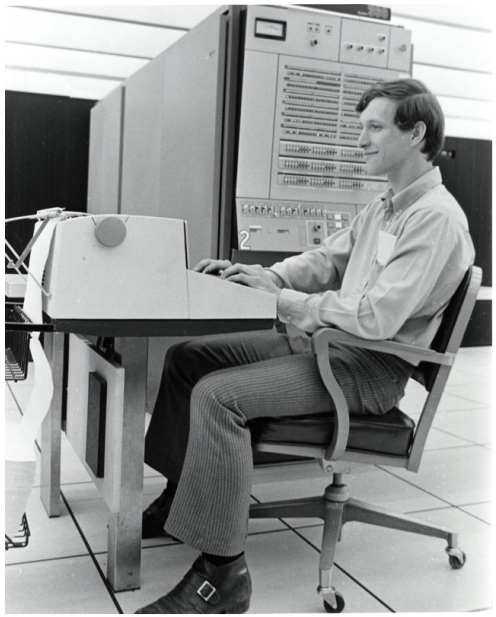
AICC-I

- **Computation**
- Algorithms
- Discrete Structures

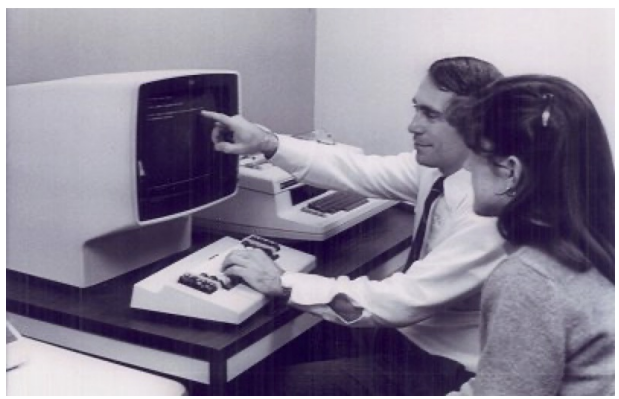
But to have interesting computations, we need data!

AICC-I

- **Computation**
- Algorithms
- Discrete Structures

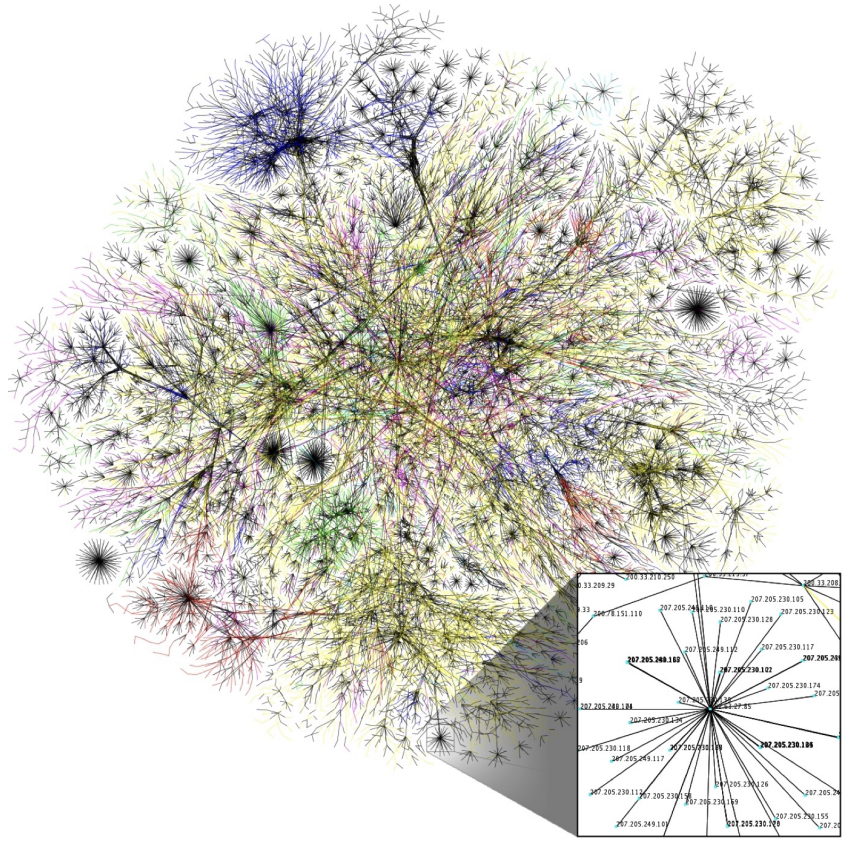


ca. 1980



AICC-I

- **Computation**
- Algorithms
- Discrete Structures



From The Opte Project

AICC-I

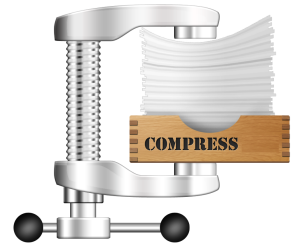
- **Computation**
- Algorithms
- Discrete Structures

AICC-II

- **Communication**
- Information and Data Science
- Cryptography, Secrecy, Privacy

In this course: three main topics

- **Source Coding:** It is about **compressing information.**



- **Cryptography:** It is about **protecting** the information against undesirable **human** activities: how to provide message **integrity** and **confidentiality**.

- **Channel Coding:** It is about **protecting** the information from **natural** damages.

All three pertain to information **storage/communication**.

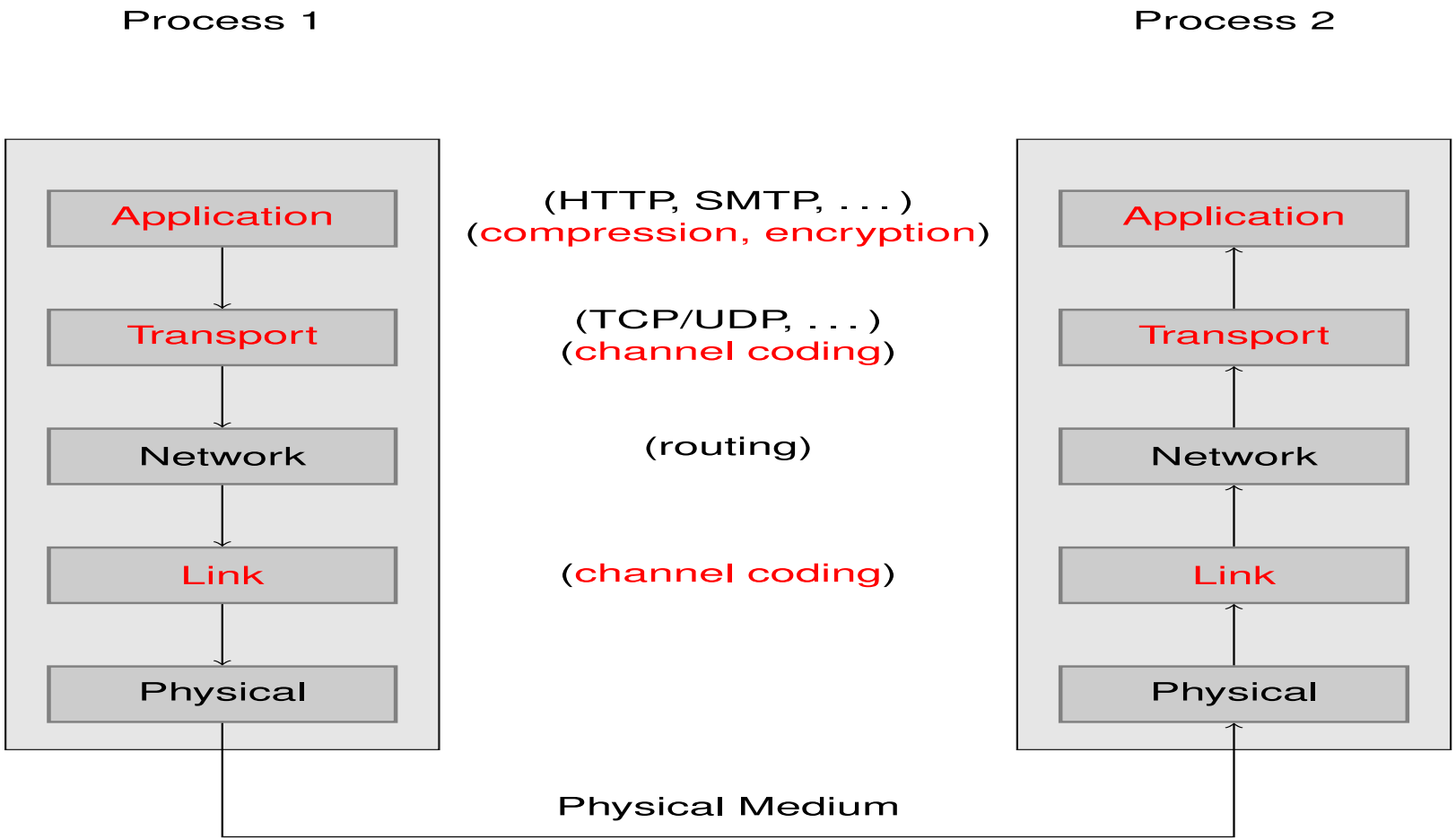
WE STUDY: SOURCE CODING, CRYPTOGRAPHY, CHANNEL CODING

Why these topics?

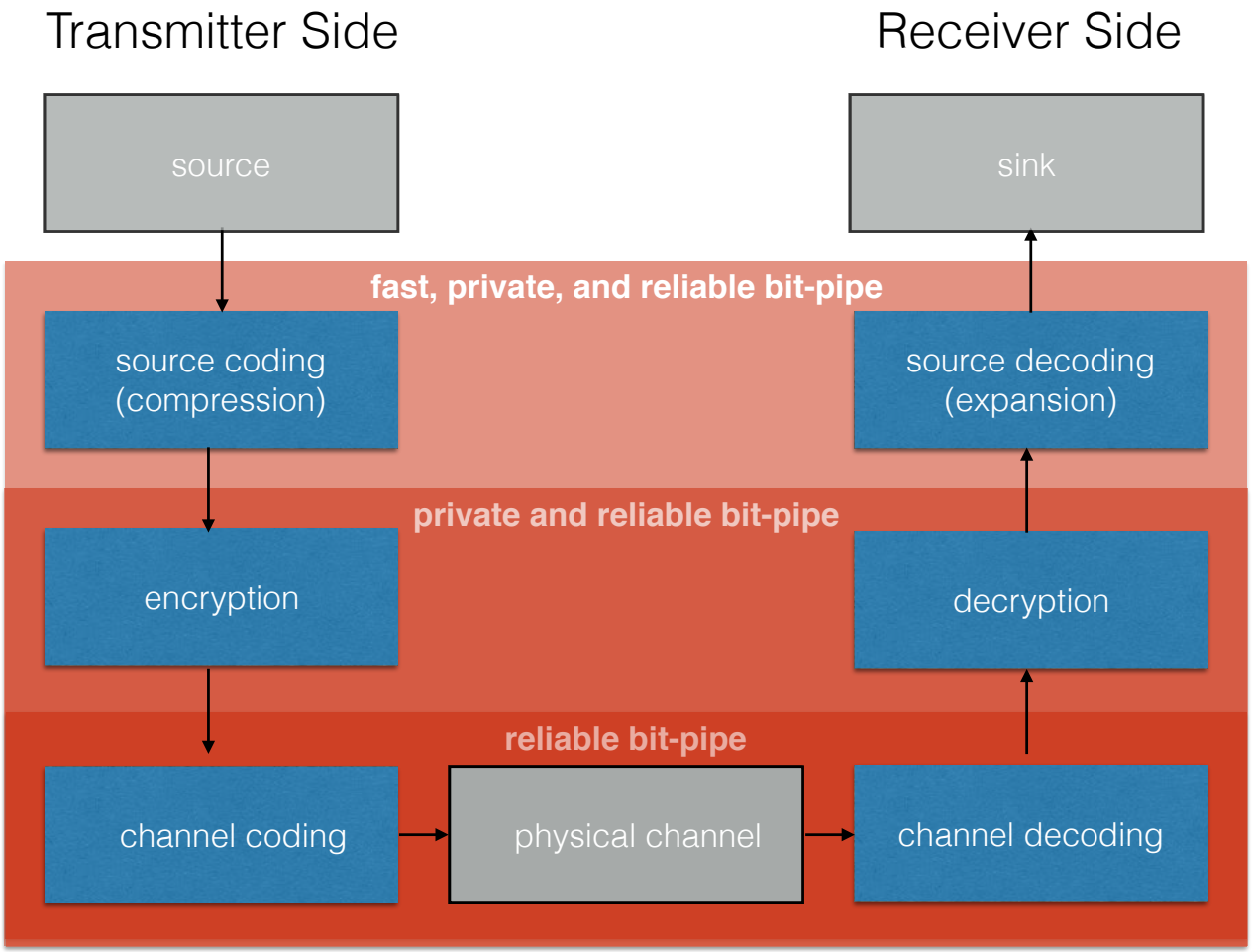
- important building blocks of communication systems
- non-evident topics and the results are often surprising
- intimately related to fundamental concepts (probability theory, linear algebra, number theory)
- have a common root: the notion of entropy
- require/promote rigorous thinking

Digital Communication: The "Big Picture"

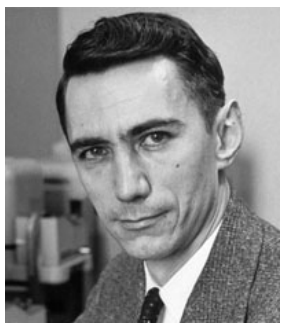
COMMUNICATION OVER THE INTERNET



POINT-TO-POINT COMMUNICATION SYSTEM

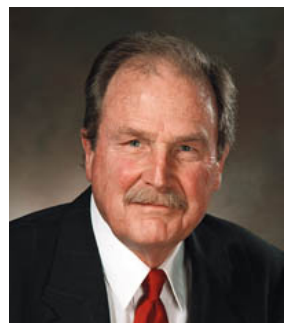


We will rely on **discrete probability theory** and on the work of various people including:



Shannon

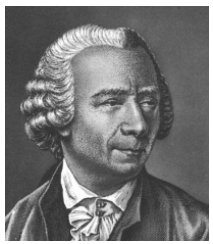
Fano



Huffman

Second Topic: Cryptography

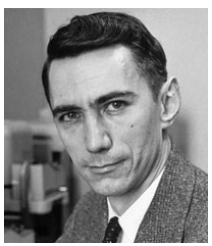
We will rely on **number theory**



Euler

Fermat

as well as on **group theory** and on the work of various people including:



Shannon

Clifford Cocks

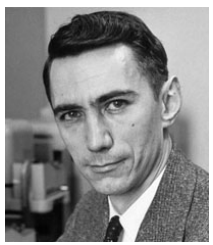
Rivest, Shamir, Adleman

THIRD TOPIC: CHANNEL CODING

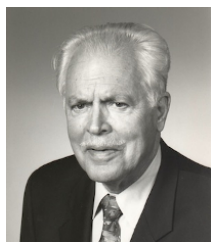
We will rely on **finite fields**

Galois

as well as on **linear algebra** and on the work of various people including:



Shannon



Reed

Solomon

Course Organization

Outline

Teaching Crew

- Professor:
  Michael Gastpar
- Senior Teaching Assistants:
  Adrien Vandenbroucque, Millen Kanabar, Yunzhen Yao
- Student TAs:

Roxanne Chevalley	Ait Lalim Adrien	Mehdi Zoghlami
Michaël Brasey	Yuki Crivelli	Valerio de Santis
Théo Hollender	Gersende Kerjan	Simon Lefort
Mattia Metzler	Emmanuel Omont	Laura Paraboschi
	Anthony Tamberg	

SCHEDULE

- Tuesdays 15:15 - 17:00
  Lecture
  RLC E1 240

- Wednesdays 13:15 - 15:00
  Lecture
  RLC E1 240

- Wednesdays 15:15 - 17:00
  Exercises
  Various rooms, see Moodle

Grading Formula

- **90%** Final exam during exam period.

  Note: No documents or electronic devices allowed during the exam.
- **10%** Quizzes (on-line on Moodle).
  - There will be 6 Quizzes. Only the best 5 count.
  - The Quiz questions are very similar to the final exam questions in style and difficulty.
  - On the Quizzes, you can update your answer as many times as you want before the deadline.
  - However, once the deadline is passed, you can no longer change your answers.

- There is also a weekly homework set:
  - The Quizzes are highly correlated with the homework.
  - If you did not do the homework, you **should not expect to be able to do the Quizzes!**
  - We do not grade the homework.

How To Be Efficient And Do Well In This Course

Before class (stay ahead):

- browse through the slides to know what to expect
- review the background material as needed

After class:

- read the notes: they are the reference
- do the review questions

Before the exercise session:

- are you up-to-date with the theory?
- solve what you can ahead of time and finish during the exercise session
- write down YOUR own solution

Course Web Site

- moodle.epfl.ch > Informatique (IN) > Bachelor > COM-102 Advanced information, computation, communication II
  (Password protected if not registered to AICC-II)

- There you'll find:
  - Lecture slides
  - Link to videos
  - Homework assignments
  - Solutions
  - Quizzes
  - Forums (news and questions/answers)

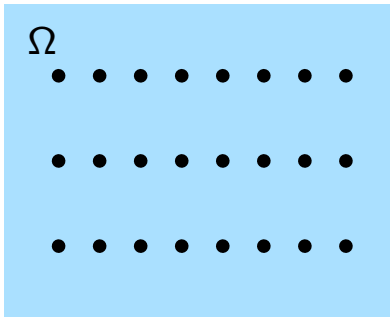
Outline

Review of Discrete Probability:
(Book Chapter 0)

Outline

Initial case: Finite Ω with Equally Likely Outcomes

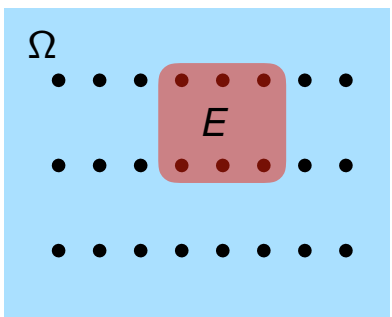
Sample space Ω: set of all possible outcomes.



$$\Omega = \{\omega_1, \ldots, \omega_n\}$$

Event E: subset of Ω. Since the outcomes are equally likely,

$$p(E) = \frac{|E|}{|\Omega|}.$$

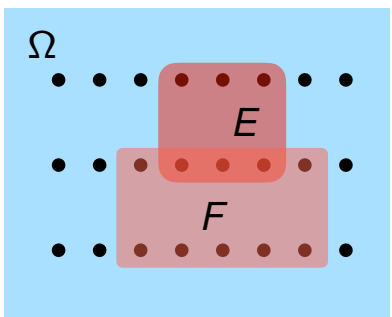


$$p(E) = \tfrac{6}{24}$$

Conditional Probability

The **conditional probability** $p(E|F)$ is the probability that E occurs, given that F has occurred (hence assuming that $|F| \neq \emptyset$):

$$p(E|F) = \frac{|E \cap F|}{|F|}.$$

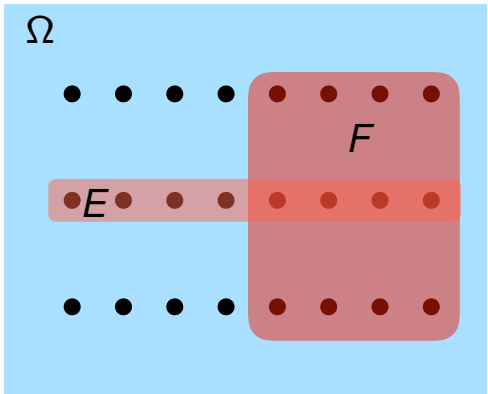


$$p(E|F) = \tfrac{3}{10}$$

We may think of F as a new sample space.

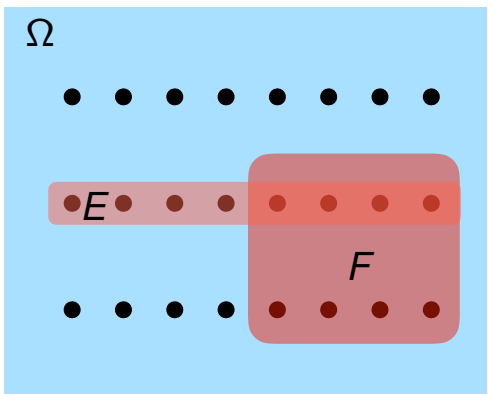
Independent Events

Events E and F are called **independent** if $p(E|F) = p(E)$.



$$p(E|F) = \frac{1}{3} = p(E)$$

E and F are independent

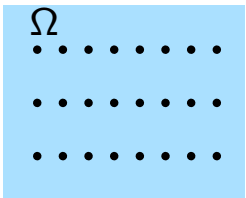


$$p(E|F) = \frac{1}{2} \neq p(E) = \frac{1}{3}$$

E and F are NOT independent

General Case: Finite Ω, arbitrary $p(\omega)$

Sample space Ω: set of all possible outcomes.

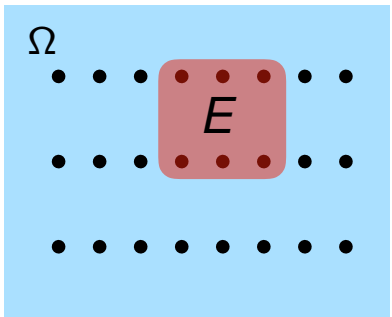


$$\Omega = \{\omega_1, \ldots, \omega_n\}$$

Probability distribution (probability mass function) p:
A function $p : \Omega \to [0, 1]$ such that

$$\sum_{\omega \in \Omega} p(\omega) = 1.$$

Event E: a subset of Ω.



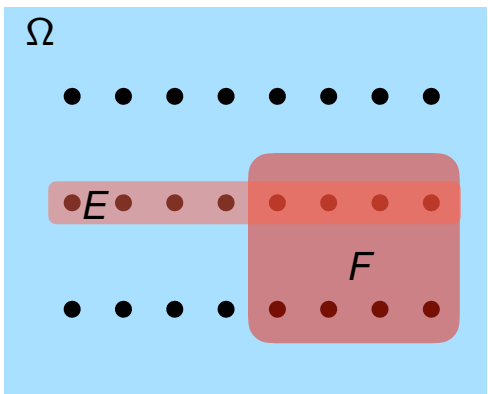
The domain of the probability mass function p is extended to the power set of Ω:

$$p(E) = \sum_{\omega \in E} p(\omega).$$

The general form for the conditional probability is

$$p(E|F) = \frac{p(E \cap F)}{p(F)}$$

for F such that $p(F) \neq 0$.

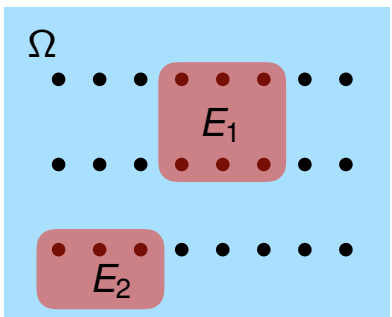


Independent Events. Exactly as before, events E and F are called independent if $p(E|F) = p(E)$. Equivalently, E and F are independent if $p(E \cap F) = p(E)p(F)$.

Disjoint Events:

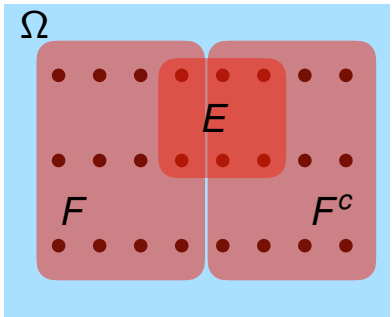
If E_1 and E_2 are disjoint events then

$$p(E_1 \cup E_2) = p(E_1) + p(E_2).$$



Law of Total Probability:

For any $F \subseteq \Omega$ and its complement F^c,



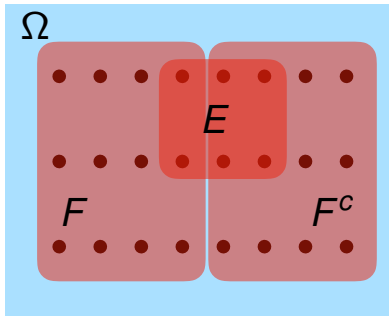
$$p(E) = p(E|F)p(F) + p(E|F^c)p(F^c).$$

More generally, if Ω is the union of disjoint events $F_1, F_2, \ldots, F_n$,

$$p(E) = p(E|F_1)p(F_1) + p(E|F_2)p(F_2) + \cdots + p(E|F_n)p(F_n).$$

(Divide and conquer.)

Proof: We prove the law of total probability for $\Omega = F \cup F^c$. (The general case follows straightforwardly.)



$$p(E) = p((\underbrace{E \cap F) \cup (E \cap F^c)}_{\text{union of disjoint sets}})$$

$$= p(E \cap F) + p(E \cap F^c)$$

$$= \frac{p(E \cap F)}{p(F)} p(F) + \frac{p(E \cap F^c)}{p(F^c)} p(F^c)$$

$$= p(E|F)p(F) + p(E|F^c)p(F^c).$$

□

EXERCISE

Example of Total Probability: Two factories supply light bulbs.

- Factory F_1's bulbs work for over 5000 hours in 99% of cases;
- Factory F_2's bulbs work for over 5000 hours in 95% of cases.
- It is known that factory F_1 supplies 60% of the total bulbs.

What is the chance that a bulb chosen at random works for longer than 5000 hours?

Solution

Answer:

Ω is the space of all bulbs.

(Optional: to picture the partitioning of Ω into subsets, you may want to imagine each bulb being labeled by the factory's name and the number of hours that it works.)

Let $E \subseteq \Omega$ be the set that consists of all bulbs that work for longer than 5000 hours and let $F_i \subseteq \Omega$ be the set of bulbs from factory $i = 1, 2$.

- $p(E|F_1) = .99$
- $p(E|F_2) = .95$
- $p(F_1) = .6$

$$p(E) = p(E|F_1)p(F_1) + p(E|F_2)p(F_2) = \frac{99}{100} \times \frac{6}{10} + \frac{95}{100} \times \frac{4}{10} = \frac{974}{1000}.$$

Sometimes we are given $p(E)$, $p(F)$ and $p(E|F)$, and we need $p(F|E)$.

In this case we use **Bayes' Rule:**

$$p(F|E) = \frac{p(E|F)p(F)}{p(E)}.$$

Proof: We use the definition of conditional probability to write $p(E \cap F)$ two ways and solve for $p(F|E)$:

$$p(F|E)p(E) = p(E \cap F) = p(E|F)p(F).$$

□

Example:

Let Ω be a population of drivers (e.g. of Switzerland, on New Year's eve).

Let A be the event that a driver has an accident.

Let D be the event that a driver is drunk.

From observations, the police knows $p(A)$, $p(D)$ as well as $p(D|A)$.

$p(A|D)$ cannot be easily obtained from observations. Yet, knowing it might discourage a drunk person to drive.

Let us be concrete (numbers are made up):

$$p(A) = 10^{-6},$$
$$p(D) = 0.1,$$
$$p(D|A) = 0.8.$$

Now

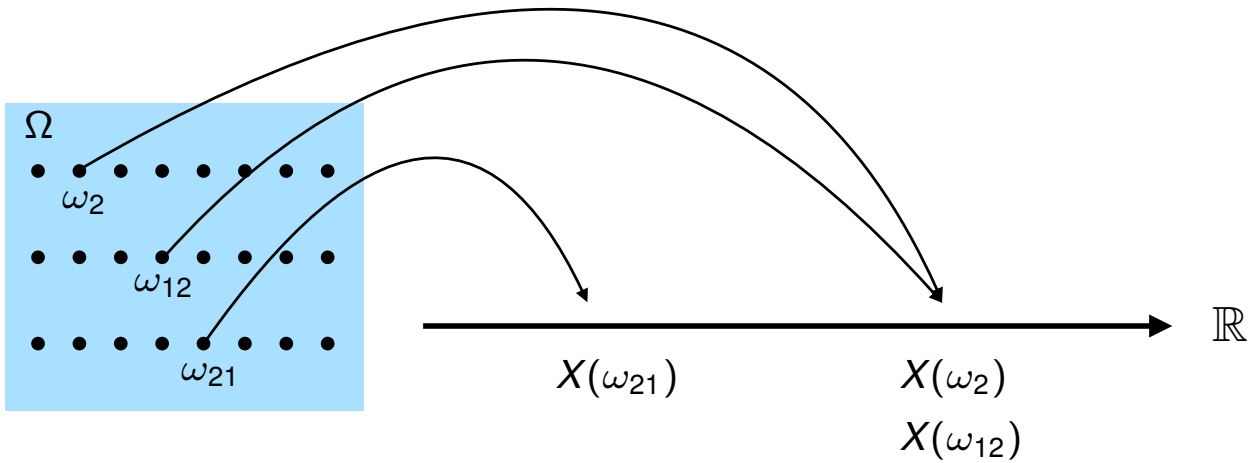
$$p(A|D) = \frac{p(D|A)p(A)}{p(D)} = \frac{0.8 \times 10^{-6}}{0.1} = 8 \times 10^{-6}.$$

We can also compute

$$p(A|D^c) = \frac{p(D^c|A)p(A)}{p(D^c)} = \frac{(1 - 0.8) \times 10^{-6}}{(1 - 0.1)} = \frac{2}{9} \times 10^{-6}.$$

Notice that, in this case, $\frac{p(A|D)}{p(A|D^c)} = 36$.

Random Variable X: A function $X : \Omega \to \mathbb{R}$.

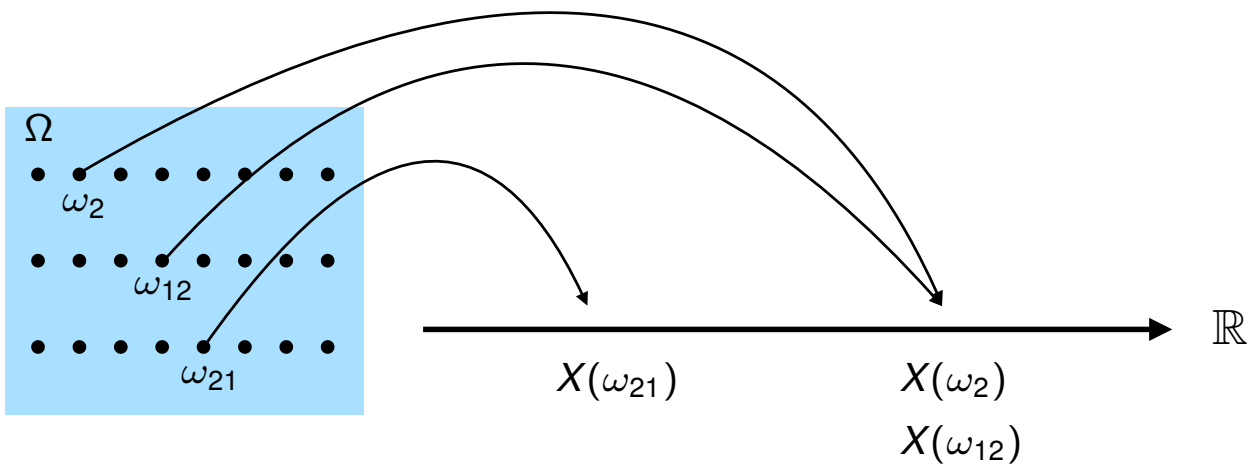


Probability distribution p_X: $p_X(x)$ is the probability that $X = x$, i.e. the probability of the event

$$E = \{\omega \in \Omega : X(\omega) = x\}.$$

Hence,

$$p_X(x) = \sum_{\omega \in E} p(\omega).$$



Example (Lucky Dice)

You roll a dice.

If the outcome is 6, you receive 10 CHF. Otherwise, you pay 1 CHF.

$$\Omega = \{1, 2, 3, 4, 5, 6\}$$

$$\text{For each } \omega, p(\omega) = 1/6.$$

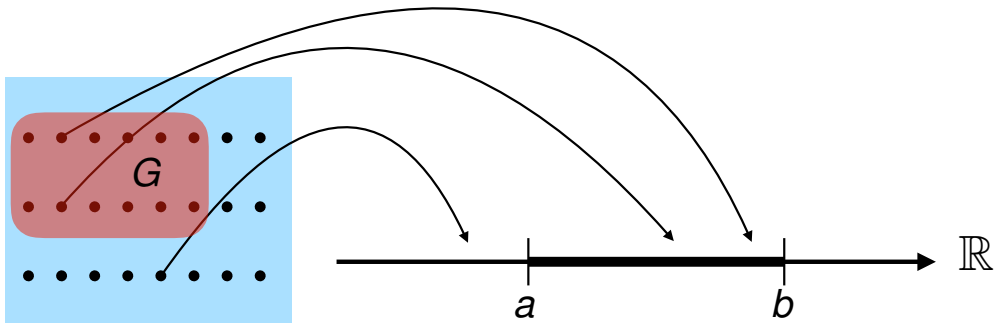
Then, define:

$$X(\omega) = \begin{cases} 10, & \omega = 6 \\ -1, & \omega \in \{1, 2, 3, 4, 5\}. \end{cases}$$

Hence, we have

$$p_X(x) = \begin{cases} \frac{1}{6}, & x = 10 \\ \frac{5}{6}, & x = -1. \end{cases}$$

How about the probability that $X \in [a, b]$?

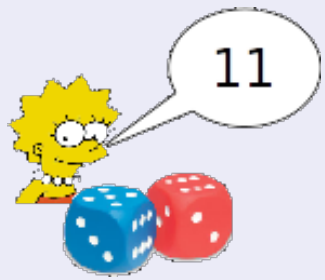


We can compute it two ways:

- using p: $\sum_{\omega \in G} p(\omega)$.
- using p_X: $\sum_{x \in [a,b]} p_X(x)$.

Exercise (Lisa rolls two dice)

- Lisa rolls two dice and announces the sum L written as a two digit number.
- The alphabet of $L = L_1 L_2$ is $\{02, 03, 04, 05, 06, 07, 08, 09, 10, 11, 12\}$.
  - The alphabet of L_1 is $\{0, 1\}$.
  - The alphabet of L_2 is $\{0, 1, \ldots, 9\}$.
- Determine p_L.



SOLUTION

$\Omega = \Big\{(i,j) : i,j \in \{1,\ldots,6\}\Big\}$.

$L : \Omega \to \mathbb{R}$ defined by $L\big((i,j)\big) = i + j$ written as a two-digit number.

$L = 02$ iff $\omega = (1,1)$, which has probability $\frac{1}{36}$.

$L = 03$ iff $\omega \in \{(1,2)\} \cup \{(2,1)\}$. The events $\{(2,1)\}$ and $\{(1,2)\}$ are disjoint, with probability $\frac{1}{36}$ each. Hence $L = 03$ with probability $\frac{2}{36}$.

Etc.

L	02	03	04	05	06	07	08	09	10	11	12
p_L	$\frac{1}{36}$	$\frac{2}{36}$	$\frac{3}{36}$	$\frac{4}{36}$	$\frac{5}{36}$	$\frac{6}{36}$	$\frac{5}{36}$	$\frac{4}{36}$	$\frac{3}{36}$	$\frac{2}{36}$	$\frac{1}{36}$

Two Random Variables:

Let $X : \Omega \to \mathbb{R}$ and $Y : \Omega \to \mathbb{R}$ be two random variables.

The probability of the event $E_{(x,y)} = \{\omega \in \Omega : X(\omega) = x \text{ and } Y(\omega) = y\}$ is

$$p_{X,Y}(x, y) = \sum_{\omega \in E_{(x,y)}} p(\omega).$$

We can compute p_X from $p_{X,Y}$:

$$p_X(x) = \sum_y p_{X,Y}(x, y).$$

p_X is called **marginal distribution** (of $p_{X,Y}(x, y)$ with respect to x).

p_Y can be computed similarly.

EXERCISE

Determine the probability p_{L_1}, knowing p_L, where $L = L_1 L_2$.

L	02	03	04	05	06	07	08	09	10	11	12
p_L	$\frac{1}{36}$	$\frac{2}{36}$	$\frac{3}{36}$	$\frac{4}{36}$	$\frac{5}{36}$	$\frac{6}{36}$	$\frac{5}{36}$	$\frac{4}{36}$	$\frac{3}{36}$	$\frac{2}{36}$	$\frac{1}{36}$

L_1	0	1
p_{L_1}		

SOLUTION

L	02	03	04	05	06	07	08	09	10	11	12
p_L	$\frac{1}{36}$	$\frac{2}{36}$	$\frac{3}{36}$	$\frac{4}{36}$	$\frac{5}{36}$	$\frac{6}{36}$	$\frac{5}{36}$	$\frac{4}{36}$	$\frac{3}{36}$	$\frac{2}{36}$	$\frac{1}{36}$

We marginalize:

$$p_{L_1}(1) = \sum_x p_{L_1,L_2}(1,x) = \frac{3}{36} + \frac{2}{36} + \frac{1}{36} = \frac{6}{36} = \frac{1}{6}.$$

Hence

L_1	0	1
p_{L_1}	$\frac{5}{6}$	$\frac{1}{6}$

The **Expected Value** $\mathbb{E}[X]$ of a random variable $X : \Omega \to \mathbb{R}$ is

$$\begin{aligned}\mathbb{E}[X] &= \sum_{\omega} X(\omega)p(\omega) \\ &= \sum_{x} xp_X(x).\end{aligned}$$

To see that these two expressions are indeed equal, we reorganize the sum:

$$\sum_{\omega} X(\omega)p(\omega) = \sum_{x} \sum_{\omega:X(\omega)=x} X(\omega)p(\omega) = \sum_{x} x \sum_{\omega:X(\omega)=x} p(\omega) = \sum_{x} xp_X(x).$$

Exercise Lucky Dice

You roll a dice.

If the outcome is 6, you receive 10 CHF. Otherwise, you pay 1 CHF.

What is your expected gain or loss?

Solution

Recall: $\Omega = \{1, 2, 3, 4, 5, 6\}$ *and for each* ω, $p(\omega) = 1/6$.

$$X(\omega) = \begin{cases} 10, & \omega = 6, \\ -1, & \omega \in \{1, 2, 3, 4, 5\}. \end{cases}$$

Then,

$$\mathbb{E}[X] = \sum_{\omega} X(\omega)p(\omega) = \frac{1}{6}(-1) + \frac{1}{6}(-1) + \frac{1}{6}(-1) + \frac{1}{6}(-1) + \frac{1}{6}(-1) + \frac{1}{6} \cdot 10$$

$$= \sum_{x} x p_X(x) = \frac{5}{6}(-1) + \frac{1}{6} \cdot 10$$

Expectation is a linear operation in the following sense:

Let $X_1, X_2, \ldots, X_n$ be random variables and $\alpha_1, \alpha_2, \ldots, \alpha_n$ be scalars. Then

$$\mathbb{E}\Big[\sum_{i=1}^{n} X_i \alpha_i\Big] = \sum_{i=1}^{n} \alpha_i \mathbb{E}[X_i].$$

(See e.g. Rosen.)

Recall that two events E and F are independent iff

$$p(E|F) = p(E)$$

or, equivalently, iff

$$p(E \cap F) = p(E)p(F).$$

Two random variables X and Y are independent iff, for all realizations x and y,

$$p(\{X = x\} \cap \{Y = y\}) = p(\{X = x\})p(\{Y = y\}),$$

or, more concisely, iff

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

Generalization to n random variables is straightforward: $X_1, \ldots, X_n$ are independent iff

$$p_{X_1,\ldots,X_n}(x_1, \ldots, x_n) = \prod_{i=1}^{n} p_{X_i}(x_i).$$

The conditional distribution of Y given X is the function $p_{Y|X}$ defined by

$$p_{Y|X}(y|x) = \frac{p_{X,Y}(x,y)}{p_X(x)}.$$

It is defined for all x such that $p_X(x) \neq 0$.

The following statements are equivalent to the statement that X and Y are independent random variables:

- $p_{X,Y} = p_X p_Y$;
- $p_{Y|X}(y|x) = p_Y(y)$ (for all x for which it is defined and for all y);
- $p_{Y|X}(y|x)$ is not a function of x;
- $p_{X|Y}(x|y) = p_X(x)$ (for all y for which it is defined and for all x);
- $p_{X|Y}(x|y)$ is not a function of y.

EXERCISE

Let L be the random variable modeling Lisa's experiment.

Let L_1 and L_2 be the first and the second digit of L, respectively.

Are L_1 and L_2 independent ?

Hint: Compute $p_L(13)$.

SOLUTION

$$p_{L_1}(1) = \frac{1}{6} \text{ (found earlier)}$$

$$p_{L_2}(3) = \frac{2}{36} \text{ (see table below)}$$

$$p_L(13) = 0 \neq p_{L_1}(1)p_{L_2}(3)$$

Hence L_1 and L_2 are NOT independent.

L	02	03	04	05	06	07	08	09	10	11	12
p_L	$\frac{1}{36}$	$\frac{2}{36}$	$\frac{3}{36}$	$\frac{4}{36}$	$\frac{5}{36}$	$\frac{6}{36}$	$\frac{5}{36}$	$\frac{4}{36}$	$\frac{3}{36}$	$\frac{2}{36}$	$\frac{1}{36}$

L_1	0	1
p_{L_1}	$\frac{5}{6}$	$\frac{1}{6}$

L_2	0	1	2	3	4	5	6	7	8	9
p_{L_2}	$\frac{3}{36}$	$\frac{2}{36}$	$\frac{2}{36}$	$\frac{2}{36}$	$\frac{3}{36}$	$\frac{4}{36}$	$\frac{5}{36}$	$\frac{6}{36}$	$\frac{5}{36}$	$\frac{4}{36}$

The expected value of a product is NOT always the product of the expected values.

Example: $X = Y \in \{-1, 1\}$, uniformly distributed.

$$\mathbb{E}[XY] = 1$$
$$\mathbb{E}[X]\mathbb{E}[Y] = 0$$

However, if X and Y are independent random variables, then

$$\mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y].$$

(See e.g. Rosen.)

Summary — Probability Review

- Random variable
- Probability distribution
  - Joint distribution of multiple random variables.
  - Marginal distribution.
  - Conditional distribution.
- Independence

Sources and Entropy
(Book Chapter 1)

Outline

How do we communicate in the digital world?

We communicate by revealing the value of a sequence of variables that we call **(information) symbols**.

The i-th symbol might represent

- the intensity of the ith pixel of a black/white digital photo
- your score in your ith exam
- the ith bit of a binary file
- the ith letter of a text
- etc.

In an article that appeared in 1928, Hartley (Bell Labs) wrote: A symbol provides information only if there had been other possibilities for its value, besides that which was revealed.

In modern language, Hartley was saying that the value of a symbol provides information only if the symbol is a (non-constant) **random variable**.

In the same article, Hartley gave a tentative answer to the following related question: How much information is carried by a symbol such as S?

Hartley's answer:

- Suppose that $S \in \mathcal{A}$ is a symbol that can take on $|\mathcal{A}|$ different values.
- The amount of information conveyed by n such symbols should be n times the information conveyed by one symbol.
- There are $|\mathcal{A}|^n$ possible values for the n symbols.
- This suggests that $\log |\mathcal{A}|^n = n \log |\mathcal{A}|$ is the appropriate measure for information, where we are free to choose the base for the logarithm.

Example

In a village that has 8 telephones, we can assign a different three-digit binary number, such as 001, to each phone.

Hence it takes 3 bits of information to identify a phone. Mathematically, the phones are represented by a uniformly distributed random variable $S \in \mathcal{A} = \{1, 2, \ldots, 8\}$.

Example

The world population in 2024 is estimated to be 8.1 billion.

Hence it takes $\log_2(8.1 \times 10^9) = 32.9$ bits of information to identify a person. A person is represented by a uniformly distributed random variable $S \in \mathcal{A} = \{1, 2, \ldots, 8.1 \times 10^9\}$.

The world population in 1970 is estimated to have been 3.7 billion. How many bits did it take back then?

The following example shows that something is not right with Hartley's measure of information.

Example

Suppose that $S_n \in \{\text{sunny}, \text{rainy}\}$ is the weather prognosis for day $n+1$, revealed on day n. Suppose that $S_n = \text{rainy}$ has probability $\frac{5}{365}$.

It seems intuitively obvious that the amount of information provided by $S_n = \text{rainy}$ is much higher than that provided by $S_n = \text{sunny}$.

Hartley's measure assigns $\log_2(2) = 1$ bit of information to both, $S_n = \text{sunny}$ and $S_n = \text{rainy}$.

In an article that appeared in 1948, Shannon fixes the problem by defining the notion of **uncertainty** or **entropy** $H(S)$ associated to a discrete random variable S.

Definition (entropy, uncertainty)

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

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

A few comments are in order regarding

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

- The condition $s \in \mathrm{supp}(p)$ is needed because $\log_b p_S(s)$ is not defined if $p_S(s) = 0$.
- To simplify the notation, we declare that $p_S(s) \log p_S(s) = 0$ when $p_S(s) = 0$. This convention allows us to simplify the notation to

$$H_b(S) = -\sum_{s \in \mathcal{A}} p_S(s) \log_b p_S(s).$$

- The choice of the base b determines the unit. Typically $b = 2$. In this case, the unit is the **bit**.

We can think of evaluating

$$H(S) = -\sum_{s \in \mathcal{A}} p_S(s) \log p_S(s)$$

by first computing $-\log p_S(s)$ for each $s \in \mathcal{A}$, and then take the average (excluding zero-probability terms).

Hence we can write

$$H(S) = \mathbb{E}[-\log p_S(S)].$$

EXAMPLE

When p_S is the uniform distribution over the alphabet $\mathcal{A}$, $p_S(s) = \frac{1}{|\mathcal{A}|}$ and

$$-\log p_S(s) = \log |\mathcal{A}|, \text{ which is constant.}$$

In this case

$$H(S) = \mathbb{E}[\log |\mathcal{A}|] = \log |\mathcal{A}|,$$

which is Hartley's information measure.

Hence Shannon's entropy equals Hartley's measure of information if (and only if as we will see) the random variable has uniform distribution.

We will see that Shannon's entropy it is indeed the answer to very practical engineering questions.

Example (Anne's lock)

A sequence of 4 decimal digits s_1, s_2, s_3, s_4 representing the number to open Anne's lock can be seen as the output of a source S_1, S_2, S_3, S_4 with $S_i \in \mathcal{A} = \{0, 1, \ldots, 9\}$.

If Anne picks each of the 4 digits at random and independently, then all 4-digit sequences are equiprobable, i.e.,

$$p_{S_1,S_2,S_3,S_4}(s_1, s_2, s_3, s_4) = \frac{1}{10^4} \quad \text{for all 4-digit numbers } s_1 s_2 s_3 s_4.$$

Notation: When no confusion can arise, we write $p(s_1, s_2, s_3, s_4)$ instead of $p_{S_1,S_2,S_3,S_4}(s_1, s_2, s_3, s_4)$.

Example (Anne's lock, alternative view)

We can also take the view that Anne's lock number is modeled by a single random variable

$$S \in \mathcal{A} = \{0000, 0001, \ldots, 9998, 9999\}$$

$$p(s) = \frac{1}{10^4} \quad \text{for all 4-digit numbers.}$$

Since the distribution is uniform over the alphabet $\mathcal{A}$,

$$H(S) = \log_2 |\mathcal{A}| = \log_2 10^4 \approx 13.3 \text{ bits.}$$

Example (Entropy of french)

A monkey produces text by selecting letters at random from a French text

$$H = 3.9425 \text{ bits.}$$

As we will see shortly, the maximum entropy of a source with $|\mathcal{A}| = 26$ is $\log_2 26 = 4.7004$ bits.

Letter	Prob.
A	0.0811
B	0.0081
C	0.0338
D	0.0428
E	0.1769
F	0.0113
G	0.0119
H	0.0074
I	0.0724
J	0.0018
K	0.0002
L	0.0599
M	0.0229
N	0.0768
O	0.0520
P	0.0292
Q	0.0083
R	0.0643
S	0.0887
T	0.0744
U	0.0523
V	0.0128
W	0.0006
X	0.0053
Y	0.0026
Z	0.0012

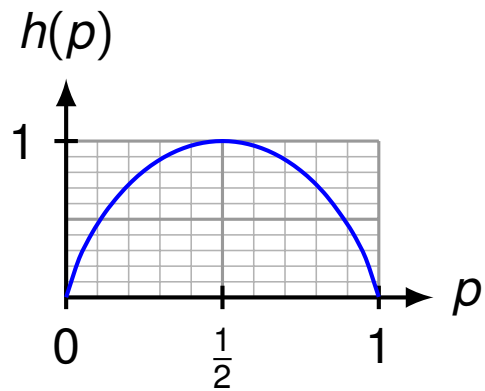
Binary Entropy Function

An interesting special case is when $|\mathcal{A}| = 2$.

In this case, p_S has only two possible values, say p and $(1-p)$.
The corresponding entropy is $H(S) = h(p)$ where

$$h(p) := -p \log_2 p - (1-p) \log_2 (1-p).$$

$h(p)$ is called the **binary entropy function**.



- For $p = 0$ and for $p = 1$, $h(p) = 0$.
- For $p = \frac{1}{2}$, $h(p) = 1$.
- For $p \in \{0.0001, 0.9999\}$, $H(S) \approx 0.001$.

EXAMPLE

Let S_n be the above weather forecast.

The probabilities of S_n are $p = \frac{5}{365}$ and $(1 - p) = \frac{360}{365}$.

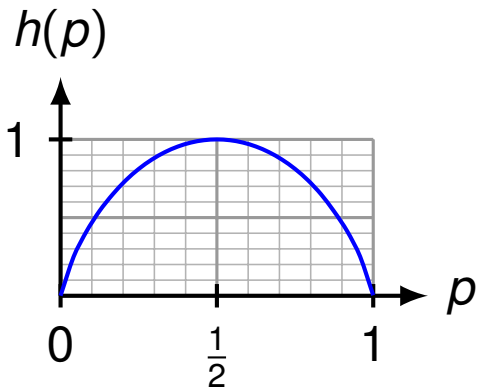
$$H_2(S_n) = h\left(\frac{5}{365}\right) = -\frac{5}{365}\log_2\frac{5}{365} - \frac{360}{365}\log_2\frac{360}{365} \approx 0.072 \text{ bits.}$$

EXAMPLE

Let S be the answer to the question "Is 8950 Anne's lock number ?".

Now $S \in \mathcal{A} = \{YES, NO\}$ is a binary random variable with $p_S(YES) = \frac{1}{10^4}$.

Hence $H(S) = h\left(\frac{1}{10^4}\right) \approx 0.001$ bits.



Information-Theory Inequality

Surprisingly many results in information theory are a direct consequence of the following key inequality.

Lemma (IT-Inequality)

For a positive real number r,

$$\log_b r \leq (r-1)\log_b(e),$$

with equality iff (if and only if) $r = 1$.

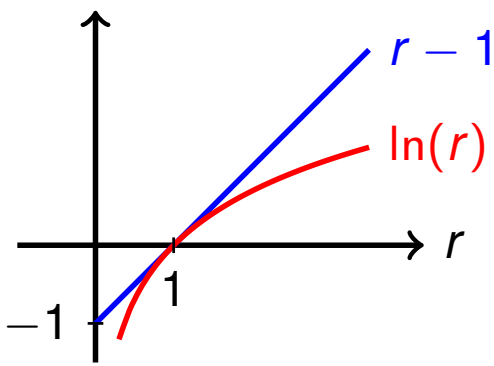
Proof [IT-Inequality]: Because $\log_b(r) = \ln(r)\log_b(e)$, it suffices to show that

$$\ln r \leq (r-1),$$

with equality iff $r = 1$.

The inequality is true (see graph below) because:

- the functions $\ln r$ and $r-1$ coincide at $r = 1$,
- the function $r - 1$ has slope 1 throughout,
- $\frac{d}{dr}\ln(r) = \frac{1}{r} < 1$ for $r > 1$,
- $\frac{d}{dr}\ln(r) = \frac{1}{r} > 1$ for $r < 1$.



Theorem (Entropy Bounds)

The entropy of a discrete random variable $S \in \mathcal{A}$ satisfies

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

with equality on the left iff $p_S(s) = 1$ for some s, and with equality on the right iff $p_S(s) = \frac{1}{|\mathcal{A}|}$ for all s.

Proof of the left inequality:

Recall that

$$H(S) = \sum_{s \in \mathcal{A}} -p_S(s) \log p_S(s),$$

and observe that

$$-p_S(s) \log p_S(s) = \begin{cases} 0 & \text{if } p_S(s) \in \{0, 1\} \\ > 0 & \text{if } 0 < p_S(s) < 1. \end{cases}$$

Thus, $H(S) \geq 0$, with equality iff $p_S(s) \in \{0, 1\}$ for all $s \in \mathcal{A}$.

$p_S(s) \in \{0, 1\}$ for all $s \in \mathcal{A}$ iff $p_S(s) = 1$ for some s. □

To prove the right inequality, we use a trick that often works in inequalities involving entropies:

To prove, say, $A \leq B$, we prove $A - B \leq 0$ by means of the IT-Inequality.

Proof of the right inequality:

$$
\begin{aligned}
H(S) - \log|\mathcal{A}| &= \mathbb{E}\Big[-\log p_S(S)\Big] - \log|\mathcal{A}| \\
&= \mathbb{E}\Big[-\log p_S(S) - \log|\mathcal{A}|\Big] \\
&= \mathbb{E}\left[\log \frac{1}{p_S(s)|\mathcal{A}|}\right] \\
&= \sum_{s\in\mathcal{A}} p_S(s)\left[\log \frac{1}{p_S(s)|\mathcal{A}|}\right] \\
&\overset{\text{(IT-Inequality)}}{\leq} \sum_{s\in\mathcal{A}} p_S(s)\left[\frac{1}{p_S(s)|\mathcal{A}|} - 1\right]\log(e) \\
&= \log(e)\sum_{s\in\mathcal{A}}\left[\frac{1}{|\mathcal{A}|} - p_S(s)\right] \\
&= \log(e)\,(1-1) = 0,
\end{aligned}
$$

with equality iff $p_S(s)|\mathcal{A}| = 1$ for all s. □

Example

Let S be Anne's lock number. Its entropy is maximized if Anne chooses at random over all 10^4 possibilities.

In this case, and only in this case,

$$H(S) = \log |\mathcal{A}| = \log 10^4 \approx 13.3 \text{ bits.}$$

Example

Let S be Anne's grandmother's lock number. She always picks $S = 0000$. Then

$$H(S) = 0$$

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

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

which means

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

EXAMPLE

Let $p_{X,Y}$ be given by the following table

x	y	$p_{XY}(x,y)$
0	0	1/8
0	1	3/8
1	0	1/4
1	1	1/4

$$H(X,Y) = -\tfrac{1}{8}\log_2\tfrac{1}{8} - \tfrac{3}{8}\log_2\tfrac{3}{8} - \tfrac{1}{4}\log_2\tfrac{1}{4} - \tfrac{1}{4}\log_2\tfrac{1}{4}.$$

We are mainly interested is sources that emit a large number of random variables. (We want to compress large amounts of data.)

The sequence of random variables can be

- finite, like in $(S_1, \ldots, S_n)$ (random vector)
- infinite, like in $S_1, S_2, \ldots$ (random sequence, random process), also denoted by $\{S_i\}_{i=1}^{\infty}$.
- sometimes it is convenient to consider $\ldots, S_{-1}, S_0, S_1, \ldots$, also denoted by $\{S_i\}$.

A collection of random variables $(S_1, \ldots, S_n)$ is specified by the joint probability distribution $p_{S_1,\ldots,S_n}$. This is all we need to compute the entropy $H(S_1, \ldots, S_n)$.

Example (Coin-Flip Source)

A sequence of coin flips can be seen as the output of a source $S_1, S_2, \ldots, S_n$ with $S_i \in \mathcal{A} = \{H, T\}$, where H stands for head, and T for tail, $i = 1, \ldots, n$.

If the coin is fair, all sequences are equally likely:

$$p(s_1, s_2, \ldots, s_n) = \prod_i p(s_i) = \frac{1}{2^n} \quad \text{for all } (s_1, s_2, \ldots, s_n) \in \mathcal{A}^n$$

Notation: $\mathcal{A}^n = \underbrace{\mathcal{A} \times \mathcal{A} \times \cdots \times \mathcal{A}}_{n \text{ times}}$ is the n-fold cartesian product of $\mathcal{A}$.

The following statement is a corollary to two fundamental results that we will prove next week.

Theorem (1.4 of textbook)

Let $S_1, \ldots, S_n$ be discrete random variables. Then

$$H(S_1, S_2, \ldots, S_n) \leq H(S_1) + H(S_2) + \cdots + H(S_n),$$

with equality iff $S_1, \ldots, S_n$ are independent.

EXAMPLE

Let S_1 and S_2 be the random variables associated to Bart's two dice rolls.

$$H(S_1) = H(S_2) = \log 6 \quad \text{(the two distributions are uniform)}$$
$$H(S_1, S_2) = \log 36 \quad \text{(the distribution of } (S_1, S_2) \text{ is uniform)}$$

We verify that

$$H(S_1, S_2) = \log 36 = \log 6^2 = 2 \log 6 = H(S_1) + H(S_2).$$

Example (Entropies in Lisa's experiment)

- $H(L_1) = 0.65$ bits
- $H(L_2) = 3.22$ bits
- $H(L_1, L_2) = 3.27$ bits

$H(L_1, L_2) < H(L_1) + H(L_2)$. Hence L_1 and L_2 are **not** independent.

We will see that entropy is fundamental in all three topics:

- Source coding: To derive the limit to how much a source can be compressed.
- Cryptography: To derive the length of the shortest key for which perfect secrecy is possible.
- Channel coding: To derive the highest rate at which we can communicate reliably across an unreliable communication channel.

Stay tuned!

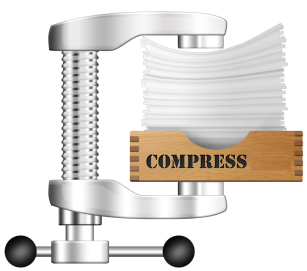
Week 2: Source Coding (Compression)

Prof. Michael Gastpar

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

EPFL

Spring Semester 2025



Outline

DEFINITION (ENTROPY, UNCERTAINTY)

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

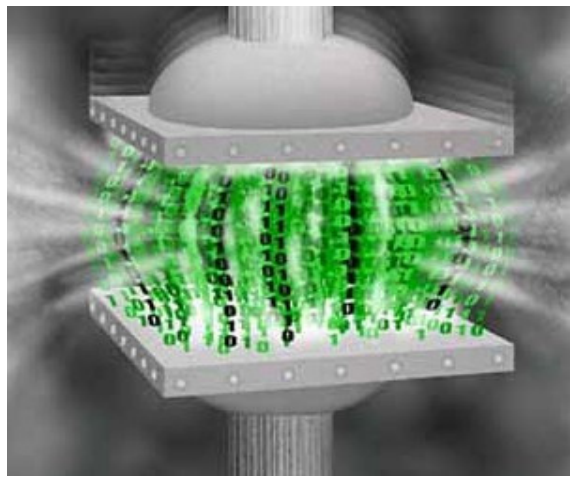
where $\mathrm{supp}(p_S) = \{s : p_S(s) > 0\}$.

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.*

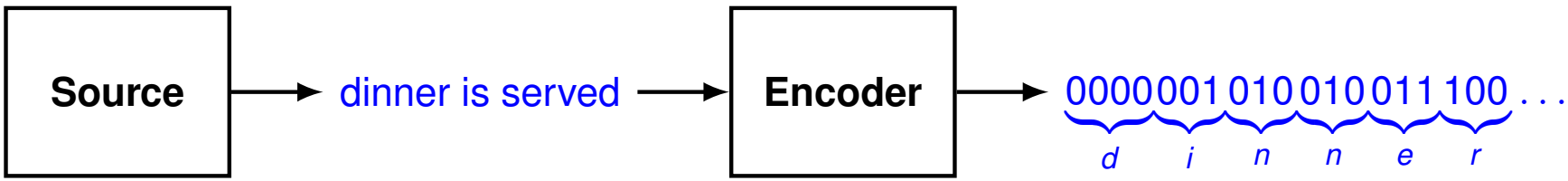
SOURCE CODING PURPOSE

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.

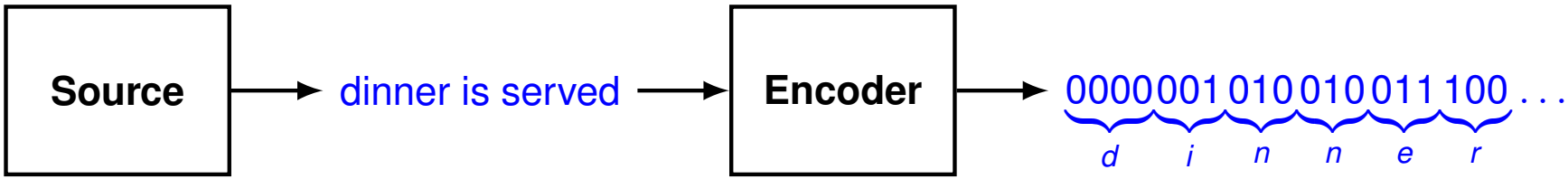
For a fixed description alphabet (often binary), we want to minimize the average description length.



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:

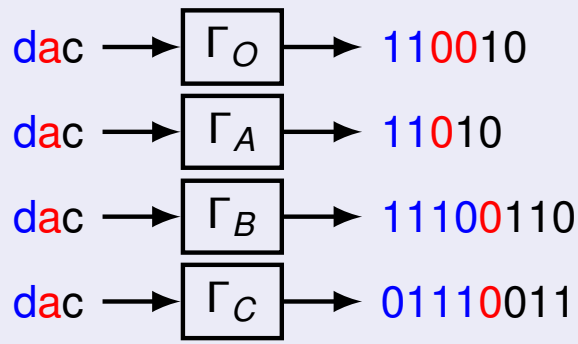
- the input alphabet $\mathcal{A}$ (the same as the source alphabet);
- the output alphabet $\mathcal{D}$ (typically $\mathcal{D} = \{0, 1\}$);
- the codebook $\mathcal{C}$ which consists of finite sequences over $\mathcal{D}$;
- by the one-to-one encoding map $\Gamma : \mathcal{A}^k \to \mathcal{C}$,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

For now, $k = 1$.

Example (Four little codes)

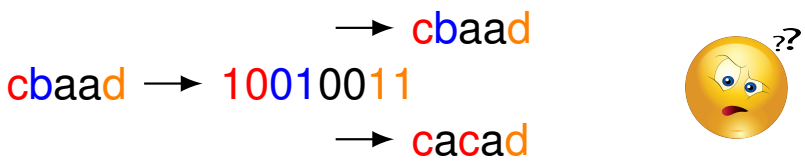
For each code, the encoding map Γ is specified in the following table:

$\mathcal{A}$	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.

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.

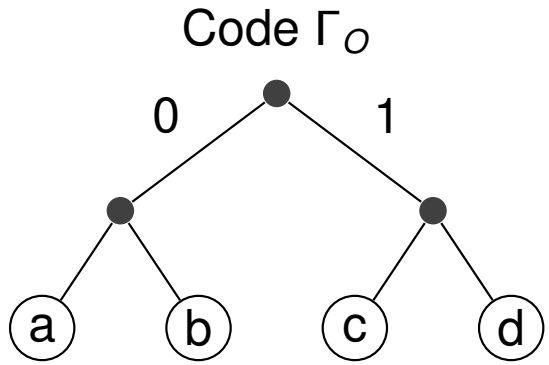
CODES FOR ONE RANDOM VARIABLE

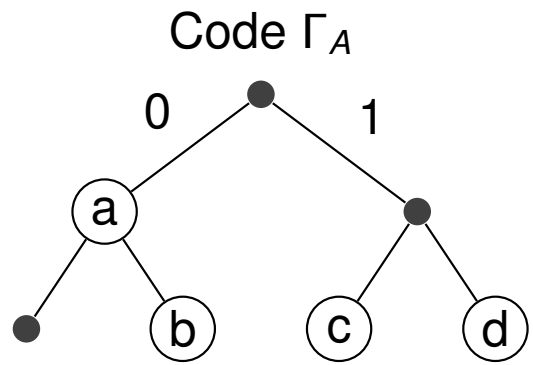
We start by considering codes that encode one single random variable $S \in \mathcal{A}$.

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

Complete tree of a code

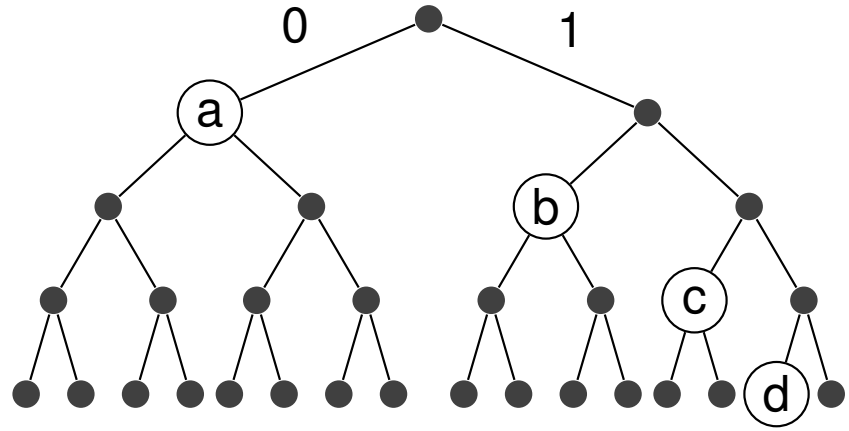
$\mathcal{A}$	Γ_O	Γ_A
a	00	0
b	01	01
c	10	10
d	11	11



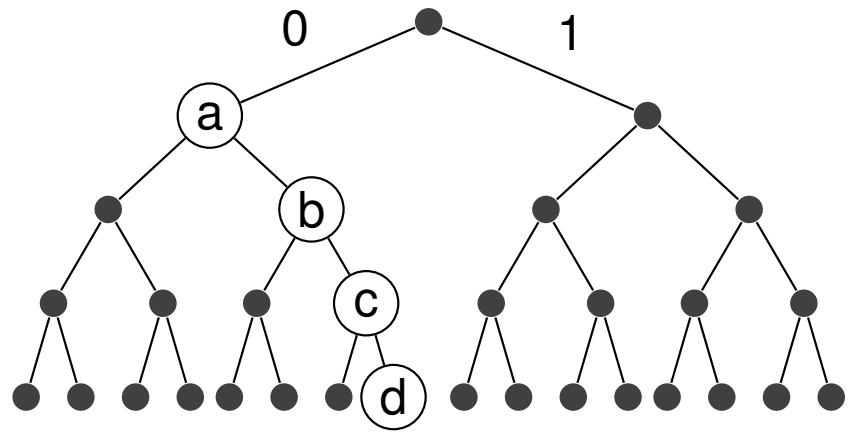


$\mathcal{A}$	Γ_B	Γ_C
a	0	0
b	10	01
c	110	011
d	1110	0111

Code Γ_B

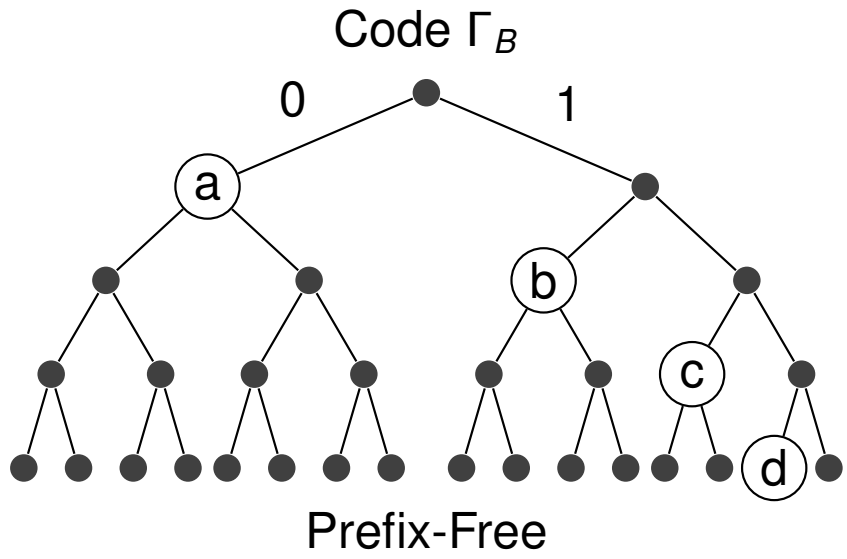


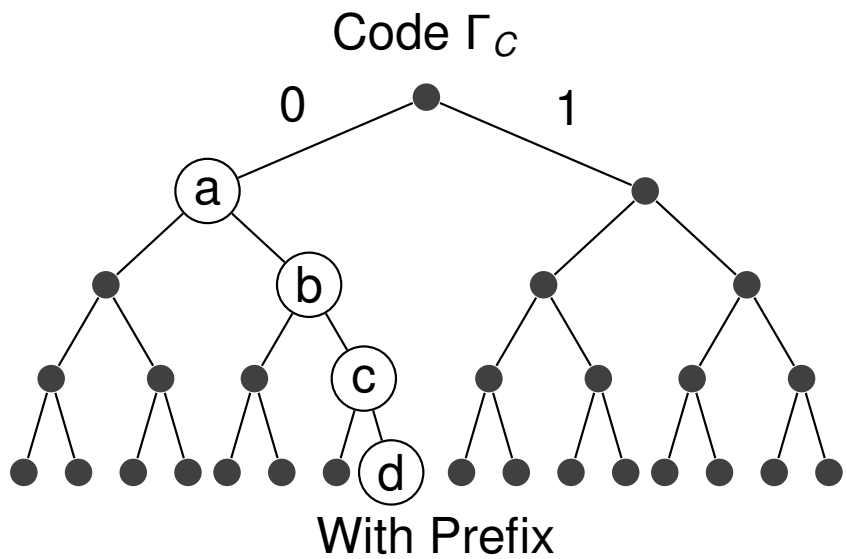
Code Γ_C



WITH/WITHOUT PREFIX

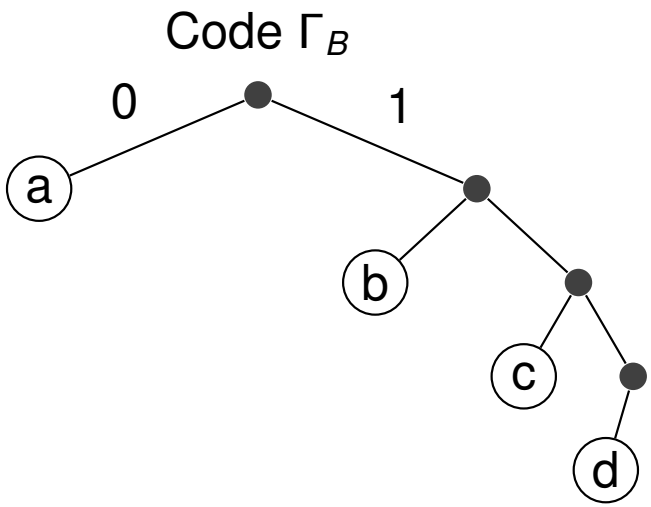
$\mathcal{A}$	Γ_B	Γ_C
a	0	0
b	10	01
c	110	011
d	1110	0111





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 $2^{-1} + 2^{-2} + 2^{-2} + 2^{-2} = 1.25 > 1$.

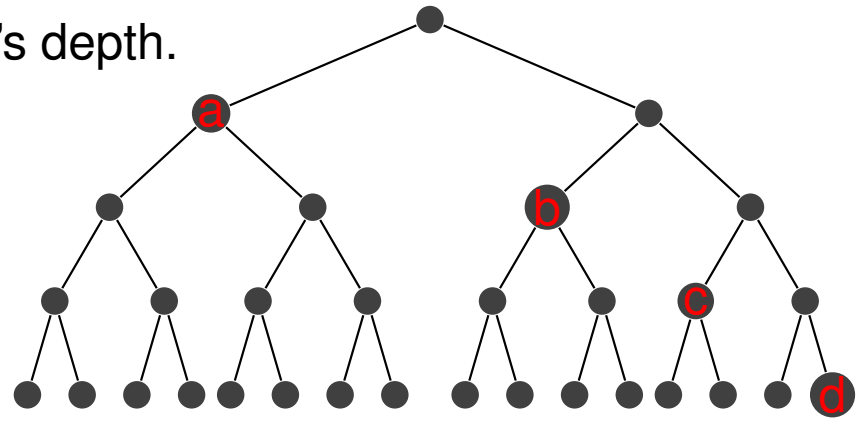
Kraft-McMillan's inequality is not fulfilled.

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)

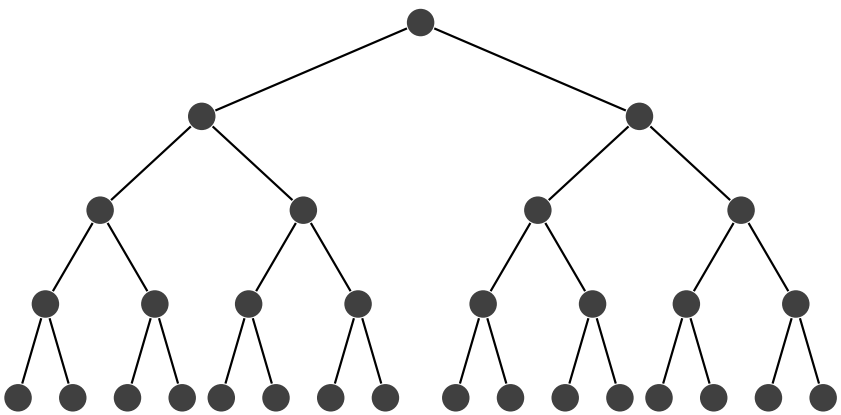
If the positive integers $l_1, \ldots, l_M$ satisfy Kraft's inequality for some positive integer D, then there exists a D-ary prefix-free code (hence uniquely decodable) that has codeword lengths $l_1, \ldots, l_M$.

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.

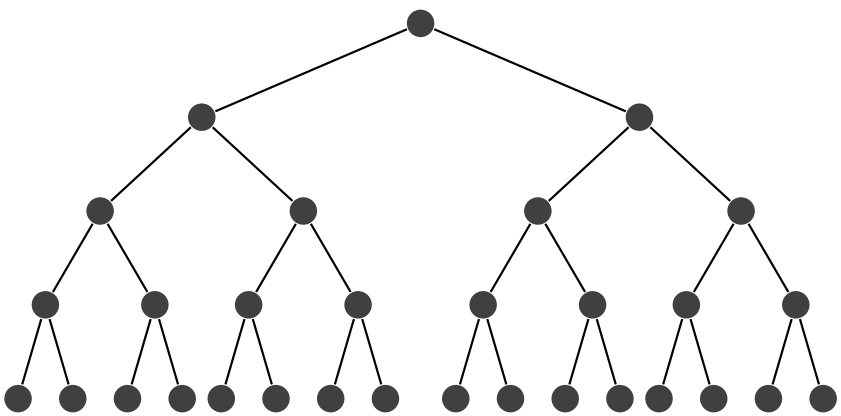
(Use the following tree as a starting point.)

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

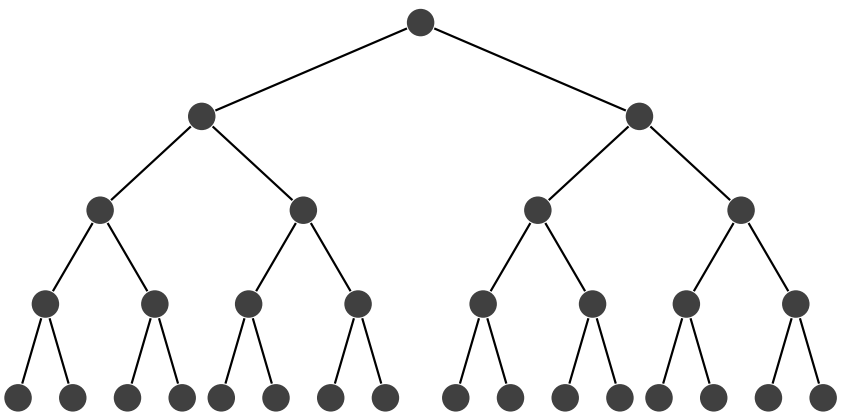


Proof Outline of K-MM part 2

- 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.

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

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

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

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. □

Example (cont.)

- $H(S) = -2 \times 0.05 \times \log 0.05 - 0.1 \times \log 0.1 - 0.8 \times \log 0.8 = 1.022$
- 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.

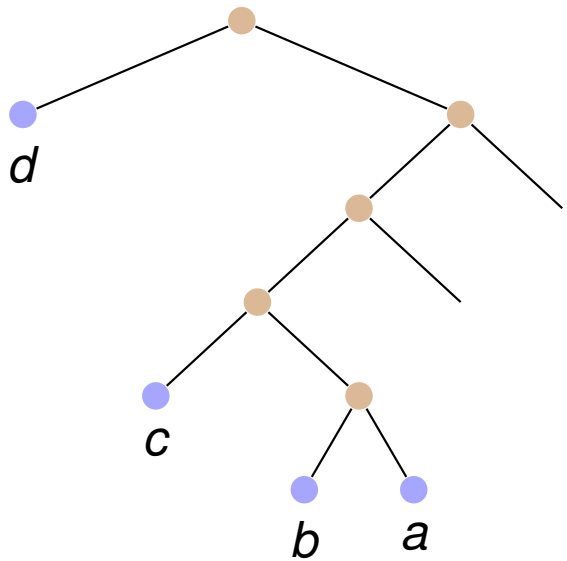
$s \in \mathcal{A}$	$p_S(s)$	$\lceil -\log_2 p_S(s) \rceil$
a	0.05	
b	0.05	
c	0.1	
d	0.8	

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

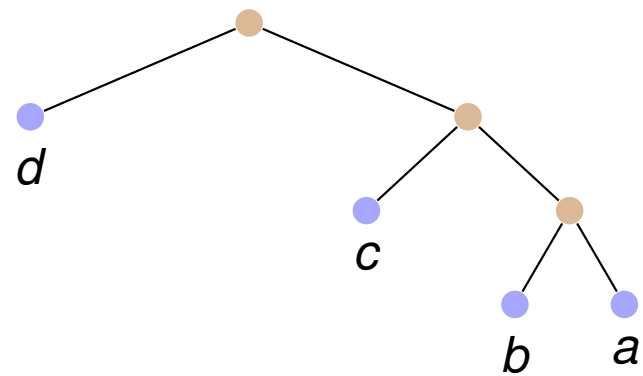
x	0.05	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
$-\log_2(x)$	4.3219	3.3219	2.3219	1.7370	1.3219	1.0000	0.7370	0.5146	0.3219	0.1520

$s \in \mathcal{A}$	$p_S(s)$	$\lceil -\log_2 p_S(s) \rceil$
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

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

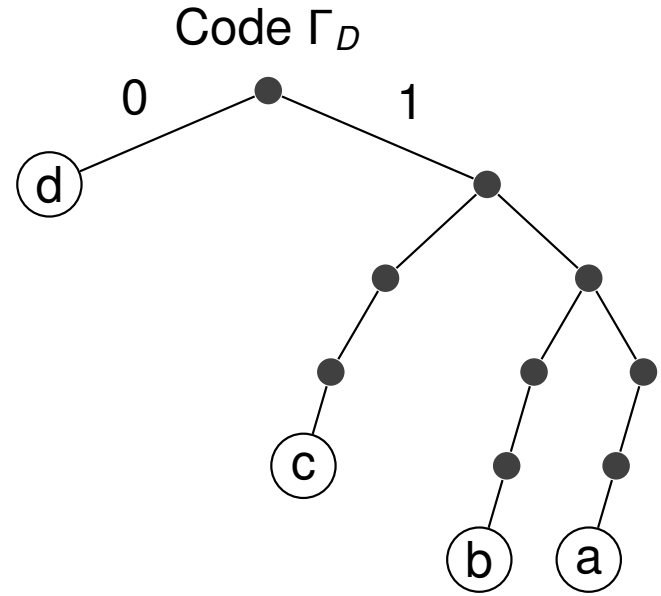
Instead we use $l_i = \lceil -\log p_i \rceil < -\log p_i + 1$.

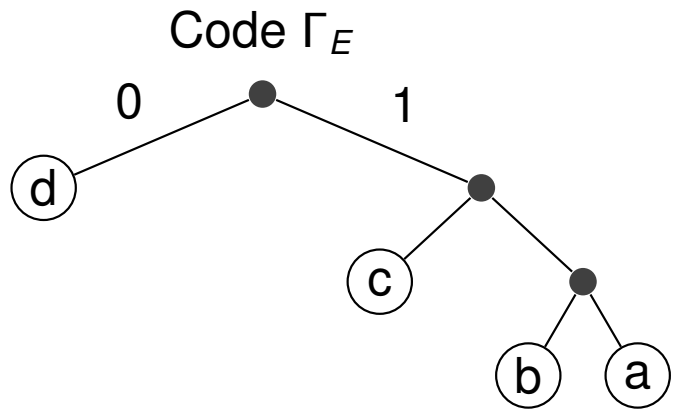
Since each term of an average increases by less than 1, the average itself increases by less than 1. □

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

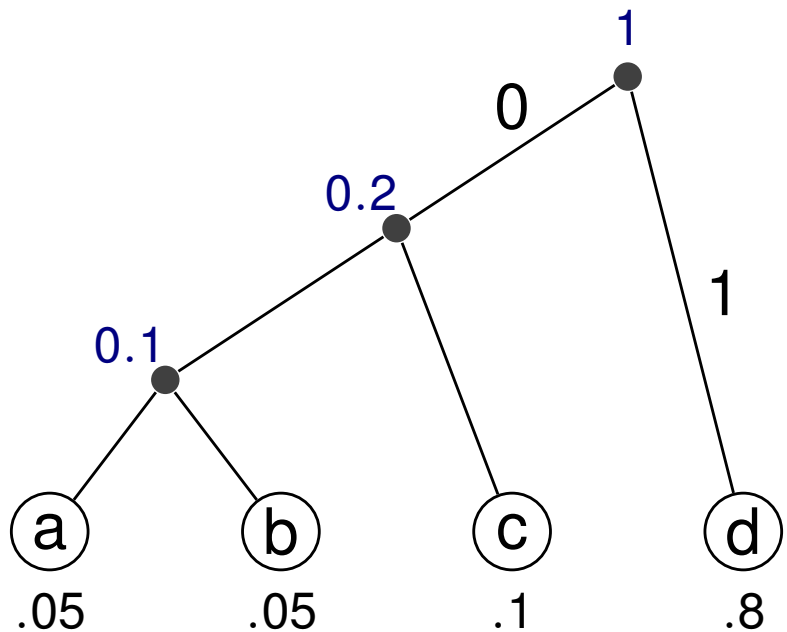




So, Shannon-Fano codes are good, but not optimal.

Optimal Code: Huffman Code

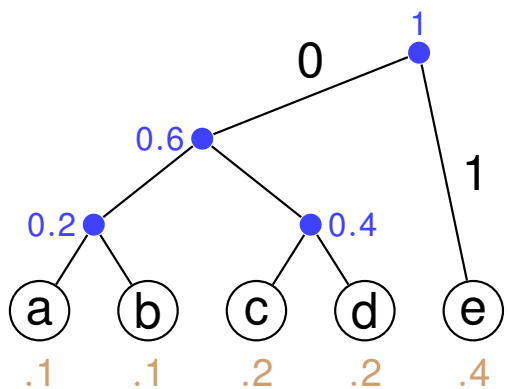
Unlike the Shannon-Fano code, the construction of the Huffman code starts from the leaves.



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.)

Tree with probabilities: A handy tool

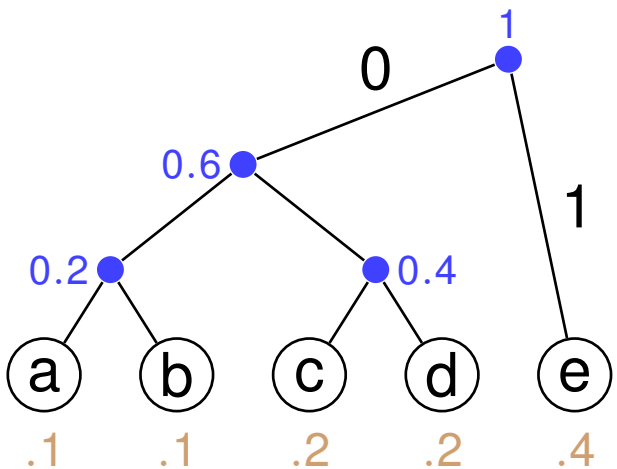
- Consider a tree with probabilities assigned to leaf nodes, like the decoding tree of a prefix-free code
- The probabilities of the leaf nodes induce probabilities to the intermediate notes (like in Huffman's construction).
- The result is called a **tree with probabilities**.



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):

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



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

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

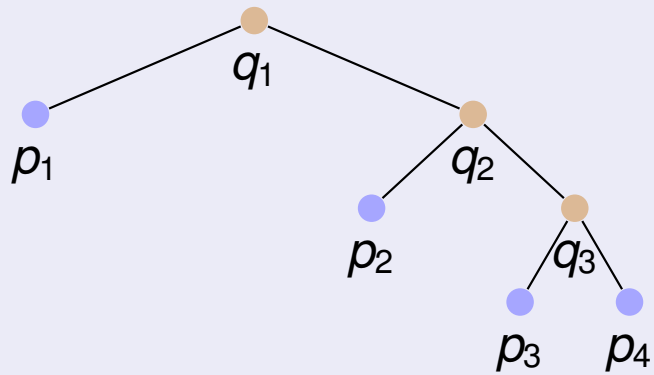
Hence

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

□

Example

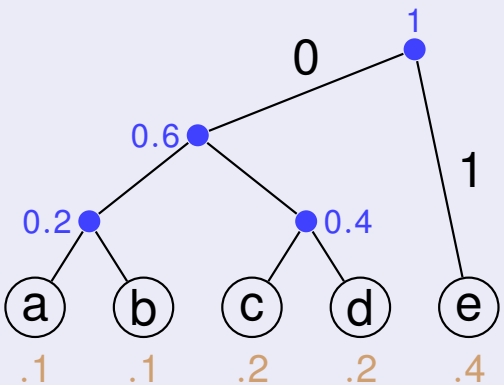
The following example mirrors the proof.



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

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)

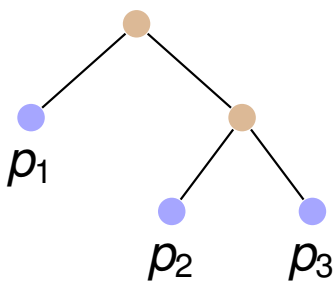
If Γ_H is a Huffman code (prefix-free by construction) and Γ is another uniquely decodable code for the same source S, then it is guaranteed that

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

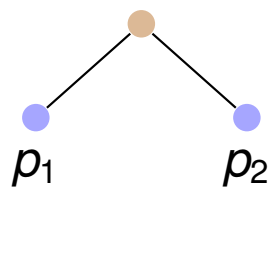
Proof: For simplicity, we consider only binary codes. Let $L = \max_i l_i$. The proof is based on the following three facts.

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

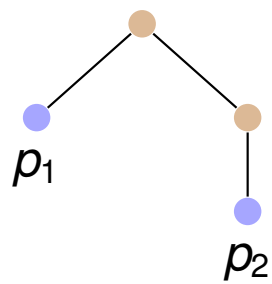
Examples:



This is OK



This is OK

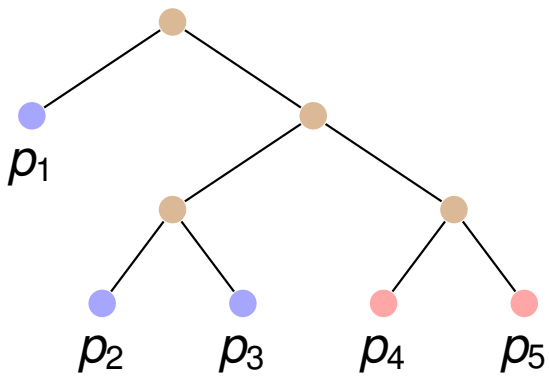


This is NOT OK

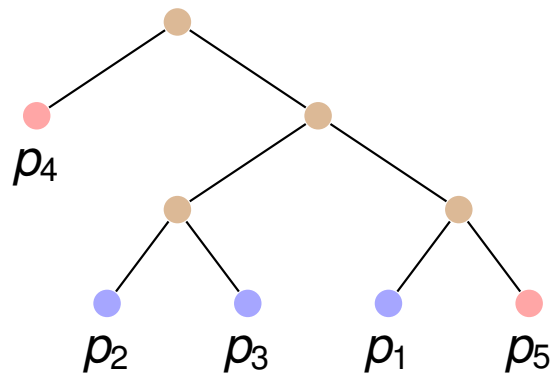
In particular, any leaf at depth L has a sibling.

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

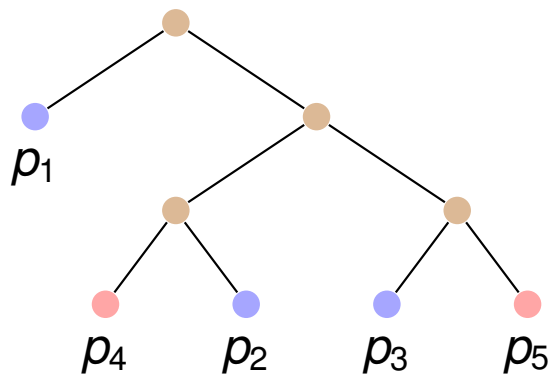


This is NOT OK

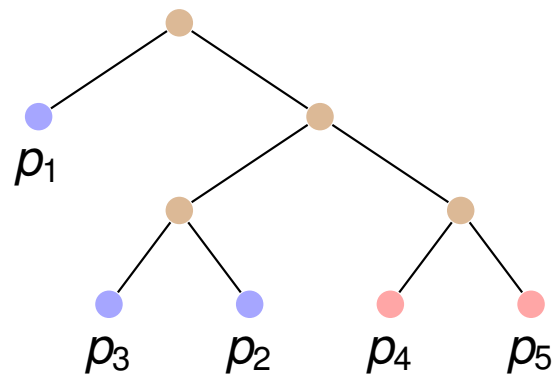
This and Fact 1 imply that two of the least likely codewords have length L.

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 . . .

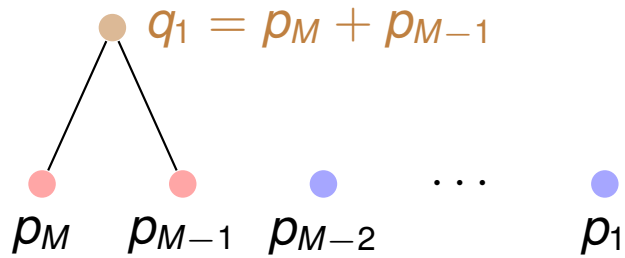


. . . then this is also optimal

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 $q_1, p_{M-2}, \ldots, p_1$.

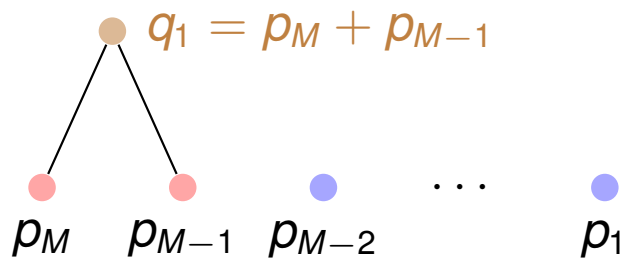
In $\tilde{\Gamma}$, q_1 is a leaf node.

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, $L = \tilde{L} + q_1$.

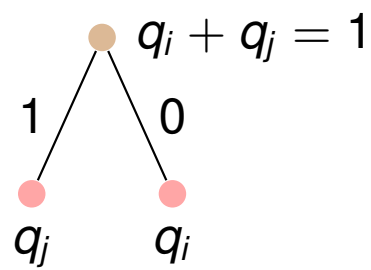
Hence, L is as small as possible iff $\tilde{L}$ is as small as possible.

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. □

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

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

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

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

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

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

Week 3: Conditional Entropy (Book Chapter 4)

Prof. Michael Gastpar

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

Spring Semester 2025

Outline

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.

Our Next Nugget

- Understand the behavior of

$$H_D(S_1, S_2, \ldots, S_n)$$

when $S_1, S_2, \ldots, S_n$ are not independent random variables following the same distribution.

Key steps to get there:

- Understand **conditional entropy**
- Understand how to model "many" random variables (a.k.a. **random processes**)

Our Next Nugget

Example: Standard text.

- After a letter “q”, we have a letter “u” with very high probability (probability 1 in some languages).
- After a letter “c”, we have a letter “h” with higher probability than many other letters.
- After a letter “i”, it is extremely unlikely to have yet another letter “i”. And so on.

Our Next Nugget

Example: Audio recoding.

- Why?

Example: Image.

Example: Video recording.

Key (Simple) Example 1 : Independent

Definition (Coin-Flip Source)

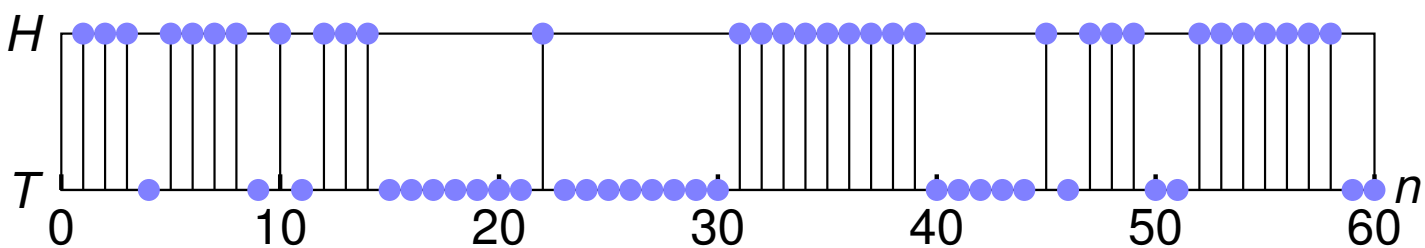
The source models a sequence $S_1, S_2, \ldots, S_n$ of n coin flips.

So $S_i \in \mathcal{A} = \{H, T\}$, where H stands for heads, T for tails, $i = 1, 2, \ldots, n$.

$p_{S_i}(H) = p_{S_i}(T) = \frac{1}{2}$ for all i, and coin flips are independent.

Hence,

$$p_{S_1,S_2,\ldots,S_n}(s_1, s_2, \ldots, s_n) = \frac{1}{2^n} \quad \text{for all } (s_1, s_2, \ldots, s_n) \in \mathcal{A}^n$$



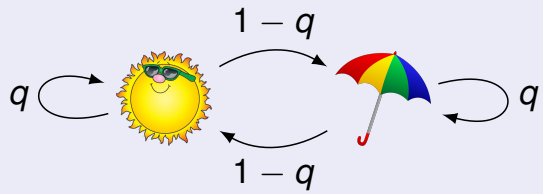
Key (Simple) Example 2 : Not Independent

Definition (Sunny-Rainy Source)

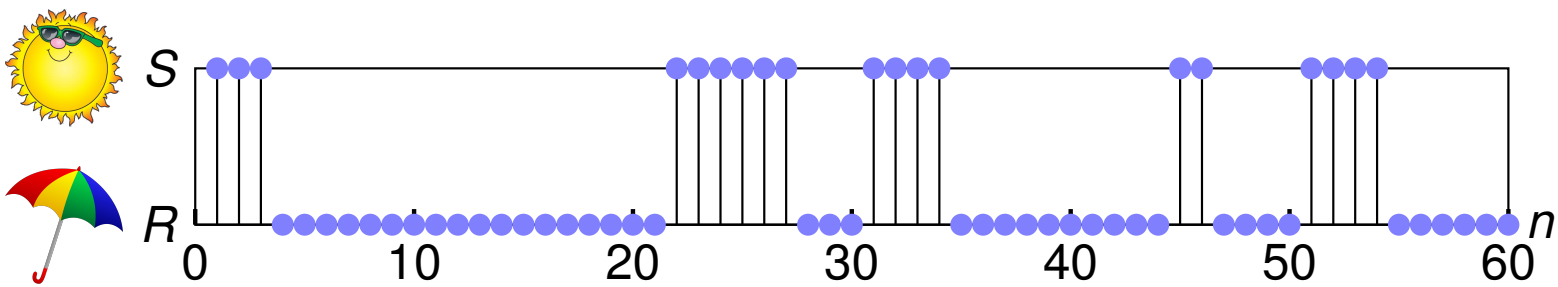
The source models a sequence $S_1, S_2, \ldots, S_n$ of weather conditions.

So $S_i \in \mathcal{A} = \{S, R\}$, where S stands for sunny, R for rainy, $i = 1, 2, \ldots, n$.

The weather on the first day is uniformly distributed in $\mathcal{A}$.



For all other days, with probability $q = \frac{6}{7}$ the weather is as for the day before.



CONDITIONAL PROBABILITY

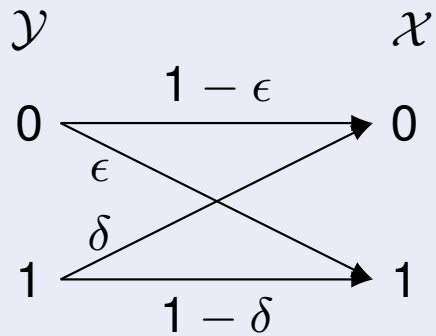
Recall how to determine the conditional probability:

$$p_{X|Y}(x|y) \stackrel{def}{=} \frac{p_{X,Y}(x,y)}{p_Y(y)}.$$

It gives the probability of the event $X = x$, given that the event $Y = y$ has occurred.

It is defined for all y for which $p_Y(y) > 0$.

EXAMPLE ("BIT FLIPPER CHANNEL")



$\Longleftrightarrow$

$p(x\|y)$	$y=0$	$y=1$
$x=0$	$1-\epsilon$	δ
$x=1$	ϵ	$1-\delta$

Suppose Y is uniformly distributed. Then, the joint distribution of X, Y is

$p(x,y)$	$y=0$	$y=1$
$x=0$	$\frac{1}{2}(1-\epsilon)$	$\frac{1}{2}\delta$
$x=1$	$\frac{1}{2}\epsilon$	$\frac{1}{2}(1-\delta)$

EXERCISE ("BIT FLIPPER CHANNEL")

As we have seen, for the bit flipper channel with uniform input Y, the joint distribution of X, Y is

$p(x,y)$	$y=0$	$y=1$
$x=0$	$\frac{1}{2}(1-\epsilon)$	$\frac{1}{2}\delta$
$x=1$	$\frac{1}{2}\epsilon$	$\frac{1}{2}(1-\delta)$

- Find the conditional distribution $p(y|x)$ (input given the output).

SOLUTION ("BIT FLIPPER CHANNEL")

The general formula is

$$p(y|x) = \frac{p(x,y)}{p(x)}.$$

Hence, we need the marginal distribution of X :

$p(x,y)$	$y=0$	$y=1$	Marginal distribution $p(x)$
$x=0$	$\frac{1}{2}(1-\epsilon)$	$\frac{1}{2}\delta$	$\frac{1}{2}(1-\epsilon)+\frac{1}{2}\delta$
$x=1$	$\frac{1}{2}\epsilon$	$\frac{1}{2}(1-\delta)$	$\frac{1}{2}\epsilon+\frac{1}{2}(1-\delta)$

Hence, we find the desired object:

$p(y\|x)$	$y=0$	$y=1$
$x=0$	$\frac{1-\epsilon}{1-\epsilon+\delta}$	$\frac{\delta}{1-\epsilon+\delta}$
$x=1$	$\frac{\epsilon}{1-\delta+\epsilon}$	$\frac{1-\delta}{1-\delta+\epsilon}$

- Convince yourself that indeed, $p(y|x)$ is a valid probability distribution for each fixed value of x.

Conditional expectation of X given $Y = y$

$p_{X|Y}(\cdot|y)$ is a probability distribution on the alphabet of X, just like $p_X(\cdot)$

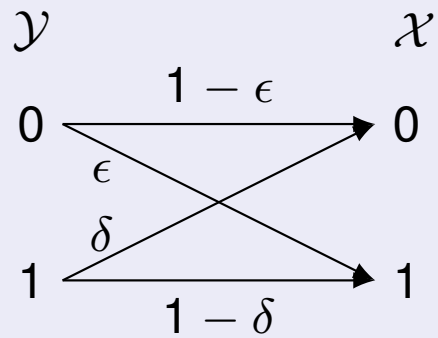
Definition

The conditional expectation of X given $Y = y$ is defined as

$$\mathbb{E}[X|Y = y] \stackrel{def}{=} \sum_{x \in \mathcal{X}} x p_{X|Y}(x|y).$$

Conditional expectation of X given $Y = y$

Exercise ("Bit flipper channel")



$\Longleftrightarrow$

$p(x\|y)$	$y=0$	$y=1$
$x=0$	$1-\epsilon$	δ
$x=1$	ϵ	$1-\delta$

- Find the conditional expectations $\mathbb{E}[X|Y=y]$ for $y=0$ and for $y=1$.

Conditional entropy of X given $Y = y$

$p_{X|Y}(\cdot|y)$ is a probability distribution on the alphabet of X, just like $p_X(\cdot)$

Every probability distribution has an entropy associated to it:

- $p_X(\cdot) \longrightarrow H(X)$
- $p_{X|Y}(\cdot|y) \longrightarrow H(X|Y = y)$

Definition

The conditional entropy of X given $Y = y$ is defined as

$$H_D(X|Y = y) \stackrel{def}{=} -\sum_{x \in \mathcal{X}} p_{X|Y}(x|y) \log_D p_{X|Y}(x|y).$$

Conditional entropy of X given $Y = y$

Exercise ("Bit flipper channel")

For the Bit flipper channel with uniform input, calculate:

- $H(X|Y = y)$ for each fixed y,
- $H(Y|X = x)$ for each fixed x.

Solution

THEOREM (BOUNDS ON CONDITIONAL ENTROPY OF X GIVEN $Y = y$)

The conditional entropy of a discrete random variable $X \in \mathcal{X}$ conditioned on $Y = y$ satisfies

$$0 \leq H_D(X|Y = y) \leq \log_D |\mathcal{X}|,$$

with equality on the left iff $p_{X|Y}(x|y) = 1$ for some x, and with equality on the right iff $p_{X|Y}(x|y) = \frac{1}{|\mathcal{X}|}$ for all x.

The proof is identical to our proof of the basic entropy bounds.

EXAMPLE ("BIT FLIPPER CHANNEL")

For the Bit flipper channel, verify the entropy bounds.

Question: Do we also have the following entropy bound:

$$H_D(X|Y=y) \overset{???}{\leq} H_D(X)?$$

Answer: No!

EXAMPLE (BIT FLIPPER WITH UNIFORM INPUT Y)

(Or "counterexample," if you prefer). Just for ease of calculation, let us set $\delta = 0$ (but this is not necessary for the example to work!). Then, we have:

$$H_D(X|Y=0) = h_D(\epsilon) \quad \text{and} \quad H_D(X|Y=1) = 0.$$

where $h_D(\cdot)$ is the binary entropy function (with $\log_D(\cdot)$). But we have

$$H_D(X) = h_D\left(\frac{1-\epsilon}{2}\right).$$

(Set, for example, $\epsilon = 3/8$, thus $\frac{1-\epsilon}{2} = 5/16$.)

Conditional entropy of X given Y

The most useful and impactful definition is the *average* conditional entropy of X given $Y = y$, averaged over all values of y under the marginal distribution $p_Y(y)$. Formally, we thus define:

Definition

The conditional entropy of X given Y is defined as

$$H_D(X|Y) \stackrel{def}{=} \sum_{y \in \mathcal{Y}} p_Y(y) \left(- \sum_{x \in \mathcal{X}} p_{X|Y}(x|y) \log_D p_{X|Y}(x|y) \right).$$

Conditional entropy of X given Y

Example ("Bit flipper channel")

For the Bit flipper channel, we have

$$H_D(X|Y) = p(Y=0)H_D(X|Y=0) + p(Y=1)H_D(X|Y=1).$$

We have already calculated

$$H_D(X|Y=0) = h_D(\epsilon) \quad \text{and} \quad H_D(X|Y=1) = h_D(\delta).$$

For example, when Y is uniform, we have

$$H_D(X|Y) = \frac{h_D(\epsilon) + h_D(\delta)}{2}.$$

THEOREM (BOUNDS ON CONDITIONAL ENTROPY OF X GIVEN Y)

The conditional entropy of a discrete random variable $X \in \mathcal{X}$ conditioned on Y satisfies

$$0 \leq H_D(X|Y) \leq \log_D |\mathcal{X}|,$$

with equality on the left iff for every y there exists and x such that $p_{X|Y}(x|y) = 1$, and with equality on the right iff $p_{X|Y}(x|y) = \frac{1}{|\mathcal{X}|}$ for all x and all y.

This follows directly from our bounds on $H_D(X|Y = y)$.

Note: Having $p_{X|Y}(x|y) = \frac{1}{|\mathcal{X}|}$ for all x and all y implies that X and Y are independent random variables.

EXERCISE ("BIT FLIPPER CHANNEL")

Verify the bounds for the bit-flipper channel.

Entropy Bounds: "Conditioning Reduces Entropy"

The following bound is important and impactful (and also intuitively pleasing!):

Theorem (Conditioning reduces Entropy)

For any two discrete random variables X and Y,

$$H_D(X|Y) \leq H_D(X)$$

with equality iff X and Y are independent random variables.

In words: **On average,** the uncertainty about X can only become smaller if we know Y.

Note Bene: As we have seen, this is *not true point-wise:* We may have $H_D(X|Y = y) > H_D(X)$ for some values of y.

Proof [Conditioning reduces entropy]:

$$H_D(X|Y) - H_D(X) = \mathbb{E}\left[\log_D \frac{1}{p_{X|Y}(X|Y)}\right] + \mathbb{E}[\log_D p_X(X)]$$

$$= \mathbb{E}\left[\log_D \frac{p_X(X)}{p_{X|Y}(X|Y)}\right]$$

$$= \mathbb{E}\left[\log_D \frac{p_X(X)p_Y(Y)}{p_{X|Y}(X|Y)p_Y(Y)}\right] = \mathbb{E}\left[\log_D \frac{p_X(X)p_Y(Y)}{p_{X,Y}(X,Y)}\right]$$

$$\overset{\text{(IT-Inequality)}}{\leq} \mathbb{E}\left[\frac{p_X(X)p_Y(Y)}{p_{X,Y}(X,Y)} - 1\right]\log_D(e)$$

$$= \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} \left[p_X(x)p_Y(y) - p_{X,Y}(x,y)\right]\log_D(e)$$

$$= [1-1]\log_D(e) = 0.$$

The condition for equality is $\frac{p_X(x)p_Y(y)}{p_{X,Y}(x,y)} = 1$ for all x and y, i.e., equality holds iff X and Y are independent random variables. □

Entropy Bounds: "Conditioning Reduces Entropy"

A generalization of the previous bound is also of interest to us:

Theorem (Conditioning reduces Entropy)

For any three discrete random variables X, Y and Z,

$$H_D(X|Y,Z) \leq H_D(X|Z)$$

with equality iff X and Y are conditionally independent random variables given Z (that is, if and only if $p(x,y|z) = p(x|z)p(y|z)$ for all x, y, z).

Proof [Conditioning reduces entropy, generalized version]:

$$H_D(X|Y,Z) - H_D(X|Z) = \mathbb{E}\left[\log_D \frac{1}{p_{X|Y,Z}(X|Y,Z)}\right] + \mathbb{E}[\log_D p_{X|Z}(X|Z)]$$

$$= \mathbb{E}\left[\log_D \frac{p_{X|Z}(X|Z)}{p_{X|Y,Z}(X|Y,Z)}\right]$$

$$= \mathbb{E}\left[\log_D \frac{p_{X|Z}(X|Z)p_{Y|Z}(Y|Z)p_Z(Z)}{p_{X|Y,Z}(X|Y,Z)p_{Y|Z}(Y|Z)p_Z(Z)}\right]$$

$$= \mathbb{E}\left[\log_D \frac{p_{X|Z}(X|Z)p_{Y|Z}(Y|Z)p_Z(Z)}{p_{X,Y,Z}(X,Y,Z)}\right]$$

$$\overset{\text{(IT-Inequality)}}{\leq} \mathbb{E}\left[\frac{p_{X|Z}(X|Z)p_{Y|Z}(Y|Z)p_Z(Z)}{p_{X,Y,Z}(X,Y,Z)} - 1\right]\log_D(e)$$

$$= \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}}\sum_{z\in\mathcal{Z}}\left[p_{X|Z}(x|z)p_{Y|Z}(y|z)p_Z(z) - p_{X,Y,Z}(x,y,z)\right]\log_D(e)$$

$$= [1-1]\log_D(e) = 0.$$

The condition for equality is $\frac{p_{X|Z}(X|Z)p_{Y|Z}(Y|Z)p_Z(Z)}{p_{X,Y,Z}(X,Y,Z)} = 1$ for all x, y, z, i.e., equality holds iff X and Y are conditionally independent random variables given Z. □

Entropy Bounds: "Conditioning Reduces Entropy"

Recall: When we simply write $H(X)$, suppressing the subscript D, then we mean $D = 2$.

Example

Let $X \in \{0, 1\}$ be uniformly distributed and let $Y = X$. Then

$$H(X|Y) = 0 \text{ and } H(X) = 1.$$

Example

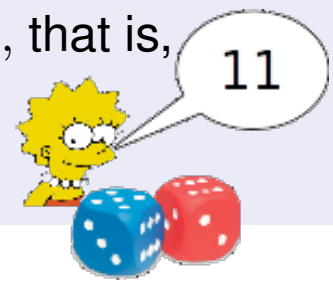
Let $X \in \{0, 1\}$ and $Y \in \{0, 1\}$ be uniformly distributed and independent. Then

$$H(X|Y) = 1 \text{ and } H(X) = 1.$$

EXERCISE (LISA ROLLS TWO DICE)

- Lisa rolls two dice and announces the sum L written as a two digit number.
- The alphabet of $L = L_1 L_2$ is $\{02, 03, 04, 05, 06, 07, 08, 09, 10, 11, 12\}$.
  - The alphabet of L_1 is $\{0, 1\}$.
  - The alphabet of L_2 is $\{0, 1, \ldots, 9\}$.
- Determine the probability that $L_2 = 2$, knowing that $L_1 = 1$, that is,

$$p_{L_2|L_1}(2|1).$$



SOLUTION

Using the definition (and calculations from Lecture 1),

$$p_{L_2|L_1}(2|1) = \frac{p_{L_1,L_2}(1,2)}{p_{L_1}(1)} = \frac{1/36}{1/6} = \frac{1}{6}.$$

LISA ROLLS TWO DICE

After running over all possible values for (i, j), we obtain

$L_2 = j$	$L_1 = i$ / $p_{L_1,L_2}(i,j)$	0	1	$p_{L_2}(j)$
0		0	3/36	3/36
1		0	2/36	2/36
2		1/36	1/36	2/36
3		2/36	0	2/36
4		3/36	0	3/36
5		4/36	0	4/36
6		5/36	0	5/36
7		6/36	0	6/36
8		5/36	0	5/36
9		4/36	0	4/36
	$p_{L_1}(i)$	5/6	1/6	

$L_2 = j$	$p_{L_2\|L_1}(j\|0)$	$p_{L_2\|L_1}(j\|1)$
0	0	3/6
1	0	2/6
2	1/30	1/6
3	2/30	0
4	3/30	0
5	4/30	0
6	5/30	0
7	6/30	0
8	5/30	0
9	4/30	0

Lisa rolls two dice

Example

$$H(L_2|L_1=1) = \frac{3}{6}\log\frac{6}{3} + \frac{2}{6}\log\frac{6}{2} + \frac{1}{6}\log 6$$
$$= 1.459 \text{ bits}$$
$$H(L_2|L_1=0) = \cdots = 2.857 \text{ bits}$$

$L_2 = j$	$p_{L_2\|L_1}(j\|0)$	$p_{L_2\|L_1}(j\|1)$
0	0	3/6
1	0	2/6
2	1/30	1/6
3	2/30	0
4	3/30	0
5	4/30	0
6	5/30	0
7	6/30	0
8	5/30	0
9	4/30	0

EXAMPLE

$$H(L_2|L_1) = p_{L_1}(0)H(L_2|L_1 = 0) + p_{L_1}(1)H(L_2|L_1 = 1)$$
$$= \frac{5}{6} \times 2.857 + \frac{1}{6} \times 1.459 = 2.624 \text{ bits}$$

Now, we can observe that

$$2.624 = H(L_2|L_1) \leq H(L_2) = 3.22,$$

exactly like it has to be according to our theorems.

The Chain Rule for Entropy

Recall that the joint entropy of two random variables X, Y is completely naturally defined as

$$H_D(X, Y) = -\sum_x \sum_y p_{X,Y}(x, y) \log_D p_{X,Y}(x, y).$$

Using the fact that $p_{X,Y}(x, y) = p_X(x)p_{Y|X}(y|x)$, we can write this as

$$\begin{aligned} H_D(X, Y) &= -\sum_x p_X(x) \left(\sum_y p_{Y|X}(y|x) \log_D \left(p_X(x)p_{Y|X}(y|x) \right) \right) \\ &= -\sum_x p_X(x) \left(\sum_y p_{Y|X}(y|x) \left(\log_D p_X(x) + \log_D p_{Y|X}(y|x) \right) \right) \\ &= -\sum_x p_X(x) \left\{ \left(\sum_y p_{Y|X}(y|x) \log_D p_X(x) \right) \right. \\ &\qquad \left. + \left(\sum_y p_{Y|X}(y|x) \log_D p_{Y|X}(y|x) \right) \right\} \end{aligned}$$

But now, we observe:

$$
\begin{aligned}
H_D(X,Y) &= -\sum_x p_X(x) \left\{ \left(\sum_y p_{Y|X}(y|x) \log_D p_X(x) \right) \right. \\
&\qquad \left. + \left(\sum_y p_{Y|X}(y|x) \log_D p_{Y|X}(y|x) \right) \right\} \\
&= \underbrace{-\sum_x p_X(x) \left(\sum_y p_{Y|X}(y|x) \log_D p_X(x) \right)}_{H_D(X)} \\
&\qquad + \underbrace{\sum_x p_X(x) \left(-\sum_y p_{Y|X}(y|x) \log_D p_{Y|X}(y|x) \right)}_{H_D(Y|X)} \\
&= H_D(X) + H_D(Y|X).
\end{aligned}
$$

Let us write this once more and enjoy it properly:

$$H_D(X, Y) = H_D(X) + H_D(Y|X).$$

In words: To find the joint entropy of two random variables, we can first calculate the entropy of one of the two, and then add to it the conditional entropy of the second, given the first.

Of course, what we could do once, we can do again!

THEOREM (CHAIN RULE FOR ENTROPIES)

Let $S_1, \ldots, S_n$ be discrete random variables. Then

$$H_D(S_1, S_2, \ldots, S_n) = H_D(S_1) + H_D(S_2|S_1) + \cdots + H_D(S_n|S_1, \ldots, S_{n-1}).$$

The above result says that the uncertainty of a collection of random variables (in any order) is the uncertainty of the first, plus the uncertainty of the second when the first is known, plus the uncertainty of the third when the first two are known, etc.

Proof [Chain rule for entropy]:

$$p_{S_1,S_2,\ldots,S_n}(s_1,\ldots,s_n) = p_{S_1}(s_1)\prod_{i=2}^{n} p_{S_i|S_1,\ldots,S_{i-1}}(s_i|s_1,\ldots,s_{i-1})$$

$$-\log_D\left(p_{S_1,S_2,\ldots,S_n}(s_1,\ldots,s_n)\right) = -\log p_{S_1}(s_1) - \sum_{i=2}^{n} \log_D\left(p_{S_i|S_1,\ldots,S_{i-1}}(s_i|s_1,\ldots,s_{i-1})\right)$$

The expected value of the LHS is $H_D(S_1, S_2, \ldots, S_n)$.

The expected value of the RHS is
$H_D(S_1) + H_D(S_2|S_1) + \cdots + H_D(S_n|S_1,\ldots,S_{n-1})$. □

EXAMPLE

Let X, Y, Z be discrete random variables. We have:

$$\begin{aligned} H(X,Y,Z) &= H(X) + H(Y|X) + H(Z|X,Y) \\ &= H(X) + H(Z|X) + H(Y|X,Z) \\ &= H(Y) + H(X|Y) + H(Z|X,Y) \\ &= H(Y) + H(Z|Y) + H(X|Y,Z) \\ &= H(Z) + H(X|Z) + H(Y|X,Z) \\ &= H(Z) + H(Y|Z) + H(X|Y,Z), \end{aligned}$$

where we omitted the subscript D for compact notation, but these relationships hold for all integers $D \geq 2$.

The chain rule for entropy and the fact that conditioning reduces entropy, proves the following theorem which was stated last week without proof:

THEOREM

Let $S_1, \ldots, S_n$ be discrete random variables. Then

$$H(S_1, S_2, \ldots, S_n) \leq H(S_1) + H(S_2) + \cdots + H(S_n),$$

with equality iff $S_1, \ldots, S_n$ are independent.

Sometimes it is convenient to compute the conditional entropy using the chain rule for entropies. For instance:

$$H(X|Y) = H(X, Y) - H(Y).$$

COROLLARY

$$H(X, Y) \geq H(X);$$
$$H(X, Y) \geq H(Y).$$

The above inequalities follow from the chain rule for entropies and the fact that entropy (conditional or not) is nonnegative.

EXAMPLE (LISA ROLLS TWO DICE)

From

$$H(L_1, L_2) = 3.2744 \text{ bits}$$
$$H(L_1) = 0.6500 \text{ bits}$$
$$H(L_2) = 3.2188 \text{ bits},$$

we compute

$$H(L_2|L_1) = H(L_1, L_2) - H(L_1) = 3.2744 - 0.6500 = 2.624 \text{ bits}$$
$$H(L_1|L_2) = H(L_1, L_2) - H(L_2) = 3.2744 - 3.2188 = 0.056 \text{ bits},$$

and verify that indeed

$$H(L_1|L_2) \leq H(L_1) \leq H(L_1, L_2)$$
$$H(L_2|L_1) \leq H(L_2) \leq H(L_1, L_2).$$

EXERCISE

Determine $H(L_1, L_2|S_1, S_2)$.

SOLUTION

L_1 and L_2 are deterministic functions of S_1 and S_2.
Hence $H(L_1, L_2|S_1, S_2) = 0$.

LISA ROLLS TWO DICE

EXAMPLE

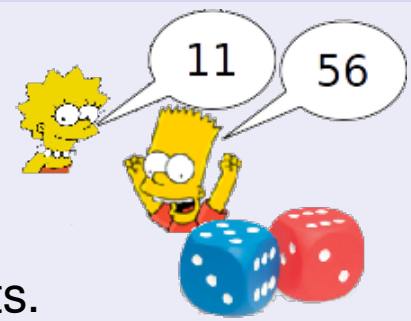
Determine $H(S_1, S_2|L_1, L_2)$ knowing that $H(S_1, S_2) = 5.1699$ bits and $H(L_1, L_2) = 3.2744$ bits.

SOLUTION

$H(S_1, S_2|L_1, L_2) = H(S_1, S_2, L_1, L_2) - H(L_1, L_2)$.

But $H(S_1, S_2, L_1, L_2) = H(S_1, S_2)$. (Can you say why?)

Hence $H(S_1, S_2|L_1, L_2) = H(S_1, S_2) - H(L_1, L_2) = 1.896$ bits.



Definition (Coin-Flip Source)

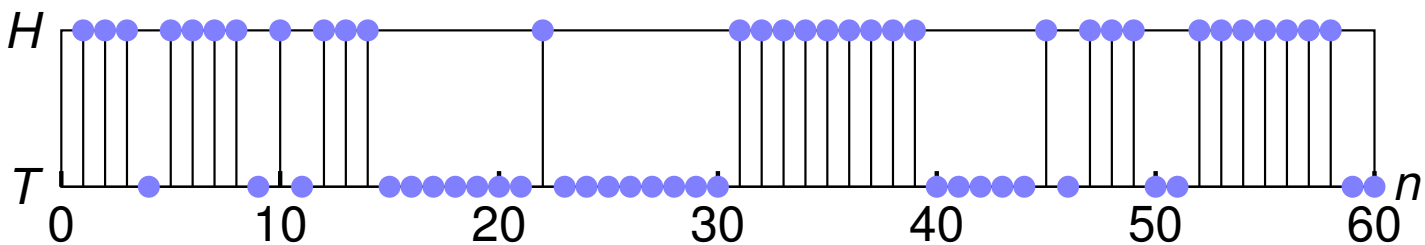
The source models a sequence $S_1, S_2, \ldots, S_n$ of n coin flips.

So $S_i \in \mathcal{A} = \{H, T\}$, where H stands for heads, T for tails, $i = 1, 2, \ldots, n$.

$p_{S_i}(H) = p_{S_i}(T) = \frac{1}{2}$ for all i, and coin flips are independent.

Hence,

$$p_{S_1,S_2,\ldots,S_n}(s_1, s_2, \ldots, s_n) = \frac{1}{2^n} \quad \text{for all } (s_1, s_2, \ldots, s_n) \in \mathcal{A}^n$$

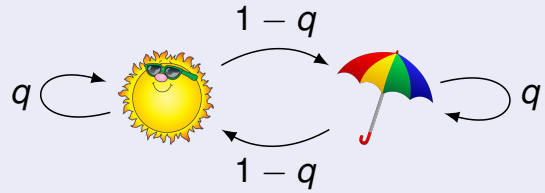


Definition (Sunny-Rainy Source)

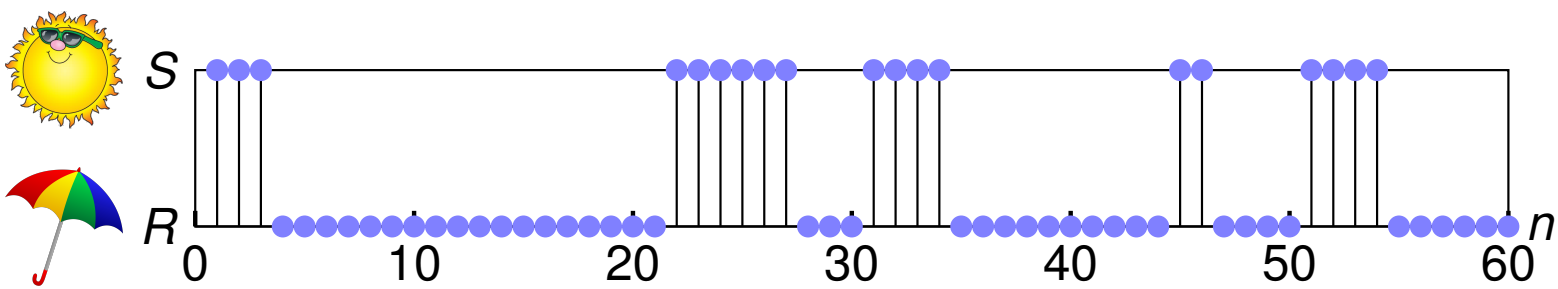
The source models a sequence $S_1, S_2, \ldots, S_n$ of weather conditions.

So $S_i \in \mathcal{A} = \{S, R\}$, where S stands for sunny, R for rainy, $i = 1, 2, \ldots, n$.

The weather on the first day is uniformly distributed in $\mathcal{A}$.



For all other days, with probability $q = \frac{6}{7}$ the weather is as for the day before.



Example

For the Sunny-Rainy source:

- $p_{S_1}(S) = \frac{1}{2}$
- $p_{S_1,S_2}(R,R) = p_{S_1}(R)p_{S_2|S_1}(R|R) = \frac{1}{2}q$
- $p_{S_1,S_2}(R,S) = p_{S_1}(R)p_{S_2|S_1}(S|R) = \frac{1}{2}(1-q)$
- $p_{S_1,S_2,S_3,S_4}(R,S,S,R) = \frac{1}{2}(1-q)q(1-q) = \frac{1}{2}q(1-q)^2$

In general, if c is the number of weather changes ($0 \leq c \leq n-1$), then

$$p_{S_1,S_2,\ldots,S_n}(s_1,s_2,\ldots,s_n) = \frac{1}{2}q^{n-1-c}(1-q)^c.$$

EXERCISE

Let $i = 2, 3, \ldots$

For the Sunny-Rainy source:

- Find $p_{S_i}(s_i)$
- Find $p_{S_i|S_{i-1}}(s_i|s_{i-1})$
- Are S_i and S_{i-1} independent?

Solution (Sunny-Rainy source)

By definition, $p_{S_i|S_{i-1}}(j|k) = q$ if $j = k$ and $(1-q)$ otherwise.

Hence S_{i-1} and S_i are not independent.

To determine the statistic of the marginals, we use the law of total probability and induction to show that p_{S_i} is uniform.

It is true by definition for $i = 1$.

Suppose that p_{S_i} is uniform for $i = 1, \ldots, n-1$. We show that it is uniform also for $i = n$:

$$p_{S_n}(j) = \sum_{k \in \{S,R\}} p_{S_n|S_{n-1}}(j|k) p_{S_{n-1}}(k) = \frac{1}{2} \sum_{k \in \{S,R\}} p_{S_n|S_{n-1}}(j|k)$$
$$= \frac{1}{2}(q + (1-q)) = \frac{1}{2}.$$

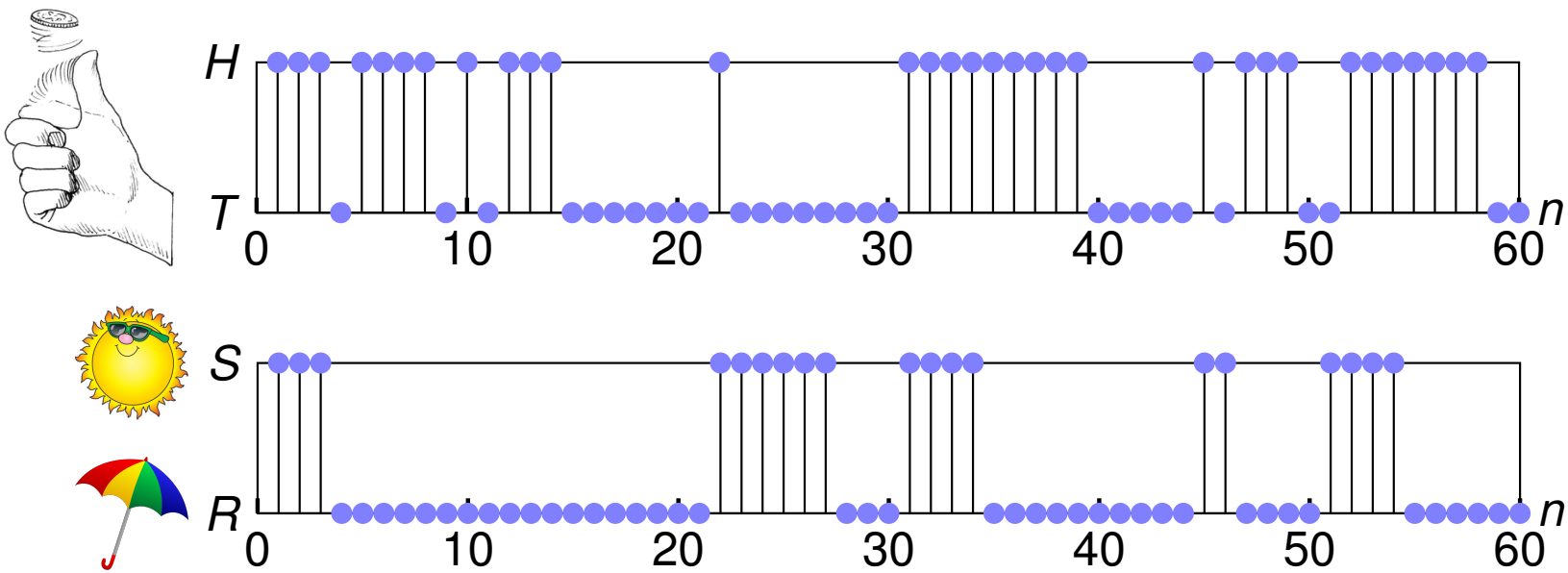
Hence the marginals are uniformly distributed (like for the Coin-Flip source).

EXERCISE

Let $i = 2, 3, \ldots$

For the Coin-Flip (CF) and Sunny-Rainy (SR) sources:

- Compute $H(S_i)$
- Compute $H(S_i|S_1, \ldots, S_{i-1})$



Solution ($H(S_i)$)

The entropy depends only on the distribution, and for a uniform distribution, it is the log of the alphabet's cardinality. Hence

$$H_{CF}(S_i) = H_{SR}(S_i) = \log 2 = 1$$

Solution ($H(S_i|S_1, \ldots, S_{i-1})$ for the Coin-Flip source)

S_i is independent of $S_1, \ldots, S_{i-1}$

Hence, $H(S_i|S_1, \ldots, S_{i-1}) = H(S_i)$.

Solution ($H(S_i|S_1, \ldots, S_{i-1})$ for the Sunny-Rainy source)

S_i depends only on S_{i-1}. Hence

$$H_{SR}(S_i|S_1 = s_1, \ldots, S_{i-1} = s_{i-1}) = H_{SR}(S_i|S_{i-1} = s_{i-1}).$$

When $S_{i-1} = k \in \{S, R\}$, the probabilities for S_i are q and $(1 - q)$. Hence

$$H_{SR}(S_i|S_{i-1} = s_{i-1}) = -q \log q - (1 - q) \log(1 - q).$$

Taking the average on both sides yields

$$H_{SR}(S_i|S_{i-1}) = -q \log q - (1 - q) \log(1 - q).$$

For $q = \frac{6}{7}$, we have

$$H_{SR}(S_i|S_{i-1}) = -q \log q - (1 - q) \log(1 - q) = 0.592.$$

Exercise

Determine $H(S_1, S_2, \ldots, S_n)$ for the Coin-Flip source.

Solution

The source produces independent and identically distributed symbols. Hence

$$H(S_1, S_2, \ldots, S_n) \overset{\text{(indep.)}}{=} H(S_1) + H(S_2) + \cdots + H(S_n) \overset{\text{(identically distributed)}}{=} nH(S_1)$$

Moreover, the distribution is uniform, therefore $H(S_1) = 1$ bit. Putting things together,

$$H(S_1, S_2, \ldots, S_n) = n \text{ bits}$$

EXERCISE

Determine $H(S_1, S_2, \ldots, S_n)$ for the Sunny-Rainy source with $q = \frac{6}{7}$.

SOLUTION

$$H(S_1, S_2, \ldots, S_n) = H(S_1) + H(S_2|S_1) + \cdots + H(S_n|S_1, \ldots, S_{n-1})$$

For $i = 2, 3, \ldots, n$, the statistic of S_i depends only on S_{i-1}. Hence

$$H(S_i|S_1, S_2, \ldots, S_{i-1}) = H(S_i|S_{i-1})$$

$$H(S_1, S_2, \ldots, S_n) = H(S_1) + H(S_2|S_1) + \cdots + H(S_n|S_{n-1})$$

We have already determined that $H(S_1) = 1$ bit and $H(S_i|S_{i-1}) = 0.592$ bits. Therefore

$$H(S_1, S_2, \ldots, S_n) = 1 + 0.592(n-1) \text{ bits}$$

Summary : The Main Result of Source Coding / Data Compression

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

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

- And in many cases, as n becomes large, the upper and the lower bound are arbitrarily close!

Additional Questions of interest include:

- What if the source alphabet is not finite?
- What if we do not know the source distribution $p_X(x)$? (Universal source coding)

What if the source alphabet is infinite?

- In all of our previous discussion on actual codes, we have assumed that the source alphabet is discrete and finite.
- What if it is discrete but infinite?
- ... is this just an academic endeavour?
- In this class, we only touch the top of this iceberg...

BINARY PREFIX-FREE CODE FOR POSITIVE INTEGERS

The set of positive integers is infinite and no probability is assigned to its elements. Hence we cannot use Huffman's construction to encode integers.

First Attempt to Encode Positive Integers: "Standard Method"

n	$c(n)$
1	1
2	10
3	11
4	100
5	101
$\vdots$	$\vdots$

The code is not prefix-free.

The length of $c(n)$ is $l(n) = \lfloor \log_2 n \rfloor + 1$.

Note: The first digit is always 1.

Second Attempt: “Elias Code 1"

We prefix code $c(n)$ with $l(n) - 1$ zeros.

n	$c_1(n)$
1	1
2	010
3	011
4	00100
5	00101
$\vdots$	$\vdots$

The code is prefix-free. (Codewords of different length cannot have the same number of leading zeros.)

The length of $c_1(n)$ is

$$l_1(n) = l(n) - 1 + l(n) = 2\lfloor \log_2 n \rfloor + 1.$$

Note: we are essentially doubling the length to make the code prefix-free.

Third Attempt: “Elias Code 2"

Instead of $l(n) - 1$ zeros followed by a 1, we prefix with $c_1\big(l(n)\big)$, which is also prefix-free (hence can be identified). Like the zeros, it tells the length of the codeword.

Notation: $\tilde{c}(n)$ is $c(n)$ without the leading 1.

n	$c(n)$	$l(n)$	$c_1(n)$	$c_1\big(l(n)\big)\tilde{c}(n)$
1	1	1	1	$c_1(1) = 1$
2	10	2	010	$c_1(2)\,0 = 010\,0$
3	11	2	011	$c_1(2)\,1 = 010\,1$
4	100	3	00100	$c_1(3)\,00 = 011\,00$
5	101	3	00101	$c_1(3)\,01 = 011\,01$
⋮	⋮			

The code is prefix-free.

The codeword length is
$l_2(n) = l_1\big(l(n)\big) + l(n) - 1 = 2\lfloor \log_2(\lfloor \log_2 n \rfloor + 1) \rfloor + 1 + \lfloor \log_2 n \rfloor$.

What if the source distribution is not known?

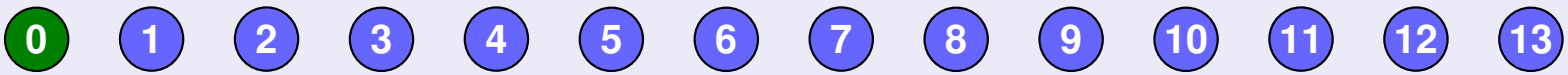
- Universal source coding.

- Practically important algorithms: “Lempel-Ziv” (LZ77, LZ78). Time permitting, we briefly discuss how they work. An analysis is beyond the scope of AICC-2.

Challenge for next lecture

Exercise

There are 14 billiard balls numbered as shown:



Among balls 1 - 13, at most one **could** be heavier/lighter than the others.

What is the minimum number of weightings to simultaneously determine:

- if one ball is different . . .
- if there is such a ball, which one, . . .
- and whether the different ball is heavier/lighter.

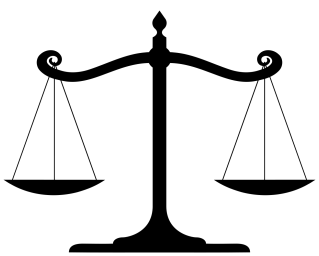
Week 4, Part 1:
Entropy and Algorithms

Prof. Michael Gastpar

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

EPFL

Spring Semester 2025



Outline

Entropy and Algorithms

In today's lecture, we explore the role of entropy in algorithms beyond data compression.

Specifically, we will briefly look at the following examples:

- "20 Questions Problem"
- Sorting
- "Billiard Balls" Puzzle

THE 20 QUESTIONS PROBLEM

Let X be a random variable.

What is the minimum number of "Yes/No questions" needed to identify X?

And which questions should be asked?

SOLUTION

- Consider a binary code Γ for $X \in \mathcal{X}$.
- Once Γ is fixed, we know $x \in \mathcal{X}$ iff we know the codeword $\Gamma(x)$.
- The strategy consists in asking the ith question so as to obtain the ith bit of the codeword $\Gamma(x)$.
- The average number of questions is $L(X, \Gamma)$, which is minimized if Γ is the encoding map of a Huffman code.
- We will see that we cannot do better.

First an example.

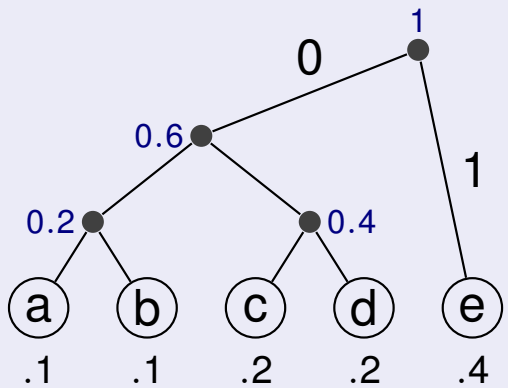
EXAMPLE

Let X be a random variable in $\mathcal{A} = \{a, b, c, d, e\}$ having distribution p_X:

X	a	b	c	d	e
p_X	0.1	0.1	0.2	0.2	0.4

Example (cont.)

We construct a binary Huffman code for X:



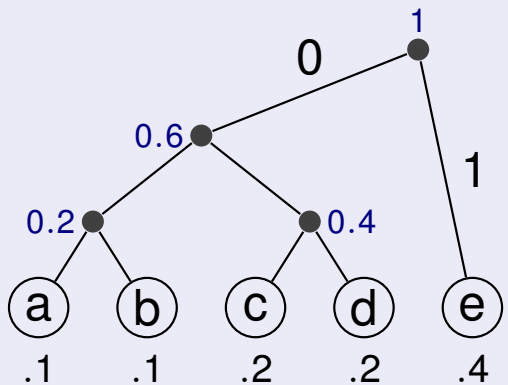
$\mathcal{A}$	Γ_H
a	000
b	001
c	010
d	011
e	1

Suppose that the realization is $X = b$ (but we do not know it).

Example (cont.)

The strategy is to identify b via its binary codeword.

With the first question we try to find the first letter of the codeword:



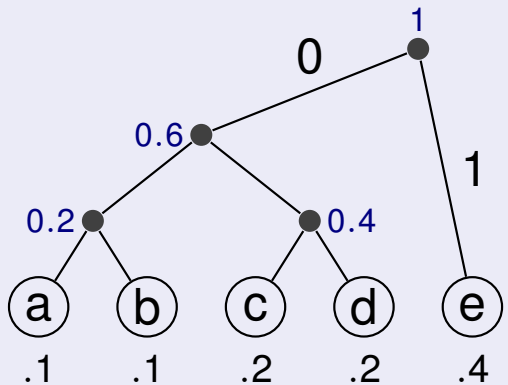
$\mathcal{A}$	Γ_H
a	000
b	001
c	010
d	011
e	1

We ask the question: Is $X \in \{e\}$?

The answer is NO. We know that the first letter of the codeword is 0.

Example (cont.)

With the second question we find the second codeword letter:



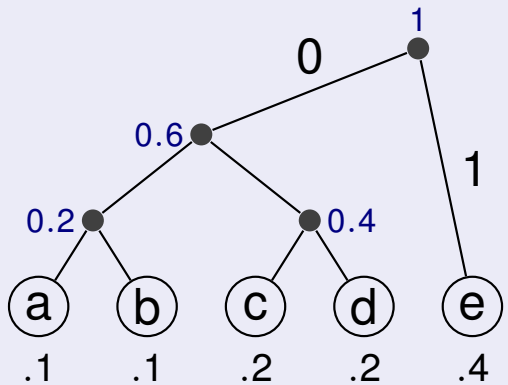
$\mathcal{A}$	Γ_H
a	000
b	001
c	010
d	011
e	1

We ask the question: Is $X \in \{c, d\}$?

The answer is NO. We know that the second letter of the codeword is 0.

Example (cont.)

With the third question we find the third codeword letter:



$\mathcal{A}$	Γ_H
a	000
b	001
c	010
d	011
e	1

We ask the question: is $X = b$?

The answer is Yes. We know that the third letter of the codeword is 1 and that $X = b$.

Optimality of the Huffman querying strategy

We have seen that a prefix-free code for $X \in \mathcal{X}$ leads to a querying strategy to find the realization of X.

Similarly, a deterministic querying strategy leads to a binary prefix-free code for X. Here is why:

- Before the first question we know that $x \in \mathcal{X}$.
- Without loss of generality, the first question can be formulated in terms of "Is $x \in \mathcal{A}$?" for some $\mathcal{A} \subset \mathcal{X}$. (The choice of $\mathcal{A}$ is determined from the strategy, that we fix once and for all.)
- If the answer is YES, then we know that $x \in \mathcal{A} \subset \mathcal{X}$. Otherwise $x \in \mathcal{A}^c \subset \mathcal{X}$. Either way we have reduced the size of the set that contains x.
- We continue asking similar questions until the value of x is fully determined, then we stop.

- The sequence of YES/NO answers is a binary codeword associated to x.
- The code obtained when we consider all possible values of x is a binary prefix-free code.
- Since the tree is prefix-free, its average codeword-length cannot be smaller than that of a Huffman code.

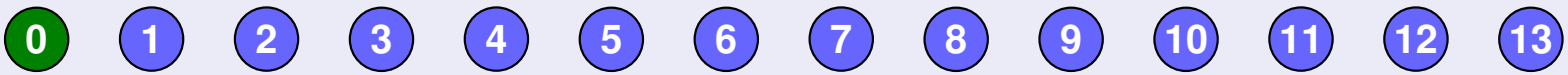
Entropy and Sorting

- Sorting by pairwise comparisons with binary output.
  That is, for each comparison, the answer is either " $<$ " or " $\geq$ ".
- Suppose we have an unordered list of n objects that need to be sorted.
- How many binary comparisons are needed?
- Let us tackle this question via entropy.

Billiard Balls

Exercise

There are 14 billiard balls numbered as shown:



Among balls 1 - 13, at most one **could** be heavier/lighter than the others.

What is the minimum number of weightings to simultaneously determine:

- if one ball is different . . .
- if there is such a ball, which one, . . .
- and whether the different ball is heavier/lighter.

EXERCISE

Can we use the 20 questions approach to solve the 14 billiard balls riddle?

SOLUTION

No, because the kind of questions that we can "ask", when we are weighing, is quite limited.

For instance, the first question cannot be "is 1 or 2 heavy?"

BILLIARD BALLS

Here is a way to formalize the Billiard Balls problem.

Let us identify the solution with the variable X :

"all balls equal"	$X = 0$
"Ball 1 heavy"	$X = 1$
"Ball 1 light"	$X = -1$
"Ball 2 heavy"	$X = 2$
"Ball 2 light"	$X = -2$
$\vdots$	$\vdots$
"Ball 13 heavy"	$X = 13$
"Ball 13 light"	$X = -13$

Hence, $X \in \mathcal{X} = \{ -13, -12, \ldots, -1, 0, 1, \ldots, 12, 13\}$.

Evidently, $|\mathcal{X}| = 27$.

The results of the weighings uniquely specify the value of X.

Hence, in the billiard balls problem, we implicitly specify a ternary code for a certain source.

We know that the number of ternary symbols needed to represent the source is *at least*

$$N \geq H_{D=3}(X).$$

Our code should work *irrespective of the source distribution.* In other words, it must work for all source distributions, thus

$$N \geq \max_{p(x)} H_{D=3}(X) = \log_3 |\mathcal{X}| = \log_3 27 = 3.$$

Hence, conclusion: **We need at least 3 weighings.**

Billiard Balls : Strategies

But is there a strategy that requires only 3 weighings?

From source compression, we can establish the following facts:

- For each weighing, the three outcomes must be *equally likely*.
- The weighings must be independent of each other.

In class, we will together formally prove these two statements.

Then, leveraging these two insights, we will construct the weighing strategy.

Remark: It is because we carefully selected the numbers (alphabet size of 27; each weighing has 3 possible outcomes) that there is a strategy that *exactly* matches the entropy lower bound of 3 weighings. If you change the numbers, it will not generally be true that there is a strategy that *exactly* matches the lower bound.

Week 4, Part 2:
Prediction, Learning, and Cross-Entropy Loss

Prof. Michael Gastpar

Slides by Prof. M. Gastpar

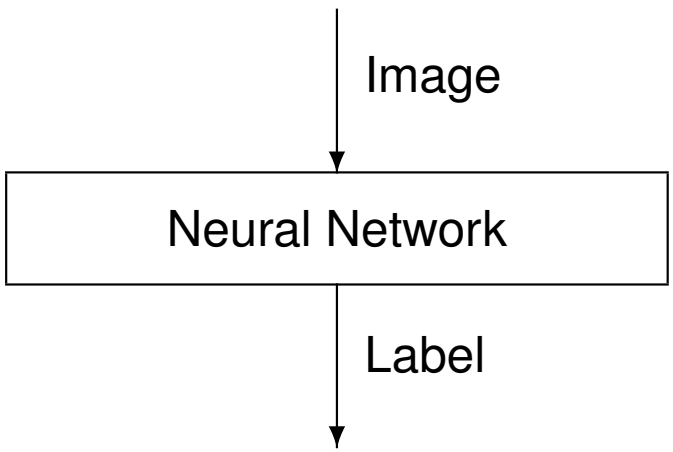
Spring Semester 2025

Outline

Prediction, Learning, and Cross-Entropy Loss

In today's lecture, we explore the role of entropy in prediction and learning problems.

Example : Classify Images



Label	*Probability*
Ibex	0.98
Kangaroo	0.005
Lynx	0.002
Wombat	0.002
Dog	0.001
Cat	0.001
Turtle	0.001
Dolphin	0.001
Elephant	0.001
Kookaburra	0.001
Other	0.005

Example : Classify Images

- Our Neural Network produces

$$P_{machine}(label|image).$$

- The true label distribution is

$$P_{true}(label|image) = \begin{cases} 1, & \text{correct label,} \\ 0, & \text{wrong label.} \end{cases}$$

(assuming for simplicity that for each image, there is a single correct label).

- Ideally, we would like

$$P_{machine}(label|image) = P_{true}(label|image)$$

for *every* pair $(image, label)$.

- Clearly, this is not going to happen in the real world!

Example : Classify Images

- Instead, people like to consider **cross entropy loss**.
- That is, we wish for our $P_{machine}(label|image)$ to **minimize**

$$\begin{aligned} & L(P_{true}(label|image), P_{machine}(label|image)) \\ & = -\sum_{label} P_{true}(label|image) \log_D P_{machine}(label|image) \end{aligned}$$

- Given training data $(image_i, label_i)$, for $i = 1, 2, \ldots, n$, we select $P_{machine}(label|image)$ to minimize the cross entropy loss.

Cross Entropy Loss

- Cross Entropy Loss:

$$L(P, Q) = -\sum_{y} P(y) \log_D Q(y).$$

  where
  - P is the true distribution
  - Q is our approximation (via the neural network).

Why is it popular?

- Good properties for training with "gradient descent" in certain standard architectures.
- Theoretical properties.

We will now discuss these in turn.

Example : Classify Images, Training with cross entropy loss

- The Neural Network takes in an image. Let us call this x.
- It outputs a label distribution $Q(y|x)$ over the set of labels.
- Let us restrict to just two labels. Only "Ibex" ($y = 0$) and "Kangaroo" ($y = 1$).
- In simplified terms, the Neural Network outputs:

$$Q(y = 0|x) = \frac{e^{z_0}}{e^{z_0} + e^{z_1}},$$
$$Q(y = 1|x) = \frac{e^{z_1}}{e^{z_0} + e^{z_1}} = 1 - Q(y = 0|x).$$

  where

$$z_0 = w_0 x + b_0$$
$$z_1 = w_1 x + b_1$$

  where w_0 and w_1 are called *weights* and b_0 and b_1 are called *biases*.
- The key is to select the weights and biases cleverly.
- This is done by *training* with data.

EXAMPLE : CLASSIFY IMAGES, TRAINING WITH CROSS ENTROPY LOSS

Label = 0
("Ibex")



Label = 1
("Kangaroo")



Example : Classify Images, Training with cross entropy loss

- For fixed weights and biases, calculate the loss over all n training samples:

$$L_{training} = -\sum_{i=1}^{n} \left(P(y=0|x_i) \log_D \frac{e^{z_{0,i}}}{e^{z_{0,i}} + e^{z_{1,i}}} + P(y=1|x_i) \log_D \frac{e^{z_{1,i}}}{e^{z_{0,i}} + e^{z_{1,i}}} \right)$$

  where $z_{0,i} = w_0 x_i + b_0$ and $z_{1,i} = w_1 x + b_1$.

- Suppose images $i = 1, 2, \ldots, k$ are ibexes (label 0), and images $i = k+1, k+2, \ldots, n$ are kangaroos (label 1). Then, we can write

$$L_{training} = -\sum_{i=1}^{k} \log_D \frac{e^{w_0 x_i + b_0}}{e^{w_0 x_i + b_0} + e^{w_1 x_i + b_1}} - \sum_{i=k+1}^{n} \log_D \frac{e^{w_1 x_i + b_1}}{e^{w_0 x_i + b_0} + e^{w_1 x_i + b_1}}$$

- Now minimize this over all weights and biases!

Example : Classify Images, Training with cross entropy loss

$$L_{training} = -\sum_{i=1}^{k} \log_D \frac{e^{w_0 x_i + b_0}}{e^{w_0 x_i + b_0} + e^{w_1 x_i + b_1}} - \sum_{i=k+1}^{n} \log_D \frac{e^{w_1 x_i + b_1}}{e^{w_0 x_i + b_0} + e^{w_1 x_i + b_1}}$$

- Find gradient (derivative) with respect to weights (and biases).
- Most commonly, *gradient descent* is used.
  - Start with a **random choice** of the weights.
  - Then, proceed in "small" steps against the gradient.

Cross Entropy Loss : Theoretical Properties

- Cross Entropy Loss:

$$L(P, Q) = -\sum_{y} P(y) \log_D Q(y).$$

Theorem

For a fixed probability distribution P, the minimum

$$\min_{Q} L(P, Q)$$

is attained if and only if we select $Q^* = P$, and in this case,

$$L(P, Q^*) = L(P, P) = H(P),$$

where $H(P)$ is the entropy of the probability distribution P.

The proof, which will be done in class, uses once again the "IT inequality."

Outline

Entropy:

$$H_D(X) = -\sum_x p(x) \log_D p(x).$$

For $D = 2$, we simply write $H(X)$, and we call the unit *bits.*

Entropy has many useful properties, including:

- $0 \leq H_D(X) \leq \log_D |\mathcal{X}|$
- $H_D(X|Y) \leq H_D(X)$ with equality if and only if X and Y are independent.
- $H_D(X, Y) = H_D(X) + H_D(Y|X)$

Data Compression:

- Every uniquely decodable binary code must use at least $H(X)$ bits per symbol on average.
- There exists a binary code that uses between $H(X)$ and $H(X)+1$ bits per symbol on average.
- Hence, for a source string of length n :
  - every uniquely decodable binary code must use at least $H(S_1, S_2, \cdots, S_n)/n$ bits per source symbol, and
  - there exists a binary code that uses between $H(S_1, S_2, \cdots, S_n)/n$ and $H(S_1, S_2, \cdots, S_n)/n + 1/n$ bits per source symbol.

Entropy and Algorithms

- We explored examples where entropy can give a lower bound on algorithmic performance.
  - *Example:* in search-type problems, give a lower bound on the minimum number of necessary queries.

Cross-Entropy Loss

- Machine (e.g., Neural Network) outputs a distribution $Q(y)$ over all possible labels.
- Cross-Entropy Loss: Select $Q(y)$ to minimize $L(P, Q) = -\sum_y P(y) \log_D Q(y)$.

Outline

Week 5: Introduction to Cryptography
One-Time Pad, Perfect Secrecy, and Public-Key Cryptography (Diffie-Hellman)
(Textbook Chapter 6)

Prof. Michael Gastpar

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

EPFL

Spring Semester 2025

Outline

Why Cryptography

Cryptography serves two purposes:

- Privacy: Preventing that sensitive information lands in the wrong hands.
- Authenticity: Preventing that information is falsified.

Before the Internet:

- Cryptography was essentially a tool for diplomats and generals.
- Common people would sign a letter (for authenticity), put it in an envelope (for privacy) and trust the postal service for the delivery to the intended recipient (reliability).

The Internet has changed that:

- Now we send sensitive information over public channels on a daily basis. We need to control who can decipher such information (privacy). People and businesses can be destroyed if private information leaks out.

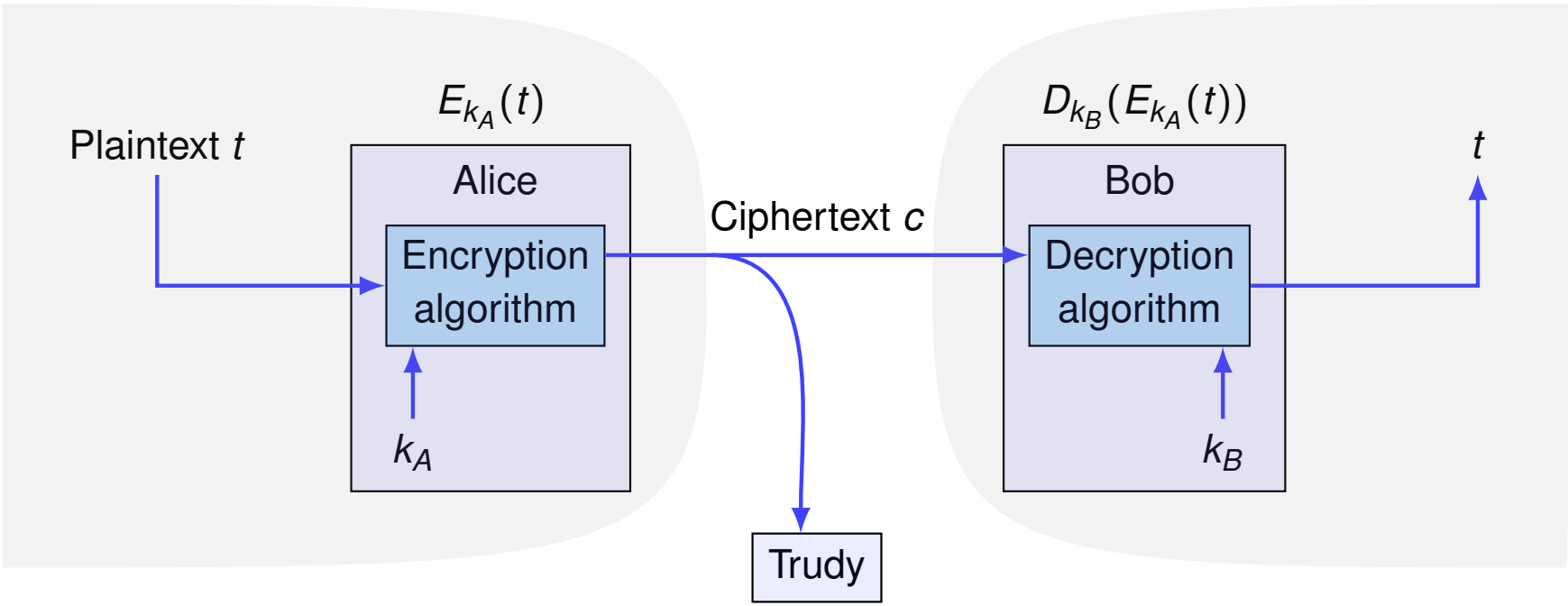
- We have the ability to post information that can be read by anybody — hence that can have a huge impact. We need to be able to verify who is posting (authenticity). People and businesses can be destroyed if information is falsified.

Cryptography gives us the tools to:

- authenticate the sender and the receiver
- verify the integrity of the message
- keep the message confidential

All these problems are related. Our initial focus is on how to keep a message confidential.

BASIC SETUP FOR CONFIDENTIALITY



Alice wants to sent the plaintext t to Bob:

- She encrypts t using her key k_A. The result is the ciphertext $c = E_{k_A}(t)$.
- She sends c to Bob over a public channel.
- Bob decrypts c using his key k_B. The result is $D_{k_B}(E_{k_A}(t)) = t$.
- For Trudy, it is nearly impossible to recover t from c without knowing k_B.

Basic Terminology

- plaintext, ciphertext (also called cryptogram), key, encrypter, decrypter: already defined.
- cryptography: the art of composing cryptograms.
- cryptanalysis: the art of breaking cryptograms.
- a cryptanalyst has broken the system when he can quickly determine the plaintext from the cryptogram, no matter what key is used.
- attacker: same as cryptanalyst.

Caesar's Cipher (Julius Caesar (1st century BC))

Suppose that we are using the English alphabet augmented by a few special characters, say "space", "comma", and "period".

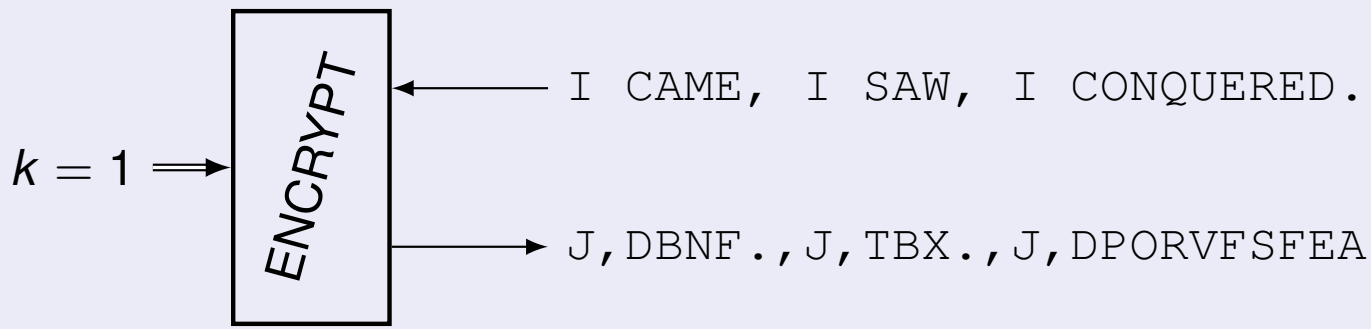
An alphabet of 29 characters, represented by the integers $0, 1, \ldots, 28$.

- the key k is an integer between 0 and 28, known to Alice and Bob and to nobody else.
- the encryption algorithm substitutes the i-th letter of the alphabet with the $(i + k)$-th letter $\pmod{29}$.
- the decryption algorithm substitutes the j-th letter with the $(j - k)$-th $\pmod{29}$.

Example (Caesar's cipher)

The alphabet is

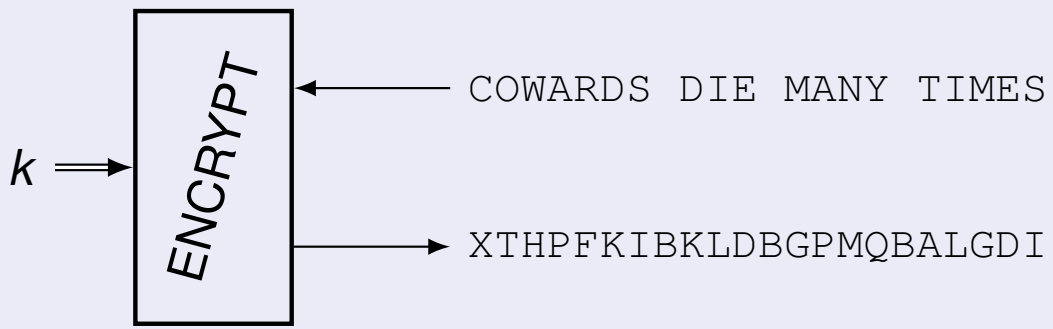
$$\{A, B, C, D, E, \ldots, W, X, Y, Z, \text{space}, \text{comma}, \text{period}\}$$



Monoalphabetic Cipher

Caesar's cipher is a special case of a monoalphabetic cipher. A more general monoalphabetic cipher uses an arbitrary permutation of the alphabet.

Example (Monoalphabetic cipher)



k
A → P
B → V
C → X
D → K
E → D
F → C
G → O
H → J
I → L
J → W
K → Z
L → E
M → G
N → M
O → T
P → Y
Q → S
R → F
S → I
T → A
U → N
V → U
W → H
X → R
Y → Q
Z → space
space → B

Polyalphabetic Cipher

A monoalphabetic cipher uses a fixed substitution table over the entire message. A polyalphabetic cipher uses multiple substitution tables.

A key specifies which table is used for which position of the message.

Example (Polyalphabetic cipher: Vigenère's cipher)

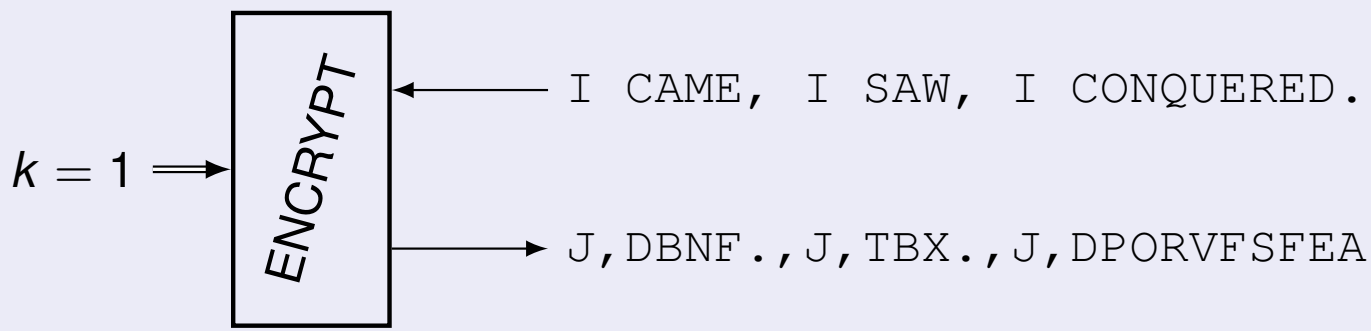
- It uses multiple Caesar ciphers.
- So if the key is 5,9,20, it means
  - the offset for the first letter of the message is 5
  - that for the second letter is 9
  - for the third letter it is 20
  - for the fourth letter it is 5 (we start over with the first offset of the key)
  - etc.

Key Assumption In Modern Cryptography

The security is based on the secret key (not on the secrecy of the algorithm).

Example (counterexample)

Caesar was evidently relying on the secrecy of the algorithm.



VARIOUS ATTACKS POSSIBLE

We distinguish between the following attacks:

- ciphertext-only: one or more cryptograms available to the cryptanalyst, known to have been encrypted with the same key.
- known plaintext: the cryptanalyst has one or more plaintexts and the resulting cryptograms, known to have been encrypted with the same key.
- chosen plaintext: for any plaintext that he requires, the cryptanalyst can obtain the cryptogram under the same key.

What kind of security do we expect?

- Ideally, a cryptographic system should be secure against a chosen plaintext attack.
- At the very least, it should be secure against a ciphertext-only attack.

EXAMPLE (CAESAR'S CIPHER)

- chosen plaintext attack: encrypt one letter and you get the key
- known plaintext attack: compare one letter and get the key
- ciphertext-only attack: try all the 29 possible keys

Caesar's cipher is not at all secure against a contemporary attacker.

Example (Generic Monoalphabetic cipher)

- chosen plaintext attack: encrypt each letter of the alphabet
- known plaintext attack: compare input/output over a text that uses all letters
- ciphertext-only attack:
  - brute-force approach: try all $29! = 8.84 \times 10^{30}$ permutations
  - letter-frequency approach: use the fact that for a given language we know the frequency of each letter

A brute-force approach is challenging.

With a modern computer, the key can easily be found using the letter-frequency attack.

How to make the letter-frequency attack unfruitful?

Example (Vigenère's cipher, with an n-length key)

- chosen plaintext attack: encode the same letter until you have the n-length key
- known plaintext attack: compare input/output until you have the n-length key
- ciphertext-only attack:
  - brute-force approach: try all 29^n keys if you know n. (Many more otherwise.)
    - for $n = 21$, the number of keys is 5.13^{30}
    - for $n = 100$, the number of keys is 1.73^{146}
  - if you know n, you can partition input/output into n parts, each of which is a Caesar cipher with its own key.
  - letter-frequency approach: effective if the plaintext-length to key-length ratio is sufficiently large.

Preliminary assumptions:

- The plaintext t, the key k and the cryptogram c are n-length binary sequences over the alphabet $\mathcal{A} = \{0, 1\}$.
- The key k is produced by selecting each bit independently and with uniform distribution.
- Alice and Bob use a private channel to exchange the key ahead of time.

Encryption: $c = t \oplus k$ (component-wise binary sum)

Decryption: $c \oplus k = (t \oplus k) \oplus k = t \oplus (k \oplus k) = t$

Example (one-time pad)

Encryption:

t	=	1	0	1	1	0	1
k	=	0	1	0	0	0	1
c	=	1	1	1	1	0	0

Decryption:

c	=	1	1	1	1	0	0
k	=	0	1	0	0	0	1
t	=	1	0	1	1	0	1

We get back the plaintext because $b + b = 0 \pmod 2$ for $b \in \{0, 1\}$.

Generalizing to a non-binary alphabet is straightforward.

DEFINITION (PERFECT SECRECY)

A cryptosystem has **perfect secrecy** if the plaintext T and the cryptogram C are statistically independent.

Perfect secrecy is the ultimate kind of security against a ciphertext-only attack: The attacker cannot do better than guessing the plaintext T.

- The n-length key k is selected at random (uniform distribution over $\{0,1\}^n$).
- The key k and the message t are selected independently.
- The ciphertext is $c = t \oplus k$.

$$p_{C|T}(c|t) = p_{K|T}(c \ominus t|t) = p_K(c \ominus t) = \frac{1}{2^n}.$$

(n is known by assumption.)

Hence C and T are independent: knowledge of C is useless in guessing T.

A Weakness of the one-time pad

Example (one-time pad)

An cryptanalyst that has the plaintext t and the corresponding cryptogram c, immediately gets the key:

$$k = c \ominus t$$

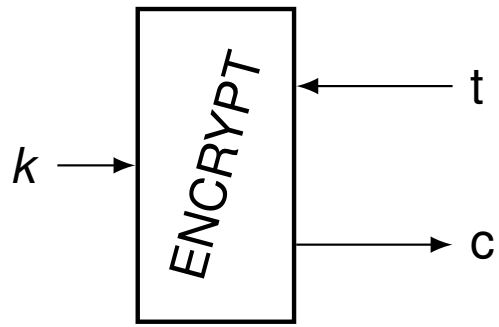
Hence the pad (the key) should be used only once.

ONE-TIME PAD: ADVANTAGES AND DRAWBACKS

+ very simple algorithm
+ as secure as it gets against a ciphertext-only attack and key used once
+ of instructional value to prove that perfect secrecy is possible
- the key is as long as the plaintext (this is fundamental, see later)
- the key needs to be exchanged ahead of time over a private channel
- a ciphertext-only attack can break the system if the key is used twice (see homework)
- a known plaintext attack reveals the key

The "one-time pad" has been used extensively in diplomatic and espionage circles.

The following Theorem makes no assumption on the encryption algorithm.



THEOREM (PERFECT SECRECY)

Perfect secrecy implies

$$H(T) \leq H(K).$$

Proof:

Perfect secrecy $\big(H(T) = H(T|C)\big)$ and decodability $\big(H(T|K,C) = 0\big)$ imply

$$\begin{aligned} H(T) &= H(T|C) \\ &\leq H(T,K|C) \\ &= H(K|C) + H(T|K,C) \\ &= H(K|C) \\ &\leq H(K). \end{aligned}$$

□

NB: Entropy plays a key role also in cryptography.

EXERCISE

Determine the minimum average length of the binary key for a cryptosystem that has the following characteristics:

- the message is an uncompressible binary string of length n
- the system achieves perfect secrecy

Solution

- $H(T)$ must be (essentially) n bits (otherwise further compression is possible).
- perfect secrecy requires $H(T) \leq H(K)$.
- hence $H(K)$ is at least n.
- the average blocklength of the binary key is at least n bits.

Symmetric-Key Cryptosystems: Key-distribution problem

A symmetric-key cryptosystem is one for which both ends use the same key ($k_A = k_B = k$). All examples considered so far rely on a symmetric key.

There exists fast (and secure) symmetric-key cryptosystems, but:

- Anybody that has the key can encrypt and/or decrypt.
- The key cannot be sent over an insecure channel.
- In an n-user network, each user needs $n - 1$ keys to communicate privately with every other user. Key distribution is a problem as it has to be done over a secure channel. And keys have to be changed frequently!
- We have a real problem: see e.g. the first 6 min. and 20 sec. of `http://www.youtube.com/watch?v=YEBfamv-_do&sns=em`

Public Key-Distribution (Diffie and Hellman)

Is there a way to distribute keys over a public channel?

In 1976, Diffie and Hellman came up with a solution.

Setup:

- Fix a large prime number p. Hereafter all the numbers are in $\{0, 1, \ldots, p-1\}$ and arithmetic is modulo p (more on it later).
- Pick a generator g. A generator has the property that g^i generates all elements in $\{1, 2, \ldots, p-1\}$ when $i = 0, 1, \ldots, p-2$.
- *Note:* Towards the end of this chapter, after introducing all of the algebra necessary, we will see that a generator always exists since we are in what is called a *cyclic group.*

Example

$p = 5$. The numbers are $\{0, 1, 2, 3, 4\}$.

$g = 2$ is a generator. Indeed:

i	g^i
0	1
1	2
2	4
3	3

- Alice picks a number a, kept secret.
- Bob picks a number b, kept secret.
- Alice and Bob send the number $A = g^a$ and $B = g^b$ to the public directory, respectively. This can be done over a non-private channel.

  The public directory, readable by everyone, looks like this

User	Public Key
Alice	A
Bob	B
$\vdots$	$\vdots$

Shared key generation:

Suppose that Alice and Bob want to communicate using a symmetric-key cryptosystem (the only kind of cryptosystem that we have studied so far).

They need a shared key k that nobody else knows.

Here is how they proceed:

- Alice gets B from the public directory and computes $k = B^a = g^{ba}$.
- Bob gets A from the public directory and computes $k = A^b = g^{ab}$.

We see that Alice and Bob have come up with a shared key k.

Eve wants to listen in:

Assuming that the cryptosystem used by Bob and Alice is secure, the best option for Eve is to find the key k.

She knows p, g, A, and B.

In general, there seems to be no better way than finding the number a for which $g^a = A$, and then compute $k = B^a$.

This is a problem. Let us check out some numbers: Suppose p is a 2048-bit number. (It must be prime, but let us neglect this and assume $p = 2^{2048}$.)

- It takes roughly
$$2 \log_2 p = 4096$$
multiplications to perform $a \to g^a$ (called discrete exponentiation). With a computer that performs 10^{10} multiplications per second, the exponentiation is done seamlessly.

- It takes roughly
$$\exp\left(\left(\frac{64}{9}\right)^{\frac{1}{3}} (\ln p)^{\frac{1}{3}} \ln\ln p)^{\frac{2}{3}}\right) \approx 10^{35}$$
multiplications to perform $g^a \to a$ (called discrete logarithm to the base g). With the same computer, it takes about 10^{25} seconds, which is about 7×10^7 times the age of the Earth. (The age of the Earth is about 4.5×10^9 years, i.e., 14.3×10^{16} seconds.)

Conclusion: Diffie and Hellman's public key-distribution scheme is clever, efficient, and it seems to be secure.

A Paradigm Shift

The perceived security of the DH public key-distribution algorithm relies on the solution to a problem considered to be difficult to solve.

We call this computational security. Even though it seems unlikely, someone could find a very fast algorithm to compute the discrete logarithm. The DH system would instantly become insecure.

To the contrast, perfect secrecy offers provable security even when the enemy has infinite time and computing power.

Most cryptographic systems rely on computational security.

This leads to the notion of a **one-way function**.

One-Way Functions

Discrete exponentiation is an example of a one-way function: a function for which a fast algorithm exists and no fast algorithm is known for the function's inverse.

(More precisely, to be considered as a one-way function, the modulus p needs to be a large prime number such that $n = p - 1$ has a large prime factor.)

In the DH protocol:

- Alice uses the function $f_a : g \mapsto g^a$ (with a kept secret)
- Bob uses the function $f_b : g \mapsto g^b$ (with b kept secret)

The functions commute: $f_a(f_b(g)) = f_b(f_a(g))$. Hence

$$f_a(B) = f_b(A).$$

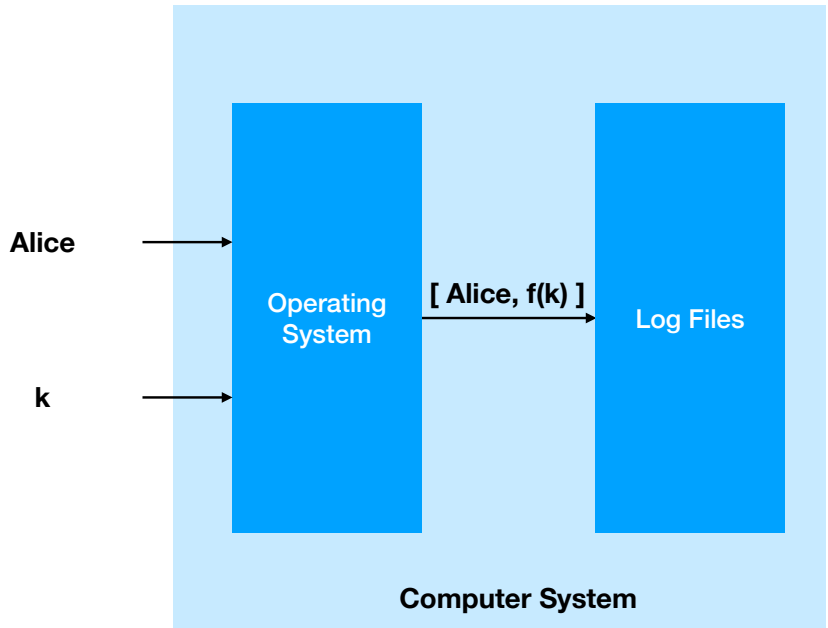
An attacker needs to invert the map $a \mapsto g^a$ (or, equivalently, invert the map $b \mapsto g^b$). This is hard to do, because discrete exponentiation is believed to be a one-way function.

The following is another application of a one-way function.

Example (Application of a one-way function)

If a computer were to save user's names and passwords, a system manager would have access to both.

This is not the case if the operating system stores, along the name, a one-way function f of our password. (The password itself is never stored.)



Trapdoor One-Way Functions

A trapdoor one-way function is a one-way function with an extra feature called the trapdoor information: with this information, the hard-to-carry-out inverse computation becomes easy.

Diffie and Hellmann realized that with such a tool the key-distribution problem would disappear.

Let us first take a look at what Diffie and Hellman proposed to do with a trapdoor one-way function (if such a function could be found).

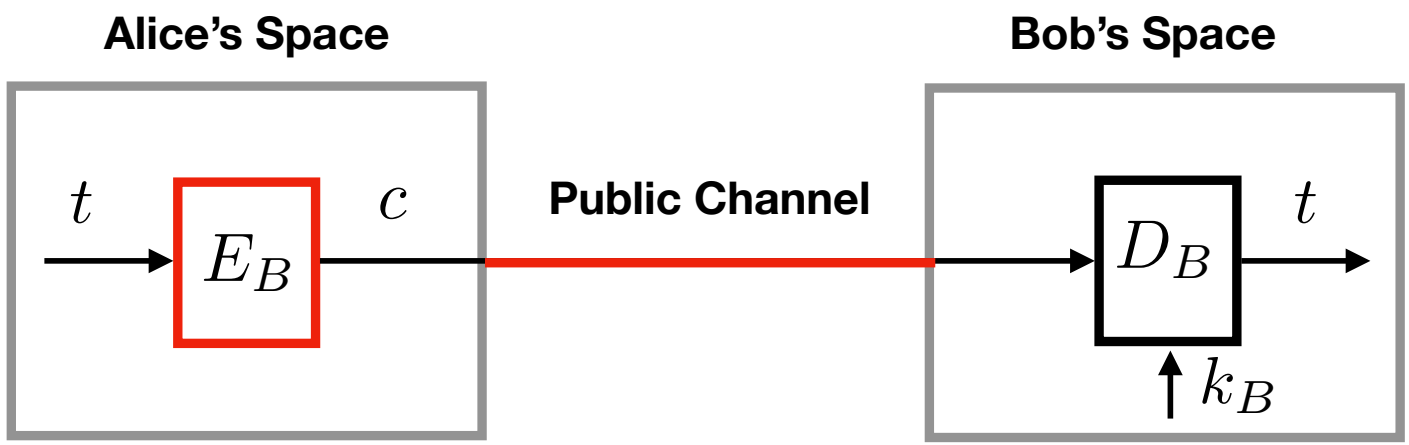
PUBLIC-KEY CRYPTOGRAPHY (ASYMMETRIC CRYPTOGRAPHY)

Suppose that Alice wants to send private information to Bob.

Bob has a trapdoor one-way function, implemented by an algorithm E_B that he publishes in a open directory.

He is the only one who has the trapdoor information k_B. Hence he has the algorithm D_B that implements the inverse function.

Alice and Bob no longer need a shared key (see figure).



The first trapdoor one-way function was found by Rivest, Shamir, and Adleman in 1978.

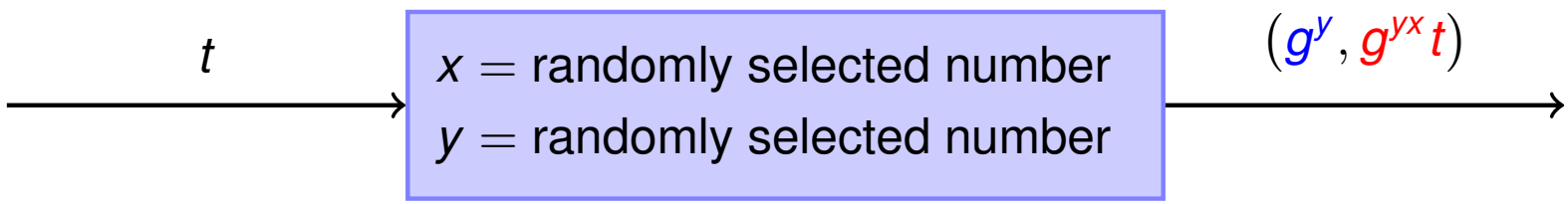
We start with ElGamal's trapdoor one-way function because, as for the DH protocol, it relies only on the assumption that the exponentiation is a one-way function.

ElGamal's trapdoor one-way function

Setup: (same as for the DH protocol)

- Fix a large prime number p. Hereafter all the numbers are in $\{0, 1, \ldots, p-1\}$ and arithmetic is modulo p (more on it later).
- Pick a generator g.
- Pick randomly selected numbers x and y. Unlike p and g, x and y are kept secret.

Here is a trapdoor one-way function, with **trapdoor information** x.



Given the trapdoor information x, we can invert the function as follows:

- compute the inverse of $\left(g^y\right)^x = g^{yx}$.
- multiply the result with $g^{yx}t$. The result is t.

ElGamal's Encryption Scheme

It is based on the above trapdoor one-way function. Let p and g be fixed and known to everyone.

Here is how Alice sends encrypted text to Bob:

Alice:
$t =$ plaintext, t;
$y =$ random number, y;

Bob:
$x =$ random number, x;

Bob sends g^x to Alice

$\longleftarrow$

Alice sends the cryptogram $(g^y, g^{yx}\, t)$ to Bob

$\longrightarrow$

Note: x and y are transaction specific.

Next goal: The RSA public key cryptosystem. It will take two weeks to build up the necessary background.

Week 6: Rudiments of Number Theory (Textbook Chapter 7)

Prof. Michael Gastpar

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

EPFL

Spring Semester 2025

Outline

Why Number Theory

Much of public-key cryptography is based on number theory.

More generally, in the digital world, the information is represented by the elements of a finite set, and we should be able to do math with them. Which means that the finite set should be a finite field. Our bigger goal of the next few lectures is to develop the tools to understand when and how we can turn a finite set into a finite field.

Operations with integers

Within $\mathbb{Z}$ (the set of integers) we can

- add, subtract, multiply
- but not divide: $\frac{7}{2}$ is not an integer
- what comes closest to the (regular) division is the Euclidean division

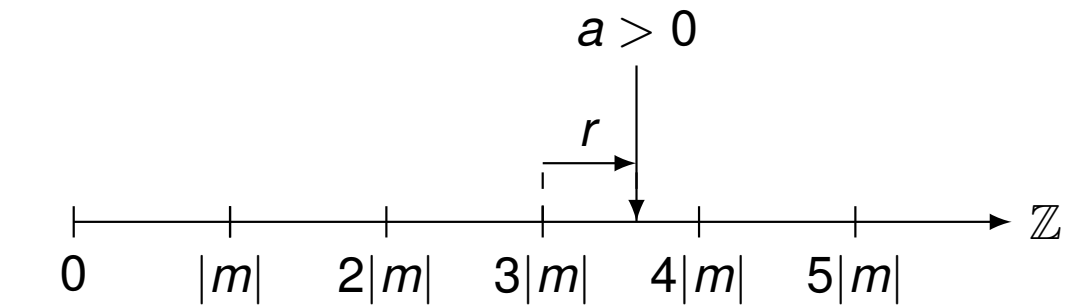
Euclidean Division

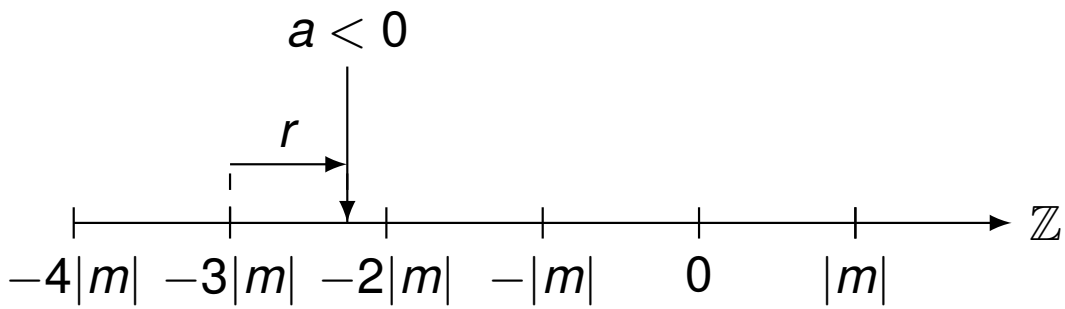
The Division Algorithm: Given integers a (the dividend) and m (the divisor), there exist unique integers q (quotient) and r (remainder), such that

$$a = mq + r, \quad 0 \leq r < |m|.$$

Note: The computation of q and r as above is called Euclidean division.

In spite of its name, the above should be seen as a theorem. It's proof is obvious from a drawing: find the mq to the left of a.





- The Euclidean division of 8 by 3 yields

$$8 = 3 \times 2 + 2$$

- The Euclidean division of -8 by 3 yields

$$-8 = 3 \times (-3) + 1$$

- The Euclidean division of 8 by -3 yields

$$8 = -3 \times (-2) + 2$$

- The Euclidean division of -8 by -3 yields

$$-8 = -3 \times 3 + 1$$

EUCLIDEAN DIVISION IN MAINSTREAM PROGRAMMING LANGUAGES

In `C`/`C++`/`Java`/`Python` we use the operator % to compute r as follows.

If a and m are both positive, then $r = a \,\%\, m$.

If one or the other or both are negative, different languages behave differently, but the general rule is:

- if $a \,\%\, m$ is nonnegative, then $r = a \,\%\, m$;
- if $a \,\%\, m$ is negative, then $r = a \,\%\, m + m$.

More precisely about the value of $a \,\%\, m$:

- `C/C++/Java`: $a \,\%\, m$ has the same sign as a.
- `Python`: $a \,\%\, m$ has the same sign as m.

Example

a	m	$a \,\%\, m$ in `C/C++/Java`	$a \,\%\, m$ in `Python`	r
8	3	2	2	2
−8	3	−2	1	1
8	−3	2	−1	2
−8	−3	−2	−2	1

- Wolfram Alpha: `https://www.wolframalpha.com`

EXAMPLE

```
5%3
```

- Python in browser: `https://trinket.io/python`

EXAMPLE

```
a= 5%3
print a
```

Both behave like Python, i.e., the sign of $a \,\%\, m$ is that of m.

From now on, unless otherwise specified, the divisor will be a positive integer m.

By

$$r = a \mod m,$$

we denote the remainder r when the integer a is divided by m.

Example

A pie that has 7 slices has to be divided evenly among 3 people. Then $7 \mod 3$ is the number of slices left over.

Example

The arrival time of a trip that starts at time 13 h and lasts 40 hours is $5 = 13 + 40 \mod 24$.

Congruence

Sometimes we are interested in knowing if two numbers have the same remainder when divided by m.

Definition

Two integers a and b are said to be **congruent modulo** m, denoted

$$a \equiv b \pmod{m},$$

if $m \mid a - b$.
(Read m divides $a - b$.)

Note: do not confuse the relation $a \equiv b \pmod{m}$ and the function $a \to a \bmod m$.

Example

- $23 \equiv 21 \pmod 2$
- $23 \equiv 3 \pmod 5$
- $x \equiv 0 \pmod 5$ means that x is a multiple of 5

Exercise

Which of these are true statements?

1. If $x \equiv 3 \pmod 5$, then x is not a multiple of 5.
2. If $x \equiv 25 \pmod 5$, then x is not a multiple of 5.
3. If $x \equiv 0 \pmod 5$, then x is a multiple of 5.

Solution

1. (true:) $x \equiv 3 \pmod 5$ means that $x - 3$ is divisible by 5.
   $\Rightarrow x$ is not a multiple of 5.

2. (false:) $x \equiv 25 \pmod 5$ means that $x - 25$ is divisible by 5.
   $\Rightarrow x$ is divisible by 5.

3. (true:) $x \equiv 0 \pmod 5$ means that x is divisible by 5.

The following statements are equivalent:

- $a \equiv b \pmod{m}$
- $(a - b) \mod m = 0$
- $a \mod m = b \mod m$

Congruence is an equivalence relation

A binary relation $\sim$ on a set is an **equivalence relation** iff the following three axioms are satisfied:

- $a \sim a$ (reflexivity)
- if $a \sim b$ then $b \sim a$ (symmetry)
- if $a \sim b$ and $b \sim c$ then $a \sim c$ (transitivity)

Substitute $a \sim a$ with $a \equiv a \pmod{m}$ etc. to see that congruence is an equivalence relation.

One of the consequences is that we can form equivalence classes and we can work with one representative of each class. (This will become useful later.)

Useful Rules (Thm 7.9 of textbook)

If

$$a \equiv a' \pmod{m}$$
$$b \equiv b' \pmod{m}$$

then

$$a + b \equiv a' + b' \pmod{m}$$
$$ab \equiv a'b' \pmod{m}$$
$$a^n \equiv (a')^n \pmod{m}$$

In particular, if $a' = (a \mod m)$ and $b' = (b \mod m)$, then we obtain the following facts (useful in $\mod$ calculations)

- $(a+b) \equiv \big((a \mod m) + (b \mod m)\big) \pmod m$
- hence
- $(a+b) \mod m = \big((a \mod m) + (b \mod m)\big) \mod m$

- $ab \equiv \big((a \mod m)(b \mod m)\big) \pmod m$
- hence
- $(ab) \mod m = \big((a \mod m)(b \mod m)\big) \mod m$

- $a^n \equiv \big(a \mod m\big)^n \pmod m$
- hence
- $a^n \mod m = \big(a \mod m\big)^n \mod m$

Bottom line: If the final result is mod m, then intermediate results can be reduced mod m.

Example

- $23 \equiv 3 \pmod 5$
- $2 \equiv 2 \pmod 5$
- Hence $23 + 2 \equiv 5 \pmod 5$

Example

- $(123 + 97) \pmod 2 = (1 + 1) \pmod 2 = 0$
- $(123 \cdot 97) \pmod 2 = (1 \cdot 1) \pmod 2 = 1$
- $\big((1234 \cdot 333) + 41(76 + 5)\big) \mod 2 = \big((0 \cdot 1) + 1(0 + 1)\big) \mod 2 = 1$

Exercise

Which of these is/are correct?

1. $23 \equiv 3 \pmod 5$
2. $-23 \equiv -3 \pmod 5$
3. $-23 \equiv 2 \pmod 5$

Solution

1. $23 \equiv 3 \pmod 5$ is correct: $23 - 3$ is divisible by 5
2. $-23 \equiv -3 \pmod 5$ is correct: multiply the above on both sides by -1
3. $-23 \equiv 2 \pmod 5$ is correct: use item 2 and $0 \equiv 5 \pmod 5$.

Less Trivial example

Example (Is $2 + 2^{1000}$ divisible by 3?)

- $2 \equiv -1 \pmod 3$
- $2^{1000} \equiv (-1)^{1000} \equiv 1 \pmod 3$
- $2 + 2^{1000} \equiv -1 + 1 \equiv 0 \pmod 3$

Hence $2 + 2^{1000}$ is divisible by 3.

Attention: we cannot reduce the exponent!

$$2^2 \mod 2 = 0 \neq 2^0 \mod 2 = 1.$$

Exercise

Is $9^{1000} + 9^{10^6}$ divisible by 5?

Solution

- $9 \equiv -1 \pmod 5$
- $9^{1000} + 9^{10^6} \equiv (-1)^{1000} + (-1)^{10^6} \equiv 1 + 1 \equiv 2 \pmod 5$

Hence $9^{1000} + 9^{10^6}$ is not divisible by 5.

Even Numbers

- $10 \equiv 0 \pmod 2$
- $10^n \equiv 0^n \equiv 0 \pmod 2$, n positive integer

$$\begin{aligned} 1234 &= 1 \cdot 10^3 + 2 \cdot 10^2 + 3 \cdot 10^1 + 4 \\ &\equiv 1 \cdot 0 + 2 \cdot 0 + 3 \cdot 0 + 4 \pmod 2 \\ &\equiv 4 \pmod 2 \\ &\equiv 0 \pmod 2 \end{aligned}$$

Hence 1234 is divisible by 2.

We see that a decimal number is divisible by 2 iff the last digit is divisible by 2. (Not new to us, but the method generalizes.)

Remainder after division by 9

- $10 \equiv 1 \pmod 9$
- $10^n \equiv 1^n \equiv 1 \pmod 9$, n positive integer

$$\begin{aligned} 1234 &= 1 \cdot 10^3 + 2 \cdot 10^2 + 3 \cdot 10^1 + 4 \\ &\equiv 1 \cdot 1 + 2 \cdot 1 + 3 \cdot 1 + 4 \pmod 9 \\ &\equiv 1 + 2 + 3 + 4 \pmod 9 \\ &\equiv 10 \pmod 9 \\ &\equiv 1 \pmod 9 \end{aligned}$$

Hence the remainder after division of 1234 by 9 is 1.

To obtain the rest after division of a decimal number by 9, we can substitute the number with the sum of its digits.

Example (mod 9)

$$
\begin{aligned}
1234567890 &\equiv 1+2+3+4+5+6+7+8+9+0 \pmod 9 \\
&\equiv 45 \pmod 9 \\
&\equiv 0 \pmod 9
\end{aligned}
$$

Check Digits Mod 97

- Write down an integer in decimal notation, e.g.,

$$021\,235\,1234$$

- Compute its remainder after division by 97:

$$021\,235\,1234 \mod 97 = 95$$

- Append the remainder to the number, as a check digit:

$$021\,235\,1234\,95$$

- A common mistake consists in transposing two digits:

$$021\,253\,1234\,95$$

- The check digits are no longer consistent:

$$021\,253\,1234 \mod 97 = 63$$

Procedure Mod 97 – 10

It is a variant of the previous one:

1. Append 00 (i.e., multiply the number by 100)
2. Let r be the remainder after division by 97
3. The check digits are $c = 98 - r$ (written as a 2-digit number, e.g., 03)
4. Replace 00 with c
5. Check: the resulting number mod 97 equals 1

Procedure Mod 97 – 10: why the check is as stated

Encoding:

$$n \longmapsto 100n + \underbrace{98 - (100n \mod 97)}_{\text{check digits}}$$

Check: we compute the resulting number mod 97:

$$\begin{aligned} & 100n + 98 - (100n \mod 97) \mod 97 \\ = \; & 100n + 98 - 100n \mod 97 \\ = \; & 98 \mod 97 \\ = \; & 1 \end{aligned}$$

Example (mod 97 − 10)

1. $n = 212351234$
2. $n \longmapsto 21235123400 + \underbrace{(98 - 91)}_{\text{check digits}} = 21235123407$
3. Check: $21235123407 \mod 97 = 1$. Check passed
4. If we transpose: 21253123407
5. Check: $21253123407 \mod 97 = 2$. Check **not** passed

Next lecture we will see why Mod 97 − 10 always detects a transposition.

IBAN (International Bank Account Number)

Main difference to MOD 97 − 10: The check digits are in position 3 and 4

Example:

1. Account number: $\overbrace{00243}^{\text{bank, 5 digits}}\ \overbrace{0001\,2345\,6789}^{\text{account, 12 digits}}$
2. Append CH (for a Swiss bank account): 00243 0001 2345 6789 CH
3. Convert into numbers according to: $A \mapsto 10, \ldots, Z \mapsto 35$:

   00243 0001 2345 6789 1217

4. MOD 97 − 10 procedure:

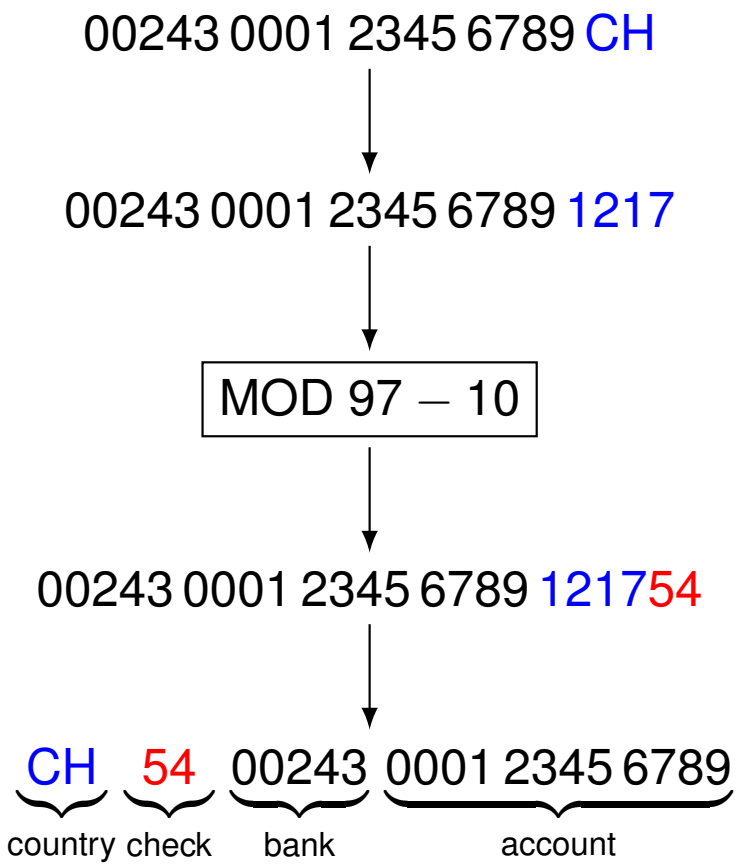
   00243 0001 2345 6789 121754

5. Reposition:

   CH54 00243 0001 2345 6789

6. To verify, we undo the repositioning and do the MOD 97 − 10 check.

IBAN CONSTRUCTION



Exercise

Is the following statement correct?

$$2 + x \equiv 2 + y \pmod{12} \implies x \equiv y \pmod{12}$$

Solution

We are allowed to add and multiply on both sides as we do when we solve equations over the reals.

By adding -2 on both sides:

$$2 + x \equiv 2 + y \pmod{12} \Rightarrow x \equiv y \pmod{12}$$

The statement is true.

(Which property of the "useful rules" have we used?)

Exercise

Is the following statement correct?

$$2x \equiv 2y \pmod{12} \implies x \equiv y \pmod{12}$$

Solution

No. The multiplicative inverse of 2 does not exist $\pmod{12}$.

For instance

$$2 \times 9 \equiv 2 \times 3 \pmod{12},$$

however

$$9 \not\equiv 3 \pmod{12}$$

(Why can't we say that $\frac{1}{2} \equiv \frac{1}{2} \pmod{12}$ and multiply both sides by $\frac{1}{2}$?)

DEFINITION

A **prime number** (or a **prime**) is an integer > 1 that has no positive divisors other than 1 and itself.

$2, 3, 5, 7, 11, 13, 17, 19, 23, 29, \ldots$ are prime numbers.

Non-primes are called **composites**.

Exercise

Many people forget if 1 is prime or not. Why is it not?

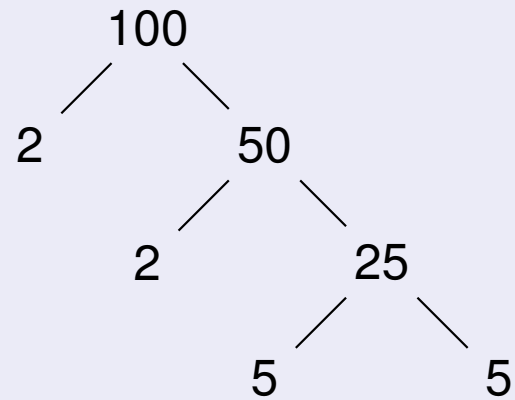
Solution

Because if we declare 1 to be a prime number, then the following fundamental theorem is no longer valid.

THEOREM (PRIME FACTORIZATION: SHORT VERSION)

Every integer greater than 1 has a unique prime factorization (except for order).

Example



$\Rightarrow 100 = 2 \times 2 \times 5 \times 5$

Prime Factorization: A difficult task

- a number as big as 2^{700} (700 bits) can be factored
- a 1000 bits number cannot be factored (with today's technology)

  "Among the b-bit numbers, the most difficult to factor in practice using existing algorithms are those that are products of two primes of similar size. For this reason, these are the integers used in cryptographic applications. The largest such semiprime yet factored was RSA-250, an 829-bit number with 250 decimal digits, in February 2020. The total computation time was roughly 2700 core-years of computing using Intel Xeon Gold 6130 at 2.1 GHz. Like all recent factorization records, this factorization was completed with a highly optimized implementation of the general number field sieve run on hundreds of machines."

  [Wikipedia, March 23, 2023]

Theorem (Textbook Thm 7.3)

Let a and b be positive integers. a divides b iff all prime factors of a are present in the prime factorization of b with an equal or greater exponent.

Example

$$168 = 2^3 \cdot 3 \cdot 7$$

$$12 = 2^2 \cdot 3$$

Hence 12 divides 168.

Example

$$30 = 2 \cdot 3 \cdot 5$$

$$12 = 2^2 \cdot 3$$

Hence 12 does not divide 30.

Definition

Let a and b be integers, not both zero. The largest integer that divides both is called the **greatest common divisor** of a and b. It is denoted by $\gcd(a, b)$.

Theorem (Textbook Thm 7.4)

Let a and b be positive integers, not both zero, and let $p_1 < p_2 < \cdots < p_k$ be the sequence of prime numbers that divide a or b. Write

$$a = p_1^{\alpha_1} \cdots p_k^{\alpha_k}$$
$$b = p_1^{\beta_1} \cdots p_k^{\beta_k},$$

with $0 \leq \alpha_i$ and $0 \leq \beta_i$. Then

$$\gcd(a, b) = p_1^{\gamma_1} \cdots p_k^{\gamma_k},$$

with $\gamma_i = \min(\alpha_i, \beta_i)$.

Example

$$\begin{aligned} 12 = 2^2 \cdot 3 &= 2^2 \cdot 3^1 \cdot 5^0 \\ 30 = 2 \cdot 3 \cdot 5 &= 2^1 \cdot 3^1 \cdot 5^1 \\ \gcd(12, 30) &= 2^1 \cdot 3^1 \cdot 5^0 = 6 \end{aligned}$$

It is an immediate consequence of the above theorem that $\gcd(a, b) = 1$ iff a and b have no common factor.

Definition

When $\gcd(a, b) = 1$, we say that a and b are **coprime** (or **relatively prime**, or **mutually prime**).

Example

$$9 = 3^2$$

$$100 = 2^2 \cdot 5^2$$

$$\gcd(9, 100) = 1$$

9 and 100 are thus coprime.

Theorem (Textbook Thm 7.6)

Let p be a prime number and let a be an integer such that $0 < a < p$. Then

$$\gcd(p, a) = 1$$

Proof

The prime factorization of a cannot contain p.

Exercise (True or False?)

If $ab \mid c$, then $a \mid c$ and $b \mid c$.

Solution

If $ab \mid c$, then we can write $c = abd$ for some integer d.

Clearly both a and b divide c.

Exercise (True or False?)

If $a \mid c$ and $b \mid c$, then $ab \mid c$.

Solution

$a \mid c$ and $b \mid c$ does not imply $ab \mid c$.

In fact, ab could exceed c.

Example: $a = b = c$.

However

Theorem

If $a \mid c$ and $b \mid c$ and $\gcd(a, b) = 1$, then $ab \mid c$.

Proof

- the prime factorization of c contains all the prime factors of a.
- the prime factorization of c contains all the prime factors of b.
- a and b have distinct prime factors.

$\Rightarrow$ $\frac{c}{a}$ is an integer that has all the prime factors of b in it.

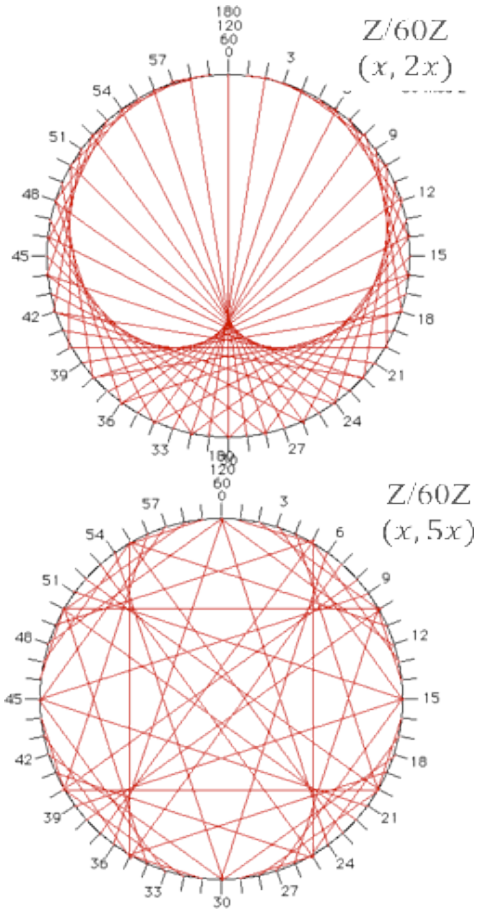
$\Rightarrow$ Hence it is divisible by b.

- This proves that $ab \mid c$.

Example

- $a = 2 \cdot 3$
- $b = 5 \cdot 7$
- $c = 2 \cdot 3 \cdot 5 \cdot 7 \cdot 11$

Week 7: Modular Arithmetic (Textbook Chapter 8)



Prof. Michael Gastpar

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

EPFL

Spring Semester 2025

Outline

Why Modular Arithmetic

Modular arithmetic is a foundation of number theory.

We need number theory for cryptography and for channel coding.

Introducing $\mathbb{Z}/m\mathbb{Z}$

Instead of considering integers and congruences $\pmod m$, and write "equations" like

$$a + b \equiv c \pmod m$$
$$a \cdot b \equiv d \pmod m,$$

we would like to write the "usual" kind of equations like

$$a + b = c$$
$$a \cdot b = d,$$

even when the operations are $\bmod m$.

This can be done, if we give new meaning to a, b, c and d, namely we make them the congruence classes $[a]_m$, $[b]_m$, $[c]_m$ and $[d]_m$.

Definition (Congruence Classes)

Let $m > 1$ be an integer, called the modulus.

The set of all integers congruent to $a \pmod{m}$ is called the **congruence class of** a **modulo** m.

It is denoted by $[a]_m$.

Example

- $[24]_2$ is the set of even integers. Same as $[0]_2$, $[2]_2$, etc.
- $[23]_2$ is the set of odd integers. Same as $[1]_2$, $[3]_2$, etc.
- $[a]_m = [b]_m$ iff $a \equiv b \pmod{m}$.

Definition ($\mathbb{Z}/m\mathbb{Z}$)

The set of all congruence classes modulo m is denoted by $\mathbb{Z}/m\mathbb{Z}$ (which is read "$\mathbb{Z}$ mod m") .

Note: Some authors use the notation $\mathbb{Z}_m$.

Example

- $\mathbb{Z}/2\mathbb{Z} = \{[0]_2, [1]_2\}$.
- $\mathbb{Z}/3\mathbb{Z} = \{[0]_3, [1]_3, [2]_3\}$.
- etc.

NB: An element of $\mathbb{Z}/m\mathbb{Z}$ can be written in many ways

$$[a]_m = [a+m]_m = [a+2m]_m = \cdots$$

In particular:

- if $a = mq + r$, with $0 \leq r \leq m-1$, then

  $$[a]_m = [r]_m.$$

  We say that $[r]_m$ is in reduced form.

- every element of $\mathbb{Z}/m\mathbb{Z}$ has a unique representation in reduced form;

- $[b]_m$ is in reduced form iff $0 \leq b \leq m-1$.

Example

Which statements are correct?

1. $[-13]_9 = [5]_9$
2. $[13]_9 = [-5]_9$
3. $[13]_9 = [5]_9$
4. $[-13]_9 = [-5]_9$

Solution

1. $[-13]_9 = [5]_9$ is correct: $9 \mid -18$
2. $[13]_9 = [-5]_9$ is correct: $9 \mid 18$
3. $[13]_9 = [5]_9$ is incorrect: 9 does not divide 8
4. $[-13]_9 = [-5]_9$ is incorrect: 9 does not divide -8

In $\mathbb{Z}/m\mathbb{Z}$ we define the sum and the product as follows:

- $[a]_m + [b]_m = [a+b]_m$
- $[a]_m[b]_m = [ab]_m$

The result is the same regardless the choice of representatives. In fact:

- If we choose $[a+km]_m$ instead of $[a]_m$
- and $[b+lm]$ instead of $[b]_m$
- then we obtain $[a+km]_m + [b+lm]_m = [a+km+b+lm]_m$ which is the same as $[a+b]_m$.

Idem for multiplication.

Example (addition and multiplication in $\mathbb{Z}/3\mathbb{Z}$)

If the value of m is implicit, e.g. $m = 3$, then we may write a instead of $[a]_3$.

The addition and multiplication tables are:

$\mathbb{Z}/3\mathbb{Z}$	$+$	0	1	2
	0	0	1	2
	1	1	2	0
	2	2	0	1

$\mathbb{Z}/3\mathbb{Z}$	$\times$	0	1	2
	0	0	0	0
	1	0	1	2
	2	0	2	1

Example (addition and multiplication in $\mathbb{Z}/4\mathbb{Z}$)

$\mathbb{Z}/4\mathbb{Z}$	$+$	0	1	2	3
	0	0	1	2	3
	1	1	2	3	0
	2	2	3	0	1
	3	3	0	1	2

$\mathbb{Z}/4\mathbb{Z}$	$\times$	0	1	2	3
	0	0	0	0	0
	1	0	1	2	3
	2	0	2	0	2
	3	0	3	2	1

Properties of $+$ in $\mathbb{Z}/m\mathbb{Z}$

The sum has the following properties:

- associativity:

$$[a]_m + ([b]_m + [c]_m) = ([a]_m + [b]_m) + [c]_m;$$

- there exists an additive identity, namely $[0]_m$:

$$[a]_m + [0]_m = [0]_m + [a]_m = [a]_m;$$

- there exists an inverse with respect to addition: every $[a]_m$ has an inverse, denoted $(-[a]_m)$, such that

$$[a]_m + (-[a]_m) = (-[a]_m) + [a]_m = [0]_m;$$

  the inverse of $[a]_m$ is $[-a]_m$;
- commutativity:

$$[a]_m + [b]_m = [b]_m + [a]_m;$$

Properties of $\times$ in $\mathbb{Z}/m\mathbb{Z}$

The multiplication has the following properties:

- associativity:
$$[a]_m([b]_m[c]_m) = ([a]_m[b]_m)[c]_m;$$
- multiplicative identity, namely $[1]_m$:
$$[a]_m[1]_m = [1]_m[a]_m = [a]_m;$$
- commutativity:
$$[a]_m[b]_m = [b]_m[a]_m;$$

Mixed Property in $\mathbb{Z}/m\mathbb{Z}$

The two operations have the following property:

- distributivity:

$$[a]_m([b]_m + [c]_m) = [a]_m[b]_m + [a]_m[c]_m;$$

THE NOTATION $k[a]_m$ IN $\mathbb{Z}/m\mathbb{Z}$

For an arbitrary positive integer k, $k[a]_m$ is a short hand for

$$\underbrace{[a]_m + [a]_m + \cdots + [a]_m}_{k \text{ times}}.$$

We can easily verify that

$$k[a]_m = [ka]_m = [k]_m[a]_m.$$

THE MULTIPLICATIVE INVERSE

Some elements of $\mathbb{Z}/m\mathbb{Z}$ have the **multiplicative inverse**.

The multiplicative inverse of $[a]_m$, if it exists, is an element $[b]_m$ such that

$$[a]_m[b]_m = [b]_m[a]_m = [1]_m.$$

The multiplicative inverse, if it exists it is unique, and it is denoted by $([a]_m)^{-1}$.

Furthermore $\left(([a]_m)^{-1}\right)^{-1} = [a]_m$.

Proof that the inverse, if it exists, is unique:

- Suppose $ab = 1$ and $ac = 1$.
- Then $ab = ac$. Multiplying both sides by b yields
- $bab = bac$. But $ba = ab = 1$. Hence $b = c$. □

Proof that if b is the inverse of a, then the inverse of b is a.

If b is the inverse of a, then $ab = ba = 1$, which implies that a is the inverse of b. □

Exercise ($\mathbb{Z}/4\mathbb{Z}$)

Which elements of $\mathbb{Z}/4\mathbb{Z}$ have the multiplicative inverse? What is it?

$\mathbb{Z}/4\mathbb{Z}$	$\times$	0	1	2	3
	0	0	0	0	0
	1	0	1	2	3
	2	0	2	0	2
	3	0	3	2	1

Solution

We see that

- $[1]_4$ and $[3]_4$ have the inverse ($[1]_4$ and $[3]_4$, respectively).
- $[2]_4$ has no inverse.
- $[0]_m$ has no inverse, regardless of m.

For any positive integer k,

- $([a]_m)^k$ is a short hand for $\underbrace{[a]_m[a]_m \cdots [a]_m}_{k \text{ times}}$;
- $([a]_m)^0 = [1]_m$ (empty product).
- Note that we do not consider negative exponents $([a]_m)^{-k}$ because it is problematic in general, with the exception of $([a]_m)^{-1}$, if course, which is simply the multiplicative inverse of $[a]_m$ whenever it exists.

Example

$$([3]_7)^{12} = \left(([3]_7)^2\right)^6 = ([2]_7)^6 = \left(([2]_7)^3\right)^2 = ([1]_7)^2 = [1]_7.$$

An equation of the form

$$[a]_m x = [b]_m$$

has a **unique** solution iff $[a]_m$ has the inverse. In this case,

$$x = ([a]_m)^{-1}[b]_m.$$

We prove a more general statement.

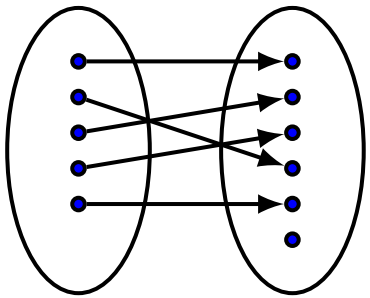
First a brief terminology review.

Terminology Review

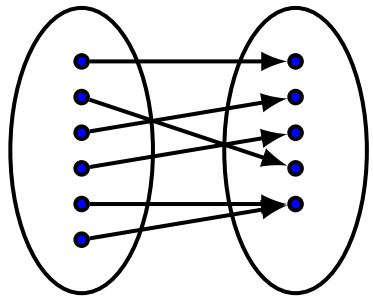
Recall that for a function $f : \mathcal{E} \to \mathcal{F}$

- $\mathcal{E}$ is the domain
- $\mathcal{F}$ is the codomain
- $f(\mathcal{E})$ is the image
- (the word *range* is sometimes used for the codomain, and sometimes for the image)

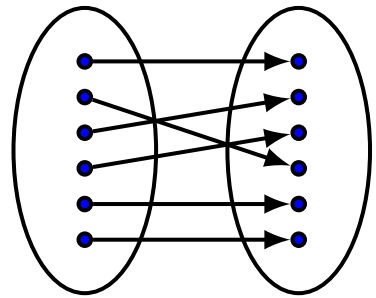
PIGEONHOLE PRINCIPLE



injective
(one-to-one)



surjective
(onto)



bijective
(one-to-one and onto)

Let $f : \mathcal{E} \to \mathcal{F}$, where $\mathcal{E}$ and $\mathcal{F}$ are finite sets.

- f injective $\Rightarrow |\mathcal{E}| \leq |\mathcal{F}|$
- f surjective $\Rightarrow |\mathcal{E}| \geq |\mathcal{F}|$
- f bijective $\Rightarrow |\mathcal{E}| = |\mathcal{F}|$

THEOREM

In $\mathbb{Z}/m\mathbb{Z}$, the following statements are equivalent:

(1) $[a]_m$ has the inverse;

(2) For all $[b]_m$, $[a]_m x = [b]_m$ has a unique solution;

(3) There exists a $[b]_m$, such that $[a]_m x = [b]_m$ has a unique solution.

Proof:

(1) $\Rightarrow$ (2): We multiply both sides of $[a]_m x = [b]_m$ by $[a]_m^{-1}$ and obtain the equivalent equation $x = [a]_m^{-1}[b]_m$, showing that there is a solution and the solution is unique.

(2) $\Rightarrow$ (1): For $[b]_m = [1]_m$ we obtain $[a]_m x = [1]_m$, which has a solution by assumption. The solution is the inverse of a.

(2) $\Rightarrow$ (3): True since (3) is a weaker statement than (2).

(3) $\Rightarrow$ (2): We prove the contrapositive, i.e., we assume that there is a $[\tilde{b}]_m$ such that $[a]_m x = [\tilde{b}]_m$ has either no solution or multiple solutions, and we prove that for no $[b]_m$, $[a]_m x = [b]_m$ has a unique solution.

- So suppose that $[a]_m x = [\tilde{b}]_m$ has no solution or multiple solutions.
- By the pigeonhole principle, the map $x \to ax$ is neither injective nor surjective.
- We can find a $[b^*]_m$ such that $[a]_m x = [b^*]_m$ has multiple solutions, say x_1 and x_2. Define $x_3 = x_1 - x_2 \neq [0]_m$.
- Hence, $[a]_m x_3 = [a]_m x_1 - [a]_m x_2 = [b^*]_m - [b^*]_m = [0]_m$.
- So the equation $[a]_m x = [0]_m$ has at least two solutions, x_3 and $[0]_m$.
- If $[a]_m x = [b]_m$ has a solution, say x_4, then $x_4 + x_3$ is also a solution.
- We conclude that for no $[b]_m$, $[a]_m x = [b]_m$ has a unique solution. $\square$

Exercise ($\mathbb{Z}/9\mathbb{Z}$)

If it exists, find the solution of $[4]_9 x = [3]_9$

Solution

x	0	1	2	3	4	5	6	7	8
$[4]_9 x$	0	4	8	3	7	2	6	1	5

Pedestrian solution: From the above table we see that the solution is $[3]_9$. This approach requires having the above $[4]_9 x$ table.

Preferable solution (when possible): If it exists, we find the inverse of $[4]_9$. For now, we use the table to find $([4]_9)^{-1} = [7]_9$. Hence

$$x = [7]_9[3]_9 = [3]_9.$$

We will see how to find the inverse without constructing the table.

Example

If it exists, find the solution of $[2]_7 x + [3]_7 = [1]_7$

1. $\Leftrightarrow [2]_7 x = [1]_7 + (-[3]_7)$ (adding on both sides the negative of $[3]_7$ — always exists)
2. $\Leftrightarrow [2]_7 x = [-2]_7$
3. $\Leftrightarrow x = ([2]_7)^{-1}[5]_7$ (multiplying both sides by the inverse of $[2]_7$, which exists)
4. $\Leftrightarrow x = [4]_7[5]_7$ $(([2]_7)^{-1} = [4]_7)$
5. $\Leftrightarrow x = [20]_7$
6. $\Leftrightarrow x = [6]_7$

Example

If it exists, find the solution of $[3]_9 x + [2]_9 = [5]_9$

1. $\Leftrightarrow [3]_9 x = [5]_9 + [-2]_9$
2. $\Leftrightarrow [3]_9 x = [3]_9$

$([3]_9)^{-1}$ does not exist (see table below).

x	0	1	2	3	4	5	6	7	8
$[3]_9 x$	0	3	6	0	3	6	0	3	6

Yet, from the above table, we see that there are three solutions, namely

$$x = [1]_9,\ x = [4]_9,\ x = [7]_9.$$

THEOREM

Let $m > 1$ be integer.

The element $[a]_m \in \mathbb{Z}/m\mathbb{Z}$ has a multiplicative inverse iff $\gcd(a, m) = 1$.

The proof is postponed (see Bézout's identity).

Example (Multiplicative inverses in $\mathbb{Z}/4\mathbb{Z}$)

$\mathbb{Z}/4\mathbb{Z}$	$\times$	0	1	2	3
	0	0	0	0	0
	1	0	1	2	3
	2	0	2	0	2
	3	0	3	2	1

$\gcd(a, 4) = 1$ for $a = 1, 3$.

Theorem ($\mathbb{Z}/p\mathbb{Z}$ with p prime)

If p is prime, all elements of $\mathbb{Z}/p\mathbb{Z}$ except $[0]_p$ have a multiplicative inverse.

Proof:

$$\gcd(a, p) = 1 \text{ for } a = 1, 2, \ldots, p-1$$

$$\gcd(0, p) = p.$$

□

Recall the Mod 97 − 10 procedure

1. Append 00 (i.e., multiply the number by 100)
2. Let r be the remainder after division by 97
3. The check digits are $c = 98 - r$ (written as a 2-digit number)
4. Replace 00 with c
5. Check: the resulting number mod 97 equals 1

Recall the example

1. $n = 212351234$
2. $n \longmapsto 21235123400 + 98 - 91 = 21235123407$
3. Check: $21235123407 \mod 97 = 1$. Check passed
4. If we transpose: 21253123407
5. Check: $21253123407 \mod 97 = 2$. Check **not** passed

WHY IT DETECTS TRANSPOSITIONS

Let us use the new notation to remind ourselves why an unmodified number passes the check:

Recall that $n \to 100n + 98 - (100n \mod 97)$.

The test is passed if [number with check digits]$_{97} = [1]_{97}$

This is the case:

$$[100n+98-(100n \mod 97)]_{97} = [[100n]_{97}+98-[100n]_{97}]_{97} = [98]_{97} = [1]_{97}.$$

Two consecutive digits ba of a decimal number are worth $10^k(a+10b)$ for some nonnegative integer k.

After we transpose them they are worth $10^k(b+10a)$.

The check detects the transposition, unless

$$[10^k(b+10a) - 10^k(a+10b)]_{97} = [0]_{97}$$

$$\Leftrightarrow\ [10^k(9a-9b)]_{97} = [0]_{97}$$

$$\Leftrightarrow\ [10^k 9(a-b)]_{97} = [0]_{97}$$

$$\Leftrightarrow\ ([10]_{97})^k [9]_{97} [a-b]_{97} = [0]_{97}$$

$$\Leftrightarrow\ [a-b]_{97} = [0]_{97}$$ (all non-zero elements of $\mathbb{Z}/97\mathbb{Z}$ have an inverse)

We conclude that the transposition is not detected iff $a = b$, i.e., if there is no transposition.

What for?

- Recall that $[a]_m$ has an inverse (in $\mathbb{Z}/m\mathbb{Z}$) iff $\gcd(a, m) = 1$.
- The Euclidean algorithm is a technique for quickly finding the gcd of two integers. (Much faster than via the prime factor decomposition, which is hard to do for large numbers.)
- When $\gcd(a, m) = 1$, Bézout's identity gives us the inverse of $[a]_m$.

THEOREM (EUCLID, TEXTBOOK THM 8.3)

Let a and b be integers, not both zero. Then, for any integer k

$$\gcd(a, b) = \gcd(b, a - kb)$$

Proof:

If d divides a and b, then it divides b and $a - kb$.

Similarly, if d divides both b and $a - kb$, then it divides b and $a - kb + kb = a$.

Since the set of divisors of a and b is the same as the set of divisors of b and $a - kb$, the greatest divisor is the same in both cases. □

Basic ingredients to compute the gcd

- $\gcd(a, b) = \gcd(\pm a, \pm b) = \gcd(b, a)$.
- Hence we can focus on the computation of $\gcd(a, b)$ with $0 \leq b \leq a$.
- If $a = qb + r$ is the Euclidean division, then

$$\gcd(a, b) = \gcd(b, a - qb) = \gcd(b, r),$$

  with $0 \leq r < b$. This is progress.

- Hence $\gcd(a, b) = \gcd(b, r_1) = \gcd(r_1, r_2) = \cdots = \gcd(r_n, 0) = r_n$, where

$$\begin{aligned} a &= bq_1 + r_1, \quad 0 \leq r_1 < b \\ b &= r_1 q_2 + r_2, \quad 0 \leq r_2 < r_1 \\ r_i &= r_{i+1} q_{i+2} + r_{i+2}, \quad 0 \leq r_{i+2} < r_{i+1}. \end{aligned}$$

Example

$\gcd(a,b)$	$a = bq + r$
$= \gcd(b,r)$	
$= \gcd(122,22)$	$122 = 22 \times 5 + 12$
$= \gcd(22,12)$	$22 = 12 \times 1 + 10$
$= \gcd(12,10)$	$12 = 10 \times 1 + 2$
$= \gcd(10,2)$	$10 = 2 \times 5 + 0$
$= \gcd(2,0)$	
$= 2$	

EUCLIDEAN ALGORITHM (RECURSIVE)

Algorithm 1 gcd(a, b : positive integers)

1: **if** $a < b$ **then**
   return gcd(b, a)
2: **else if** $b = 0$ **then**
   return a
3: **else**
   return gcd($b, a \% b$)
4: **end if**

Exercise

Compute $\gcd(12345678906, 12345678901)$

Solution

$$\begin{aligned}\gcd(12345678906, 12345678901) &= \gcd(12345678901, 5)\\ &\overset{(*)}{=} \gcd(5, 1)\\ &= \gcd(1, 0)\\ &= 1,\end{aligned}$$

where in $(*)$ we use the fact that a number $xxxxxxx0$ is divisible by 5.

Bézout

Theorem (Bézout's identity (Textbook theorem 8.4))

Let a and b be integers, not both zero.

There exist integers u and v, such that

$$\gcd(a, b) = au + bv$$

We prove Bézout's identity by means of the extended Euclidean algorithm, which finds solutions to Bézout's identity

$$\gcd(a, b) = au + bv,$$

where a and b are given, and u, v and $\gcd(a, b)$ are returned by the algorithm.

Note: if $\gcd(a, b) = au + bv$, then $\gcd(-a, b) = au + bv$ and $\gcd(a, -b) = au + bv$.

Hence it suffices that we consider nonnegative numbers a, b, not both zero.

Proof:

- Iteration step: Suppose $a \geq b$.
  - $\gcd(a, b) = \gcd(b, r)$, where $a = bq + r$;
  - suppose we have found $\tilde{u}$ and $\tilde{v}$ such that $\gcd(b, r) = b\tilde{u} + r\tilde{v}$;
  - use $r = (a - bq)$ to rewrite

    $$\gcd(a, b) = \gcd(b, r) = b\tilde{u} + r\tilde{v} = b\tilde{u} + (a - bq)\tilde{v} = a\tilde{v} + b(\tilde{u} - q\tilde{v}) \stackrel{!}{=} au + bv;$$

  - comparing terms: $u = \tilde{v}$ and $v = (\tilde{u} - q\tilde{v})$.

- Final step: $\gcd(a, 0) = a \Rightarrow \tilde{u} = 1$, $\tilde{v} = 0$.
  Note: in this last step, $\tilde{v}$ is not unique.

- Via successive applications of the above iteration, eventually we reach the form $\gcd(a, b) = au + bv$. □

Example

$\gcd(a,b)$	$a = bq + r$	$u = \tilde{v}$	$v = (\tilde{u} - q\tilde{v})$
$\gcd(122,22)$	$122 = 22 \times 5 + 12$		
$\gcd(22,12)$	$22 = 12 \times 1 + 10$		
$\gcd(12,10)$	$12 = 10 \times 1 + 2$		
$\gcd(10,2)$	$10 = 2 \times 5 + 0$		
$\gcd(2,0) = 2$			

Example (cont.)

$\gcd(a, b)$	$a = bq + r$	$u = \tilde{v}$	$v = (\tilde{u} - q\tilde{v})$
$\gcd(122, 22)$	$122 = 22 \times 5 + 12$		
$\gcd(22, 12)$	$22 = 12 \times 1 + 10$		
$\gcd(12, 10)$	$12 = 10 \times 1 + 2$		
$\gcd(10, 2)$	$10 = 2 \times 5 + 0$	0	1
$\gcd(2, 0) = 2$		1	0

Example (cont.)

$\gcd(a, b)$	$a = bq + r$	$u = \tilde{v}$	$v = (\tilde{u} - q\tilde{v})$
$\gcd(122, 22)$	$122 = 22 \times 5 + 12$		
$\gcd(22, 12)$	$22 = 12 \times 1 + 10$		
$\gcd(12, 10)$	$12 = 10 \times 1 + 2$	1	$(0 - 1 \times 1) = -1$
$\gcd(10, 2)$	$10 = 2 \times 5 + 0$	0	1
$\gcd(2, 0) = 2$		1	0

Example (cont.)

$\gcd(a, b)$	$a = bq + r$	$u = \tilde{v}$	$v = (\tilde{u} - q\tilde{v})$
$\gcd(122, 22)$	$122 = 22 \times 5 + 12$		
$\gcd(22, 12)$	$22 = 12 \times 1 + 10$	-1	$(1 - 1(-1)) = 2$
$\gcd(12, 10)$	$12 = 10 \times 1 + 2$	1	-1
$\gcd(10, 2)$	$10 = 2 \times 5 + 0$	0	1
$\gcd(2, 0) = 2$		1	0

Example (cont.)

$\gcd(a, b)$	$a = bq + r$	$u = \tilde{v}$	$v = (\tilde{u} - q\tilde{v})$	sporadic checks
$\gcd(122, 22)$	$122 = 22 \times 5 + 12$	2	$-1 - 5 \times 2 = -11$	
$\gcd(22, 12)$	$22 = 12 \times 1 + 10$	-1	2	$-22 + 12 \cdot 2 \overset{\checkmark}{=} 2$
$\gcd(12, 10)$	$12 = 10 \times 1 + 2$	1	-1	$12 - 10 \overset{\checkmark}{=} 2$
$\gcd(10, 2)$	$10 = 2 \times 5 + 0$	0	1	
$\gcd(2, 0) = 2$		1	0	

$$\gcd(122, 22) = 122 \times 2 + 22 \times (-11)$$

Extended Euclidean algorithm (recursive)

Algorithm 2 Euclid(a, b : nonnegative integers, not both zero)

1: **if** $a < b$ **then**
   $(u, v, d) = \text{Euclid}(b, a)$
   return (v, u, d)
2: **else if** $b = 0$ **then**
   return $(1, 0, a)$
3: **else**
   $(q, r) \leftarrow$ quotient & remainder
   $(u, v, d) = \text{Euclid}(b, r)$
   return $(v, u - v * q, d)$
4: **end if**

Now we are in the position to prove the following result (stated earlier without proof).

THEOREM

Let $m > 1$ be integer.

The element $[a]_m \in \mathbb{Z}/m\mathbb{Z}$ has a multiplicative inverse iff $\gcd(a, m) = 1$.

Proof: $\gcd(a, m) = 1$ implies the existence of integers u and v such that $1 = au + mv$ (Bézout). Hence

$$[1]_m = [au + mv]_m = [au]_m = [a]_m[u]_m,$$

proving that $[u]_m$ is the inverse of $[a]_m$ in $\mathbb{Z}/m\mathbb{Z}$.

For the other direction, if $[u]_m$ is the inverse of $[a]_m$ in $\mathbb{Z}/m\mathbb{Z}$, then $[a]_m[u]_m = [1]_m$ or, equivalently, $[au]_m = [1]_m$. This implies that

$$au + mv = 1$$

for some integer v. If d is a divisor of both a and m, then we can write

$$\frac{a}{d}u - \frac{m}{d}v = \frac{1}{d}.$$

The left hand side is an integer, whereas the right hand side is an integer iff $d = \pm 1$. Hence 1 is the greatest integer that divides a and m. □

Corollary

$\gcd(a, m) = 1$ iff there exist integers u and v such that $1 = au + mv$.

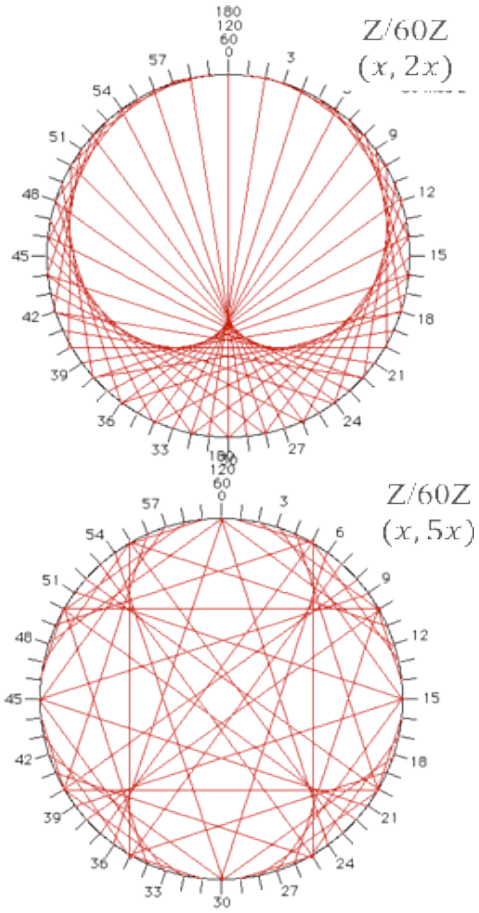
Proof:

If $\gcd(a, m) = 1$, by Bézout, there exist integers u and v such that $1 = au + mv$.

For the other direction, suppose that $1 = au + mv$, where u and v are integers. Then $[1]_m = [au + mv]_m = [au]_m = [a]_m[u]_m$, showing that $[u]_m$ is the inverse of $[a]_m$ in $\mathbb{Z}/m\mathbb{Z}$.

By the theorem that we just proved, $\gcd(a, m) = 1$. □

Week 8: Commutative Groups (Textbook Chapter 9)



Prof. Michael Gastpar

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

EPFL

Spring Semester 2025

Outline

After $\mathbb{Z}/m\mathbb{Z}$ we could proceed in two directions:

- focus on finite groups, which are finite sets with one operation, like $(\mathbb{Z}/m\mathbb{Z}, +)$. We do so now because we need them for cryptography.

- focus on finite fields, which are finite sets with two operations, like $(\mathbb{Z}/m\mathbb{Z}, +, \cdot)$, with the extra property that every non-zero element has a multiplicative inverse. We do so later as we need finite fields for channel coding.

We care about commutative groups because:

- they lead to exponentiation and logarithms
- which are the building blocks of various cryptographic algorithms, including DH, RSA, and ElGamal's encryption scheme.

Definition (Commutative group)

A **commutative group** (also called Abelian group) is a set G endowed with a binary operation $\star$ that combines any two elements a and b to form another element denoted $a \star b$. The group operation $\star$ must satisfy the following five axioms:

- (Closure:) For all $a, b \in G$, the result of the operation $a \star b$ is also in G.
- (Associativity:) For all $a, b \in G$, $a \star (b \star c) = (a \star b) \star c$.
- (Identity element:) There exists an element $e \in G$, such that for all $a \in G$, $a \star e = e \star a = a$.
- (Inverse element:) For all $a \in G$, there exists $b \in G$, such that $a \star b = b \star a = e$.
- (Commutativity:) For all $a, b \in G$, $a \star b = b \star a$.

Exercise

Which are commutative groups?

1. $(\mathbb{R}, +)$
2. $(\mathbb{R}, \cdot)$
3. $(\mathbb{R}\backslash\{0\}, \cdot)$
4. $(\mathbb{C}, +)$
5. $(\mathbb{Z}/m\mathbb{Z}, +)$
6. $(\mathbb{Z}/m\mathbb{Z}, \cdot)$
7. $(\mathbb{Z}/m\mathbb{Z}\backslash\{[0]_m\}, \cdot)$
8. $(\mathbb{N}, +)$
9. $(\mathbb{Z}, +)$
10. $(\mathbb{Z}\backslash\{0\}, \cdot)$

Solution

Which are commutative groups?

1. $(\mathbb{R}, +)$: Yes.
2. $(\mathbb{R}, \cdot)$: No, 0 has no inverse.
3. $(\mathbb{R}\backslash\{0\}, \cdot)$: Yes.
4. $(\mathbb{C}, +)$: Yes.
5. $(\mathbb{Z}/m\mathbb{Z}, +)$: Yes.
6. $(\mathbb{Z}/m\mathbb{Z}, \cdot)$: No, $[0]_m$ has no inverse.
7. $(\mathbb{Z}/m\mathbb{Z}\backslash\{[0]_m\}, \cdot)$: Only if m is prime.
8. $(\mathbb{N}, +)$: No.
9. $(\mathbb{Z}, +)$: Yes.
10. $(\mathbb{Z}\backslash\{0\}, \cdot)$: No, only 1 is invertible.

$(\mathbb{Z}/m\mathbb{Z}^*, \cdot)$

To obtain a commutative group with the modulo multiplication, we take only the elements of $\mathbb{Z}/m\mathbb{Z}$ that have a multiplicative inverse. The resulting set is denoted $\mathbb{Z}/m\mathbb{Z}^*$.

Theorem (Textbook Thm 9.1)

For every integer $m > 1$, $(\mathbb{Z}/m\mathbb{Z}^*, \cdot)$ is a commutative group.

Proof

Check the axioms: **closure**, associativity, identity element, inverse element, commutativity.

Definition (Textbook Def. 8.5)

Euler's $\phi(n)$ function (also called **Euler's totient function**) is the number of positive integers in $\{1, \ldots, n\}$ that are relatively prime to n.

Observations:

- Recall that two integers a and b are relatively prime iff $\gcd(a, b) = 1$.
- Hence 1 is relatively prime with every integer.
- $\phi(m)$ is the cardinality of $\mathbb{Z}/m\mathbb{Z}^*$.
- If p is prime, $\phi(p) = p - 1$.

Example

- $\phi(1) = 1$
- $\phi(2) = 1, \quad \mathbb{Z}/2\mathbb{Z}^* = \{1\}$
- $\phi(3) = 2, \quad \mathbb{Z}/3\mathbb{Z}^* = \{1, 2\}$
- $\phi(4) = 2, \quad \mathbb{Z}/4\mathbb{Z}^* = \{1, 3\}$
- $\phi(5) = 4, \quad \mathbb{Z}/5\mathbb{Z}^* = \{1, 2, 3, 4\}$
- $\phi(6) = 2, \quad \mathbb{Z}/6\mathbb{Z}^* = \{1, 5\}$
- $\phi(7) = 6, \quad \mathbb{Z}/7\mathbb{Z}^* = \{1, 2, 3, 4, 5, 6\}$

EXERCISE

Prove the following:

- If p is prime and k is a positive integer, $\phi(p^k) = p^k - p^{k-1}$.
- If p and q are distinct primes, $\phi(pq) = \phi(p)\phi(q) = (p-1)(q-1)$.

Solution (outline)

- In $\{1, 2, \ldots, p^k\}$, only the numbers $p, 2p, 3p, \ldots, p^{k-1}p$ are divisible by p.

  Hence $p^k - p^{k-1}$ elements of $\{1, 2, \ldots, p^k\}$ are not divisible by p.

- In $\{1, 2, \ldots, pq\}$, only pq is divisible by both, p and q.

  Hence, there are q elements that are divisible by p, p elements that are divisible by q, and one which is divisible by both.

  $pq - p - q + 1 = (p - 1)(q - 1)$ elements are divisible by neither.

EXERCISE

Below is the multiplication table of $(\mathbb{Z}/5\mathbb{Z}^*, \cdot)$. Every element of $\mathbb{Z}/5\mathbb{Z}^*$ shows up exactly once in every row. Is it surprising?

$\mathbb{Z}/5\mathbb{Z}^*$	$\times$	1	2	4	3
	1	1	2	4	3
	2	2	4	3	1
	4	4	3	1	2
	3	3	1	2	4

Solution

We have seen that in $\mathbb{Z}/m\mathbb{Z}$, when a^{-1} exists, the map $\mathbb{Z}/m\mathbb{Z} \to \mathbb{Z}/m\mathbb{Z}$

$$x \to ax$$

is a bijection.

Each row of the above table is such a map. (The same is true for each column.) □

Nota Bene:

- In $(\mathbb{Z}/m\mathbb{Z}, +)$, the identity element is $[0]_m$.
- In $(\mathbb{Z}/m\mathbb{Z}^*, \cdot)$, the identity element is $[1]_m$.

Recall that if $\mathcal{A}_1$ and $\mathcal{A}_2$ are sets, the cartesian product $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2$ is the set

$$\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2 = \{(a_1, a_2) : a_1 \in \mathcal{A}_1, a_2 \in \mathcal{A}_2\}.$$

Similarly, $(G, \star) = (G_1, \star_1) \times (G_2, \star_2)$ is the set $G = G_1 \times G_2$ endowed with the binary operation $\star$ defined by

$$(a_1, a_2) \star (b_1, b_2) = (a_1 \star_1 b_1, a_2 \star_2 b_2).$$

Example $((\mathbb{Z}/2\mathbb{Z}, +) \times (\mathbb{Z}/3\mathbb{Z}, +))$

$\mathbb{Z}/2\mathbb{Z}$	$+$	0	1
	0	0	1
	1	1	0

$\mathbb{Z}/3\mathbb{Z}$	$+$	0	1	2
	0	0	1	2
	1	1	2	0
	2	2	0	1

$+$	00	01	02	10	11	12
00	00	01	02	10	11	12
01	01	02	00	11	12	10
02	02	00	01	12	10	11
10	10	11	12	00	01	02
11	11	12	10	01	02	00
12	12	10	11	02	00	01

The Cartesian Product of Commutative Groups is a Commutative Group

Recall the axioms of a commutative group:

- (Closure:) For all $a, b \in G$, the result of the operation $a \star b$ is also in G.
- (Associativity:) For all $a, b \in G$, $a \star (b \star c) = (a \star b) \star c$.
- (Identity element:) There exists an element $e \in G$, such that for all $a \in G$, $a \star e = e \star a = a$.
- (Inverse element:) For all $a \in G$, there exists $b \in G$, such that $a \star b = b \star a = e$.
- (Commutativity:) For all $a, b \in G$, $a \star b = b \star a$.

and check that they apply to elements of the form $(a_1, a_2) \in (G_1, \star_1) \times (G_2, \star_2)$.

$(G_1, \star_1) \times (G_2, \star_2)$ is called the **product group**.

EXERCISE

Consider $(G, \star) = (G_1, \star_1) \times (G_2, \star_2)$, where $(G_1, \star_1) = (\mathbb{Z}/4\mathbb{Z}, +)$ and $(G_2, \star_2) = (\mathbb{Z}/3\mathbb{Z}^*, \cdot)$:

- evaluate $(3, 2) \star (1, 2)$;
- find the identity element;
- find the inverse element of $(3, 2)$.

Solution

In $(\mathbb{Z}/4\mathbb{Z}, +) \times (\mathbb{Z}/3\mathbb{Z}^*, \cdot)$:

- $(3, 2) \star (1, 2) = (0, 1)$;
- $e = (0, 1)$;
- the inverse of $(3, 2)$ is $(1, 2)$.

The operation $\star$ of a product group is called **product operation**.

NB: The product operation can be a component-wise addition, as in

Example $((\mathbb{Z}/2\mathbb{Z}, +) \times (\mathbb{Z}/3\mathbb{Z}, +))$

$\mathbb{Z}/2\mathbb{Z}$	$+$	0	1
	0	0	1
	1	1	0

$\mathbb{Z}/3\mathbb{Z}$	$+$	0	1	2
	0	0	1	2
	1	1	2	0
	2	2	0	1

$+$	00	01	02	10	11	12
00	00	01	02	10	11	12
01	01	02	00	11	12	10
02	02	00	01	12	10	11
10	10	11	12	00	01	02
11	11	12	10	01	02	00
12	12	10	11	02	00	01

Exercise

Which of the following are product groups?

- $(\mathbb{Z}/2\mathbb{Z}, \cdot) \times (\mathbb{Z}/3\mathbb{Z}, \cdot)$.
- $(\mathbb{Z}/2\mathbb{Z}^*, \cdot) \times (\mathbb{Z}/3\mathbb{Z}^*, \cdot)$.

Solution

- $(\mathbb{Z}/2\mathbb{Z}, \cdot) \times (\mathbb{Z}/3\mathbb{Z}, \cdot)$: Not a commutative group, because $(0, 0)$ has no inverse.
- $(\mathbb{Z}/2\mathbb{Z}^*, \cdot) \times (\mathbb{Z}/3\mathbb{Z}^*, \cdot)$: Indeed a commutative group.

EXERCISE

Let m and n be integers greater than 1.

- Is it true that the subset of $(\mathbb{Z}/m\mathbb{Z}, \cdot) \times (\mathbb{Z}/n\mathbb{Z}, \cdot)$ that consists of elements that have an inverse is a commutative group?
- If yes, is it the same commutative group as $(\mathbb{Z}/m\mathbb{Z}^*, \cdot) \times (\mathbb{Z}/n\mathbb{Z}^*, \cdot)$?

Solution

Yes to both questions.

In fact, $(G_1, \star_1) \times (G_1, \star_1)$ is a group iff both $(G_1, \star_1)$ and $(G_1, \star_1)$ are groups.

The subset of $(\mathbb{Z}/m\mathbb{Z}, \cdot)$ that contains all the elements of $(\mathbb{Z}/m\mathbb{Z}, \cdot)$ that have an inverse is a group, denoted $(\mathbb{Z}/m\mathbb{Z}^*, \cdot)$.

Similarly, ... (same argument with n instead of m).

Isomorphism

Some sets endowed with an operation might look different, but they are actually the same once their elements are re-labeled.

Definition

Let $(G, \star)$ and $(H, \otimes)$ be sets, each endowed with a binary operation.

An **isomorphism** from $(G, \star)$ to $(H, \otimes)$ is a bijection $\psi : G \to H$ such that

$$\psi(a \star b) = \psi(a) \otimes \psi(b)$$

holds for all $a, b \in G$.

We say that $(G, \star)$ and $(H, \otimes)$ are **isomorphic** if there exists an isomorphism between them.

Suppose that ψ is an isomorphism from $(G, \star)$ to $(H, \otimes)$. The following properties hold:

- If $(G, \star)$ is a commutative group, so is $(H, \otimes)$.
- If e is the identity element of $(G, \star)$, then $\psi(e)$ is the identity element of $(H, \otimes)$.
- If a, b are inverse of one another in $(G, \star)$, then $\psi(a), \psi(b)$ are inverse of one-another in $(H, \otimes)$.

From a group-theoretic viewpoint, isomorphic groups are the same object.

Proofs: For the first point, we show that if $(G, \star)$ is a commutative group, so is $(H, \otimes)$. To do so, each element of H is written as $\psi(x)$ for some $x \in G$.

- Closure: $\psi(a) \otimes \psi(b) = \psi(a \star b) \in H$;
- Associativity: No matter in which order we perform the operations on the LHS (left-hand side), $\psi(a) \otimes \psi(b) \otimes \psi(c) = \psi(a \star b \star c)$;
- Identity Element: $\psi(e) \otimes \psi(a) = \psi(e \star a) = \psi(a)$, proving that $\psi(e)$ is the identity element in $(H, \otimes)$;
- Inverse Element: $\psi(a) \otimes \psi(a^{-1}) = \psi(a \star a^{-1}) = \psi(e)$, showing that the inverse of $\psi(a)$ is $\psi(a^{-1})$;
- Commutativity: $\psi(a) \otimes \psi(b) = \psi(a \star b) = \psi(b \star a) = \psi(b) \otimes \psi(a)$.

We have also proved the other two points of the previous slide.

Example

$(\mathbb{Z}/2\mathbb{Z}, +)$ and $(\mathbb{Z}/4\mathbb{Z}^*, \cdot)$ are isomorphic.

$\mathbb{Z}/2\mathbb{Z}$	$+$	0	1
	0	0	1
	1	1	0

$\mathbb{Z}/4\mathbb{Z}^*$	$\times$	1	3
	1	1	3
	3	3	1

$$\psi : \mathbb{Z}/2\mathbb{Z} \to \mathbb{Z}/4\mathbb{Z}^*$$
$$0 \to 1$$
$$1 \to 3$$

- Check that $\psi([0]_2)$ is the identity element in $(\mathbb{Z}/4\mathbb{Z}^*, \cdot)$.
- Check that $\psi(-[1]_2)$ is the (multiplicative) inverse of $\psi([1]_2)$ in $(\mathbb{Z}/4\mathbb{Z}^*, \cdot)$.

EXERCISE

Are $(\mathbb{Z}/4\mathbb{Z}, +)$ and $(\mathbb{Z}/5\mathbb{Z}^*, \cdot)$ isomorphic?

$\mathbb{Z}/4\mathbb{Z}$	$+$	0	1	2	3
	0	0	1	2	3
	1	1	2	3	0
	2	2	3	0	1
	3	3	0	1	2

$\mathbb{Z}/5\mathbb{Z}^*$	$\times$	1	2	3	4
	1	1	2	3	4
	2	2	4	1	3
	3	3	1	4	2
	4	4	3	2	1

- Hint 1: match up identity elements.
- Hint 2: $[2]_4$ is the inverse of itself in $(\mathbb{Z}/4\mathbb{Z}, +)$.

Solution

The following correspondence is not negotiable:

- $0 \to 1$ (identity elements must match);
- $2 \to 4$ (inverses must match).

There are two ways to complete:

- $1 \to 2$ and $3 \to 3$

or

- $1 \to 3$ and $3 \to 2$.

Both form an isomorphism.

Exercise

Say why the following cannot be isomorphic:

- $(\mathbb{Z}/2\mathbb{Z}, +) \times (\mathbb{Z}/2\mathbb{Z}, +)$ and $(\mathbb{Z}/3\mathbb{Z}, +)$;
- $(\mathbb{Z}/2\mathbb{Z}, +) \times (\mathbb{Z}/2\mathbb{Z}, +)$ and $(\mathbb{Z}/4\mathbb{Z}, +)$.

Solution

- $(\mathbb{Z}/2\mathbb{Z}, +) \times (\mathbb{Z}/2\mathbb{Z}, +)$ and $(\mathbb{Z}/3\mathbb{Z}, +)$:
  They do not have the same cardinality.

- $(\mathbb{Z}/2\mathbb{Z}, +) \times (\mathbb{Z}/2\mathbb{Z}, +)$ and $(\mathbb{Z}/4\mathbb{Z}, +)$:
  They do have the same cardinality.
  In $(\mathbb{Z}/2\mathbb{Z}, +) \times (\mathbb{Z}/2\mathbb{Z}, +)$, the inverse of x is x.
  Not the case for $(\mathbb{Z}/4\mathbb{Z}, +)$.

Exercise

Find an isomorphism from $((0, +\infty), \cdot)$ to $(\mathbb{R}, +)$.

Solution

An isomorphism from $((0, +\infty), \cdot)$ to $(\mathbb{R}, +)$ is:

$$\begin{aligned} \psi : (0, +\infty) &\to \mathbb{R} \\ x &\mapsto \log(x) \end{aligned}$$

$$\psi : (x \cdot y) \mapsto \log(x) + \log(y).$$

Theorem (textbook Thm 9.4)

Let $(G, \star)$ be a finite commutative group with identity element e.

For every $a \in G$, there exists an integer $k \geq 1$, such that

$$\underbrace{a \star a \star \cdots \star a}_{k \text{ terms}} = e.$$

For the proof, we use the notation $a^k := \underbrace{a \star a \star \cdots \star a}_{k \text{ terms}}$.

For instance, in $(\mathbb{Z}, +)$, $a^3 = a + a + a$.

Proof:

- The commutative group is finite, hence the sequence

$$a, a^2, a^3, a^4, \ldots$$

  must contain repetitions.

- Suppose $a^i = a^j$ with $i < j$.
- By multiplying both sides by $(a^{-1})^i$ we obtain $e = a^{j-i}$. □

The order of a Group Element

Definition (textbook definition 9.4)

Let $(G, \star)$ be a finite commutative group with identity element e, and let $a \in G$.

The smallest positive integer k such that

$$\underbrace{a \star a \star \cdots \star a}_{k \text{ terms}} = e$$

is called the **order** of a.

Sometimes it is called the **period** of a. ("Période de a" in French.)

Example

The order of $[a]_{12} \in (\mathbb{Z}/12\mathbb{Z}, +)$ is the smallest k such that

$$\underbrace{[a]_{12} + [a]_{12} + \cdots + [a]_{12}}_{k \text{ terms}} = [0]_{12}.$$

- For $a = 3$, the order is 4.
- For $a = 4$, the order is 3.
- For $a = 5$, the order is 12.

Example

The order of $[a]_8 \in (\mathbb{Z}/8\mathbb{Z}^*, \cdot)$ is the smallest k such that

$$\underbrace{[a]_8 \cdot [a]_8 \cdots [a]_8}_{k \text{ terms}} = [1]_8.$$

Mind that $\mathbb{Z}/8\mathbb{Z}^* = \{[1]_8, [3]_8, [5]_8, [7]_8\}$).

- For $a = 1$, the order is 1.
- For $a = 3$, the order is 2.
- For $a = 5$, the order is 2.
- For $a = 7$, the order is 2.

Exercise

Find the order of every element in $((\mathbb{Z}/2\mathbb{Z})^2, +)$.

Solution

In $((\mathbb{Z}/2\mathbb{Z})^2, +)$, the identity is $([0]_2, [0]_2)$.

- $([0]_2, [0]_2)$ has order 1.
- $([0]_2, [1]_2)$ has order 2.
- idem for $([1]_2, [0]_2)$.
- idem for $([1]_2, [1]_2)$.

Example

$\mathbb{Z}/10\mathbb{Z}^* = \{1, 3, 7, 9\}$. Find the order of each element in $(\mathbb{Z}/10\mathbb{Z}^*, \cdot)$.

Hint: it is recommended to reduce intermediate results.

Solution

x	x^2	x^3	x^4	order
1				1
3	9	7	1	4
7	9	3	1	4
9	1			2

or, for instance,

x	x^2	x^3	x^4	order
1				1
3	-1	-3	1	4
7	-1	3	1	4
9	1			2

- Recall: An isomorphism ψ from $(G, \star)$ to $(H, \otimes)$ maps the identity element of $(G, \star)$ to the identity element of $(H, \otimes)$.
- This implies that the order of $g \in (G, \star)$ is the same as the order of $\psi(g) \in (H, \otimes)$.

Example

- In $(\mathbb{Z}/2\mathbb{Z}^2, +)$, the orders are $1, 2, 2, 2$.
- In $(\mathbb{Z}/10\mathbb{Z}^*, \cdot)$, the orders are $1, 4, 4, 2$.
- Hence the two commutative groups cannot be isomorphic.

The following result is given without proof:

THEOREM

Two finite commutative groups are isomorphic iff they have the same set of orders.

Let e be the identity element of a commutative group $(G, \star)$ and let $a \in G$.

Find the integers k such that $\underbrace{a \star a \star \cdots \star a}_{k \text{ terms}} = e$.

Example (addition)

$(G, \star) = (\mathbb{Z}/12\mathbb{Z}, +)$, $e = [0]_{12}$, $a = [2]_{12}$

k	1	2	3	4	5	6	7	8	9	$\cdots$
$([2]_{12})^k = k[2]_{12}$	2	4	6	8	10	0	2	4	6	$\cdots$

The values of k are the integer multiples of the order of a, which is 6.

Example (multiplication)

$(G, \star) = (\mathbb{Z}/10\mathbb{Z}^*, \cdot)$, $e = [1]_{10}$, $a = [3]_{10}$

k	1	2	3	4	5	6	7	8	9	$\cdots$
$([3]_{10})^k$	3	9	7	1	3	9	7	1	3	$\cdots$

The values of k are the integer multiples of the order of a, which is 4.

It is always like that: For $a \in (G, \star)$, $a^k = e$ when k is an integer multiple of the order of a.

This is not surprising: if q is the order of a and $k = qn$, we can write $a^k = (a^q)^n = e^n = e$. The following theorem states an even stronger result.

THEOREM

Let $(G, \star)$ be a commutative group and $a \in G$.

An integer k satisfies $\underbrace{a \star a \star \cdots \star a}_{k \text{ terms}} = e$ iff the order of a divides k.

Proof

Recall the notation: a^k means $\underbrace{a \star a \star \cdots \star a}_{k \text{ terms}}$.

- Let p be the order of a and write $k = pq + r, \;\; 0 \leq r < p$.
- $e = a^k = a^{pq+r} = (a^p)^q \star a^r = a^r$.
- $r = 0$, because p is the smallest positive integer such that $a^p = e$.
- Hence k is a multiple of p.

Example

- the order of $[2]_{12} \in (\mathbb{Z}/12\mathbb{Z}, +)$ is 6:

l	1	2	3	4	5	6	7	8	$\cdots$
$l[2]_{12}$	2	4	6	8	10	0	2	4	$\cdots$

- the order of $[3]_{10} \in (\mathbb{Z}/10\mathbb{Z}^*, \cdot)$ is 4:

l	1	2	3	4	5	6	$\cdots$
$([3]_{10})^l$	3	9	7	1	3	9	$\cdots$

$\mathbb{Z}/12\mathbb{Z}$ has cardinality 12 and the cardinality of $\mathbb{Z}/10\mathbb{Z}^* = \{1, 3, 7, 9\}$ is 4.

In both cases, the order divides the cardinality of the commutative group. A coincidence?

Theorem (Lagrange, Textbook Thm 9.3)

Let $(G, \star)$ be a finite commutative group of cardinality n. The order of each of its elements divides n.

Equivalence Relation and Equivalence Classes

To be ready for the elegant proof of Lagrange's theorem, we review the concept and the implication of an **equivalence relation** .

Relationships occur in many contexts in life. In math, they are represented by the structure called a **binary relation**.

Example

To relate people to their car, we can define

- a set A of all people;
- a set B of all cars;
- a set $R \subset A \times B$ that contains (a, b) iff person a owns car b.

The set R is called **a binary relation from A to B**.

The shorthand notations $a \sim b$ and $a\,R\,b$ mean the same as $(a, b) \in R$.

If the sets A and B are the same, then we speak of a **relation on** A.

An equivalence relation is a special case of a relation on a set. It is used to relate objects that are similar in some way, like in $\mathbb{Z}$, we may relate a and b if, for a specified m, $[a]_m = [b]_m$.

Definition

A relation on a set A is called an **equivalence relation** if it is *reflexive, symmetric, and transitive*.

Example

Let A be the set of all EPFL students.

Define $R = \{(a, b) \in A \times A : a \text{ and } b \text{ graduated from the same high school}\}$

R is an equivalence relation. In fact

- $a \sim a$ (reflexive);
- if $a \sim b$ then $b \sim a$ (symmetric);
- if $a \sim b$ and $b \sim c$ then $a \sim c$ (transitive).

Exercise

Let A be a set of people.

Define $R = \{(a, b) \in A \times A : a \text{ trusts } b\}$.,

Is this an equivalence relation?

Solution

No, this relation on A is not symmetric.

Exercise

Let A be the students of AICC-II.

Define
$R = \{(a, b) \in A \times A : a \text{ and } b \text{ got the same score in AICC-I or AICC-II}\}$.

Is this an equivalence relation?

Solution

No, this relation on A is not transitive.

Let R be an equivalence relation on A and $a \in A$.

By $[a]$ we denote the **equivalence class of** a:

$$[a] = \{b \in A : b \sim a\}.$$

Any element of an equivalence class can be used to represent the class: if $b \in [a]$ then $[b]$ and $[a]$ are the same class.

Every $a \in A$ is in one and only one equivalence class. In fact, if $a \in [b]$ and $a \in [c]$ then $[b] = [a] = [c]$.

To say it in a different way, an equivalence relation on A partitions A into equivalence classes: they are disjoint subsets of A and their union is A.

Example (continuation)

Let A be the set of all EPFL students.

Define $R = \{(a, b) \in A \times A : a \text{ and } b \text{ graduated from the same high school}\}$.

We can partition A into sets of students that graduated from the same high school. Each student is in exactly one such subset.

The following example is a special case of the construction used in the proof of Lagrange's Theorem.

Example

Let $(G, \star)$ be the group $\left(\mathbb{Z}/20\mathbb{Z}^*, \times\right) = \left(\{1, 3, 7, 9, 11, 13, 17, 19\}, \times\right)$.

Pick an arbitrary group element, e.g., $h = 7$.

Let $H = \{7, 9, 3, 1\}$ be the set that consists of all the powers of h.

We use H to define an equivalence relation on $G = \{1, 3, 7, 9, 11, 13, 17, 19\}$:

$$a \sim b \text{ if } ah^i = b \text{ for some } h^i \in H.$$

(This is an equivalence relation. We prove it later.) Let us construct the equivalence classes:

- $[1] = H = \{7, 9, 3, 1\}$;
- $[11] = \{17, 19, 13, 11\}$.

$G = [1] \cup [11]$. It is not a coincidence that all equivalence classes have the same cardinality. The cardinality of G must be a multiple of the cardinality of H.

Proof of Lagrange's Theorem:

- Let $(G, \star)$ be a finite commutative group of cardinality n.
- Let p be the order of $h \in G$.
- Let $H = \{h, h^2, h^3, \ldots, h^p = e\}$. (Note that $(H, \star)$ is itself a group, and is called a subgroup of G of cardinality p.)
- Define a relation on G:

$$a \sim b \quad \Leftrightarrow \quad \exists h^i \in H \text{ such that } a \star h^i = b.$$

- It is reflexive (H contains the identity element), symmetric (H contains the inverse of each of its elements), and transitive (the product of elements of H is in H) — hence $\sim$ is an equivalence relation.
- An equivalence relation splits G into equivalence classes.
- H is one such equivalence class.

- It suffices to show that each equivalence class has the same cardinality p. Then p must divide n.
- We show that there is a one-to-one map between H and each equivalence class.
- The equivalence class of b is $[b] = \{b \star h, b \star h^2, \ldots, b \star h^p\}$.
- Clearly the cardinality of $[b]$ is at most p.
- It is p because the map $f : H \to [b]$ that sends h^i to $b \star h^i$ is one-to-one.
- Proof by contradiction: $b \star h^i = b \star h^k$ implies $h^i = h^k$ (b has an inverse). But for $1 \leq i, k \leq p$, $h^i = h^k$ holds if and only if $i = k$.
- Hence all equivalence classes have the same cardinality p, which must divide n.

□

Example (Something Old)

- The cardinality of $(\mathbb{Z}/m\mathbb{Z}, +)$, is m.
- For each element $[a]_m \in \mathbb{Z}/m\mathbb{Z}$, $m[a]_m = [0]_m$.
- Hence the period of each element of $(\mathbb{Z}/m\mathbb{Z}, +)$ divides m.

In $(\mathbb{Z}/m\mathbb{Z}, +)$, Lagrange's Theorem says nothing new to us.

In $(\mathbb{Z}/m\mathbb{Z}^*, \cdot)$, Lagrange's Theorem is non-trivial.

Using the fact that the cardinality of $\mathbb{Z}/m\mathbb{Z}^*$ is Euler's $\phi(m)$, we obtain:

Corollary (Euler's Theorem, textbook corollary 9.4)

Let $m \geq 2$ be an integer. For all $a \in (\mathbb{Z}/m\mathbb{Z}^*, \cdot)$

$$a^{\phi(m)} = [1]_m.$$

Equivalently, for all integers a that are relatively prime with m,

$$a^{\phi(m)} \equiv 1 \pmod{m}.$$

The above theorem underlies the cryptographic method studied in the next chapter.

Corollary (Fermat's Theorem, textbook corollary 9.5)

Let p be prime. For all $a \in (\mathbb{Z}/p\mathbb{Z}, \cdot)$

$$a^p = a.$$

Equivalently, for all integers a ,

$$a^p \equiv a \pmod{p}.$$

Proof: It follows from Euler's Theorem, and $\phi(p) = p - 1$, that

$$a^{(p-1)} = [1]_p$$

holds for all $a \in (\mathbb{Z}/p\mathbb{Z}, \cdot)$, except for $a = [0]_p$.

By multiplying both sides by a we obtain

$$a^p = a,$$

which holds also for $a = [0]_p$. □

Example

- $2^3 \equiv 2 \pmod 3$
- $4^3 \equiv 4 \pmod 3$
- $5^3 \equiv 5 \pmod 3$
- etc.

Recall the Diffie-Hellman Setup

- Fix a large prime number p. Hereafter all the numbers are in $\{0, 1, \ldots, p-1\}$ and arithmetic is modulo p (more on it later).
- Pick a generator g. A generator has the property that g^i generates all elements in $\{1, 2, \ldots, p-1\}$ when $i = 0, 1, \ldots, p-2$.
- *Note:* Towards the end of this chapter, after introducing all of the algebra necessary, we will see that a generator always exists since we are in what is called a *cyclic group.*

Example

$p = 5$. The numbers are $\{0, 1, 2, 3, 4\}$.

$g = 2$ is a generator. Indeed:

i	g^i
0	1
1	2
2	4
3	3

We are now in a position to deliver on this.

Specifically: Exponentiation can be defined on any finite group, but its inverse, the logarithm, is well-defined only for cyclic groups.

Next, we define cyclic groups and study their properties.

Cyclic Groups

Given a finite commutative group $(G, \star)$, we can take any of its elements, say $g \in G$, and compute g^2, g^3, ..., until for some n (the order of g), $g^n = e$, where e is the identity in G.

The result is the group $H = \{e, g, g^2, \ldots, g^{n-1}\}$.

H is the cycle of a single element, g. Any finite group of cardinality n, that consists of the cycle of a group element g is called a **cyclic group of order n**, and g is called a **generator**. A generator is not necessarily unique.

Note: even if $(G, \star)$ is infinite and non-commutative, $(H, \star)$ is finite (by construction) and commutative. Indeed, $g^i \star g^k = g^{i+k} = g^k \star g^i$.

Example (cyclic group)

$(\mathbb{C}, \cdot)$ is an infinite group that contains $j = \sqrt{-1}$.

$$\left(H = \left\{j, j^2, j^3, j^4 = 1\right\}, \cdot\right)$$

is a cyclic group, and j as well as $-j$ are generators.

Example (cyclic group)

$\left(\mathbb{Z}/m\mathbb{Z}, +\right)$ is a cyclic group of order m and $g = 1$ is one of its generators.

Example (cyclic group)

$\left(\mathbb{Z}/5\mathbb{Z}^*, \times\right)$ is a finite commutative group. Its elements are $\{1, 2, 3, 4\}$. The group can be generated by the powers of 2. Hence the group is a cyclic group of order $n = 4$ and $g = 2$ is one of its generators.

All cyclic groups that have the same order are isomorphic.

Proof: Let $(G, \star)$ and $(H, *)$ be cyclic groups of order n generated by g and h, respectively.

The map

$$\begin{array}{rccc} \psi: & G & \to H & \\ & g^i & \mapsto & h^i. \end{array}$$

is an isomorphism: In fact

- it is a bijection and
- for $a = g^i$ and $b = g^j$ we have

$$\psi(a \star b) = \psi(g^i \star g^j) = \psi(g^{i+j}) = h^{i+j} = h^i * h^j = \psi(a) * \psi(b).$$

□

Let $(G, \star)$ be a cyclic group of order n generated by g.

Let $b = g^i$ be one of its elements, $1 \leq i \leq n$.

The order of b is the smallest k such that $b^k = g^{ik}$ equals e.

ik is the smallest multiple of n and i, i.e.,

$$k = \frac{\mathrm{lcm}(i, n)}{i} = \frac{n}{\gcd(i, n)}.$$

Example

$\left(\mathbb{Z}/5\mathbb{Z}^*, \times\right)$ is a cyclic group of order $n = 4$, and $g = 2$ is a generator.

Let $i = 2$ and consider the group element $b = g^i = 4$. The order of b is

$$\frac{n}{\gcd(i, n)} = \frac{4}{\gcd(2, 4)} = 2.$$

(Let us verify: $b^2 = \left(2^2\right)^2 = 1$, as it should.)

g^i is another generator iff it has order n, i.e. iff $\gcd(i, n) = 1$.

The number of such i in $\{1, \ldots, n\}$ is Euler's $\phi(n)$.

EXAMPLE

The elements of $\left(\mathbb{Z}/5\mathbb{Z}^*, \times\right)$ are $\{1, 2, 3, 4\}$, and $g_1 = 2$ is a generator.

Hence $\left(\mathbb{Z}/5\mathbb{Z}^*, \times\right)$ is a cyclic group of order 4.

There are $\phi(4) = 2$ generators, one for each i such that $\gcd(i, 4) = 1$. Those i are $i = 1$ and $i = 3$. The other generator is $g_2 = g_1^3 = 3$.

Recall that we have proved the following: a cyclic group of order n has $\phi(n)$ generators.

However, not all groups are cyclic.

Example (A non-cyclic group)

The elements of the group $\left(\mathbb{Z}/24\mathbb{Z}^*, \times\right)$ are $\{1, 5, 7, 11, 13, 17, 19, 23\}$.

The cardinality of this group is $n = 8$. However, it would be a mistake to conclude that the group has $\phi(8) = 4$ generators.

All we can say is that if it has a generator (in this case the group is a cyclic group of order 8), then it has 4 generators.

But in fact, this group has no generator: except for 1, all the elements have order 2.

For any element h of a finite commutative group $(G, \star)$, the discrete exponentiation h^i is well-defined for any integer i. (Note that i is an integer, not an element of $(G, \star)$.)

The discrete logarithm to the base $b \in G$ of $h \in G$ is the integer i such that $b^i = h$. This is well-defined (for every $h \in G$) iff $(G, \star)$ is a **cyclic group, and b is one of its generators**.

Let $(G, \star)$ be a cyclic group of order n generated by b. The discrete exponentiation to the base b is the map

$$\begin{array}{rccc} f: & \mathbb{Z}/n\mathbb{Z} & \to & G \\ & [i]_n & \mapsto & b^i. \end{array}$$

We prove that it is well-defined and that it is an isomorphism from $(\mathbb{Z}/n\mathbb{Z}, +)$ to $(G, \star)$.

Proofs:

We show that the map is well-defined: suppose that $[i]_n = [j]_n$, then $j = i + nk$ for some integer k, and

$$f([j]_n) = g^{i+nk} = g^i \star g^{nk} = g^i = f([i]_n).$$

Next we show tha the map is one-to-one: If $f([i]_n) = f([j]_n)$, then:

- $g^i = g^j$;
- $g^{i-j} = e$;
- $i - j \in \{0, n, 2n, \dots\}$;
- $[i]_n = [j]_n$.

By the pigeonhole principle, the map is also onto, hence it is a bijection.

Finally we prove that $f : \mathbb{Z}/n\mathbb{Z} \to G$ is an isomorphism:

$$f([i]_n + [j]_n) = g^{i+j} = g^i \star g^j = f([i]_n) \star f([j]_n).$$

□

The inverse map

$$\begin{array}{rccc} f^{-1}: & G & \to & \mathbb{Z}/n\mathbb{Z} \\ & a = b^i & \mapsto & [i]_n, \end{array}$$

is called the **discrete logarithm to the base** b. Naturally, we write

$$[i]_n = \log_b a.$$

Note that the usual rules for exp and log apply: Specifically, for any group generator b of order n, we have:

- $(a^i)^j = a^{ij}$;
- $a^i a^j = a^{i+j}$;
- $\log_b(c \star d) = \log_b c + \log_b d$;
  Mind that on the RHS we have elements of $(\mathbb{Z}/n\mathbb{Z}, +, \cdot)$;
- $\log_b a^k = [k]_n \log_b a$.

Complexity of the Discrete Exponentiation

For an element of a group of order n, discrete exponentiation requires at most $2\log_2 n$ operations. Let us count them:

- to compute a^k, $1 < k < n$, we write k in binary form using $L = \log_2 n$ bits:

$$k = \sum_{i=0}^{L-1} b_i 2^i, \text{ with } b_i \in \{0, 1\};$$

- now

$$\begin{aligned} a^k = a^{\sum_{i=0}^{L-1} b_i 2^i} &= \prod_{i=0}^{L-1} a^{b_i 2^i} \\ &= \prod_{i=0}^{L-1} \left(a^{2^i}\right)^{b_i} \\ &= \prod_{i=0}^{L-1} a_i^{b_i}, \end{aligned}$$

where $a_i = a^{2^i}$ is computed as follows:

$$\begin{aligned} a_0 &= a \\ a_1 &= a_0^2 \\ a_2 &= a_0^4 = a_1^2 \\ a_3 &= a_0^8 = a_2^2 \\ &\vdots \\ a_{L-1} &= a_0^{2^{L-1}} = a_{L-2}^2. \end{aligned}$$

- It takes $L-1$ operations to compute $a_1, \ldots, a_{L-1}$ It takes at most $L-1$ operations to compute $\prod_{i=0}^{L-1} a_i^{b_i}$. (No computation required to perform $a_i^{b_i}$.)

- The total number of operations is at most $2(L-1) < 2\log_2 n$. □

FINDING THE INVERSE

Recall that in ElGamal's scheme, to invert the function we compute the inverse of g^{yx}. To compute the multiplicative inverse of a number $[b]_m \in (\mathbb{Z}/m\mathbb{Z}^*, \cdot)$, we can proceed two ways:

1. we use Bézout to write $1 = \gcd(b, m) = bu + mv$, hence $[u]_m$ is the inverse;

2. we use the fact that $[b]_m^{\phi(m)} = 1$, hence $[b]_m^{\phi(m)-1}$ is the inverse.

Often Bézout is more efficient, but if m is prime, we know that $\phi(m) = m - 1$. Exponentiation can be done efficiently.

If we are in a cyclic group of order n, then we know that $b^n = 1$. Hence the inverse of b is b^{n-1}.

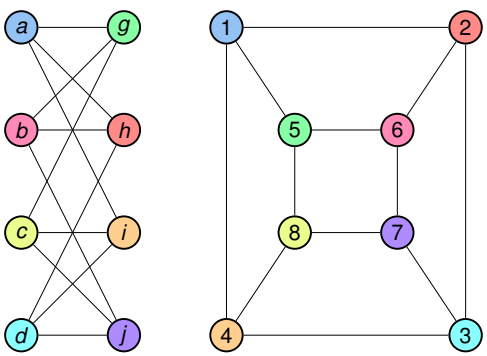
Week 9: Public-Key Cryptography (Textbook Chapter 10)

Prof. Michael Gastpar

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

EPFL

Spring Semester 2025

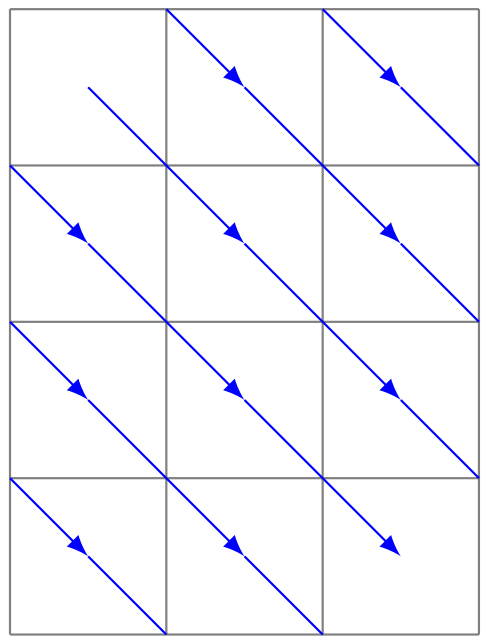


$f(a) = 1$
$f(b) = 6$
$f(c) = 8$
$f(d) = 3$
$f(g) = 5$
$f(h) = 2$
$f(i) = 4$
$f(j) = 7$

Outline

The Chinese Remainders Theorem

- Consider filling a table, going down diagonals, following the "torus rule"
- i.e., you start on the main diagonal . . .
- and when you drop off from an edge, you re-enter from the opposite edge.



Example (filled table)

Consider filling the table with the integers $0, 1, 2, \ldots, 23, \ldots$

0,12	4,16	8,20
9,21	1,13	5,17
6,18	10,22	2,14
3,15	7,19	11,23

Example (unfilled table)

Consider filling the table with the integers $0, 1, 2, \ldots, 7, \ldots$

0,4		2,6	
	1,5		3,7

In this case, the table will never be filled.

Question: under which conditions will the table eventually fill?

Mathematical formulation:

- we have $m_1 m_2$ numbers to be placed in $m_1 \times m_2$ drawers (m_1 rows and m_2 columns, matrix convention);
- we can see the numbers as elements of $\mathbb{Z}/m_1 m_2\mathbb{Z}$;
- and we can index the drawers with the elements of $\mathbb{Z}/m_1\mathbb{Z} \times \mathbb{Z}/m_2\mathbb{Z}$.

The placing

$$[k]_{m_1 m_2} \mapsto \left([k]_{m_1}, [k]_{m_2}\right)$$

can be seen as the action of a map

$$\psi : \mathbb{Z}/m_1 m_2\mathbb{Z} \to \mathbb{Z}/m_1\mathbb{Z} \times \mathbb{Z}/m_2\mathbb{Z}.$$

Is this map onto? (In which case it is a bijection).

EXAMPLE ($m_1 = 3$, $m_2 = 4$)

	$[0]_4$	$[1]_4$	$[2]_4$	$[3]_4$
$[0]_3$	$[0]_{12}$	$[9]_{12}$	$[6]_{12}$	$[3]_{12}$
$[1]_3$	$[4]_{12}$	$[1]_{12}$	$[10]_{12}$	$[7]_{12}$
$[2]_3$	$[8]_{12}$	$[5]_{12}$	$[2]_{12}$	$[11]_{12}$

map ψ

$$
\begin{aligned}
[0]_{12} &\mapsto ([0]_3, [0]_4) \\
[1]_{12} &\mapsto ([1]_3, [1]_4) \\
[2]_{12} &\mapsto ([2]_3, [2]_4) \\
[3]_{12} &\mapsto ([3]_3, [3]_4) = ([0]_3, [3]_4) \\
&\vdots \\
[7]_{12} &\mapsto ([7]_3, [7]_4) = ([1]_3, [3]_4) \\
[8]_{12} &\mapsto ([8]_3, [8]_4) = ([2]_3, [0]_4) \\
&\vdots
\end{aligned}
$$

Theorem (Chinese Remainders)

If m_1 and m_2 are **relatively prime**, the map ψ defined by

$$\psi : \mathbb{Z}/m_1 m_2\mathbb{Z} \to \mathbb{Z}/m_1\mathbb{Z} \times \mathbb{Z}/m_2\mathbb{Z}$$
$$[k]_{m_1 m_2} \mapsto ([k]_{m_1}, [k]_{m_2})$$

is

1. bijective
2. an isomorphism with respect to "$+$" and with respect to "$\cdot$".

If m_1 and m_2 are **not relatively prime**, ψ is neither onto nor one-to-one.

Example (bijective yes/no)

	0	1	2
0	0	4	8
1	9	1	5
2	6	10	2
3	3	7	11

$\gcd(m_1, m_2) = \gcd(4, 3) = 1$

$\Rightarrow$ bijective ψ

	0	1	2	3
0	0,4		2,6	
1		1,5		3,7

$\gcd(m_1, m_2) = \gcd(2, 4) \neq 1$

$\Rightarrow \psi$ is neither surjective nor injective

Example (Bijection $\Rightarrow$ isomorphism)

$\mathbb{Z}/12\mathbb{Z}$

0	1	2	3	4	5	6	7	8	9	10	11

$\mathbb{Z}/4\mathbb{Z} \times \mathbb{Z}/3\mathbb{Z}$

	0	1	2
0	0	4	8
1	9	1	5
2	6	10	2
3	3	7	11

isomorphism w.r.t. "$+$":

$$[8]_{12} + [10]_{12} = [6]_{12}$$
$$([0]_4, [2]_3) + ([2]_4, [1]_3) = ([2]_4, [0]_3)$$

isomorphism w.r.t. "$\cdot$":

$$[8]_{12} \cdot [2]_{12} = [4]_{12}$$
$$([0]_4, [2]_3) \cdot ([2]_4, [2]_3) = ([0]_4, [1]_3)$$

Proof that: m_1 and m_2 coprime $\Rightarrow$ ψ is bijective.

First we prove that ψ is one-to-one:

- suppose $[k]_{m_1} = [k']_{m_1}$ and $[k]_{m_2} = [k']_{m_2}$;
- $\Leftrightarrow$ m_1 and m_2 divide $(k - k')$;
- because m_1 and m_2 have no common factors, $m_1 m_2$ divides $(k - k')$;
- hence $[k]_{m_1 m_2} = [k']_{m_1 m_2}$;
- the map is one-to-one.

The function is bijective because it is one-to-one and the co-domain has the same cardinality as the domain. □

Proof that: m_1 and m_2 coprime $\Rightarrow$ Isomorphism w.r.t. "$+$"

By the definition of ψ and the modulo arithmetic,

$$\begin{aligned}[k+l]_{m_1 m_2} &\mapsto ([k+l]_{m_1}, [k+l]_{m_2}) \\ &([k]_{m_1} + [l]_{m_1}, [k]_{m_2} + [l]_{m_2}) \\ &([k]_{m_1}, [k]_{m_2}) + ([l]_{m_1}, [l]_{m_2})\end{aligned}$$

□

Proof that: m_1 and m_2 coprime $\Rightarrow$ Isomorphism w.r.t. "$\cdot$"

By the definition of ψ and the modulo arithmetic,

$$\begin{aligned}[k \cdot l]_{m_1 m_2} &\mapsto ([k \cdot l]_{m_1}, [k \cdot l]_{m_2}) \\ &\quad ([k]_{m_1} \cdot [l]_{m_1}, [k]_{m_2} \cdot [l]_{m_2}) \\ &\quad ([k]_{m_1}, [k]_{m_2}) \cdot ([l]_{m_1}, [l]_{m_2}).\end{aligned}$$

□

Proof that: $\gcd(m_1, m_2) \neq 1 \;\Rightarrow\;$ neither One-To-One nor Onto

We show that if $m_1 = aq$ and $m_2 = bq$, the map is not one-to-one.

- Consider $k = abq$
- Properties of k: $\quad 0 < k < m_1 m_2 = abq^2; \quad k = m_1 b; \quad k = m_2 a$
- Hence ψ maps $[k]_{m_1 m_2} \mapsto ([0]_{m_1}, [0]_{m_2})$
- But it maps also $[0]_{m_1 m_2} \mapsto ([0]_{m_1}, [0]_{m_2})$
- ψ is not one-to-one

Since the co-domain has the same cardinality as the domain, the map is not onto either. □

EXAMPLE

Find all solutions of $x^3 = [7]_{12}$, $x \in \mathbb{Z}/12\mathbb{Z}$.

Solution

Since $12 = 3 \times 4$ and $\gcd(3, 4) = 1$, $\psi : \mathbb{Z}/12\mathbb{Z} \to \mathbb{Z}/3\mathbb{Z} \times \mathbb{Z}/4\mathbb{Z}$ is an isomorphism w.r.t. $+$ and $\times$.

Instead of solving $x^3 = [7]_{12}$, we can work in $\mathbb{Z}/3\mathbb{Z} \times \mathbb{Z}/4\mathbb{Z}$ and solve

$$(x_1, x_2)^3 = ([7]_3, [7]_4).$$

Same as solving

$$\begin{cases} x_1^3 = [1]_3 & x \in \mathbb{Z}/3\mathbb{Z} \\ x_2^3 = [3]_4 & x \in \mathbb{Z}/4\mathbb{Z}. \end{cases}$$

The solution (by inspection) is

$$\begin{cases} x_1 = [1]_3 \\ x_2 = [3]_4 \end{cases} \Rightarrow x = [7]_{12}.$$

Example (recalling some old stuff)

Recall the following: $([x]_{96})^2 = [0]_{96} \quad \not\Rightarrow [x]_{96} = [0]_{96}$.
Reason:

- $96 = 2^5 \cdot 3$
- We can find a number, such as $k = 2^3 \cdot 3$, which fulfills $k < 96$ and k^2 is a multiple of 96.
- Hence $[k]_{96} \neq [0]_{96}$ and $[k^2]_{96} = [0]_{96}$.

However, for a prime modulus, like 97: $([x]_{97})^2 = [0]_{97} \quad \Rightarrow [x]_{97} = [0]_{97}$.
Reason:

- If $[x]_{97} = 0$ we are done. Otherwise, $\Rightarrow [x]_{97}$ has an inverse;
- $\Rightarrow [x]_{97} \cdot [x]_{97} = [0]_{97}$ implies $[x]_{97} = [0]_{97}$.

Example (How about this one)

$([x]_{77})^2 = [0]_{77}$ implies $[x]_{77} = [0]_{77}$?

Solution

- $77 = 7 \cdot 11$
- $(\mathbb{Z}/77\mathbb{Z}, \cdot)$ is isomorphic to $(\mathbb{Z}/7\mathbb{Z}, \cdot) \times (\mathbb{Z}/11\mathbb{Z}, \cdot)$
- Hence $[x]_{77} \cdot [x]_{77} = [0]_{77}$

  $\Leftrightarrow ([x]_7, [x]_{11}) \cdot ([x]_7, [x]_{11}) = ([0]_7, [0]_{11})$

  $\Leftrightarrow ([x]_7 \cdot [x]_7, [x]_{11} \cdot [x]_{11}) = ([0]_7, [0]_{11})$
- We are back to the prime modulus case

  which implies $[x]_7 = [0]_7$ and $[x]_{11} = [0]_{11}$

  which implies $[x]_{77} = [0]_{77}$.

If $\gcd(m_1, m_2) = 1$,

the Chinese remainders theorem says that

we can calculate in $\mathbb{Z}/m_1 m_2\mathbb{Z}$

or in $\mathbb{Z}/m_1\mathbb{Z} \times \mathbb{Z}/m_2\mathbb{Z}$

whichever is more convenient.

The Inverse Map

The map to

$$\psi : \mathbb{Z}/m_1 m_2\mathbb{Z} \to \mathbb{Z}/m_1\mathbb{Z} \times \mathbb{Z}/m_2\mathbb{Z}$$
$$[k]_{m_1 m_2} \mapsto ([k]_{m_1}, [k]_{m_2}),$$

is easy to compute.

How about the inverse map?

$$\psi^{-1} : \mathbb{Z}/m_1\mathbb{Z} \times \mathbb{Z}/m_2\mathbb{Z} \to \mathbb{Z}/m_1 m_2\mathbb{Z}.$$

We use the extended Euclid's algorithm to find integers u and v such that

$$1 = \gcd(m_1, m_2) = m_1 u + m_2 v.$$

Let

$$a = m_2 v,$$
$$b = m_1 u.$$

Notice that

$$[a]_{m_2} = [m_2 v]_{m_2} = [0]_{m_2},$$
$$[a]_{m_1} = [m_2 v]_{m_1} = [1 - m_1 u]_{m_1} = [1]_{m_1}.$$

Similarly,

$$[b]_{m_1} = [m_1 u]_{m_1} = [0]_{m_1},$$
$$[b]_{m_2} = [m_1 u]_{m_2} = [1 - m_2 v]_{m_2} = [1]_{m_2}.$$

Hence, for any integers k_1 and k_2,

$$\psi\big([ak_1 + bk_2]_{m_1 m_2}\big) = ([k_1]_{m_1}, [k_2]_{m_2}),$$

implying that

$$\begin{aligned}\psi^{-1} : \mathbb{Z}/m_1\mathbb{Z} \times \mathbb{Z}/m_2\mathbb{Z} &\to \mathbb{Z}/m_1 m_2\mathbb{Z} \\ ([k_1]_{m_1}, [k_2]_{m_2}) &\mapsto ([ak_1 + bk_2]_{m_1 m_2})\end{aligned}$$

is the inverse map.

Fermat + Chinese Remainders

- Let p and q be distinct primes
- let k be a multiple of both $(p-1)$ and $(q-1)$
- for all non-negative integers l,

$$([a]_p)^{lk+1} = [a]_p$$
$$([a]_q)^{lk+1} = [a]_q$$

- using the Chinese remainders theorem, we combine into

$$([a]_{pq})^{lk+1} = [a]_{pq}.$$

We have proved the following result:

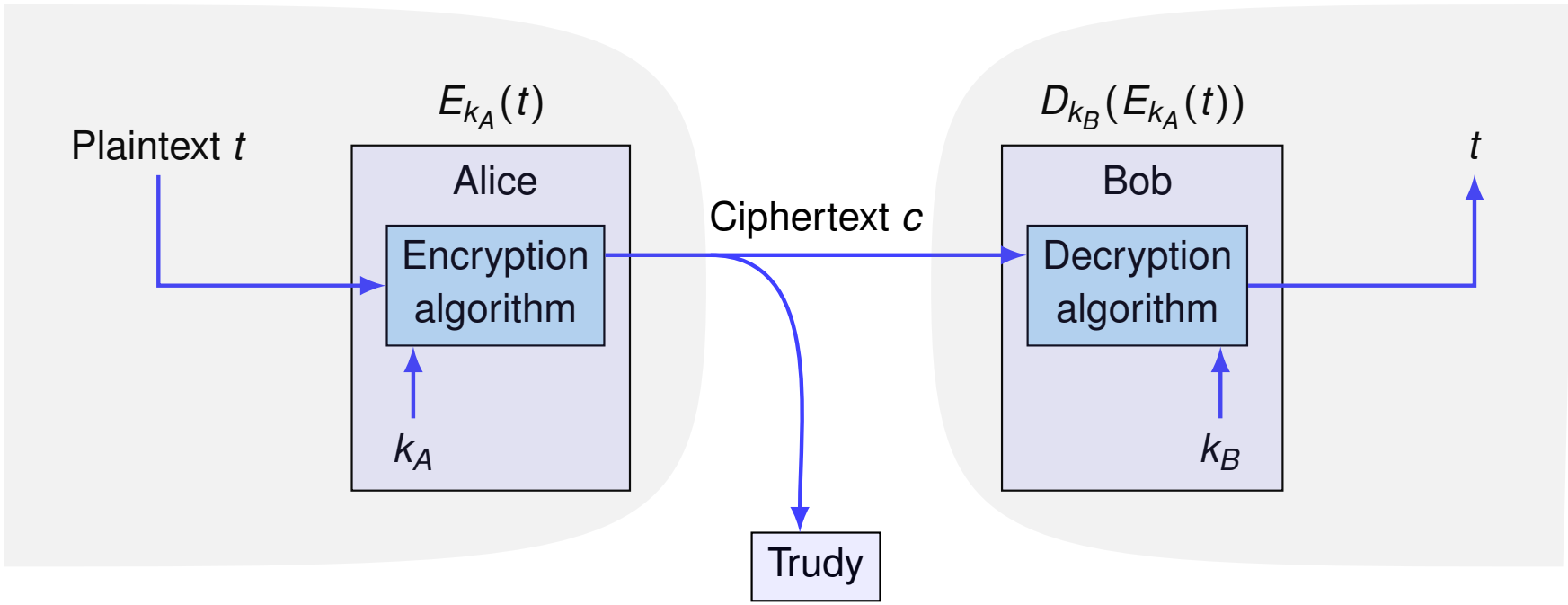
Theorem (Textbook Thm 10.3)

Let p and q be distinct prime numbers and let k be a multiple of both $(p-1)$ and $(q-1)$.

For every integer a, and every non-negative integer l,

$$([a]_{pq})^{lk+1} = [a]_{pq}.$$

- RSA: Rivest, Shamir, Adleman, 1977 (first public-key cryptosystem).



RSA HIGH-LEVEL: THE VERY ESSENCE OF IT

Suppose that we can find:

- integer m (modulus),
- integer e (encoding exponent),
- integer d (decoding exponent),

such that, for all integers $t \in \mathbb{Z}/m\mathbb{Z}$ (plaintext),

$$[(t^e)^d]_m = [t]_m.$$

Then:

- the receiver generates m, e, d (we will see how).
- (m, e) is the public encoding key — announced in a phone-like public directory.
- (m, d) is the private decoding key — d never leaves the receiver.
- To send the plaintext $t \in \mathbb{Z}/m\mathbb{Z}$,
- the encoder forms the cryptogram $c = t^e \mod m$. Exponentiation is easy.
- The intended decoder performs $c^d \mod m$ and obtains the plaintext t. Again, this is easy.

Example

$m = 33$, $e = 7$, $d = 3$

- suppose that the plaintext is $t = 2$
- encryption: $c = t^e \mod m = 2^7 \mod 33 = 128 \mod 33 = 29$
- decryption: $c^d \mod m = 29^3 \mod 33 = \cdots = 2$, as expected.

It works similarly with any $t \in \{0, \ldots, 32\}$.

NB: we may want to exclude $t = 0$, because from $c = 0$ we immediately infer $t = 0$.

RSA: KEYS GENERATION (AT THE RECEIVER)

- generate large primes p and q at random
- $m = pq$ is the modulus used for encoding and decoding
- let k be a multiple of $(p-1)$ and $(q-1)$, to be kept secret
- for instance, $k = \phi(pq)$ or $k = \mathrm{lcm}(p-1, q-1)$
- produce the public (encoding) exponent e such that $\gcd(e, k) = 1$
- (a common choice is $e = 65537 = 2^{16} + 1$ which is a prime number. No need for e to be distinct for each recipient)
- the public key is (m, e)
- k is kept secret. Using Bézout, the receiver produces the positive decoding exponent d such that

$$de + kl = 1.$$

- (m, d) is the private key.

RSA: HOW DECODING WORKS

$[t]_m \in \mathbb{Z}/m\mathbb{Z}$, with $m = pq$. Hence

$$\begin{aligned}
\left([t]_m^e\right)^d &= [t]_m^{ed} \\
&= [t]_{pq}^{1-kl} \\
&= [t]_{pq} \quad \text{Fermat + CRs} \\
&= [t]_m.
\end{aligned}$$

Example ("toy-key" generation)

- $p = 3,\ q = 11,\ m = 33,\ \ k = \mathrm{lcm}(2, 10) = 10$
- $e = 7$ which is relatively prime with k
- $d = 3$ (check that $ed \mod k = 1$)
- the public key is $(m, e) = (33, 7)$
- the private key is $(m, d) = (33, 3)$

Example (encrypting and decrypting with the "toy key")

- Each letter of the alphabet is converted into a number in $\{1, 2, \ldots, m - 1 = 32\}$ (we avoid 0, to avoid $c = 0 = t$).
- we use the natural order: $a \mapsto 1$, $b \mapsto 2$, etc.
- suppose we want to send the letter "b"
- the encoder maps it into the plaintext $t = 2$
- and encrypts: $c = t^e \mod 33 = 29$
- the decoder decrypts: $t = c^d \mod m = 2$
- and maps back $t = 2$ to the letter b.

In practice, m is very large, and the mapping

$$\text{text} \mapsto \mathbb{Z}/m\mathbb{Z}$$

is done in blocks of letters.

RSA: POSSIBLE ATTACKS

How to decrypt not knowing d? Here the possibilities (that we know of):

- factor m to find p and q. Very hard to do if m is large (say $\approx 2^{500}$).
- in $\mathbb{Z}/m\mathbb{Z}$, solve $c = x^e$ for x. Very hard to do if m is large.
- guess k (good luck!)
- guess t (good luck!)
- guess d (good luck!)

THE TRAPDOOR ONE-WAY FUNCTION BEHIND RSA

- The trapdoor one-way function is

$$t \mapsto c = t^e \mod m,$$

  where e is called the encoding exponent.

- Instead of publishing the function, it suffices to publish (m, e). This is called the public key.

- Someone that knows (m, d) can perform

$$c \mapsto t = c^d \mod m,$$

  where d is called the decoding exponent.

- Hence the trapdoor information is (m, d). It is called the private key.

We have used trapdoor one-way functions for privacy.

In conjunction with hash functions, they are equally suited for authenticity.

Hash Function

A hash function is a many-to-one function, used to map a sequence of arbitrary length to a fixed-length bit sequence of, say, 200 bits.

What we expect from a hash function, is that even the smallest change in the input produces a different output.

Ideally it should be so that one has to try about 2^{200} alternative inputs to hope to find a sequence that produces a given output.

To sign a document, we append to the document a hash function of the document in such a way that only the signee could have done it. This is done using a trapdoor one-way function as follows:

- let t be Alice's plaintext that she wants to sign;
- let f_A be Alice's trapdoor one-way function (publicly available);
- let h be a hash function (publicly available, the same function for everyone);
- the digital signature is $s = f_A^{-1}(h(t))$;
- the signed document is (t, s).

Alice sends t and signature:
$\left(t, s = f_A^{-1}(h(t))\right)$

$$\xrightarrow{(t,s)}$$

Bob verifies:
$h(t) \stackrel{?}{=} f_A(s)$

If $h(t)$ equals $f_A(s)$, Bob trusts that the plaintext t is authentic, since for anybody other than Alice, it is nearly impossible to compute s.

Note 1: For privacy, Alice can encrypt (t, s) using Bob's trapdoor one-way function f_B.

Note 2: Privacy relies on f_B; authenticity relies on f_A.

Trusted Agency

How do we know that the directory storing all the public keys has not been tampered with?

Example

Alice queries the public directory for Bob's public key.

The directory sends the message "Bob's public key is k".

Eve, who is sitting on the wire, substitutes "Bob's public key is k" with "Bob's public key is $\tilde{k}$", where $\tilde{k}$ is her own public key.

By using $\tilde{k}$ to encrypt, Alice believes that only Bob will be able to decrypt.

But in fact, Eve is the only person that can decrypt.

How to prevent this from happening?

The directory information is signed by a trusted agency, say Symantec. Here is how:

- Symantec's public key is distributed once and for all via a channel that cannot be tampered with (e.g. hard-coded into the crypto hardware).
- Each directory entry is digitally signed by Symantec. We call the result a certificate.
- Anybody that has Symantec's public key can verify that the information received from the directory is authentic.
- Once verified, Alice can be confident that she is using Bob's public key.

Here are examples of widely-used standards:

- SHA-1 through SHA-3 (Secure Hash Algorithm) family: cryptographic hash functions.
- DSA (Digital Signature Algorithm), ECDSA (Elliptic Curve DSA): standards for digital signature.
- DES (Data Encryption Standard), AES (Advanced Encryption Standard): symmetric-key encryption standards. They are faster than RSA and require less memory.
- RSA (Rivest Shamir Adleman): public-key crypto.

WHY NOT JUST RSA?

With RSA we can encrypt (provide privacy) and sign (verify authenticity). Why do we need other cryptographic standards?

- DSA is faster than RSA in signing (and ECDSA a more recent standard than DSA). When keys have the same length, DSA leads to a shorter signature. RSA 512 bits has been cracked, only a DSA 280 bits has been cracked.

- The symmetric-key standards (DES, AES) are faster than RSA and require less memory. Most CPUs now include hardware that makes AES very fast.

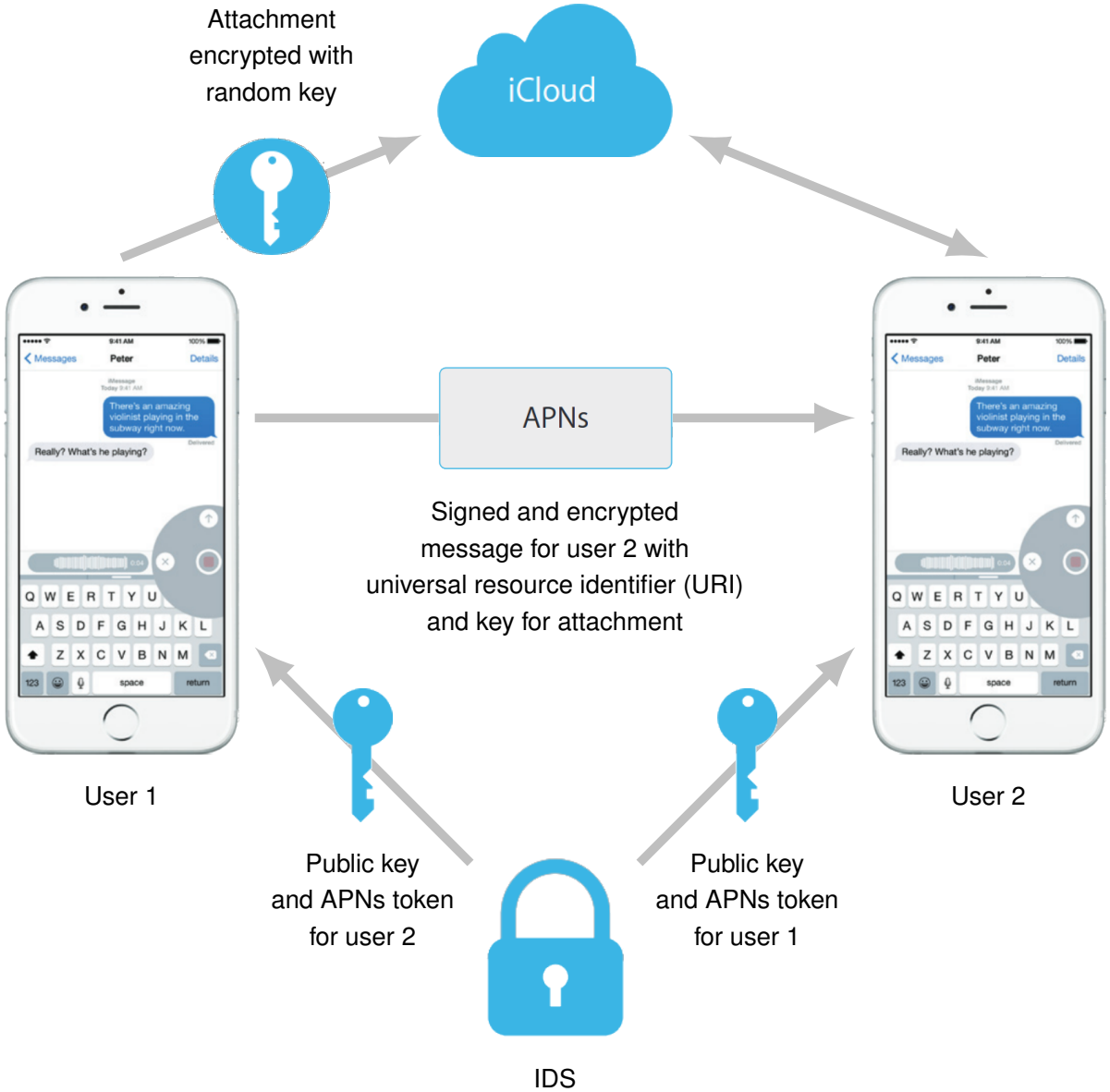
Cryptographic implementations, such as PGP (Pretty Good Privacy), available as a computer program, use symmetric-key and public-key cryptography, as well as digital-signature algorithms.

A COMPREHENSIVE EXAMPLE: HOW APPLE SENDS IMESSAGES

Apple uses all of the four standards mentioned above (SHA-1, ECDSA, AES, RSA). Let's see how.

To send an iMessage, Apple uses three services:

- IDS (Apple's directory service): It is here that public keys and device addresses are stored.
- APNs (Apple's Push Notification Service): outgoing messages are sent to this service. It is designed for short messages (like SMS).
- iCloud: to temporarily store what exceeds a maximum length. (Typically the case for a photo attachment.)


Attachment
encrypted with
random key
iCloud
APNs
Signed and encrypted
message for user 2 with
universal resource identifier (URI)
and key for attachment
User 1
User 2
Public key
and APNs token
for user 2
Public key
and APNs token
for user 1
IDS

When the iMessage service is enabled on an Apple device (iPhone, iPad, Mac):

- The device produces the RSA keys (public, private, each 1280 bits) and the ECDSA keys (public, private, each 256 bits).
- The two public keys are sent to the IDS.
- IDS associates the keys to the device's APN address, and lists the APN address(es) under the user's email address (or phone number).

We can think that Bob's IDS entry looks like this:

bob.cryptoexpert@epfl.ch	APN addr. (iMac)	RSA pub k
		ECDSA pub k
	APN addr. (iPhone)	RSA pub k
		ECDSA pub k
	APN addr. (iPad)	RSA pub k,
		ECDSA pub k
	APN addr. (MacBook Pro)	RSA pub k
		ECDSA pub k

When Alice sends a message to Bob using her Apple device, the following happens:

- The app looks in her contacts to find Bob's email address (or phone number),
- The app sends a request to the IDS, asking for Bob's APN addresses and corresponding RSA public keys.

For each of Bob's APN addresses, the following is done:

- The text, say t, is AES-encrypted with a randomly-generated symmetric key k to produce the cryptogram c_t;
- the key k is RSA-encrypted using Bob's public key, producing c_k;
- (c_t, c_k) are SHA-1-hashed and the result ECDSA-signed using Alice's private key, producing s;
- (c_t, c_k, s) is dispatched to the APN for delivery to the intended device.

Upon reception of (c_t, c_k, s), Bob's device does the following:

- Using Alice's public ECDSA key, the integrity of (c_t, c_k) is verified;
- using Bob's private RSA key, the cryptogram c_k is decrypted to obtain the AES symmetric key k;
- the cryptogram c_t is AES-decrypted to obtain the message t.

The APN can only relay messages up to a certain size (4 KB or 16 KB, depending on iOS).

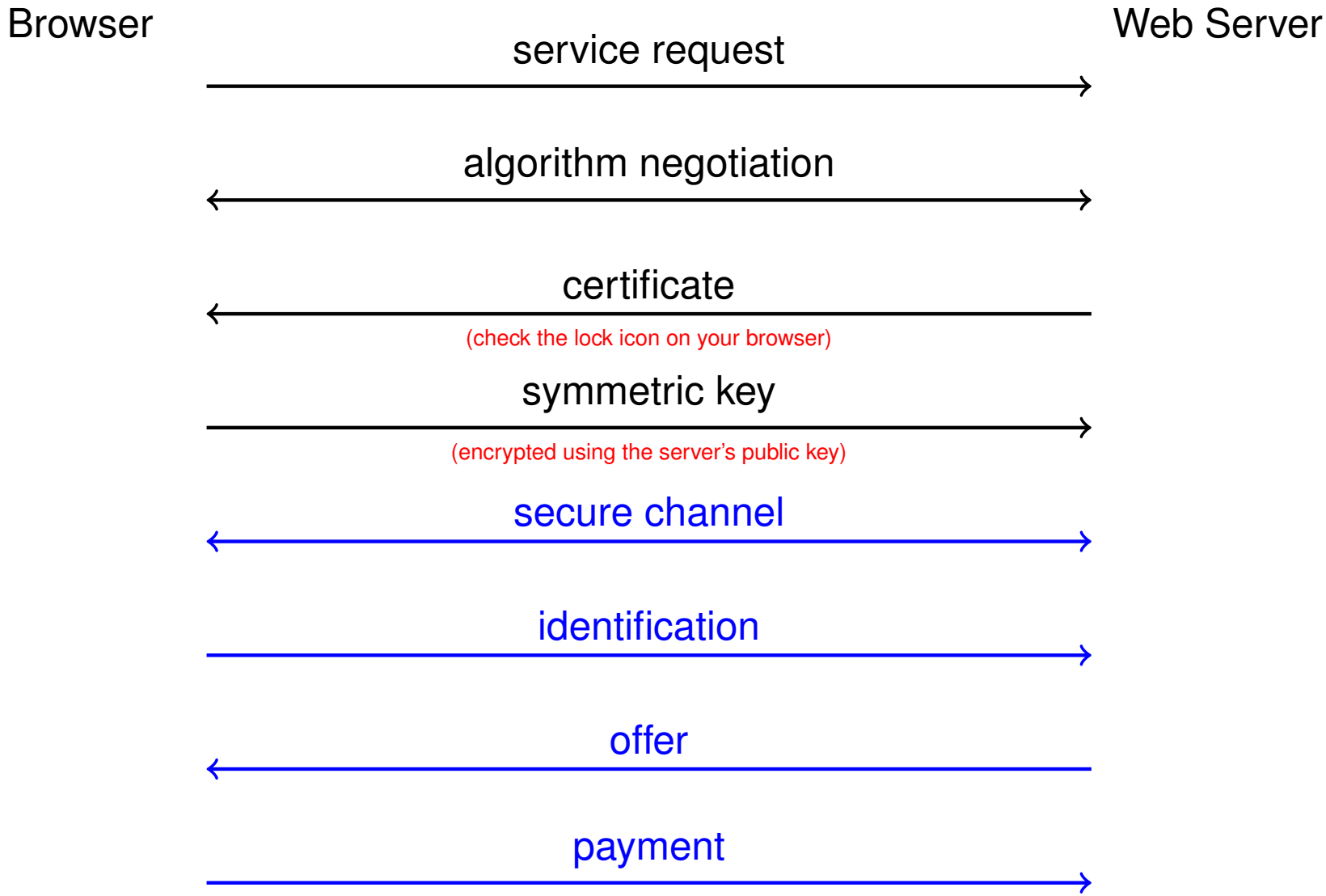
What exceeds this length, (e.g. a photo attachment), is AES-encrypted with a randomly-generated symmetric key, and the cryptogram is uploaded to iCloud.

The key, the URL, and the SHA-1 hash of the cryptogram are part of an iMessage sent to the recipient.

For further details, see the document: *iOS Security, iOS 9.0 or later, Sept. 2015.*

ANOTHER EXAMPLE: HTTPS

https (Hyper Text Transfer Protocol Secure) is the protocol used to exchange data between a browser and a web server. Sample transaction:



Outline

Summary of Chapter 2

Perfect secrecy is possible, but requires long keys.

- One-time pad

$$\text{Cryptogram} = \text{PlainText} \oplus \text{SharedKey}$$

  - If the SharedKey is perfectly (uniformly) random and shared between encrypter and decrypter ahead of time
  - and the SharedKey is kept secret from anyone else,
  - then the One-time Pad offers perfect secrecy.
  - Hence: It is expensive to implement. Only worth it for spies and such.

Summary of Chapter 2

- Practical cryptography is based on algorithmic/computational complexity.
- Public-key cryptography. Most public-key cryptographic algorithms fall into one of the following two categories:
  - those that are based on the belief that discrete exponentiation (in a multiplicative cyclic group) is a one-way function (e.g. Diffie-Hellman and ElGamal);
  - those that are based on the difficulty of factoring (e.g. RSA).
- To understand RSA and Diffie-Hellman, we need Number Theory and Algebra.

Summary of Chapter 2

Number Theory and Algebra

- Modulo operation, Euclid's algorithm
- Groups.
  - $\mathbb{Z}/m\mathbb{Z}$ with addition is always a group.
  - $\mathbb{Z}/m\mathbb{Z}$ with multiplication: need to retain only those elements that have a multiplicative inverse: $\mathbb{Z}/m\mathbb{Z}^*$
  - Finding multiplicative inverses in $\mathbb{Z}/m\mathbb{Z}$: Bézout's identity; Extended Euclid algorithm.
  - How many elements in $\mathbb{Z}/m\mathbb{Z}$ have a multiplicative inverse? Euler's totient function.
  - Group isomorphism.
  - Order of group elements. Lagrange's theorem: Order of any group element must divide the cardinality of the group.
- Product Groups. Main theorem: Cartesian product of groups is again a group.
- The special isomorphism between $\mathbb{Z}/m_1m_2\mathbb{Z}$ and $\mathbb{Z}/m_1\mathbb{Z} \times \mathbb{Z}/m_2\mathbb{Z}$ when m_1 and m_2 are coprime.
  - Holds for both addition and multiplication, including for elements that do not have a multiplicative inverse.
  - Hence, this is more than just a group isomorphim.

Computationally hard problem 1: Discrete logarithm.

- leads to **Diffie Hellman** (and, by slight extension, El Gamal)
  - Encryption: $A = g^a, B = g^b$.
  - Leads to a shared key: $C = A^b = B^a$.
  - To understand that it works, we need *cyclic groups.*

Computationally hard problem 2: Factorization of large integers.

- leads to **Cocks/RSA**
  - Encryption: $t^e \mod m$, where t is the plaintext and $m = pq$, where p and q are primes.
  - Decryption: $(t^e)^d \mod m$
  - To understand that it works (meaning that $(t^e)^d \mod m = t$ for all plaintexts t), we need to understand $\mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z}$.

Summary of Chapter 2

- Authenticity: Digital Signatures.
  - Can be done with the same algorithm!

- In practice, so-called symmetric-key cryptosystems are important. The common secret key is typically only a few hundred bits, distributed e.g. via Diffie-Hellman. Encryption/decryption can be implemented more efficiently (faster algorithms, smaller hardware). Think: one-time pad, but with an imperfect key. There is no proof that the resulting algorithm is secure.

Outline

Week 10: Error detection and error correction codes (Textbook Chapter 11)

Prof. Michael Gastpar

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

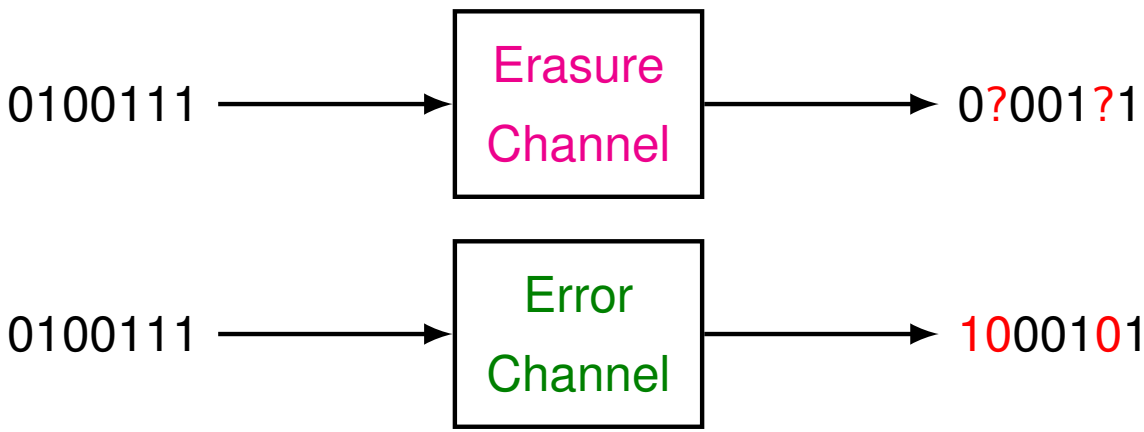
EPFL

Spring Semester 2025

Outline

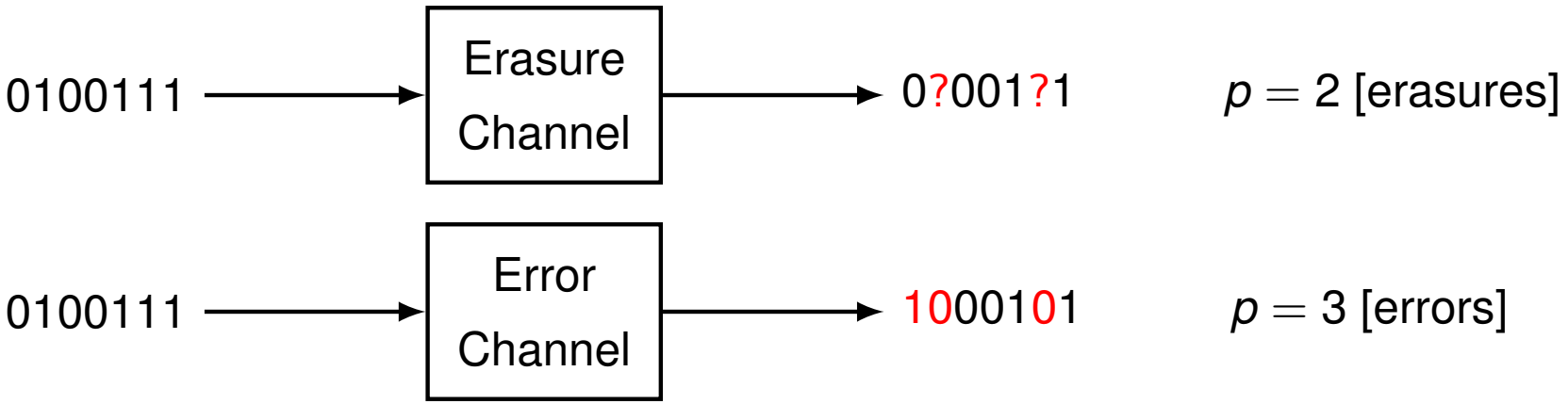
- The Internet often drops packets due to congestion.
- Not all the bits on a storage device can be retrieved.
- Wireless signals are very noisy.

We consider two types of channel models:



(The channel input alphabet is not necessarily binary.)

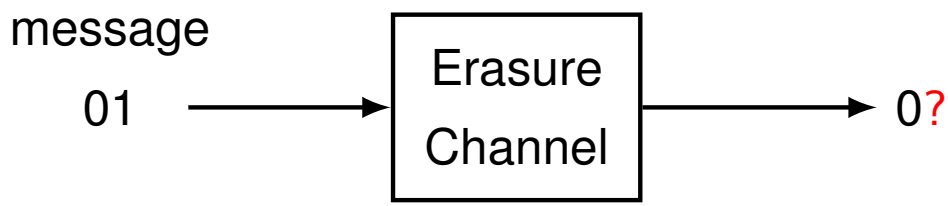
We define the **erasure weight** p (resp. **error weight** p) as the total number of erasures (resp. errors).



Erasures are easier to deal with: they are essentially channel errors of known location.

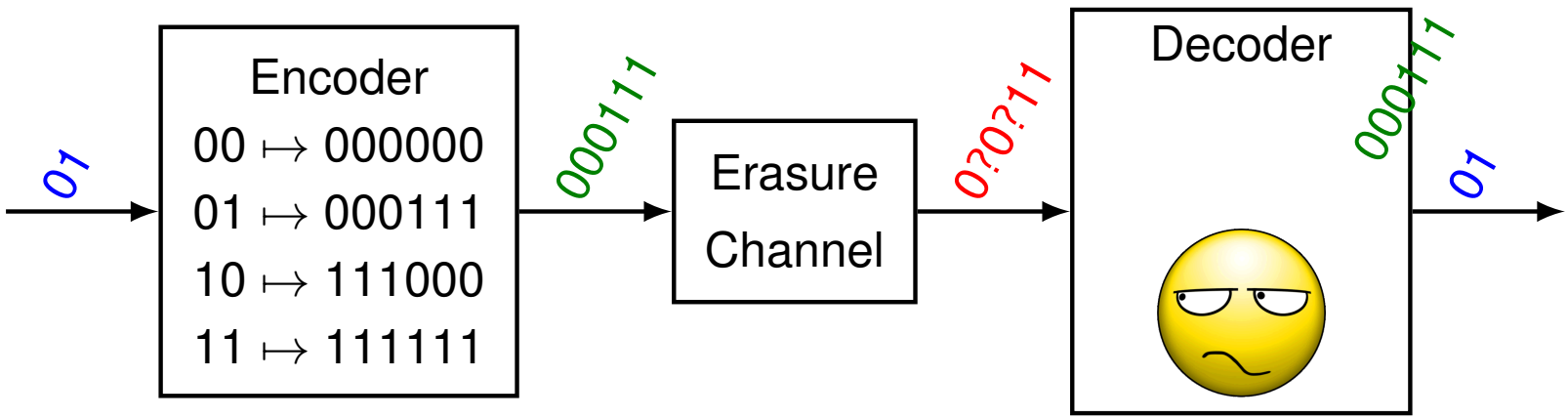
CHANNEL CODING TO DEAL WITH ERASURES

Suppose that the source outputs 2 bits, and we store them as is (no channel coding):



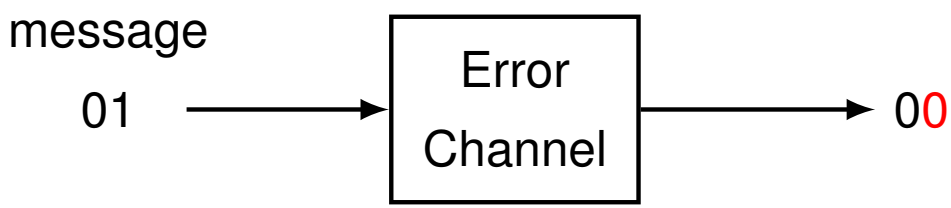
If any bit is erased, there is no way to determine the original message. (All hypotheses are equally valid.)

Now suppose that we do some channel coding:



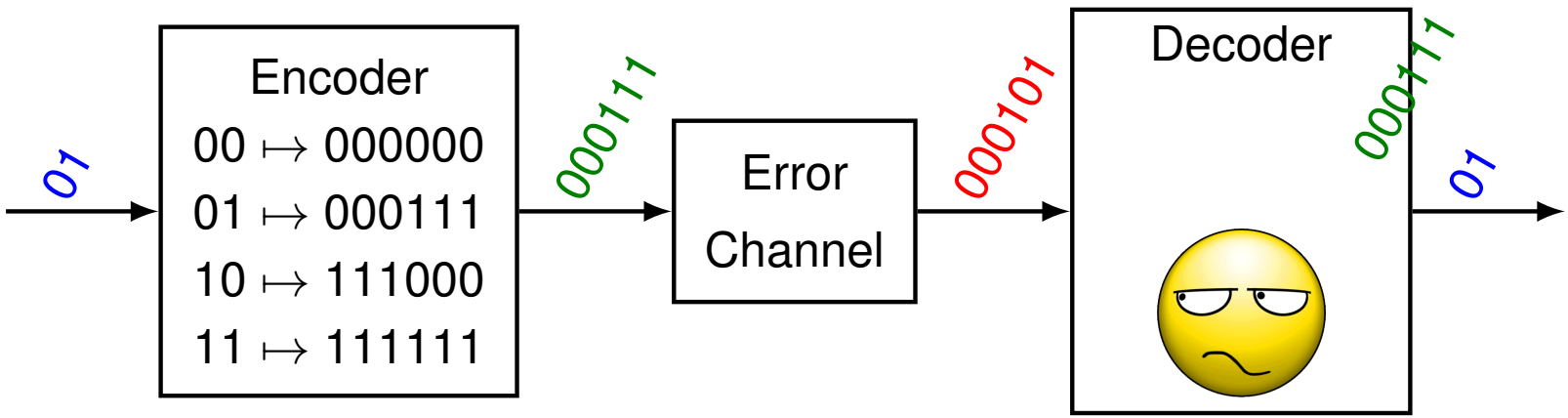
The decoder is able to fill the erased positions, because only one codeword is consistent with the observed channel output.

Suppose that the source outputs 2 bits, and we store them as is (no channel coding):



There is no way to tell that the channel flipped a bit.

Now suppose that we do channel coding:



The channel output is not a valid codeword. The decoder recognizes it, and assumes that the transmitted codeword is the one that agrees in most positions with the observed channel output.

WE STUDY ONLY BLOCK CODES

The above is an (n, k) **block code** with $n = 6$ and $k = 2$: each k source symbols are substituted by n channel symbols over the same alphabet.

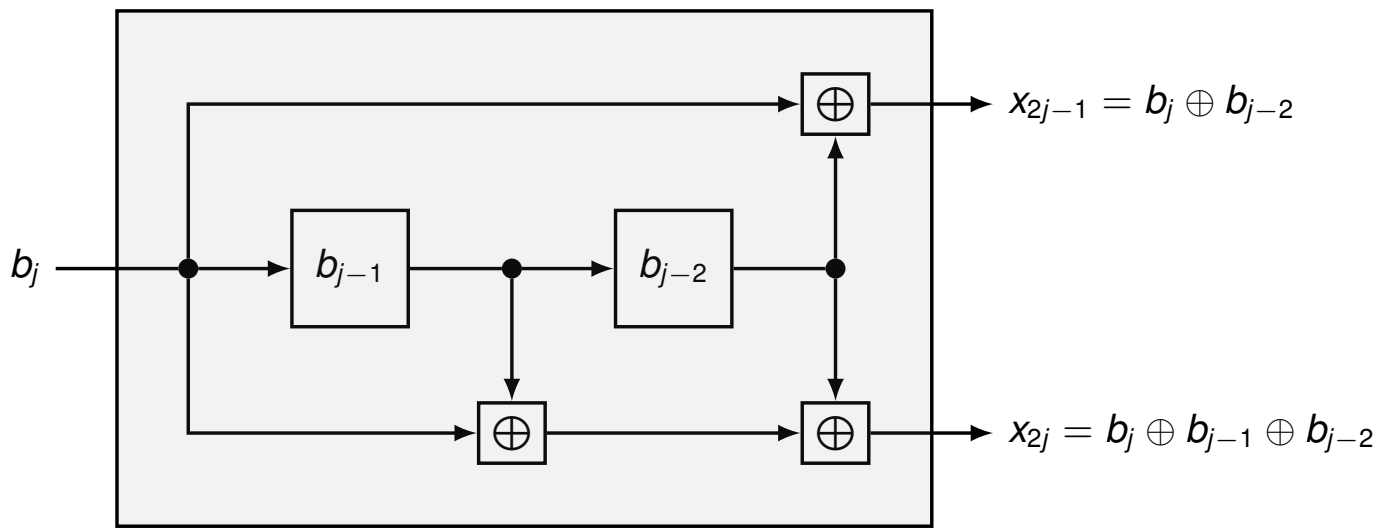
Since the alphabet is $\{0, 1\}$, we call it a **binary** (n, k) block code.

We consider only block codes.

EXAMPLE OF A NON-BLOCK CODE

The following is a convolutional encoder: every encoder input symbol produces two encoder output symbols.

The output pair produced at any given time is a linear function of the corresponding encoder input and encoder state (the previous two inputs).



(The name comes from linear system theory, done in your second year.)

TERMINOLOGY

- The code $\mathcal{C}$ is the set of codewords.
- A codeword c is an element of $\mathcal{A}^n$. (The alphabet $\mathcal{A}$ is $\{0, 1\}$ in our example).
- The block-length is n.
- Each codeword carries $k = \log_2 |\mathcal{C}|$ information bits. ($k = \log_2 8 = 3$ bits in our example.)
- The rate is $\frac{k}{n}$ bits per symbol.

Encoder

$k = 3$		$n = 7$
000	$\mapsto$	0000000
001	$\mapsto$	0011100
010	$\mapsto$	0111011
100	$\mapsto$	1110100
011	$\mapsto$	0100111
101	$\mapsto$	1101000
110	$\mapsto$	1001111
111	$\mapsto$	1010011

The Hamming distance $d(x, y)$ between two n-tuples x and y is the number of positions in which they differ.

Example (Hamming distance)

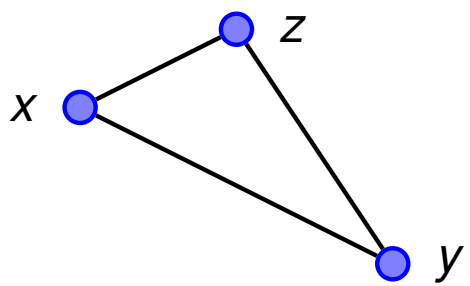
- $x = (101110)$, $y = (100110)$, $d(x, y) = 1$
- $x = (0427222)$, $y = (1227986)$, $d(x, y) = 5$
- $x = (0427222)$, $y = (0427222)$, $d(x, y) = 0$
- $x = (00)$, $y = (22)$, $d(x, y) = 2$

The Hamming distance is indeed a distance

In math, a function of two variables is a **distance if it satisfies the following axioms:**

Definition (distance axioms)

1. **non-negativity:** $d(x, y) \geq 0$ **with equality iff** $x = y$**.**
2. **symmetry:** $d(x, y) = d(y, x)$**.**
3. **triangle inequality:** $d(x, z) \leq d(x, y) + d(y, z)$**.**



Proof: We need to check that the triangle inequality holds.

Let $x = (x_1, \ldots, x_n)$, $y = (y_1, \ldots, y_n)$, $z = (z_1, \ldots, z_n)$.

The Hamming distance is additive in the sense $d(x, z) = \sum_{i=1}^{n} \underbrace{d(x_i, z_i)}_{0 \text{ or } 1}$.

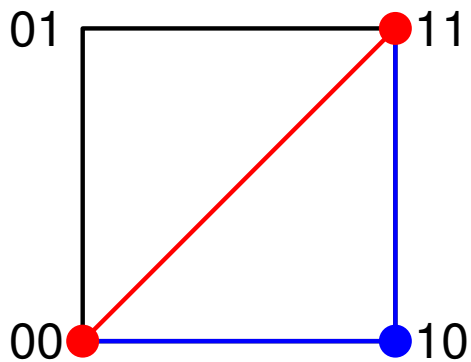
- if $d(x_i, z_i) = 0$, then $d(x_i, z_i) \leq d(x_i, y_i) + d(y_i, z_i)$.
- if $d(x_i, z_i) = 1$, then either $d(x_i, y_i) = 1$ or $d(y_i, z_i) = 1$ or both.
- Hence $d(x_i, z_i) \leq d(x_i, y_i) + d(y_i, z_i)$.
- By adding over all i,

$$d(x, z) = \sum_{i=1}^{n} d(x_i, z_i) \leq \sum_{i=1}^{n} \big(d(x_i, y_i) + d(y_i, z_i)\big) = d(x, y) + d(y, z).$$

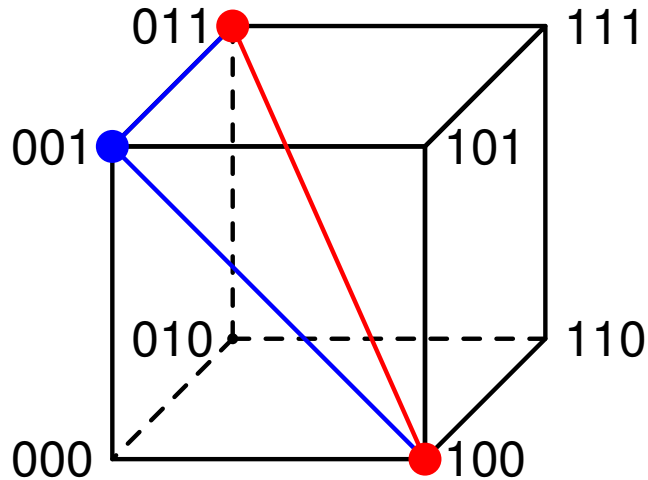
□

Geometrical Interpretation

An n-length sequence of integers may be seen as an element of $\mathbb{R}^n$.



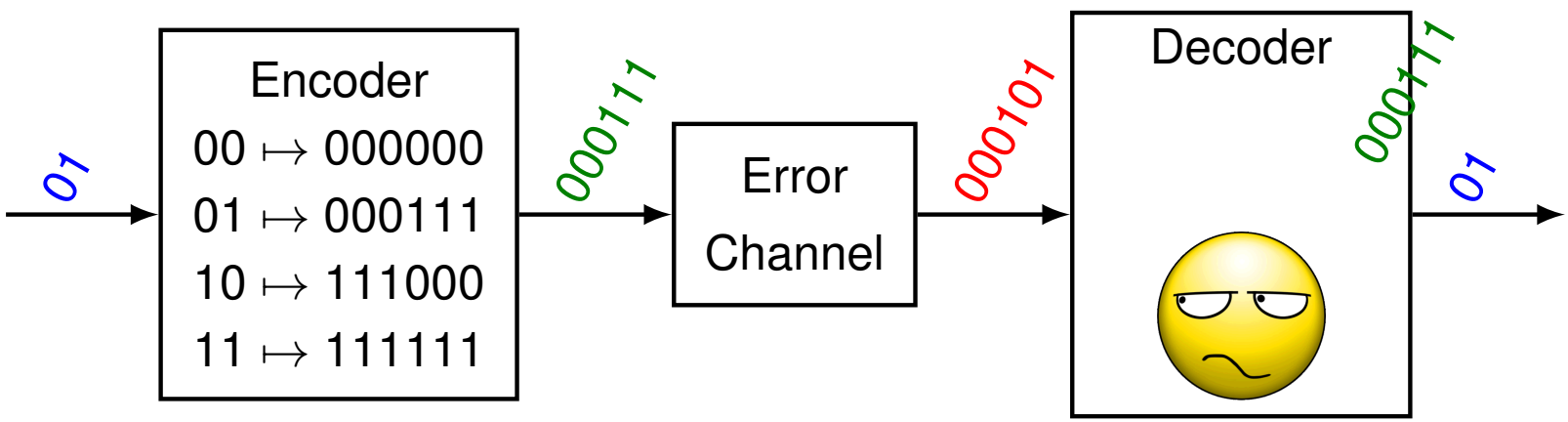
$d(00, 11) \leq d(00, 10) + d(10, 11)$



$d(011, 100) \leq d(011, 001) + d(001, 100)$

MINIMUM-DISTANCE DECODER

- The decoder guesses the encoder input based on the channel output.
- Here we consider only **minimum-distance decoders**.



Let y be the channel output observed by the decoder. A **minimum-distance decoder** decides that the channel input is (one of) the $\hat{c} \in \mathcal{C}$ for which $d(y, \hat{c})$ is minimized:

$$\hat{c} = \arg\min_{x \in \mathcal{C}} d(y, x)$$

The justification is that a small error weight is more likely than a large one.

Example (minimum-distance decoder)

Let $y = (0110111)$ be the channel output.

The encoder decides that the channel input was $\hat{c} = (0100111)$.

	Encoder	
$k = 3$		$n = 7$
000	$\mapsto$	0000000
001	$\mapsto$	0011100
010	$\mapsto$	0111011
100	$\mapsto$	1110100
011	$\mapsto$	0100111
101	$\mapsto$	1101000
110	$\mapsto$	1001111
111	$\mapsto$	1010011

Definition (minimum distance)

The minimum distance of a code $\mathcal{C}$ is

$$d_{min}(\mathcal{C}) = \min_{x,y \in \mathcal{C}; x \neq y} d(x, y)$$

Example

$\mathcal{C} = \{000000, 100110, 011001, 111111\} \Longrightarrow d_{min}(\mathcal{C}) = 3.$

WHAT TO EXPECT FROM A DECODER

For an error channel:

(1) channel-error correction: the best is if the decoder recognizes and corrects the channel errors. In this case, the encoder input is recovered error-free.

(2) channel-error detection: in some circumstances, the encoder is able to detect the presence of channel errors but it is unable to correct them. The receiver may or may not ask for retransmission.

(3) decoding error: the worse is if the decoder tries to do as in (1) and makes the wrong decision.

For an erasure channel:

(1) erasure-correction: the best is if the decoder is capable of filling in the erased positions. In this case, the encoder input is recovered error-free.

(2) (erasure detection: unlike errors, erasures are always detected.)

(3) decoding error: the worse is if the decoder fills-in one or more erased positions with incorrect symbols.

Error detection: how it relates to $d_{min}(\mathcal{C})$?

Theorem (Error detection: textbook theorem 11.2)

1. Channel errors of weight $p < d_{min}(\mathcal{C})$ do not lead to a codeword. Hence they are detected.
2. Some channel errors of weight $p \geq d_{min}(\mathcal{C})$ do lead to another codeword. Hence they cannot be detected by a minimum-distance decoder.

Note: Erasures are always detected (by definition).

Proof:

1.
   - Let $c \in \mathcal{C}$ be transmitted and y be received.
   - If $p = d(c, y) < d_{min}(\mathcal{C})$, y cannot be a codeword, therefore the error is detected.

2.
   - We construct an example in which a channel error of weight $p = d_{min}(\mathcal{C})$ cannot be detected.
   - Let c and c' be codewords at distance $d_{min}(\mathcal{C})$.
   - Suppose that c is the channel input and the channel output is $y = c'$.
   - y is a codeword. A minimum-distance decoder will decide that no channel error has occurred.

□

Example (Error Detection)

Let the encoding map be the MOD 97-10 procedure:

$$u \mapsto v = (100 \cdot u) + \left(98 - [100 \cdot u]_{97}\right)$$

Recall that v is considered as valid if $[v]_{97} = 1$.

For example, $u = 0216936631 \mapsto v = 021693663165$.

Suppose v is transmitted and v' is received, $d(v, v') = 1$.

We can always write $v' = v + a10^k$ with $a \in \{-9, \ldots, -1, 1, \ldots, 9\}$.

The only way for v' to be a valid codeword is if $\left[a10^k\right]_{97} = 0$.

Since $[10]_{97}$ is invertible, so is $[10^k]_{97}$, hence $a = 0$.

Therefore all weight 1 errors are detected, implying that the minimum distance is at least 2.

ERASURE CORRECTION: HOW IT RELATES TO $d_{min}(\mathcal{C})$?

THEOREM (ERASURE CORRECTION: TEXTBOOK THEOREM 11.3)

A minimum-distance decoder for a code $\mathcal{C}$ **corrects** (fills in) **all the erasures** of weight p iff $p < d_{min}(\mathcal{C})$.

Proof

$\Leftarrow$: Suppose that $p < d_{min}(\mathcal{C})$.

Let c and y be the input and the output of an erasure channel, respectively, with $d(c, y) = p$.

We show that there is only one way to fill in the erased positions.

Let $c \in \mathcal{C}$ and $\tilde{c} \in \mathcal{C}$ be two codewords that agree with y in the non-erased positions.

Clearly $d(c, \tilde{c}) \leq p < d_{min}(\mathcal{C})$. This is possible only if $c = \tilde{c}$.

□

Proof

$\Rightarrow$: We use contraposition.

Suppose that $p = d_{min}(\mathcal{C})$.

We construct an example where the decoder will not always decode correctly.

Let c and c' be codewords at distance $d_{min}(\mathcal{C})$.

Let c be the channel input, and suppose that the channel outputs the y obtained by erasing the p components of c that differ from c'.

Notice that $d(c, y) = p = d(c', y)$.

If c is a minimum-distance codeword, then so is c'.

□

ERROR CORRECTION: HOW IT RELATES TO $d_{min}(\mathcal{C})$?

THEOREM (ERROR CORRECTION: TEXTBOOK THEOREM 11.4)

A minimum-distance decoder for a code $\mathcal{C}$ **corrects all channel errors** of weight p iff $p < \frac{d_{min}(\mathcal{C})}{2}$.

Proof of $\Leftarrow$:

Let c and y be the input and the output of an error-channel, and suppose that $d(c, y) = p < \frac{d_{min}(\mathcal{C})}{2}$.

Let $\hat{c} \in \mathcal{C}$ be the guess made by a minimum-distance decoder that observes y.

We prove that $\hat{c} = c$.

$d(y, \hat{c}) \leq p$ because $d(y, c) = p$.

By the triangle inequality, $d(c, \hat{c}) \leq d(c, y) + d(y, \hat{c}) \leq 2p < d_{min}(\mathcal{C})$.

Hence $\hat{c} = c$. □

Proof of $\Rightarrow$: We use contraposition.

We have seen that if $p = d_{min}$, then the channel output can be a different codeword.

Hence it suffices to consider $d_{min} > p \geq \frac{d_{min}(\mathcal{C})}{2}$.

We construct an error pattern of weight p that cannot be corrected.

Let c and c' be codewords at distance $d_{min}(\mathcal{C})$.

Let y be obtained as follows: of the $d_{min}(\mathcal{C})$ positions where c and c' disagree, p positions are chosen as in c'. All the remaining positions are chosen as in c. By construction,

$$d(c, y) = p$$
$$d(c', y) = d_{min}(\mathcal{C}) - p \leq 2p - p = p.$$

We see that c' is at least as close to y as c. □

Detection/Correction Summary

	detection guaranteed if	correction guaranteed if
erasure channel	(not applicable)	$p < d_{min}$
error channel	$p < d_{min}$	$p < \frac{d_{min}}{2}$

EXERCISE
What is the minimum distance of code $\mathcal{C}$?

SOLUTION
$d_{min} = d(c_0, c_1) = 3$.

(Many other pairs (c_i, c_j) satisfy $d(c_i, c_j) = 3$ as well.)

code $\mathcal{C}$
$c_0 = 0000000$
$c_1 = 0011100$
$c_2 = 0111011$
$c_3 = 1110100$
$c_4 = 0100111$
$c_5 = 1101000$
$c_6 = 1001111$
$c_7 = 1010011$

Note: For notational convenience, we often write codewords without parenthesis or commas. For instance, 0000000 is a shorthand notation for $(0, 0, 0, 0, 0, 0, 0)$.

code $\mathcal{C}$
$c_0 = 0000000$
$c_1 = 0011100$
$c_2 = 0111011$
$c_3 = 1110100$
$c_4 = 0100111$
$c_5 = 1101000$
$c_6 = 1001111$
$c_7 = 1010011$

Exercise (cont.)

How many erasures can $\mathcal{C}$ correct?

If $y_1 = ?01110?$, what was the transmitted codeword?
If $y_2 = 11???00$, what was the transmitted codeword?
If $y_3 = ???0011$, what was the transmitted codeword?

Solution

$d_{min}(\mathcal{C}) = 3$, so the code can correct all erasures of weight $p < d_{min}(\mathcal{C}) = 3$.

For y_1, $p = 2 < d_{min}(\mathcal{C})$: correction is guaranteed. y_1 is decoded to c_1.

For y_2, $p = 3 \not< d_{min}(\mathcal{C})$: correction is not guaranteed in general. In fact, y_2 cannot be corrected by a minimum-distance decoder, because c_3 and c_5 are at the same distance from y.

For y_3, $p = 3 \not< d_{min}(\mathcal{C})$: correction is not guaranteed in general, but only one codeword is compatible with this y, namely c_7.

code $\mathcal{C}$
$c_0 = 0000000$
$c_1 = 0011100$
$c_2 = 0111011$
$c_3 = 1110100$
$c_4 = 0100111$
$c_5 = 1101000$
$c_6 = 1001111$
$c_7 = 1010011$

Exercise (cont.)

How many errors can $\mathcal{C}$ correct?

If $y_1 = c_1 + 0100000$, what codeword is decoded?

If $y_2 = c_4 + 0010100 = 0110011$, what codeword is decoded?

code $\mathcal{C}$
$c_0 = 0000000$
$c_1 = 0011100$
$c_2 = 0111011$
$c_3 = 1110100$
$c_4 = 0100111$
$c_5 = 1101000$
$c_6 = 1001111$
$c_7 = 1010011$

SOLUTION

$d_{min}(\mathcal{C}) = 3$, so the code can correct all errors of weight $p < \frac{d_{min}(\mathcal{C})}{2} = 1$.

$p(y_1) = 1 \quad \Rightarrow \arg\min_{\hat{c} \in \mathcal{C}} d(y_1, \hat{c}) = 0011100 = c_1.$

$p(y_2) = 2 \quad \Rightarrow \arg\min_{\hat{c} \in \mathcal{C}} d(y_2, \hat{c}) = c_2 \neq c_4.$

code $\mathcal{C}$
$c_0 = 0000000$
$c_1 = 0011100$
$c_2 = 0111011$
$c_3 = 1110100$
$c_4 = 0100111$
$c_5 = 1101000$
$c_6 = 1001111$
$c_7 = 1010011$

EXERCISE

Let $|\mathcal{C}| = M$. How many distances do we have to check to determine $d_{min}(\mathcal{C})$ via a brute-force approach?

Solution

Consider the following $M \times M$ matrix, with as many rows and columns as the number of codewords.

	1	2	3		$M-1$	M
1	x	$\surd$	$\surd$	$\cdots$	$\surd$	$\surd$
2	x	x	$\surd$	$\cdots$	$\surd$	$\surd$
	$\vdots$	$\vdots$				$\vdots$
$M-1$	x	x		$\cdots$	x	$\surd$
M	x	x		$\cdots$	x	x

A "$\surd$" at position (i, j) means that c_i and c_j need to be compared, whereas "x" means that they don't need to be compared.

There are $\frac{1}{2}(M^2 - M) = \frac{1}{2}M(M-1) = \binom{M}{2}$ of them.

- The code used in CDs has $M = 2^{1024} \approx 10^{300}$ codewords.
- A brute-force approach requires on the order of 10^{600} comparisons.
- (There are about 10^{50} atoms on Earth, about 10^{80} atoms in the universe, and about 5×10^{29} picoseconds since the big bang.)
- We need codes that have structure, for which we can tell the minimum distance via analytical means, rather than by brute-force computation.
- First, we derive an upper bound to $d_{min}(\mathcal{C})$.

UPPER BOUND TO $d_{min}(\mathcal{C})$

Recall the important parameters of a block code $\mathcal{C}$ over a D-ary alphabet:

- n, the block length.
- $k = \log_D |\mathcal{C}|$, the number of information symbols carried by a codeword. (Equivalently, $|\mathcal{C}| = D^k$.)
- d_{min} is the minimum distance.

Theorem (Singleton's bound: textbook theorem 11.5)

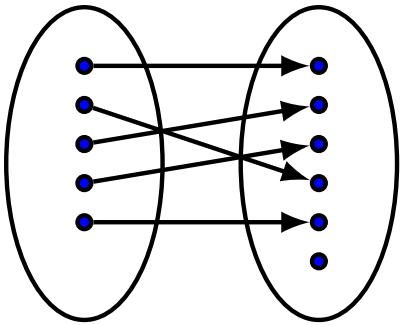
Regardless of the alphabet size, the minimum distance of a block code satisfies

$$d_{min} - 1 \leq n - k$$

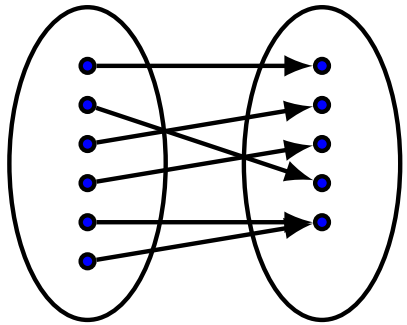
Block codes that satisfy the Singleton bound with equality are called **Maximum Distance Separable** codes. (**MDS** codes.)

Recall that for a function $f : \mathcal{E} \to \mathcal{F}$

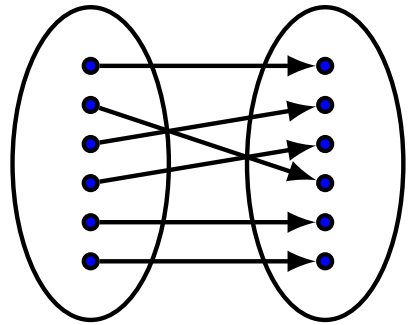
- $\mathcal{E}$ is the domain
- $\mathcal{F}$ is the codomain
- $f(\mathcal{E})$ is the image
- (range is sometimes used for the codomain, and sometimes for the image)



injective
(one-to-one)



surjective
(onto)



bijective
(one-to-one and onto)

Let $f : \mathcal{E} \to \mathcal{F}$, where $\mathcal{E}$ and $\mathcal{F}$ are finite sets.

$$f \text{ injective} \Rightarrow |\mathcal{E}| \leq |\mathcal{F}|$$

$$f \text{ surjective} \Rightarrow |\mathcal{E}| \geq |\mathcal{F}|$$

$$f \text{ bijective} \Rightarrow |\mathcal{E}| = |\mathcal{F}|$$

Proof of the Singleton Bound:

Consider the map $f : \mathcal{C} \to \mathcal{A}^{n-(d_{min}-1)}$ that removes the last $d_{min} - 1$ components of a codeword

$$f : (c_0, \ldots, c_{n-d_{min}}, \underbrace{c_{n-(d_{min}-1)}, \ldots, c_{n-1}}_{d_{min}-1 \text{ components}}) \mapsto (c_0, \ldots, c_{n-d_{min}})$$

The code has minimum distance d_{min}, so f is injective (one-to-one).

By the pigeonhole principle, the cardinality of its domain cannot exceed the cardinality of the codomain:

$$\begin{aligned} |\mathcal{C}| &\leq |\mathcal{A}|^{n-(d_{min}-1)} \\ D^k &\leq D^{n-(d_{min}-1)} \\ k &\leq n - (d_{min} - 1). \end{aligned}$$

Exercise

Write down $d_{min} - 1$ and $n - k$ for this code. Verify that Singleton's bound is satisfied.

code $\mathcal{C}$
$c_0 = 0000000$
$c_1 = 0011100$
$c_2 = 0111011$
$c_3 = 1110100$
$c_4 = 0100111$
$c_5 = 1101000$
$c_6 = 1001111$
$c_7 = 1010011$

Solution

$d_{min} - 1 = 2$.

$n - k = 7 - \log_2 8 = 4$.

Since $d_{min} - 1 \leq n - k$, Singleton's bound is satisfied.

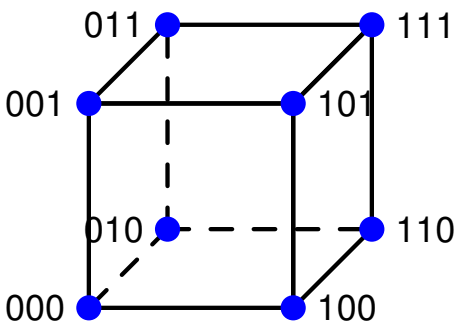
Week 11: Finite Fields and Vector Spaces (Textbook Chapter 12)

Prof. Michael Gastpar

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

EPFL

Spring Semester 2025



Outline

- Our next goal is to bring **algebraic structure** into code design and decoding.
- Encoding, decoding, and computing d_{min} become easier if the code forms a vector space.
- Vector spaces are defined over fields.
- For coding, we care about **finite** fields.

A *field* is the triplet $(\mathcal{K}, +, \times)$ where $\mathcal{K}$ is a set, and $+$, $\times$ are two binary operators called addition and multiplication, such that the following axioms hold:

1. **Associativity**: $\forall a, b, c \in \mathcal{K}$,

$$a + (b + c) = (a + b) + c$$
$$a \times (b \times c) = (a \times b) \times c$$

2. **Commutativity**: $\forall a, b \in \mathcal{K}$,

$$a + b = b + a$$
$$a \times b = b \times a$$

3. **Identity under** $+$: $\mathcal{K}$ contains an element, typically denoted by 0, such that $\forall a \in \mathcal{K}$,

$$a + 0 = a$$

4. **Inverse under** $+$: $\forall a \in \mathcal{K}$, there exists a unique $b \in \mathcal{K}$ such that

$$a + b = 0$$

b is the additive inverse of a, typically denoted by $-a$.

5. **Identity under** $\times$: $\mathcal{K}$ contains an element, typically denoted by 1, such that $\forall a \in \mathcal{K}$,

$$a \times 1 = a$$

6. **Inverse under** $\times$: $\forall a \in \mathcal{K}$, $a \neq 0$, there exists a unique $b \in \mathcal{K}$, such that

$$a \times b = 1$$

b is the multiplicative inverse of a, typically denoted by a^{-1}.

7. **Distributivity**: $\forall a, b, c \in \mathcal{K}$,

$$a \times (b + c) = (a \times b) + (a \times c).$$

Some remarks:

- If $\mathcal{K}$ is finite, then $(\mathcal{K}, +, \times)$ is a *finite field*.
- Instead of $(+, \times)$ the binary operations of a field may be denoted by $(+, \cdot)$, $(\star, \circ)$, $(\oplus, \wedge)$, $\ldots$
- ab is a short hand for $a \times b$.
- $a - b$ is a short hand for $a + (-b)$.
- If n is a positive integer and $b \in \mathcal{K}$, nb means $\underbrace{b + b + \cdots + b}_{n \text{ times}}$.
- If k is a positive integer and $a \in \mathcal{K}$, a^k means $\underbrace{a \times a \times \cdots \times a}_{k \text{ times}}$.

Example

Well-known examples of fields:

- $(\mathbb{R}, +, \cdot)$, the (field of) real numbers
- $(\mathbb{C}, +, \cdot)$, the (field of) complex numbers
- $(\mathbb{Q}, +, \cdot)$, the (field of) rational numbers

Well-known examples that are *not* fields:

- $(\mathbb{N}, +, \cdot)$, the (set of) non-negative integers
- $(\mathbb{Z}, +, \cdot)$, the (set of) integers

Exercise

Are these finite fields?

- $(\mathbb{Z}/16\mathbb{Z}, +, \cdot)$, the integers modulo 16
- $(\mathbb{Z}/17\mathbb{Z}, +, \cdot)$, the integers modulo 17

Solution

- $(\mathbb{Z}/16\mathbb{Z}, +, \cdot)$ is not a field because some non-zero elements do not have the multiplicative inverse.
- $(\mathbb{Z}/17\mathbb{Z}, +, \cdot)$ is a field because all its non-zero elements have an inverse.

Any algebraic manipulation in a finite field behaves similarly to $\mathbb{R}$. For instance:

- we can solve equations
- we can do linear algebra (define vectors and matrices, compute determinants, etc.)

The following statements can be deduced from the field axioms:

- if $x \in \mathcal{K} \setminus 0$, then $x \cdot y = 0 \Rightarrow y = 0$
- $\forall x \in \mathcal{K}$, $0 \cdot x = 0$
- $\forall x \in \mathcal{K}$, $k \in \mathbb{N}$, $x^k = 0 \Rightarrow x = 0$
- $(-1) \cdot x = -x$
- $(a + b)^2 = a^2 + 2ab + b^2$

Example

Find the solution of $3x + 6 = 4$ in $(\mathbb{Z}/7\mathbb{Z}, +, \cdot)$.

Solution

$$\begin{aligned} 3x + 6 = 4 &\Leftrightarrow 3x + 6 + (-6) = 4 + (-6) \\ &\Leftrightarrow 3x = 5 \\ &\Leftrightarrow 3^{-1} \cdot 3x = 3^{-1} \cdot 5 \\ &\Leftrightarrow x = 5 \cdot 5 = 25 = 4 \end{aligned}$$

EXAMPLE

In $(\mathbb{Z}/7\mathbb{Z}, +, \cdot)$, we have

$$\underbrace{1 + 1 + \cdots + 1}_{\text{7 times}} = 0$$

Hence, the order of 1 with respect to $+$ is 7.

- Every field contains the special number 1.
- For a finite field, the order of 1 with respect to $+$ is a prime number p called the field characteristic.

EXAMPLE

Let p be prime, so that $(\mathbb{Z}/p\mathbb{Z}, +, \cdot)$ is a finite field. Its characteristic is p.

EXERCISE

Can you prove that the characteristic of a finite field $(F, +, \cdot)$ is a prime number?

Hint: $(F, +)$ is a commutative group, hence there exists a smallest integer $m > 1$ such that

$$\underbrace{1 + 1 + \cdots + 1}_{m \text{ times}} = 0.$$

Solution

- Let m be the smallest number such that $\underbrace{1+1+\cdots+1}_{m \text{ times}} = 0$
- Suppose that $m = ab$ with $a > 1$ and $b > 1$
- One of the field axioms (distributivity) implies

$$\underbrace{(1+1+\cdots+1)}_{a \text{ times}} \cdot \underbrace{(1+1+\cdots+1)}_{b \text{ times}} = 0$$

- Hence, either

$$\underbrace{1+1+\cdots+1}_{a \text{ times}} = 0 \quad \text{or} \quad \underbrace{1+1+\cdots+1}_{b \text{ times}} = 0$$

- Contradiction: Hence the smallest m is a prime number.

□

Definition

An **isomorphism** between two finite fields $\mathbb{F} = (\mathcal{F}, +, \times)$ and $\mathbb{K} = (\mathcal{K}, \oplus, \otimes)$ is a bijection

$$\phi : \mathcal{F} \to \mathcal{K}$$

such that, for all $a, b \in \mathcal{F}$,

$$\phi(a + b) = \phi(a) \oplus \phi(b)$$
$$\phi(a \times b) = \phi(a) \otimes \phi(b).$$

We say $\mathbb{F}$ and $\mathbb{K}$ are **isomorphic** if there exists an isomorphism between them.

THEOREM (TEXTBOOK THEOREM 12.1, WITHOUT PROOF)

1. The cardinality of a finite field is an integer power of its characteristic. (Hence all finite fields have cardinality p^m for some prime p and some positive integer m.)

2. All finite fields of the same cardinality are isomorphic.

3. For every prime number p and positive integer m, there is a finite field of cardinality p^m.

- $(\mathbb{Z}/k\mathbb{Z}, +, \cdot)$ is a finite field iff $k = p$ for some prime p.
- A field that has p elements is isomorphic to $(\mathbb{Z}/p\mathbb{Z}, +, \cdot)$.
- In $(\mathbb{Z}/p\mathbb{Z}, +, \cdot)$, we know how to add and multiply without tables.
- A field with p^m elements is denoted by $\mathbb{F}_{p^m}$.
- Rather than developing the theory that allows us to add and multiply in $\mathbb{F}_{p^m}$, in most of our examples we stick to $(\mathbb{Z}/p\mathbb{Z}, +, \cdot)$, keeping in mind that all we do generalizes to arbitrary finite fields.

Example ($\mathbb{F}_2$)

The smallest finite field is $(\mathbb{Z}/2\mathbb{Z}, +, \cdot)$, denoted by $\mathbb{F}_2$. Its elements are 0 and 1 and the operations are

$+$	0	1
0	0	1
1	1	0

$\cdot$	0	1
0	0	0
1	0	1

Exercise ($\mathbb{F}_3$)

Define the finite field of cardinality 3.

Solution ($\mathbb{F}_3$)

$\mathbb{F}_3$ is isomorphic to $(\mathbb{Z}/3\mathbb{Z}, +, \cdot)$, with addition and multiplication defined as follows:

$+$	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

$\cdot$	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

Example ($\mathbb{F}_4$)

Because 4 is of the form p^m, there exists a finite field with 4 elements.

Let us denote the elements $0, 1, a, b$.

The axioms associated to 0 and 1 imply

$+$	0	1	a	b
0	0	1	a	b
1	1			
a	a			
b	b			

$\cdot$	0	1	a	b
0	0	0	0	0
1	0	1	a	b
a	0	a		
b	0	b		

Example (cont.)

The field characteristic is $p = 2$, therefore $1 + 1 = 0$.
Similarly, $x + x = x \cdot (1 + 1) = x \cdot 0 = 0$ for all x, so we can complete the diagonal of the $+$ table:

$+$	0	1	a	b
0	0	1	a	b
1	1	0	b	a
a	a	b	0	1
b	b	a	1	0

Finally, we can complete the remaining blanks knowing that each element has to show up exactly once in each row and each column.

Example (cont.)

Similarly, the $\cdot$ table is completed using the fact that $\mathbb{F}_4^* = (\{1, a, b\}, \cdot)$ is a group.

$+$	0	1	a	b
0	0	1	a	b
1	1	0	b	a
a	a	b	0	1
b	b	a	1	0

$\cdot$	0	1	a	b
0	0	0	0	0
1	0	1	a	b
a	0	a	b	1
b	0	b	1	a

Exercise

Is $\mathbb{F}_4$ isomorphic to $(\mathbb{Z}/4\mathbb{Z}, +, \cdot)$?

Solution

It cannot be, because $(\mathbb{Z}/4\mathbb{Z}, +, \cdot)$ is not a field.

Other reason: in $\mathbb{F}_4$, the characteristic is $p = 2$. Hence $a + a = 0$ for all $a \in \mathbb{F}_4$. Not the case for $(\mathbb{Z}/4\mathbb{Z}, +, \cdot)$.

Example (group isomorphism?)

Let $(\mathbb{F}_4, +)$ be $\mathbb{F}_4$ without multiplication. Is $(\mathbb{F}_4, +)$ isomorphic to $((\mathbb{Z}/2\mathbb{Z})^2, +)$?

Recall: $(\mathbb{Z}/2\mathbb{Z})^2 = \mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$ with addition component-wise over $(\mathbb{Z}/2\mathbb{Z}, +)$.

Solution

The answer is YES: both are finite commutative groups. In both cases, all nonzero elements have order 2. Since they have the same set of orders, they are isomorphic.

The isomorphism is: $0 \Rightarrow 00$, $1 \Rightarrow 11$, $a \Rightarrow 01$, $b \Rightarrow 10$
or $0 \Rightarrow 00$, $1 \Rightarrow 11$, $a \Rightarrow 10$, $b \Rightarrow 01$.

Example (field isomorphism?)

Is $\mathbb{F}_4$ isomorphic to $((\mathbb{Z}/2\mathbb{Z})^2, +, \cdot)$?

Solution

The answer is NO, because $((\mathbb{Z}/2\mathbb{Z})^2, +, \cdot)$ is not a field: $(0, 1)$ is a non-zero element that has no multiplicative inverse.

However, since they have the same number of elements, we can redefine the multiplication of $((\mathbb{Z}/2\mathbb{Z})^2, +, \cdot)$ so that the result is a field. It suffices to use the multiplication table from $\mathbb{F}_4$ and substitute $0 \Rightarrow 00$, $1 \Rightarrow 11$, $a \Rightarrow 01$, $b \Rightarrow 10$. (We are just re-labeling the elements of a previously established field.)

Solution (cont.)

$\mathbb{F}_4$

$+$	0	1	a	b
0	0	1	a	b
1	1	0	b	a
a	a	b	0	1
b	b	a	1	0

$\cdot$	0	1	a	b
0	0	0	0	0
1	0	1	a	b
a	0	a	b	1
b	0	b	1	a

$((\mathbb{Z}/2\mathbb{Z})^2, +, \otimes)$

$+$	00	11	01	10
00	00	11	01	10
11	11	00	10	01
01	01	10	00	11
10	10	01	11	00

$\otimes$	00	11	01	10
00	00	00	00	00
11	00	11	01	10
01	00	01	10	11
10	00	10	11	01

This is a review, since you *should* know everything that we need from linear algebra. (MATH-111e.)

We review only what we need for the chapter on linear block codes (next week).

For missing proofs, see e.g.

- Sheldon Axler, “Linear Algebra Done Right”, Springer
- Tom M. Apostol, “Linear Algebra: A First Course with Applications to Differential Equations”, Wiley.
- David C. Lay, “ Linear Algebra and Its Applications”, Addison-Wesley.

DEFINITION (VECTOR SPACE)

A nonempty set V is said to be a *vector space* over a finite field $\mathbb{F}$ if:

I. there exists an operation called addition that associates to each pair $\vec{u}, \vec{v} \in V$ a vector $\vec{u} + \vec{v} \in V$ called the sum of $\vec{u}$ and $\vec{v}$;

II. there exists an operation called scalar multiplication that associates to each $\alpha \in \mathbb{F}$ and $\vec{v} \in V$ a new vector $\alpha\vec{v} \in V$ called the product of α and $\vec{v}$;

and these operations satisfy the following axioms:

Definition (cont.)

- $\vec{u} + \vec{v} = \vec{v} + \vec{u}$ for all $\vec{u}, \vec{v} \in V$;
- $(\vec{u} + \vec{v}) + \vec{w} = \vec{u} + (\vec{v} + \vec{w})$ for all $\vec{u}, \vec{v}, \vec{w} \in V$;
- There exists an element $\vec{0} \in V$ such that $\vec{0} + \vec{v} = \vec{v}$ for all $\vec{v} \in V$;
- For all $\vec{v} \in V$, there exists an element $-\vec{v} \in V$ such that $\vec{v} + (-\vec{v}) = \vec{0}$;
- $\alpha(\vec{u} + \vec{v}) = \alpha\vec{u} + \alpha\vec{v}$ for all $\alpha \in \mathbb{F}$ and all $\vec{u}, \vec{v} \in V$;
- $(\alpha + \beta)\vec{v} = \alpha\vec{v} + \beta\vec{v}$ for all $\alpha, \beta \in \mathbb{F}$ and all $\vec{v} \in V$;
- $\alpha(\beta\vec{v}) = (\alpha\beta)\vec{v}$ for all $\alpha, \beta \in \mathbb{F}$ and all $\vec{v} \in V$;
- $1\vec{v} = \vec{v}$ for all $\vec{v} \in V$, where 1 is the (multiplicative) identity in $\mathbb{F}$.

Definition (equivalent, compact definition)

$(V, +, \times)$ is a vector space over a field $\mathbb{F}$ if:

- $(V, +)$ is a commutative (abelian) group;
- The binary operator $\times$ is between an element of V and one of $\mathbb{F}$, with the following properties:
  - (associativity) $\forall \vec{v} \in V$ and $\alpha, \beta \in \mathbb{F}$, $\alpha(\beta\vec{v}) = (\alpha\beta)\vec{v}$;
  - (identity) $1\vec{v} = \vec{v}$;
  - (distributivity) $\alpha(\vec{u} + \vec{v}) = \alpha\vec{u} + \alpha\vec{v}$ and $(\alpha + \beta)\vec{v} = \alpha\vec{v} + \beta\vec{v}$.

Example (vector space)

For every field $\mathbb{F}$ and every positive integer n, $V = \mathbb{F}^n$ is the vector space of n-tuples.

Vector-addition is done component-wise according to the addition rule of $\mathbb{F}$:

$$(u_1, \ldots, u_n) + (v_1, \ldots, v_n) = (u_1 + v_1, \ldots, u_n + v_n)$$

Multiplication of a vector by a scalar is also done component-wise according to the multiplication rule of $\mathbb{F}$:

$$\alpha(v_1, \ldots, v_n) = (\alpha v_1, \ldots, \alpha v_n)$$

Example (vector space)

For every field $\mathbb{F}$ and positive integer n, the set of polynomials of the form $p(x) = a_0 + a_1 x + \cdots + a_n x^n$ with coefficients $a_0, \ldots, a_n \in \mathbb{F}$ is a vector space, where the addition of polynomials and the multiplication of a polynomial by a scalar are done according to the "usual rules".

Example (vector space)

Let p be a prime number and consider the field $\mathbb{F} = (\mathbb{Z}/p\mathbb{Z}, +, \cdot)$.

Let n be a positive integer and consider the vector space $V = \mathbb{F}^n$. This is a vector space over (the finite field) $\mathbb{F}$.

It turns out that for all vector spaces of the form $V = \mathbb{F}^n$, $\mathbb{F}$ finite field, we can define a multiplication among vectors that fulfills all the axioms of a field.

Hence $\mathbb{F}^n$, where $\mathbb{F} = (\mathbb{Z}/p\mathbb{Z}, +, \cdot)$, is a vector space that can be made into the finite field $\mathbb{F}_{p^n}$.

In fact it has p^n elements, and its characteristic is p, and there is only one such field (up to isomorphism).

All finite fields can be put in this form. We have already seen $((\mathbb{Z}/2\mathbb{Z})^2, +, \cdot)$.

We are not proving the above result, because we will not make use of it in this course.

If V is a vector space and $S \subseteq V$ with the property that S is closed under vector addition and multiplication by a scalar, then S is itself a vector space.

(Closure of S with respect to vector addition and multiplication of a vector by a scalar are required by two axioms. Verify for yourself that the other axioms that S has to fulfill to be a vector space are automatically inherited from V.)

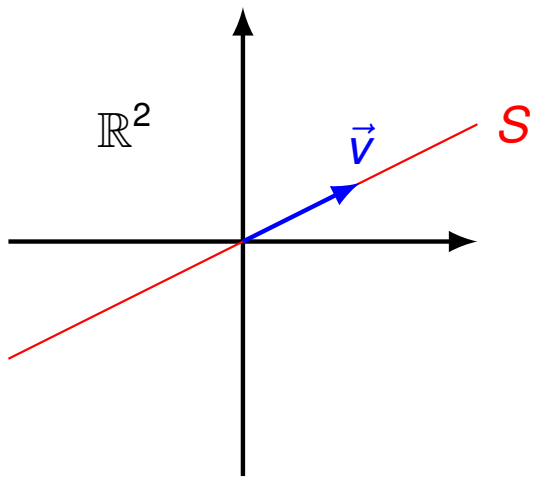
We call such an S a **subspace** of V.

Example

Let $V = \mathbb{R}^2$, $\vec{v} \in V$, and define $S = \{\vec{s} \in V : \vec{s} = a\vec{v}, a \in \mathbb{R}\}$.

- If $\vec{u} \in S$, then $b\vec{u} = b(a\vec{v}) = (ba)\vec{v} \in S$ for all $b \in \mathbb{R}$
- If $\vec{u}, \vec{w} \in S$, then $\vec{u} + \vec{w} = a_1\vec{v} + a_2\vec{v} = (a_1 + a_2)\vec{v} \in S$

Hence S is a subspace of V.



Example

Let $V = \mathbb{F}_7^3$ and define $S = \{(x_1, x_2, x_3) : x_i \in \mathbb{F}_7 \text{ and } x_1 + 2x_2 + 3x_3 = 0\}$.

S is a subspace of V. (Be sure that you see why.)

NB: there are four kinds of operations in a vector space:

1. scalar addition,
2. scalar multiplication,
3. vector addition,
4. multiplication of a vector with a scalar.

The one used is always clear from the context.

For instance, it is clear that the above equation $x_1 + 2x_2 + 3x_3 = 0$ involves additions and multiplications in $\mathbb{F}_7$.

A linear combination of a **list** $(\vec{v}_1, \ldots, \vec{v}_n)$ of vectors in V is a vector of the form $\sum_{i=1}^{n} \lambda_i \vec{v}_i$, where $\lambda_1, \ldots, \lambda_n \in \mathbb{F}$.

The set of all linear combinations of $(\vec{v}_1, \ldots, \vec{v}_n)$ is called the **span** of $(\vec{v}_1, \ldots, \vec{v}_n)$, denoted $\mathrm{span}(\vec{v}_1, \ldots, \vec{v}_n)$.

If $\mathrm{span}(\vec{v}_1, \ldots, \vec{v}_n) = V$, we say that $(\vec{v}_1, \ldots, \vec{v}_n)$ **spans** V.

A vector space is called **finite-dimensional** if some list of vectors in it spans the whole space. (A list has finite length by definition.)

The vectors $\vec{v}_i$, $i = 1, \ldots, n$ are said to be **linearly independent** iff $\sum_{i=1}^{n} \lambda_i \vec{v}_i = 0$ implies $\lambda_1 = \cdots = \lambda_n = 0$.

A **basis** of V is a list of vectors in V that is linearly independent and spans V.

THEOREM

A list $(\vec{v}_1, \ldots, \vec{v}_n)$ of vectors in V is a basis of V iff every $\vec{v} \in V$ can be written uniquely in the form

$$\vec{v} = \sum_{i=1}^{n} \lambda_i \vec{v}_i$$

Exercise (Proof of $\Rightarrow$)

Prove that if $(\vec{v}_1, \ldots, \vec{v}_n)$ is a basis of V, then for every vector $\vec{v} \in V$, there is a unique set of coefficients $\lambda_1, \ldots, \lambda_n$, such that

$$\vec{v} = \sum_{i=1}^{n} \lambda_i \vec{v}_i.$$

Solution

$(\vec{v}_1, \ldots, \vec{v}_n)$ is a basis of V, hence it spans V (by definition), which means that every $\vec{v} \in V$ can be written as

$$\vec{v} = \sum_{i=1}^{n} \lambda_i \vec{v}_i.$$

We need to prove uniqueness. Suppose that $\sum_{i=1}^{n} \lambda_i \vec{v}_i = \sum_{i=1}^{n} \beta_i \vec{v}_i$.

Then $\sum_{i=1}^{n} \lambda_i \vec{v}_i - \sum_{i=1}^{n} \beta_i \vec{v}_i = 0$.

Using the axioms, we rewrite as $\sum_{i=1}^{n} (\lambda_i - \beta_i) \vec{v}_i = 0$.

The linear independence of the basis vectors implies $\lambda_i - \beta_i = 0$, i.e., $\lambda_i = \beta_i$ for all i. □

Exercise (Proof of $\Leftarrow$)

Prove that if every vector $\vec{v} \in V$ has a unique set of coefficients $\lambda_1, \ldots, \lambda_n$, such that

$$\vec{v} = \sum_{i=1}^{n} \lambda_i \vec{v}_i,$$

then $(\vec{v}_1, \ldots, \vec{v}_n)$ is a basis of V.

Solution

By assumption, $(\vec{v}_1, \ldots, \vec{v}_n)$ spans V. It remains to be shown that the list $(\vec{v}_1, \ldots, \vec{v}_n)$ is of linearly independent vectors.

Write the zero vector as $0 = \sum_{i=1}^{n} \lambda_i \vec{v}_i$. The uniqueness of the coefficients implies that $\lambda_i = 0$ for all i.

Hence the vectors $\vec{v}_1, \ldots, \vec{v}_n$ are linearly independent. □

Theorem

Every spanning list in a vector space can be reduced to a basis of the vector space.

Proof (outline)

Remove all the zero-elements of the list.

Of the new list, remove the second element if it is in the linear span of the first. Repeat the same until we have a list in which the second element is not in the linear span of the first.

Of the new list, remove the third element if it is in the linear span of the first two.

Continue similarly.

The result is a list of vectors that span the vector space and are linearly independent (or else one vector can be written as the linear combination of vectors with smaller index). □

The above theorem implies that every finite-dimensional vector space has a basis.

Theorem (Without Proof)

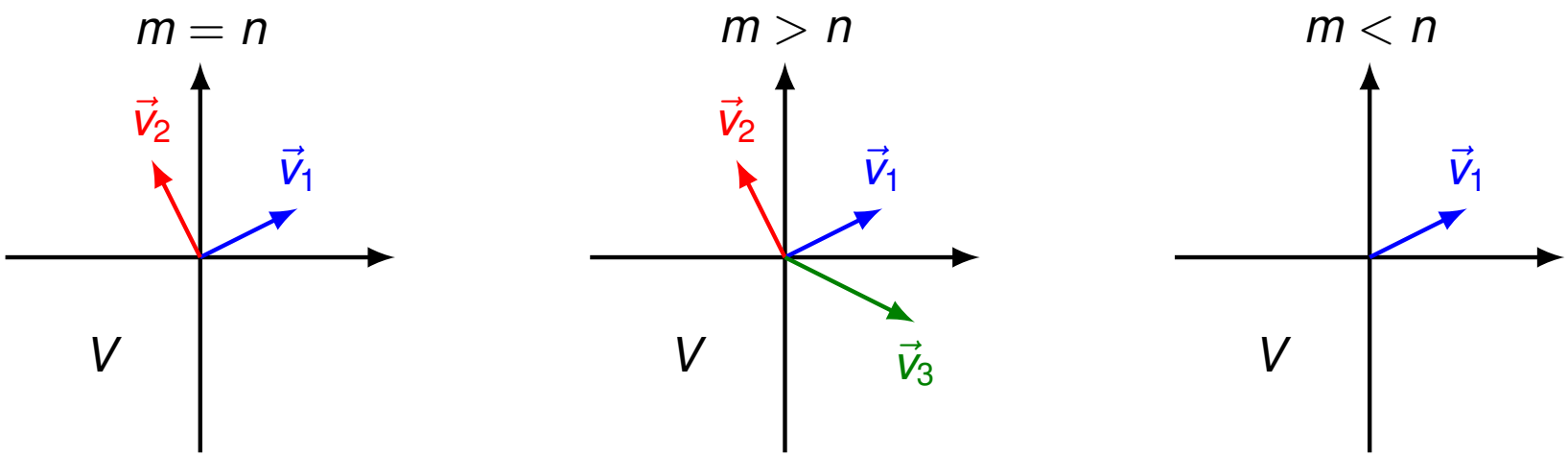
Any two bases of a finite-dimensional vector space have the same length.

The **dimension** of a finite-dimensional vector space V, denoted by $\dim(V)$, is defined to be the length of any basis of V.

A FEW PROPERTIES OF THE DIMENSION OF A VECTOR SPACE

Let V be a vector space and suppose that $\dim(V) = n$.

- If $(\vec{v}_1, \ldots, \vec{v}_n)$ is a list of linearly independent vectors in V, then it is a basis of V.
- If $(\vec{v}_1, \ldots, \vec{v}_n)$ spans V, then it is a basis of V.
- A list of $m > n$ vectors in V cannot be linearly independent.
- A list of $m < n$ vectors cannot span V.



Example

Let $\mathbb{F}$ be a finite field. A basis of $\mathbb{F}^n$ is

$$\big((1,0,\ldots,0),(0,1,0,\ldots,0),\ldots,(0,\ldots,0,1)\big).$$

It is called **canonical** basis.

$\dim(\mathbb{F}^n) = n$.

EXAMPLE

Let S be the subspace of $\mathbb{F}_7^3$ spanned by $\vec{v} = (4, 3, 1)$.

Define S by means of equations.

SOLUTION

$(x, y, z) \in S$ implies $(x, y, z) = \alpha\vec{v}$ for some $\alpha \in \mathbb{F}_7$. Equivalently, $(x, y, z) = (4\alpha, 3\alpha, \alpha)$ or

$$\begin{cases} x = 4\alpha \\ y = 3\alpha \\ z = \alpha \end{cases}$$

After eliminating α,

$$\begin{cases} x = 4z \\ y = 3z \end{cases}$$

or, using the fact that $-4 = 3$,

$$\begin{cases} x + 3z = 0 \\ y + 4z = 0 \end{cases} \tag{1}$$

Solution (cont.)

Conversely, suppose that $(x, y, z) \in \mathbb{F}_7^3$ satisfies (1). Let α be the value of z. Then

$$\begin{cases} x = -3\alpha \\ y = -4\alpha \\ z = \alpha \end{cases}$$

or, equivalently

$$\begin{cases} x = 4\alpha \\ y = 3\alpha \\ z = \alpha, \end{cases}$$

which can be written as $(x, y, z) = \alpha\vec{v}$ for some $\alpha \in \mathbb{F}_7$.

We have proved that a vector $(x, y, z) \in \mathbb{F}_7^3$ is in S iff it satisfies the system of equations (1).

Example (Finding a basis)

Let $V \subseteq \mathbb{F}_7^3$ such that

$$V = \{(x_1, x_2, x_3) \in \mathbb{F}_7^3 : 3x_1 + 2x_2 + x_3 = 0\}$$

Find $\dim(V)$.

The above equation can be described by the **vector of coefficients** $(3, 2, 1) \in \mathbb{F}_7^3$.

Specifically, the equation is satisfied for (x_1, x_2, x_3) iff $(3, 2, 1)(x_1, x_2, x_3)^T = 0$.

Solution

We must obtain a basis of V. The equation $3x_1 + 2x_2 + x_3 = 0$ has two free variables, say x_1 and x_2.

Choose $x_1 = \alpha$ and $x_2 = \beta$.

$$\begin{aligned}(x_1, x_2, x_3) &= (\alpha, \beta, -3\alpha - 2\beta) = (\alpha, \beta, 4\alpha + 5\beta) \\ &= (1, 0, 4)\alpha + (0, 1, 5)\beta\end{aligned}$$

Clearly $\vec{v}_1 = (1, 0, 4)$ and $\vec{v}_2 = (0, 1, 5)$ are linearly independent and are in V.

Moreover, since (x_1, x_2, x_3) is an arbitrary vector in V, the equation above clearly shows that $V = \mathrm{span}(\vec{v}_1, \vec{v}_2)$.

$(\vec{v}_1, \vec{v}_2)$ is both linearly independent and spans V, so it is a basis of V.

Therefore $\dim(V) = 2$. □

Theorem (without proof)

The set of solutions in $V = \mathbb{F}^n$ of m linear homogeneous equations in n variables is a subspace S of V.

Let r be the dimensionality of the vector space spanned by the coefficient vectors. Then $\dim(S) = n - r$.

In particular, if the m vectors of coefficients are linearly independent, then $\dim(S) = n - m$.

Conversely, if S is a subspace of $V = \mathbb{F}^n$ with $\dim(S) = k$, there exists a set of $n - k$ linear equations with coefficients that form linearly independent vectors in V, the solution of which are the vectors in S.

Example (revisited)

Let $V \subseteq \mathbb{F}_7^3$ such that

$$V = \{(x_1, x_2, x_3) \in \mathbb{F}_7^3 : 3x_1 + 2x_2 + x_3 = 0\}$$

Find $\dim(V)$, and then find a basis for V.

Solution

There is only one vector $\vec{v} = (3, 2, 1)$ of coefficients. It spans a vector space of dimension $r = 1$. Hence $\dim(V) = 3 - 1 = 2$.

If we choose $\vec{v}_1 = (1, 0, v_{11})$ and $\vec{v}_2 = (0, 1, v_{21})$, then we are guaranteed that they are linearly independent. We choose v_{11} so as to satisfy the above equation, i.e., $v_{11} = -3 = 4$. Hence $\vec{v}_1 = (1, 0, 4)$

Similarly we obtain $\vec{v}_2 = (0, 1, 5)$.

Example (revisited)

Let S be the subspace of $\mathbb{F}_7^3$ spanned by $\vec{v} = (4, 3, 1)$.

Define S by means of equations.

Solution

S is a one-dimensional subspace of $V = \mathbb{F}_7^3$, hence it can be described by $3 - 1 = 2$ equations.

Let us find two linearly independent vectors $\vec{c}_1, \vec{c}_2 \in V$ such that $\sum_{j=1}^{3} c_{ij} v_j = 0$, $i = 1, 2$. (The above theorem implies that they exist.)

We can choose $\vec{c}_1 = (1, 0, c_{13})$ and $\vec{c}_2 = (0, 1, c_{23})$ and complete to fulfill the above equation.

Hence, $\vec{c}_1 = (1, 0, -4) = (1, 0, 3)$ and $\vec{c}_2 = (0, 1, -3) = (0, 1, 4)$.

Therefore S is the set of vectors (x_1, x_2, x_3) that satisfy

$$\begin{cases} x_1 + 3x_3 = 0 \\ x_2 + 4x_3 = 0 \end{cases}.$$

□

Rank of a matrix

For any matrix with entries in a field $\mathbb{F}$:

- the dimension of the vector space spanned by its rows ...
- equals the dimension of the vector space spanned by its columns.

It is called the *rank* of the matrix.

Example

Let S be the subspace of $V = \mathbb{F}_7^3$ whose elements (x_1, x_2, x_3) verify

$$\begin{cases} 4x_1 + x_2 = 0 \\ x_1 + x_3 = 0 \end{cases}$$

- The coefficient matrix is $A = \begin{bmatrix} 4 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$.
- Its rank is $r = 2$.
- $\dim(S) = \dim(V) - r = 3 - 2 = 1$.

THEOREM (12.2 OF TEXTBOOK)

An n-dimensional vector space V over a finite field $\mathbb{F}$:

- is finite,
- has cardinality

$$\text{card}(V) = [\text{card}(\mathbb{F})]^n.$$

Proof:

Let $(\vec{v}_1, \ldots, \vec{v}_n)$ be a basis of V. For every $\vec{v} \in V$, there is a unique n-tuple $(\lambda_1, \ldots, \lambda_n) \in \mathbb{F}^n$ such that $\vec{v} = \sum_i \lambda_i \vec{v}_i$.

Hence the mapping

$$\mathbb{F}^n \to V$$

$$(\lambda_1, \ldots, \lambda_n) \mapsto \vec{v} = \sum_i \lambda_i \vec{v}_i$$

is a bijection.

By the pigeonhole principle,

$$\mathrm{card}(V) = \mathrm{card}(\mathbb{F}^n) = [\mathrm{card}(\mathbb{F})]^n.$$

□

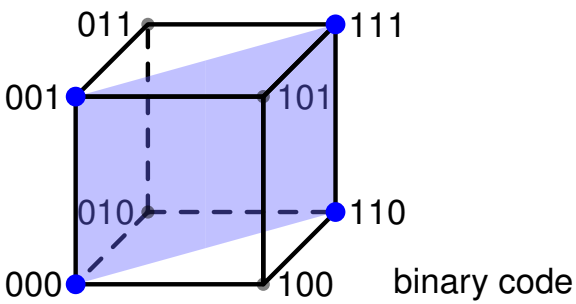
Weeks 12&13: Vector Spaces and Linear Codes (Textbook Chapter 13)

Prof. Michael Gastpar

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

EPFL

Spring Semester 2025



Outline

Why do we care about linear codes?

Linear codes have more structure.

We use that structure to simplify our tasks, notably:

- To determine the code's performance (d_{min} in particular).
- To simplify the encoding.
- To simplify the decoding.

Definition (Textbook def. 13.1)

A block code is a **linear code** if the codewords form a subspace of $\mathbb{F}^n$ for some finite field $\mathbb{F}$.

Example

Let $\mathcal{C} \subset \mathbb{F}_2^7$ be the block code that consists of the listed codewords. Is it linear?

code $\mathcal{C}$
$\vec{c}_0 = 0000000$
$\vec{c}_1 = 0011100$
$\vec{c}_2 = 0111011$
$\vec{c}_3 = 1110100$
$\vec{c}_4 = 0100111$
$\vec{c}_5 = 1101000$
$\vec{c}_6 = 1001111$
$\vec{c}_7 = 1010011$

Solution

We have

$$\vec{c}_4 = \vec{c}_1 + \vec{c}_2$$
$$\vec{c}_5 = \vec{c}_1 + \vec{c}_3$$
$$\vec{c}_6 = \vec{c}_2 + \vec{c}_3$$
$$\vec{c}_7 = \vec{c}_1 + \vec{c}_2 + \vec{c}_3$$

Therefore $\mathcal{C} = \mathsf{span}(\vec{c}_1, \vec{c}_2, \vec{c}_3) \subset \mathbb{F}_2^7$ is a linear code (over the finite field $\mathbb{F}_2$).

EXAMPLE

What is the dimension of code $\mathcal{C}$ (i.e., the dimension of the subspace formed by the codewords)?

SOLUTION

The set $(\vec{c}_1, \vec{c}_2, \vec{c}_3)$ is a basis of $\mathcal{C}$. Hence $\dim(\mathcal{C}) = 3$.

code $\mathcal{C}$
$\vec{c}_0 = 0000000$
$\vec{c}_1 = 0011100$
$\vec{c}_2 = 0111011$
$\vec{c}_3 = 1110100$
$\vec{c}_4 = 0100111$
$\vec{c}_5 = 1101000$
$\vec{c}_6 = 1001111$
$\vec{c}_7 = 1010011$

We have seen that a k-dimensional subspace of $\mathbb{F}^n$ has cardinality $[\mathrm{card}(\mathbb{F})]^k$. (Count the number of linear combinations you can form with the vectors that form the basis, with coefficients in $\mathbb{F}$.)

EXAMPLE

If the size of a binary block code is not of the form 2^k, then the code is not linear.

Hamming Weight

Definition

Let $\vec{x} = (x_1, \ldots, x_n)$ be an n-tuple with components in a finite field.

The **(Hamming) weight** of $\vec{x}$, denoted $w(\vec{x})$, is the number of its non-zero components in $(x_1, \ldots, x_n)$, i.e.

$$w(\vec{x}) = d\big((0, \ldots, 0), (x_1, \ldots, x_n)\big).$$

Exercise ("academic" question)

In the definition of Hamming weight, we are requiring that the components of $\vec{x}$ take value in a (finite) field $\mathbb{F}$. Why?

Solution

Otherwise there is no guarantee that the alphabet contains the 0 element.

Recall that in a finite field $\mathbb{F}$, no matter how we label its elements, one is the 0 element (the identity element with respect to addition). Hence the Hamming weight is well-defined.

Example

- The weight of $(1, 0, 1, 1, 0)$ is 3.
- The weight of $(3, 0, 4, 1, 1, 2)$ is 5.

THEOREM

The minimum distance of a linear code $\mathcal{C}$ is the smallest weight of a codeword in $\mathcal{C}$, zero-vector excluded, i.e.,

$$d_{min}(\mathcal{C}) = \min_{\vec{c} \in \mathcal{C}; \vec{c} \neq \vec{0}} w(\vec{c})$$

In the proof that follows, we use the following two facts:

- For all $\vec{u}, \vec{v} \in \mathbb{F}^n$,

  $$d(\vec{u}, \vec{v}) = w(\vec{u} - \vec{v})$$

  (Reason: $\vec{u}$ and $\vec{v}$ are different at position i iff $\vec{u} - \vec{v}$ is non-zero at position i.)

- Let $f : \mathcal{B} \to \mathbb{R}$ be an arbitrary function and $\mathcal{A} \subseteq \mathcal{B}$ be finite sets. Then

  $$\min_{x \in \mathcal{A}} f(x) \geq \min_{x \in \mathcal{B}} f(x)$$

  (We might find a smaller minimum if we enlarge the set.)

Proof:

$$d_{min}(\mathcal{C}) = \min_{\vec{u},\vec{v}\in\mathcal{C};\vec{u}\neq\vec{v}} d(\vec{u},\vec{v})$$

$$= \min_{\vec{u},\vec{v}\in\mathcal{C};\vec{u}\neq\vec{v}} w(\vec{u}-\vec{v})$$

$$\geq \min_{\vec{c}\in\mathcal{C};\vec{c}\neq\vec{0}} w(\vec{c}) \qquad \text{(reason: } \mathcal{C} \text{ is a vector space, so } \vec{u}-\vec{v}\in\mathcal{C}\text{).}$$

$$\min_{\vec{c}\in\mathcal{C};\vec{c}\neq\vec{0}} w(\vec{c}) = \min_{\vec{c}\in\mathcal{C};\vec{c}\neq\vec{0}} d(\vec{c},0)$$

$$\geq \min_{\vec{c},\vec{v}\in\mathcal{C};\vec{c}\neq\vec{v}} d(\vec{c},\vec{v}) \qquad \text{(reason: we have equality with } \vec{v}=0\text{)}$$

$$= d_{min}(\mathcal{C}).$$

□

EXERCISE

Find the minimum distance of the linear code $\mathcal{C}$.

code $\mathcal{C}$
$\vec{c}_0 = 0000000$
$\vec{c}_1 = 0011100$
$\vec{c}_2 = 0111011$
$\vec{c}_3 = 1110100$
$\vec{c}_4 = 0100111$
$\vec{c}_5 = 1101000$
$\vec{c}_6 = 1001111$
$\vec{c}_7 = 1010011$

Solution

- Compute the weight of each non-zero codeword:

$$w_1 = w(0011100) = 3$$
$$w_2 = w(0111011) = 5$$
$$w_3 = w(1110100) = 4$$
$$w_4 = w(0100111) = 4$$
$$w_5 = w(1101000) = 3$$
$$w_6 = w(1001111) = 5$$
$$w_7 = w(1010011) = 4$$

code $\mathcal{C}$
$\vec{c}_0 = 0000000$
$\vec{c}_1 = 0011100$
$\vec{c}_2 = 0111011$
$\vec{c}_3 = 1110100$
$\vec{c}_4 = 0100111$
$\vec{c}_5 = 1101000$
$\vec{c}_6 = 1001111$
$\vec{c}_7 = 1010011$

- $d_{min}(\mathcal{C}) = 3$.

- Note: compare this to the work needed to compute $d(\vec{v}_i, \vec{v}_j)$ for all $i \neq j$.

Exercise ((Binary) Parity-Check Code)

The parity-check code $\mathcal{C} \subset \mathbb{F}_2^n$ consists of those elements of $\mathbb{F}_2^n$ that have an even number of 1s, i.e.,

$$\mathcal{C} = \left\{ (c_1, \ldots, c_n) \in \mathbb{F}_2^n : \sum_i c_i = 0 \right\}.$$

(Addition is in $\mathbb{F}_2$, i.e., mod 2.)

Determine k and d_{min}.

Solution

- The code is a subset of $\mathbb{F}_2^n$ that satisfies an homogeneous linear equation.
- Hence the code is linear, and $k = n - 1$.
- (We can also tell that $k = n - 1$, by observing that we are free to choose the first $n - 1$ bits and satisfy the constraint with the last symbol.)
- For a linear code, d_{min} is the minimum non-zero weight.
- It is achieved by any codeword that has exactly two 1s.
- Hence $d_{min} = 2$.

Note: In this example, linearity allows us to determine d_{min} via deductive reasoning rather than by inspection.

Exercise ((Binary) Repetition Code)

It is the subset of $\mathbb{F}_2^n$ that consists of two codewords, namely $(0, \ldots, 0)$ and $(1, \ldots, 1)$.

Determine k and d_{min}.

Solution

- The code is linear: it is the subspace of $\mathbb{F}_2^n$ spanned by $(1, \ldots, 1)$.
- $k = 1$.
- $d_{min} = n$ (the weight of the only non-zero codeword).

The above two codes are not terribly useful, but they have an interesting property shared by the next code which is even less useful.

Exercise (The code $\mathbb{F}_2^n$)

$\mathbb{F}_2^n$ satisfies the definition of a linear code.

Determine k and d_{min}.

Solution

- $k = n$.
- $d_{min} = 1$.

The above three code families fulfill the Singleton bound with equality. Hence they are MDS codes. Moreover, they are also linear codes.

No other family of binary linear codes is MDS.

But there are non-binary codes that are MDS, e.g., the family of Reed-Solomon codes (studied next week).

DEFINITION (TEXTBOOK DEFINITION 13.3)

Let $(\vec{c}_1, \ldots, \vec{c}_k)$ be a basis of a linear code $\mathcal{C} \subset \mathbb{F}^n$ over some finite field $\mathbb{F}$.
The $k \times n$ matrix that has $\vec{c}_i$ as its ith row is called a **generator matrix** of $\mathcal{C}$.

EXAMPLE

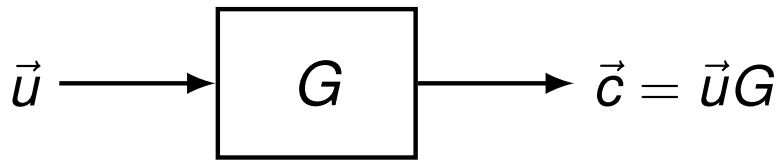
$(\vec{c}_1, \vec{c}_2, \vec{c}_3)$ form a basis for $\mathcal{C}$. Therefore

$$G = \begin{pmatrix} \vec{c}_1 \\ \vec{c}_2 \\ \vec{c}_3 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \end{pmatrix}$$

is a generator matrix of $\mathcal{C}$.

code $\mathcal{C}$
$\vec{c}_0 = 0000000$
$\vec{c}_1 = 0011100$
$\vec{c}_2 = 0111011$
$\vec{c}_3 = 1110100$
$\vec{c}_4 = 0100111$
$\vec{c}_5 = 1101000$
$\vec{c}_6 = 1001111$
$\vec{c}_7 = 1010011$

A $k \times n$ generator matrix specifies an **encoding map** that sends an **information vector** $\vec{u} \in \mathbb{F}^k$ to the corresponding codeword $\vec{c} = \vec{u}G$.



EXAMPLE

$$\vec{u} = (1,0,1) \to \vec{c} = (1,0,1)\begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \end{pmatrix}$$
$$= (1,1,0,1,0,0,0).$$

code $\mathcal{C}$
$\vec{c}_0 = 0000000$
$\vec{c}_1 = 0011100$
$\vec{c}_2 = 0111011$
$\vec{c}_3 = 1110100$
$\vec{c}_4 = 0100111$
$\vec{c}_5 = 1101000$
$\vec{c}_6 = 1001111$
$\vec{c}_7 = 1010011$

We have seen the generator matrix

$$G_1 = \begin{pmatrix} \vec{c}_1 \\ \vec{c}_2 \\ \vec{c}_3 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \end{pmatrix}.$$

Here is another one:

$$G_2 = \begin{pmatrix} \vec{c}_1 \\ \vec{c}_1 + \vec{c}_2 \\ \vec{c}_1 + \vec{c}_3 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

A linear code $\mathcal{C}$ has as many generator matrices as the number of bases of the vector space $\mathcal{C}$.

Each generator matrix determines an encoding map:

$\vec{u} \to \vec{u}G_1$	$\vec{u} \to \vec{u}G_2$
$000 \to 0000000 = \vec{c}_0$	$000 \to 0000000 = \vec{c}_0$
$001 \to 1110100 = \vec{c}_3$	$001 \to 1101000 = \vec{c}_5$
$010 \to 0111011 = \vec{c}_2$	$010 \to 0100111 = \vec{c}_4$
$011 \to 1001111 = \vec{c}_6$	$011 \to 1001111 = \vec{c}_6$
$100 \to 0011100 = \vec{c}_1$	$100 \to 0011100 = \vec{c}_1$
$101 \to 1101000 = \vec{c}_5$	$101 \to 1110100 = \vec{c}_3$
$110 \to 0100111 = \vec{c}_4$	$110 \to 0111011 = \vec{c}_2$
$111 \to 1010011 = \vec{c}_7$	$111 \to 1010011 = \vec{c}_7$

Exercise

How many generator matrices for a binary linear code of block-length $n = 7$ and dimension $k = 3$?

Solution

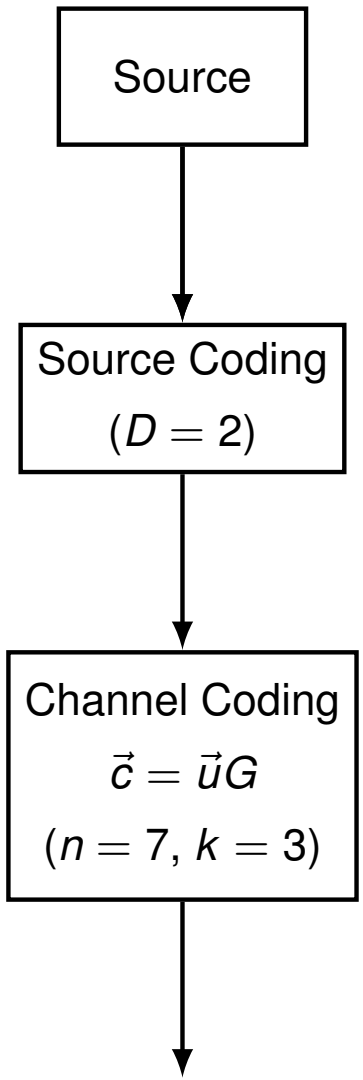
It is the number of lists that form a basis. A q-ary linear code of dimension k has q^k codewords and the number of bases is

$$(q^k - 1)(q^k - q) \cdots (q^k - q^{k-1}).$$

For a binary code ($q = 2$) we have

$$(2^3 - 1)(2^3 - 2)(2^3 - 2^2) = 7 \times 6 \times 4 = 168.$$

THE BIG PICTURE (TRANSMITTER)



dinner is served

$$\underbrace{000}_{d}\,\underbrace{0111111}_{i}\,\underbrace{00101}_{n}\ldots$$

$$\begin{cases} 000 \\ 011 \\ 111 \\ 100 \\ 101 \end{cases} \xrightarrow{G} \begin{matrix} 0000000 \\ 1001111 \\ 1010011 \\ 0011100 \\ 1101000 \end{matrix}$$

0000000100111110100110011100 1101000

EXERCISE

Is $\{\vec{c}_2 + \vec{c}_3, \vec{c}_1 + \vec{c}_2, \vec{c}_1\}$ a basis of $\mathcal{C}$?

If yes,

- Specify the generator matrix.
- Explicitly specify the map $u_1 u_2 u_3 \to c_1 c_2 c_3 c_4 c_5 c_6 c_7$.

code $\mathcal{C}$
$\vec{c}_0 = 0000000$
$\vec{c}_1 = 0011100$
$\vec{c}_2 = 0111011$
$\vec{c}_3 = 1110100$
$\vec{c}_4 = 0100111$
$\vec{c}_5 = 1101000$
$\vec{c}_6 = 1001111$
$\vec{c}_7 = 1010011$

Solution (basis)

Let $G' = \begin{pmatrix} \vec{c}_2 + \vec{c}_3 \\ \vec{c}_1 + \vec{c}_2 \\ \vec{c}_1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}$.

- From the first three columns we see that rank (G') = 3, so $\{\vec{c}_2 + \vec{c}_3, \vec{c}_1 + \vec{c}_2, \vec{c}_1\}$ are linearly independent.
- n linearly independent vectors of an n-dimensional space always form a basis of the space.
- G' is the generator matrix associated to this basis.

code $\mathcal{C}$
$\vec{c}_0 = 0000000$
$\vec{c}_1 = 0011100$
$\vec{c}_2 = 0111011$
$\vec{c}_3 = 1110100$
$\vec{c}_4 = 0100111$
$\vec{c}_5 = 1101000$
$\vec{c}_6 = 1001111$
$\vec{c}_7 = 1010011$

Solution (encoding map)

$$G' = \left(\begin{array}{ccc|cccc} 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{array}\right)$$

$$(c_1, c_2, c_3, c_4, c_5, c_6, c_7) = (u_1, u_2, u_3)G'$$

Therefore:

$$\begin{aligned} c_1 &= u_1 & c_4 &= u_1 + u_3 \\ c_2 &= u_2 & c_5 &= u_1 + u_2 + u_3 \\ c_3 &= u_3 & c_6 &= u_1 + u_2 \\ & & c_7 &= u_1 + u_2 \end{aligned}$$

$$\vec{c} = (\underbrace{c_1, c_2, c_3}_{\text{message bits}}, \underbrace{c_4, c_5, c_6, c_7}_{\text{parity bits}}).$$

code $\mathcal{C}$
$\vec{c}_0 = 0000000$
$\vec{c}_1 = 0011100$
$\vec{c}_2 = 0111011$
$\vec{c}_3 = 1110100$
$\vec{c}_4 = 0100111$
$\vec{c}_5 = 1101000$
$\vec{c}_6 = 1001111$
$\vec{c}_7 = 1010011$

The above matrix G' is in systematic form.

DEFINITION (SYSTEMATIC FORM)

A generator matrix G_s is in **systematic form** if

$$G_s = \left(I_k, P_{k\times(n-k)}\right).$$

Notice that a systematic generator matrix is a matrix in reduced echelon form.

When the generator matrix is in systematic form, each codeword is written as

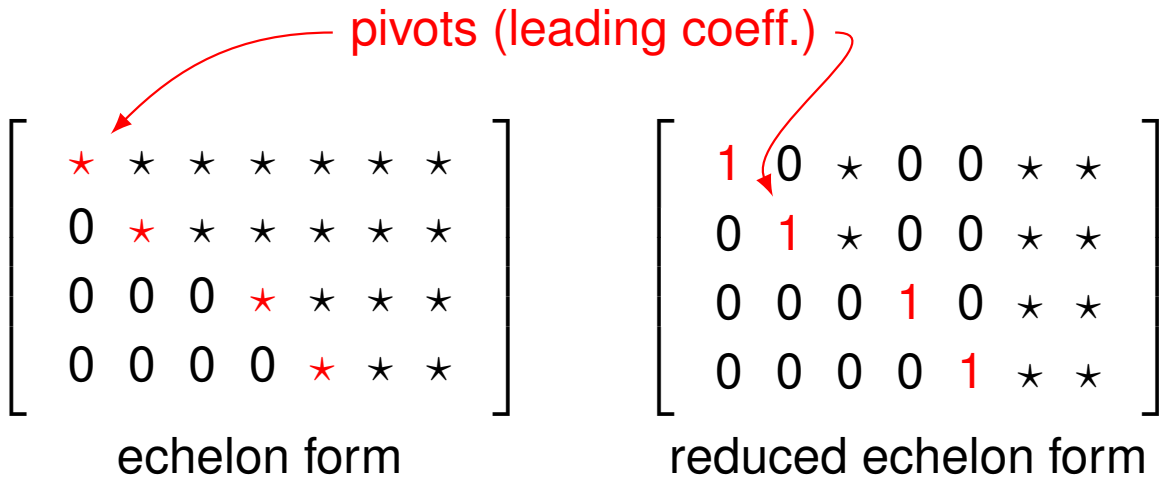
$$\vec{c} = \vec{u}G_s = (\underbrace{u_1, \ldots, u_k}_{\vec{u}I=\vec{u}}, \underbrace{c_{k+1}, \ldots, c_n}_{\vec{u}P}).$$

Given a linear code $\mathcal{C}$, how does one find the systematic form G_s?

1. Find a basis $\{\vec{c}_1, \ldots, \vec{c}_k\}$ of $\mathcal{C}$.

2. Form the generator matrix: $G = \begin{pmatrix} \vec{c}_1 \\ \vdots \\ \vec{c}_k \end{pmatrix}$.

3. Row-reduce G (Gaussian elimination on rows) to obtain a matrix in reduced echelon form.



This procedure uses the three operations below (that do *not* modify the vector space spanned by the rows of G):

T_{ij}: transpose (swap) rows i and j;

αS_i: scale row i by α;

αA_{ij}: scale row i by α and add to row j.

The result is a generator matrix for the same code.

To make sure that we have the identity matrix on the left, we may have to swap columns.

If we swap columns, we obtain a different code (different set of codewords) that has the same parameters (n, k, d_{min}) as the original code.

Example (Systematic Form)

Let G be a generator matrix of a $(5, 3)$ code on $\mathbb{F}_5$:

$$G = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 4 & 3 & 2 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \end{pmatrix}.$$

EXAMPLE (SYSTEMATIC FORM (CONT.))

$$G = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 4 & 3 & 2 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \end{pmatrix} \xrightarrow{T_{13}} \begin{pmatrix} 1 & 1 & 0 & 1 & 1 \\ 4 & 3 & 2 & 1 & 0 \\ 0 & 1 & 2 & 3 & 4 \end{pmatrix}$$

$$\xrightarrow{T_{23}} \begin{pmatrix} 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 \\ 4 & 3 & 2 & 1 & 0 \end{pmatrix} \xrightarrow{4A_{21}} \begin{pmatrix} 1 & 0 & 3 & 3 & 2 \\ 0 & 1 & 2 & 3 & 4 \\ 4 & 3 & 2 & 1 & 0 \end{pmatrix}$$

$$\xrightarrow{A_{13}} \begin{pmatrix} 1 & 0 & 3 & 3 & 2 \\ 0 & 1 & 2 & 3 & 4 \\ 0 & 3 & 0 & 4 & 2 \end{pmatrix} \xrightarrow{2A_{23}} \begin{pmatrix} 1 & 0 & 3 & 3 & 2 \\ 0 & 1 & 2 & 3 & 4 \\ 0 & 0 & 4 & 0 & 0 \end{pmatrix}$$

$$\xrightarrow{4S_3} \begin{pmatrix} 1 & 0 & 3 & 3 & 2 \\ 0 & 1 & 2 & 3 & 4 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \xrightarrow{3A_{32}} \begin{pmatrix} 1 & 0 & 3 & 3 & 2 \\ 0 & 1 & 0 & 3 & 4 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

$$\xrightarrow{2A_{31}} \begin{pmatrix} 1 & 0 & 0 & 3 & 2 \\ 0 & 1 & 0 & 3 & 4 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} = G_s.$$

EXAMPLE (SYSTEMATIC FORM (CONT.))

With the generator matrix in systematic form

$$G_s = \begin{pmatrix} 1 & 0 & 0 & 3 & 2 \\ 0 & 1 & 0 & 3 & 4 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix},$$

the map $\underbrace{(u_1, u_2, u_3)}_{\text{information word}} \mapsto \underbrace{(c_1, c_2, c_3, c_4, c_5)}_{\text{codeword}}$ is

$$\begin{aligned} c_1 &= u_1 \\ c_2 &= u_2 \\ c_3 &= u_3 \\ c_4 &= 3u_1 + 3u_2 \\ c_5 &= 2u_1 + 4u_2 \end{aligned}$$

Example (Systematic Form)

Here an example where we have to swap columns.

Let

$$G = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 2 \end{pmatrix}.$$

The steps towards the reduced echelon form are

$$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \sim \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

By swapping the second and third columns, we obtain the following generator matrix of a different (but equivalent) code.

$$\tilde{G} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}.$$

How to decode?

Decoding is about deciding the information word from the channel output.

If the channel output $\vec{y}$ is a codeword, then we assume that it equals the channel input.

In this case decoding is about inverting the encoding map. This is trivial if the generator matrix is in systematic form. (We read out the first k symbols of $\vec{y}$.)

But how to know if the channel output is a codeword?

We use the fact that a linear block code, like every subspace of a vector space, can be defined by a system of homogeneous linear equations.

The channel output is a codeword iff it satisfies those equations.

EXAMPLE

$$\begin{cases} c_4 = 3c_1 + 3c_2 \\ c_5 = 2c_1 + 4c_2 \end{cases} \Rightarrow \begin{cases} -3c_1 - 3c_2 + c_4 = 0 \\ -2c_1 - 4c_2 + c_5 = 0 \end{cases} \Rightarrow \begin{cases} 2c_1 + 2c_2 + c_4 = 0 \\ 3c_1 + c_2 + c_5 = 0 \end{cases}$$

Therefore, $\vec{y} \in \mathcal{C}$ iff

$$\begin{cases} 2y_1 + 2y_2 + y_4 = 0 \\ 3y_1 + y_2 + y_5 = 0 \end{cases}$$

i.e., iff

$$(y_1, y_2, y_3, y_4, y_5) \underbrace{\begin{pmatrix} 2 & 3 \\ 2 & 1 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}}_{H^{\mathsf{T}}} = \vec{0}.$$

Parity-Check Matrix

A **parity-check matrix** H for a linear (n, k) code is an $(n-k) \times n$ matrix that contains the coefficients of a system of homogeneous linear equations that defines the code.

Example

$$(y_1, y_2, y_3, y_4, y_5) \underbrace{\begin{pmatrix} 2 & 3 \\ 2 & 1 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}}_{H^T} = \vec{0} \quad \Leftrightarrow \quad \begin{cases} 2y_1 + 2y_2 + y_4 = 0 \\ 3y_1 + y_2 + y_5 = 0 \end{cases}$$

Theorem (Textbook Theorem 13.1)

If $G = (I_k, P)$, where P is a $k \times (n-k)$ matrix, is a generator matrix (in systematic form) of a linear (n, k) block code, then

$$H = \left(-P^{\mathsf{T}}, I_{n-k} \right)$$

is a parity-check matrix of the same code.

Proof:

$H = \left(-P^{\mathsf{T}}, I_{n-k} \right)$ has rank $(n-k)$, hence it defines a system of equations, the solution of which is a subspace of $\mathbb{F}^n$ of dimension k.

We want to show that $\vec{u}GH^T = \vec{0}$ for all information vectors $\vec{u}$.

This is true iff GH^T is the zero matrix (of size $k \times (n-k)$).

$$GH^{\mathsf{T}} = (I_k, P) \begin{pmatrix} -P \\ I_{n-k} \end{pmatrix} = -P + P = 0.$$

□

Example

$G = \begin{pmatrix} 1 & 0 & 0 & 3 & 2 \\ 0 & 1 & 0 & 3 & 4 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$ is the generator matrix of a $(5, 3)$ code over $\mathbb{F}_5$.

$$H = \begin{pmatrix} -3 & -3 & 0 & 1 & 0 \\ -2 & -4 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 2 & 0 & 1 & 0 \\ 3 & 1 & 0 & 0 & 1 \end{pmatrix}$$

is a corresponding parity-check matrix.

DEFINITION

Let H be the $(n-k) \times n$ parity-check matrix of a linear block code $\mathcal{C} \subset \mathbb{F}^n$ and let $\vec{y} \in \mathbb{F}^n$.

The **syndrome** of $\vec{y}$ is the vector

$$\vec{s} = \vec{y} H^{\mathsf{T}}.$$

By definition,

$$\vec{y} \in \mathcal{C} \quad \Longleftrightarrow \quad \vec{s} = \vec{0}.$$

Example (Hamming codes)

For every integer $m \geq 2$, there exists a binary Hamming code of parameters

$$n = 2^m - 1,$$
$$k = n - m.$$

The parity-check matrix is the $m \times n$ matrix whose columns consist of all non-zero vectors of length m.

Hamming codes are easy to encode and to decode.

Example (Cont.)

For instance, let $m = 3$.

A valid parity check matrix is

$$H = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix},$$

where, for convenience, the ith column is the binary representation of i.

The block-length is $n = 2^m - 1 = 7$.

The rank of H is m, hence the code dimension is $k = n - m = 4$.

$\vec{c} = (1, 1, 1, 0, \ldots, 0)$ is a codeword because $\vec{c}H^{\mathsf{T}} = 0$.

Hence $d_{min} \leq 3$.

We show that $d_{min} = 3$ by showing how to correct all error patterns of weight 1.

Example (Cont.)

Let $\vec{y} = \vec{c} + \vec{e}$, be the channel output, where $\vec{c} \in \mathcal{C}$, $\vec{e} \in \mathbb{F}_2^n$, and $w(\vec{e}) = 1$.

$\vec{s} = \vec{y}H^{\mathsf{T}} = \vec{c}H^{\mathsf{T}} + \vec{e}H^{\mathsf{T}} = \vec{e}H^{\mathsf{T}}$.

$\vec{s} = \vec{e}H^{\mathsf{T}}$ is the binary representation of the position of the error.

Hence we can correct the error.

EXERCISE

For the $(7, 4)$ Hamming code,

1. find a parity-check matrix of the form $H = \big(-P^{\mathsf{T}}, I_3 \big)$.
2. find the corresponding generator matrix $G = \big(I_4, P \big)$.

Solution

1. By moving to the far right the first, second, and fourth columns of the original parity-check matrix we obtain

$$H = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

   which has the desired form. Notice that the result is a different code (we have reordered the components) — still a Hamming code.

2.

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

EXERCISE

For the binary $(4, 1)$ repetition code,

1. find a generator matrix in systematic form.

2. find a parity-check matrix.

Solution

1. The code consists of the two codewords $(0,0,0,0)$ and $(1,1,1,1)$. There is only one basis of this code, hence there is only one generator matrix

$$G = (1,1,1,1).$$

2. Since the generator matrix is of the form (I_1, P), a corresponding parity-check matrix is of the form $(-P^T, I_3)$, namely

$$H = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}.$$

THEOREM

Let H be any parity check matrix of a linear code. The minimum distance of the code is the smallest positive integer d such that there are d columns of H that are linearly dependent.

Proof:

For a linear code, the minimum distance is the smallest weight of a non-zero codeword. Let $\vec{c} \neq 0$ be a codeword of smallest weight d. The fact that $\vec{c}H^T = 0$ proves that H has d linearly dependent columns.

We need to argue that fewer than d columns of H are not linearly dependent. Suppose that H has $t < d$ linearly dependent columns. Then we could find a non-zero codeword $\vec{c}$ of weight smaller than d such that $\vec{c}H^T = 0$. This is a contradiction. □

Example

The following is a parity check matrix for a Hamming code of parameters $n = 7$, $k = 4$.

$$H = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

Clearly no two columns are linearly dependent.

Column 1, 2, and 3 are linearly dependent.

Hence $d_{min} = 3$.

Equivalence Relation (review):

- $\mathcal{G}$ a set;
- $\sim$ an equivalence relation on $\mathcal{G}$;
- $[a]$ the equivalence class of $a \in \mathcal{G}$.

Key property that we will use: An equivalence relation on a set partitions the set into disjoint equivalence classes.

Example

- $\mathcal{G}$ is the set of all students in Switzerland
- $a \sim b$ if a and b attend the same university
- $[a]$ the subset of $\mathcal{G}$ that contains all the students that attend the same university as a

Note: As in the above example, equivalence classes need not have the same size.

Special Case: Group-Theoretic Construction

When $\mathcal{G}$ forms a commutative group $(\mathcal{G}, \star)$ and $(\mathcal{H}, \star)$ is a subgroup, there is a natural choice for $\sim$ defined as follows:

$a \sim b$ if there exists an $h \in \mathcal{H}$ such that $b = a \star h$.

Equivalently:
$a \sim b$ if $a^{-1} \star b \in \mathcal{H}$.

Claim: The above $\sim$ is indeed an equivalence relation:

Proof:

- (reflexive) $a \sim a$:
  true because $a^{-1} \star a \in \mathcal{H}$

- (symmetric) if $a \sim b$ then $b \sim a$:
  true because if $a^{-1} \star b = h \in \mathcal{H}$ then $b^{-1} \star a = h^{-1} \in \mathcal{H}$

- (transitive) if $a \sim b$ and $b \sim c$ then $a \sim c$:
  true because if $a^{-1} \star b = h_1 \in \mathcal{H}$ and $b^{-1} \star c = h_2 \in \mathcal{H}$, then $\mathcal{H}$ contains also $h_1 \star h_2$ which is $a^{-1} \star b \star b^{-1} \star c = a^{-1} \star c$

□

Since in this case an equivalence class has the form

$$[a] = \{a \star h : h \in \mathcal{H}\},$$

it makes sense to write

$$[a] = a \star \mathcal{H}.$$

For instance, if $\star$ is the addition, then $[a]$ is $\mathcal{H}$ translated by a.

Example

Let $(\mathcal{G}, +) = (\mathbb{Z}/10\mathbb{Z}, +)$ and let $\mathcal{H} = \{0, 5\}$.

Then $(\mathcal{H}, +)$ is a subgroup of $(\mathcal{G}, +)$, and the equivalence classes are:

$$
\begin{aligned}
[0] &= \mathcal{H} = \{0, 5\} \\
[1] &= 1 + \mathcal{H} = \{1, 6\} \\
[2] &= 2 + \mathcal{H} = \{2, 7\} \\
[3] &= 3 + \mathcal{H} = \{3, 8\} \\
[4] &= 4 + \mathcal{H} = \{4, 9\}.
\end{aligned}
$$

In the group theoretic language, $[a]$ is called the coset of $\mathcal{H}$ with respect to a.

Claim: All cosets of $\mathcal{H}$ have the same cardinality $\mathsf{card}(\mathcal{H})$.

Proof:

$$h_1, h_2 \in \mathcal{H} \text{ s.t. } h_1 \neq h_2 \implies a \star h_1 \neq a \star h_2.$$

Hence $a \star \mathcal{H}$ has the same cardinality as $\mathcal{H}$. □

Here is the group and subgroup of interest to us:

- The group $(\mathcal{G}, \star)$ is $(\mathbb{F}^n, +)$ for some finite field $\mathbb{F}$ and positive integer n;
- the subset $\mathcal{H}$ is a linear code $\mathcal{C} \subset \mathbb{F}^n$;
- then if $x, y \in \mathbb{F}^n$, $x \sim y$ iff $-x + y \in \mathcal{C}$;
- (equivalently, $x \sim y$ iff $y = x + c$ for some $c \in \mathcal{C}$).

Standard Array

The **Standard Array** is an array that has the elements of $\mathcal{C}$ in the top row, starting with the 0 codeword, and each row forms a coset of $\mathcal{C}$. Each element of $\mathbb{F}^n$ shows up exactly once in the standard array.

$$\left|\begin{array}{ccccc|l}
c_0 = 0 & c_1 & c_2 & \dots & c_{M-1} & \leftarrow [\mathcal{C}] \\
t_1 & t_1 + c_1 & t_1 + c_2 & \dots & t_1 + c_{M-1} & \leftarrow [t_1] \\
t_2 & t_2 + c_1 & t_2 + c_2 & \dots & t_2 + c_{M-1} & \leftarrow [t_2] \\
\vdots & \vdots & \vdots & & \vdots & \vdots \\
t_{L-1} & t_{L-1} + c_1 & t_{L-1} + c_2 & \dots & t_{L-1} + c_{M-1} & \leftarrow [t_{L-1}]
\end{array}\right|$$

where for each $j = 1, \dots, L-1$, t_j is such that

$$t_j \notin \left(\mathcal{C} \bigcup_{k=1}^{j-1} [t_k] \right).$$

Later we will choose the coset leaders more carefully.

Decoding Regions

Suppose that $\mathsf{card}(\mathcal{C}) = M$.

Think of the decoder as being specified by M decoding regions $\mathcal{D}_0, \ldots, \mathcal{D}_{M-1}$ that partition $\mathbb{F}^n$:

$$\mathcal{D}_i \bigcap \mathcal{D}_j = \emptyset \text{ if } i \neq j;$$

$$\bigcup_{i=0}^{M-1} \mathcal{D}_i = \mathbb{F}^n.$$

Upon observing $y \in \mathbb{F}^n$, the decoder finds the i such that $y \in \mathcal{D}_i$, and declares

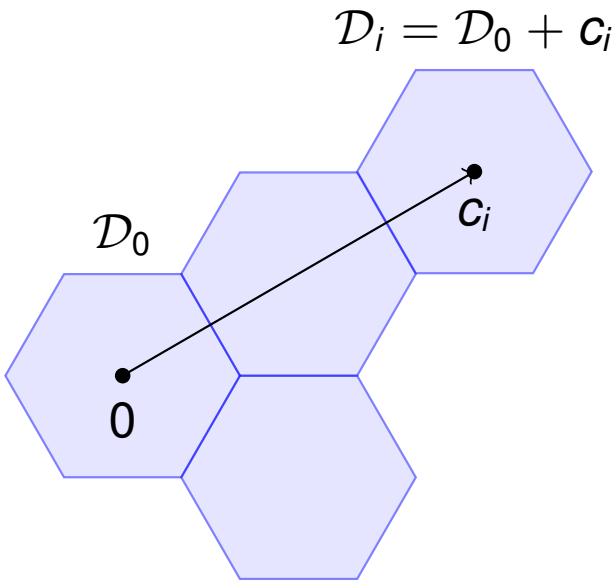
$$\hat{c} = c_i.$$

We let $\mathcal{D}_i$ be the ith column of the standard array.

$$
\begin{array}{ccccc}
c_0 = 0 & c_1 & c_2 & \dots & c_{M-1} \\
t_1 & t_1 + c_1 & t_1 + c_2 & \dots & t_1 + c_{M-1} \\
t_2 & t_2 + c_1 & t_2 + c_2 & \dots & t_2 + c_{M-1} \\
\vdots & \vdots & \vdots & & \vdots \\
t_{L-1} & t_{L-1} + c_1 & t_{L-1} + c_2 & \dots & t_{L-1} + c_{M-1} \\
\hline
\uparrow & \uparrow & \uparrow & \dots & \uparrow \\
\mathcal{D}_0 & \mathcal{D}_1 & \mathcal{D}_2 & & \mathcal{D}_{M-1}
\end{array}
$$

Note that $\mathcal{D}_i = c_i + \mathcal{D}_0$.

Geometrical Interpretation:



The union of all the $\mathcal{D}_i$ is $\mathbb{F}^n$. Hence every $y \in \mathbb{F}^n$ is in exactly one decoding region.

The Coset Decoder : How To Implement

To find the codeword associated to a channel output y, we could find y in the standard array and read out the entry on top of the same column.

Storing the whole standard array is impractical (often impossible for large codes).

The first column describes the geometry of all decoding regions. We should be able to leverage on that.

Claim: In the standard array, each element of a row has the same syndrome as the coset leader.

Proof:

- the elements of $[t_i]$ have the form $t_i + c$ for some $c \in \mathcal{C}$
- the syndrome of such an element is

$$(t_i + c)H^T = t_i H^T + cH^T = t_i H^T$$

  which is the syndrome of the coset leader t_i

□

Claim: The syndrome uniquely identifies the coset leader.

Proof:

Let t_i and t_j be coset leaders.

Suppose that $t_i H^T = t_j H^T$.

Then $(t_i - t_j)H^T = 0$.

Hence $t_i - t_j = c_k \in \mathcal{C}$.

It follows that t_i and t_j are in the same coset.

Since both are coset leaders, $t_i = t_j$.

□

In the previous slide, we have proved that the map

$$\begin{aligned} \mathcal{D}_0 &\to \mathbb{F}^{n-k} \\ t &\mapsto tH^T \end{aligned}$$

is one-to-one.

We use the pigeonhole principle to prove that it is also onto, hence it is a bijection. (We will not use this fact.)

Let $\mathbb{F} = \mathbb{F}_q$.

The standard array places the q^n elements of $\mathbb{F}^n$ into $\mathsf{card}(\mathcal{C}) = q^k$ columns and $\mathsf{card}(\mathcal{D}_0) = \frac{q^n}{q^k} = q^{n-k}$ rows.

The cardinality of $\mathbb{F}^{n-k}$ is also q^{n-k}. □

Hence the **coset decoder** can be implemented as follows:

1. we precompute and store the coset leaders and the corresponding syndrome;
2. to decode y, we compute its syndrom $s = yH^T$;
3. s encodes the row of y;
4. we use the lookup table to determine the corresponding coset leader, say, t_i;
5. t_i and y uniquely determine the column of y, namely $y = t_i + c_j$;
6. hence $c_j = y - t_i$;
7. the decoder declares that the transmitted codeword is $\hat{c} = y - t_i$.

To find the information word $\hat{u}$ that corresponds to $\hat{c}$ we solve the linear system

$$\hat{u}G = \hat{c}.$$

If G is in systematic form, then $\hat{u}$ consists of the first k components of $\hat{c}$.

Choosing Coset Leaders

There are many ways to choose the coset leaders $t_1, t_2, \cdots$.

In fact, every element of every row of the standard array can be chosen as the coset leader.

Theorem

In every row of the standard array:

- *select the coset leader to be one of the minimum-weight vectors in that row.*

Then the coset decoder is a minimum-distance decoder.

Proof:

Let y be in the ith row and jth column of the standard array, i.e., $y \in [t_i]$ and $y \in \mathcal{D}_j$.

We want to show that $d(y, c_j) \leq d(y, c_k)$ for every k.

On the LHS we have $d(y, c_j) = d(t_i + c_j, c_j) = w(t_i)$.

On the RHS we have $d(y, c_k) = d(t_i + c_j, c_k) = w(t_i + c_j - c_k) = w(t_i + c_l)$ for some l.

t_i and $t_i + c_l$ are in the same coset, and t_i is the coset leader. By choice, $w(t_i) \leq w(t_i + c_l)$. □

Example (Standard array for a Toy-Code)

Let $\mathcal{C} = \{\underbrace{(0,0,0)}_{c_0}, \underbrace{(1,1,1)}_{c_1}\}$

Standard Array:

$$\begin{array}{rr|l} c_0 = (0,0,0) & c_1 = (1,1,1) & \leftarrow \mathcal{C} \\ t_1 = (0,0,1) & t_1 + c_1 = (1,1,0) & \leftarrow t_1 + \mathcal{C} \\ t_2 = (0,1,0) & t_2 + c_1 = (1,0,1) & \leftarrow t_2 + \mathcal{C} \\ t_3 = \underbrace{(1,0,0)}_{\mathcal{D}_0} & t_3 + c_1 = \underbrace{(0,1,1)}_{\mathcal{D}_1 = \mathcal{D}_0 + c_1} & \leftarrow t_3 + \mathcal{C} \end{array}$$

Example (Decoding for the Toy-Code)

Transposed parity-check matrix and corresponding syndrome lookup table:

$$H^T = \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$$

t	s
$(0,0,0)$	$(0,0)$
$(0,0,1)$	$(0,1)$
$(0,1,0)$	$(1,0)$
$(1,0,0)$	$(1,1)$

If $y = (1,0,1)$ is received, the syndrome is $s = yH^T = (1,0)$, the coset leader is $t = (0,1,0)$, and the decoded codeword is $\hat{c} = y - t = (1,1,1)$.

Example ((6, 3) binary linear block code)

The code is defined by the following parity-check matrix

$$H = \left(\begin{array}{ccc|ccc} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \end{array}\right).$$

No two columns are linearly dependent. The first three columns are linearly dependent. Hence $d_{min} = 3$.

H has the form (P, I). The generator matrix G has the form $(I, -P^T)$:

$$G = \left(\begin{array}{ccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{array}\right).$$

Example (cont.)

The standard array has $2^k = 2^3 = 8$ columns and $2^{n-k} = 2^3 = 8$ rows.

The top row of the standard array is the code, and the left column consists of the elements of $\mathcal{D}_0$.

The code can correct all the errors of weight 1. Hence all the weight-1 words are in $\mathcal{D}_0$.

This gives:

000000	001110	010101	011011	100011	101101	110110	111000
000001							
000010							
000100							
001000							
010000							
100000							
t_7							

Example (cont.)

The next step is to fill in the rows for which we know the coset leader:

000000	001110	010101	011011	100011	101101	110110	111000
000001	001111	010100	011010	100010	101100	110111	111001
000010	001100	010111	011001	100001	101111	110100	111010
000100	001010	010001	011111	100111	101001	110010	111100
001000	000110	011101	010011	101011	100101	111110	110000
010000	011110	000101	001011	110011	111101	100110	101000
100000	101110	110101	111011	000011	001101	010110	011000
t_7							

As t_7 we choose a weight-2 word that has not yet appeared in any row, i.e., anything except 001100, 001010, 000110, 010100, 010001, 000101, 100010, 100001, 000011, 110000, 101000, 011000.

We choose $t_7 = 100100$.

Example (cont.)

000000	001110	010101	011011	100011	101101	110110	111000
000001	001111	010100	011010	100010	101100	110111	111001
000010	001100	010111	011001	100001	101111	110100	111010
000100	001010	010001	011111	100111	101001	110010	111100
001000	000110	011101	010011	101011	100101	111110	110000
010000	011110	000101	001011	110011	111101	100110	101000
100000	101110	110101	111011	000011	001101	010110	011000
100100	101010	110001	111111	000111	001001	010010	011100

The code will correct all the weight-1 channel-error patterns and the weight-2 channel-error pattern 100100. All the other channel-error patterns will lead to a decoding error.

Example (cont.)

The syndrome lookup table, i.e., the table that associates the coset leader t_i to the syndrome $s_i = t_i H^T$ is:

t_i	s_i
000000	000
000001	001
000010	010
000100	100
001000	110
010000	101
100000	011
100100	111

The code will correct all the weight-1 channel errors and the weight-2 channel error 100100. All the other error patterns will lead to a decoding error.

Example (cont.)

If $y = 011000$ is received, the decoder determines its syndrome

$$s = yH^T = 011,$$

and the corresponding coset leader

$$t = 100000.$$

t_i	s_i
000000	000
000001	001
000010	010
000100	100
001000	110
010000	101
100000	011
100100	111

The decoded word is

$$\hat{c} = y - t = 111000,$$

which is indeed a codeword.

Disclaimer:

The procedure that we have described requires to store $|\mathbb{F}|^{(n-k)}$ coset leaders and the corresponding syndrome.

While the approach is theoretically appealing, a lookup table of that size is prohibitive for most codes of practical interest.

Example (CD Rom)

The information on a CD Rom is encoded via two codes.

One code has parameters $|\mathbb{F}| = 2^8$, $n = 32$ and $k = 28$.

For it, there are $|\mathbb{F}|^{(n-k)} = \left(2^8\right)^4 = 4.29 \times 10^9$ coset leaders.

A coset leader is $8 \times 32 = 256$ bits long.

A syndrome is $8 \times 4 = 32$ bits long.

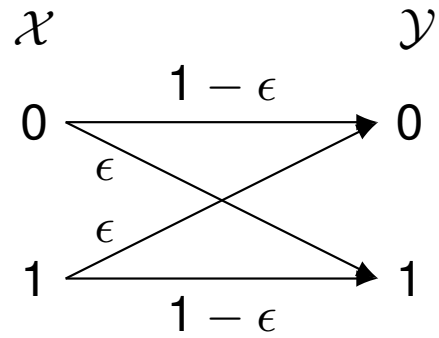
This requires more than 10^{12} bits — far more than the capacity of a CD Rom.

There are many ways to choose $\mathcal{D}_0$.

In fact, every element of every row of the standard array can be chosen as the coset leader.

Next, we learn how to choose the coset leaders so as to minimize the decoding error-probability.

Suppose that the channel is the following binary symmetric channel with input alphabet $\mathcal{X} = \{0, 1\}$ and output alphabet $\mathcal{Y} = \{0, 1\}$.



The probability of the error pattern $e \in \mathbb{F}^n$ is

$$\epsilon^{w(e)}(1-\epsilon)^{n-w(e)} = \left(\frac{\epsilon}{1-\epsilon}\right)^{w(e)}(1-\epsilon)^n.$$

We assume $\epsilon < 1/2$. Then $\frac{\epsilon}{1-\epsilon} < 1$, and the above expression is a decreasing function of $w(e)$.

Let $P_C(c_i)$ be the probability that the decoder decodes correctly, given that $c_i \in \mathcal{C}$ was transmitted.

When $c_0 \in \mathcal{C}$ is transmitted, the decoder makes the correct decision whenever $y \in \mathcal{D}_0$. But when c_0 is transmitted, the event $y \in \mathcal{D}_0$ is the same as the event $e \in \mathcal{D}_0$. Hence,

$$P_C(0) = \sum_{e \in \mathcal{D}_0} \epsilon^{w(e)}(1-\epsilon)^{n-w(e)} = \sum_{j=0}^{L-1} \epsilon^{w(t_j)}(1-\epsilon)^{n-w(t_j)},$$

where we have defined $t_0 = c_0 = 0$.

In conclusion, we maximize $P_C(0)$ if the coset leaders have the smallest possible weights.

Example (cont.)

The probability of error P_E is $1 - P_C$, where P_C is the probability that the error pattern e is in $\mathcal{D}_0$.

For the binary symmetric channel, this probability is

$$P_C = (1-\epsilon)^6 + 6(1-\epsilon)^5\epsilon + (1-\epsilon)^4\epsilon^2.$$

t_i
000000
000001
000010
000100
001000
010000
100000
100100

For $\epsilon = 0.1$, $P_C = 0.8923$.

(The probability that the channel output equals the channel input is $(1-\epsilon)^6$. This is 0.531 when $\epsilon = 0.1$.)

We have assumed for simplicity a binary symmetric channel, but the same idea applies to nonbinary channels (with input and output alphabet $\mathbb{F}$) for which the probability of an error pattern $e \in \mathbb{F}^n$ is decreasing with $w(e)$.

$P_C(c_i)$ is the probability that $y \in \mathcal{D}_i$ when c_i is transmitted.

This is the probability that $\underbrace{c_i + e}_{y} \in \underbrace{c_i + \mathcal{D}_0}_{\mathcal{D}_i}$.

It is the same as the probability that $e \in \mathcal{D}_0$. But this is $P_C(0)$.

Hence, for all i, $P_C(c_i) = P_C(0)$.

Hence, the unconditional probability of correct decoding is $P_C = P_C(0)$.

We conclude that the error probability is minimized when the coset leaders have the smallest weight.

This can be achieved by reordering each row of the standard array as follows.

Suppose that, in row j we have

$$w(t_j) > w(t_j + c_i).$$

By making $t_j + c_i$ the new coset leader of the jth row, the following happens:

1. the elements of that row are permuted;

2. the term of $P_C(0)$ that corresponds to the jth row increases whereas the other terms of $P_C(0)$ are unaffected.

To Summarize:

- A linear code $\mathcal{C}$ is a subspace of a vector space $\mathbb{F}^n$.
- The code partitions $\mathbb{F}^n$ into cosets.
- We cannot chose the cosets, but we can choose the coset leaders.
- To maximize $P_C(0)$, we let the leader of each coset be one of the smallest-weight elements of the coset.
- The cost leaders form $\mathcal{D}_0$, and $\mathcal{D}_i = c_i + \mathcal{D}_0$.
- The error probability is the same and equal to $P_C(0)$, no matter which codeword is transmitted.

Week 14: Reed Solomon Codes (Textbook Chapter 14)

Prof. Michael Gastpar

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

Spring Semester 2025

Outline

Today we learn about Reed Solomon Codes (Irving Reed and Gustave Solomon, 1960).

Why do we care?

- They are powerful (MDS)
- They are linear
- They let us choose various parameters (arbitrary $\mathbb{F}_q$ and $0 < k < n \leq q$)
- They have a nice construction
- They have elegant and efficient decoding algorithms

- They are used in many applications, namely:
  - Storage devices: tape, CDs, DVDs, bar codes, QR codes, ...
  - Digital radio/television broadcast
  - High-speed modems: ADSL, xDSL, ...
  - Deep space exploration modems (including Voyager 2, 1977, Jupiter, Saturn, Neptune)
  - Wireless mobile comm., including cellular phones and microwave links

Not surprisingly, the notion of polynomial extends to finite fields.

- Let $\vec{u} = (u_1, \ldots, u_k) \in \mathbb{F}^k$ for some finite field $\mathbb{F}$.
- We associate to $\vec{u}$ the polynomial

$$P_{\vec{u}}(x) = u_1 + u_2 x + \cdots + u_k x^{k-1}.$$

- $P_{\vec{u}}(x)$ can be evaluated at any $x \in \mathbb{F}$.
- The degree of a polynomial is the highest exponent i for which x^i has a non-zero coefficient.
- By convention, the zero polynomial has degree $-\infty$.

Example

- $\mathbb{F} = \mathbb{F}_5$;
- $P(x) = 2 + 4x + 3x^2$ is a polynomial of degree 2 over $\mathbb{F}_5$;
- $P(x) = P_{\vec{u}}(x)$ for $\vec{u} = (2, 4, 3) \in \mathbb{F}_5^3$;
- a polynomial $P(x)$ over a field $\mathbb{F}$ can be evaluated at any $x \in \mathbb{F}$: $P_{\vec{u}}(2) = 2 + 4 \cdot 2 + 3 \cdot 2^2 = 2 + 3 + 2 = 2$.

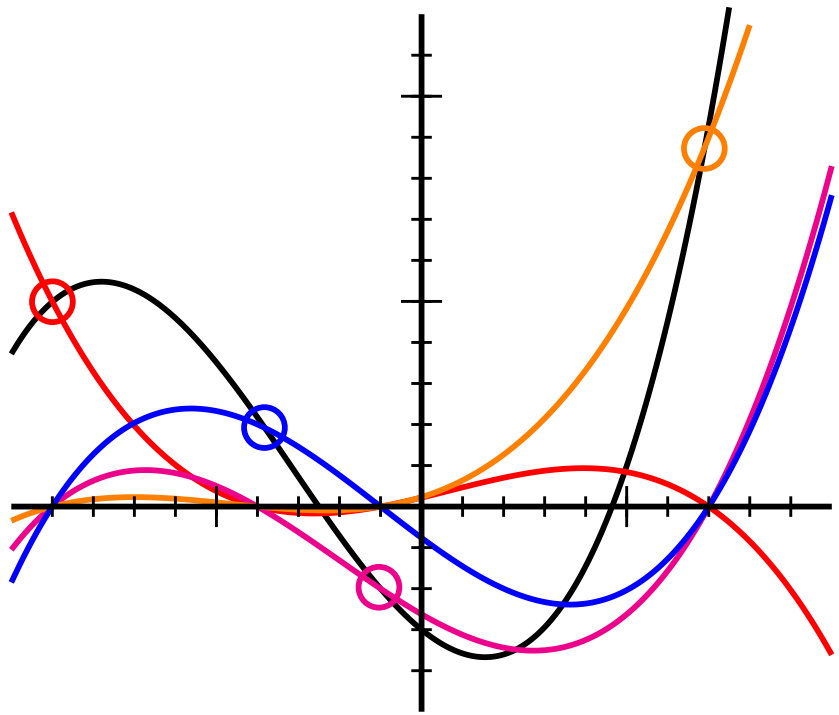
Example

- $P(x) = 3x$ has degree 1.
- $P(x) = 3$ has degree 0.
- $P(x) = 0$ has degree $-\infty$.

Interpolation via Polynomials

Problem: given a field $\mathbb{F}$ and k pairs $(a_i, y_i) \in \mathbb{F}^2$, where the a_i are all distinct, is there a polynomial $P(x)$ over $\mathbb{F}$ of degree at most $k-1$ (hence described by at most k coefficients) such that

$$P(a_i) = y_i, \quad i = 1, \ldots, k?$$



The answer is yes, obtained via Lagrange's interpolation polynomials.

To simplify notation, we demonstrate how it works by means of examples.

EXAMPLE

- Fix a field $\mathbb{F}$ and distinct field elements a_1, a_2, a_3 as well as y_1, y_2, y_3 (not necessarily distinct).
- We seek a polynomial $P(x)$ of degree at most 2 and coefficients in $\mathbb{F}$ such that $P(a_i) = y_i$.
- Suppose we can find a polynomial $Q_1(x)$ of degree at most 2, such that

$$Q_1(x) = \begin{cases} 1, & x = a_1 \\ 0, & x = a_i \neq a_1 \end{cases}$$

Example (cont.)

- Suppose that $Q_2(x)$ behaves similarly for a_2 and $Q_3(x)$ for a_3.
- The desired polynomial is then $P(x) = y_1 Q_1(x) + y_2 Q_2(x) + y_3 Q_3(x)$.
- Back to the construction of $Q_1(x)$.
- $(x - a_2)(x - a_3)$ is 0 at all the a_i except at a_1 where it is $(a_1 - a_2)(a_1 - a_3)$.
- Hence $\frac{(x-a_2)(x-a_3)}{(a_1-a_2)(a_1-a_3)}$ is the desired $Q_1(x)$.
- We construct $Q_2(x)$ and $Q_3(x)$ similarly.

Example (concrete case)

Over $\mathbb{F}_5$, find the polynomial $P(x)$ of degree not exceeding 2, for which $P(a_i) = y_i$ for

i	(a_i, y_i)
1	$(2, 3)$
2	$(1, 0)$
3	$(0, 4)$

SOLUTION

We find the Lagrange interpolation polynomials $Q_1(x)$ and $Q_3(x)$ and then form $P(x) = y_1 Q_1(x) + y_3 Q_3(x)$. (Notice that $Q_2(x)$ is not needed because $y_2 = 0$.)

$$Q_1(x) = \frac{(x-1)(x-0)}{(2-1)(2-0)} = \frac{(x-1)x}{2} = 3(x-1)x;$$

$$Q_3(x) = \frac{(x-2)(x-1)}{(0-2)(0-1)} = \frac{(x-2)(x-1)}{2} = 3(x-2)(x-1);$$

Finally

$$\begin{aligned} P(x) &= 3(3(x-1)x) + 4(3(x-2)(x-1)) \\ &= 4x^2 - 4x + 2x^2 - 6x + 4 \\ &= x^2 + 4. \end{aligned}$$

It should be obvious from the above example that we can proceed similarly for any field $\mathbb{F}$, for any positive integer k, and for any given set of k points with components in $\mathbb{F}$.

Let $P(x)$ be a polynomial over a field $\mathbb{F}$. The roots of $P(x)$ are those $b \in \mathbb{F}$ for which $P(b) = 0$.

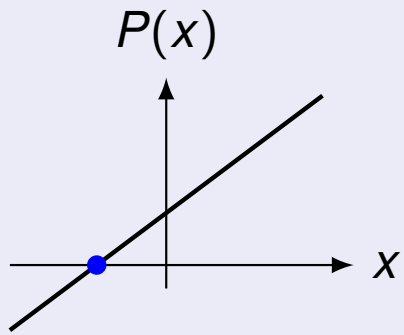
THEOREM (FUNDAMENTAL THM. OF ALGEBRA, TEXTBOOK THM 14.1)

Let $P(x)$ be a polynomial of degree at most $k-1$ over a field. If $P(x) \neq 0$ then the number of its distinct[2] roots is at most $k-1$.

[2] The theorem holds even for non-distinct roots if we account for their multiplicities. The stated version, which has an easier proof, is what we need.

Example

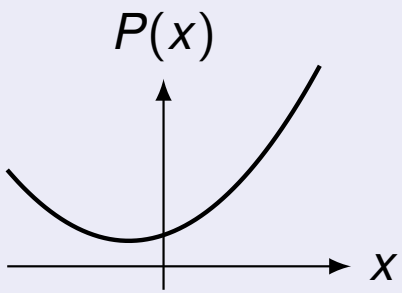
Let the field be $\mathbb{R}$ and $P(x) = ax + b$, $a \neq 0$. This polynomial has 1 root.



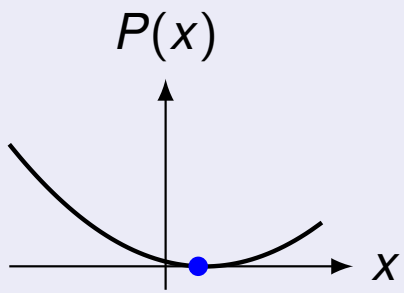
Example (cont.)

Let the field be $\mathbb{R}$ and $P(x) = ax^2 + bx + c$, $a \neq 0$.

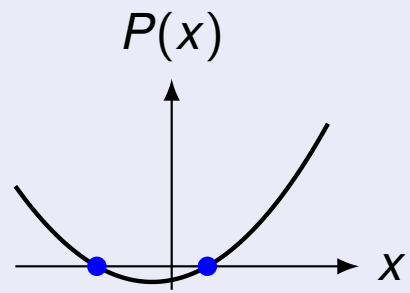
This polynomial has 0, 1, or 2 roots.



$b^2 - 4ac < 0$



$b^2 - 4ac = 0$

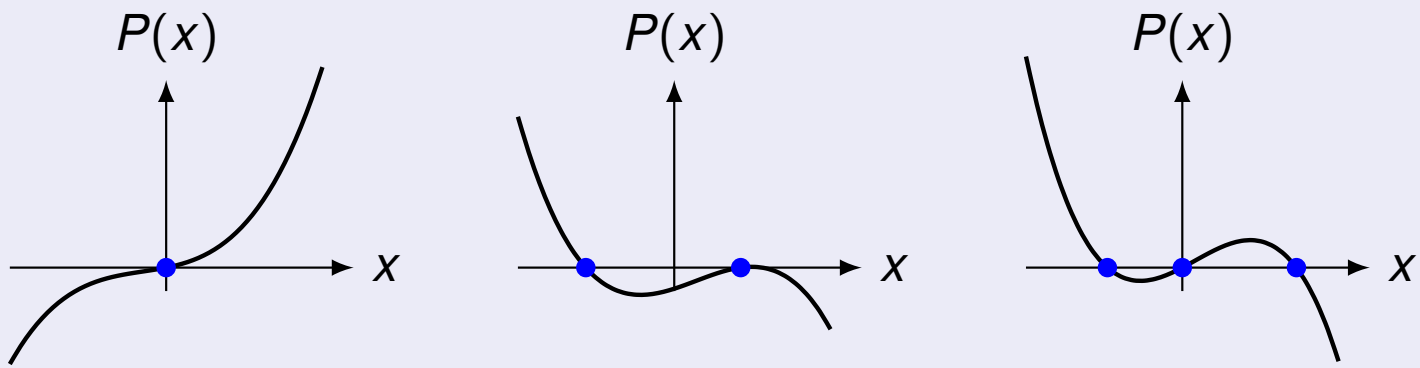


$b^2 - 4ac > 0$

Recall: The roots of $P(x)$ in $\mathbb{C}$ are $\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$.

Example (cont.)

Let the field be $\mathbb{R}$ and $P(x) = ax^3 + bx^2 + cx + d$, $a \neq 0$. This polynomial has 1, 2, or 3 roots.



Example

$P(x) = x^2 + x + 1$ has no roots in $\mathbb{F}_5$.

x	$P(x)$
0	1
1	3
2	2
3	3
4	1

Example

$P(x) = x^2 - 3x + 2$ has 2 roots in $\mathbb{F}_5$.

x	$P(x)$
0	2
1	0
2	0
3	2
4	1

Proof: Fundamental Theorem of Algebra

Let $P(x)$ be a polynomial of degree at most $k-1$. We prove that if it has more than $k-1$ distinct roots, then it is the zero polynomial (contraposition).

Let $a_1, \ldots, a_k \in \mathbb{F}$ be k distinct numbers, and consider the map

$$\psi : \mathbb{F}^k \to \mathbb{F}^k, \qquad \vec{u} \mapsto \big(P_{\vec{u}}(a_1), \ldots, P_{\vec{u}}(a_k)\big).$$

By Lagrange, for every $y_1, \ldots, y_k \in \mathbb{F}$, there exists at least one $\vec{u} \in \mathbb{F}^k$, such that $P_{\vec{u}}(a_i) = y_i$, $i = 1, \ldots, k$. Hence the above map is surjective (onto).

Because the domain of ψ and its co-domain have the same cardinality, by the pigeonhole principle, ψ is bijective.

Hence there is a single $\vec{u} \in \mathbb{F}^k$ for which $a_1, \ldots, a_k$ are roots of $P_{\vec{u}}$: it is $\vec{u} = \vec{0}$. □

Reed Solomon (RS) Code Construction

- Choose a finite field $\mathbb{F}$ and integers k, n, such that $0 < k \leq n \leq q$, where $q = \mathrm{card}(\mathbb{F})$.
- Choose n distinct elements $a_1, \ldots, a_n \in \mathbb{F}$. They exist because $n \leq q$.
- The codewords are defined via the following map:

$$\mathbb{F}^k \to \mathbb{F}^n, \qquad \vec{u} \mapsto \vec{c} = \big(P_{\vec{u}}(a_1), \ldots, P_{\vec{u}}(a_n)\big).$$

- This is a block code of length n over $\mathbb{F}$.

Example

- Let $(k, n, q) = (2, 3, 3)$. They fulfill $0 < k \leq n \leq q$.
- So we are in $\mathbb{F}_3$.
- Define $(a_1, a_2, a_3) = (0, 1, 2)$.

Write down all the codewords of the corresponding RS code.

$\vec{u}$	$P_{\vec{u}}(x)$	$\vec{c}$
00	0	000
01	x	012
02	$2x$	021
10	1	111
11	$1 + x$	120
12	$1 + 2x$	102
20	2	222
21	$2 + x$	201
22	$2 + 2x$	210

Solution

We are looking for $q^k = 3^2 = 9$ codewords of length $n = 3$.

Proof: RS Codes are Linear

Let $\vec{u}$, $\vec{v}$ be in $\mathbb{F}^k$ and $\alpha \in \mathbb{F}$. Notice that

$$P_{\alpha\vec{u}}(x) = \alpha u_1 + \alpha u_2 x + \cdots + \alpha u_k x^{k-1} = \alpha P_{\vec{u}}(x);$$
$$P_{\vec{u}+\vec{v}}(x) = (u_1 + v_1) + (u_2 + v_2)x + \cdots + (u_k + v_k)x^{k-1} = P_{\vec{u}}(x) + P_{\vec{v}}(x).$$

Hence

$$P_{\alpha\vec{u}+\vec{v}}(x) = \alpha P_{\vec{u}}(x) + P_{\vec{v}}(x).$$

This proves that if

$$\vec{u} \mapsto \vec{x} \in \mathcal{C}$$
$$\vec{v} \mapsto \vec{y} \in \mathcal{C}$$

then $\alpha\vec{u} + \vec{v} \mapsto \alpha\vec{x} + \vec{y} \in \mathcal{C}$.

Hence the code is linear. □

Proof: the design-parameter k is indeed the dimension of the RS-code

It suffices to prove that the encoding map is injective (one-to-one), implying that there are $[\mathrm{card}(\mathbb{F})]^k$ distinct codewords, hence that k is the dimension of the code.

When proving the fundamental theorem of algebra, we showed that the map

$$\begin{aligned} \psi : \mathbb{F}^k &\longrightarrow \mathbb{F}^k \\ \vec{u} &\mapsto \left(P_{\vec{u}}(a_1), \ldots, P_{\vec{u}}(a_k)\right) \end{aligned}$$

is one-to-one.

This guarantees that the encoding map

$$\mathbb{F}^k \longrightarrow \mathcal{C}, \qquad \vec{u} \mapsto \left(P_{\vec{u}}(a_1), \ldots, P_{\vec{u}}(a_k), P_{\vec{u}}(a_{k+1}), \ldots, P_{\vec{u}}(a_n)\right)$$

is one-to-one. □

Proof: RS codes are MDS

We want to prove that $d_{min} = n - k + 1$.

- Let $\vec{u} \in \mathbb{F}^k$ be a non-zero information vector. $P_{\vec{u}}(x)$ is a non-zero polynomial of degree at most $k - 1$. Hence it has at most distinct $k - 1$ roots.
- The corresponding codeword $\vec{c}$ (obtained by evaluating $P_{\vec{u}}(x)$ at n distince values) has at most $k - 1$ zeros.
- Hence $w(\vec{c}) \geq n - (k - 1) = n - k + 1$ for all $\vec{c}$.
- Since $\vec{c}$ is an arbitrary non-zero codeword, $d_{min} \geq n - k + 1$.
- By the Singleton's bound, $d_{min} \leq n - k + 1$.

Hence $d_{min} = n - k + 1$. □

To summarize, we have proved the following:

Theorem (Textbook Thm. 14.3)

A Reed Solomon code with design parameters k and n is a linear (n, k) code of minimum distance $d_{min} = n - k + 1$, i.e., it attains Singleton's bound with equality.

Note that the condition

$$\mathsf{card}(\mathbb{F}) \geq n$$

is necessary or else we can't find n distinct field elements $a_1, \ldots, a_n$.

Exercise

Find a generator matrix G for this code over $\mathbb{F}_3$.

$\vec{u}$	$P_{\vec{u}}(x)$	$\vec{c}$
00	0	000
01	x	012
02	$2x$	021
10	1	111
11	$1+x$	120
12	$1+2x$	102
20	2	222
21	$2+x$	201
22	$2+2x$	210

Solution

We need to find two codewords that are linearly independent.

Here are a few choices:

$$G = \begin{pmatrix} 0 & 1 & 2 \\ 1 & 1 & 1 \end{pmatrix} \quad G = \begin{pmatrix} 1 & 2 & 0 \\ 2 & 0 & 1 \end{pmatrix} \quad G = \begin{pmatrix} 2 & 1 & 0 \\ 1 & 0 & 2 \end{pmatrix}$$

Recall that each generator matrix defines an input/output map.

The map that we originally used to define RS codes

$$\vec{u} \to P_{\vec{u}}(x) \to \vec{c} = P_{\vec{u}}(a_1), \ldots, P_{\vec{u}}(a_n)$$

is just one of many possible maps.

In fact there are $(q^k - 1)(q^k - q) \cdots (q^k - q^{k-1})$ maps for a linear code of dimension k over $\mathbb{F}_q$.

Exercise

Which of the following is a valid parity-check matrix?

1. $A = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \end{pmatrix}$;
2. $B = \begin{pmatrix} 0 & 1 & 2 \\ 1 & 0 & 2 \end{pmatrix}$;
3. $C = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}$.

$\vec{u}$	$P_{\vec{u}}(x)$	$\vec{c}$
00	0	000
01	x	012
02	$2x$	021
10	1	111
11	$1+x$	120
12	$1+2x$	102
20	2	222
21	$2+x$	201
22	$2+2x$	210

Solution

A parity-check matrix for this code has $n - k = 1$ row, so only C is correctly sized.

Moreover, we can easily verify that $\vec{g}_i C^\mathsf{T} = 0$, where $\vec{g}_i$ is the ith row of G, hence $\vec{c} C^\mathsf{T} = 0$ for all codewords $\vec{c}$.

Therefore C is a parity-check matrix for the considered code.

$\vec{u}$	$P_{\vec{u}}(x)$	$\vec{c}$
00	0	000
01	x	012
02	$2x$	021
10	1	111
11	$1 + x$	120
12	$1 + 2x$	102
20	2	222
21	$2 + x$	201
22	$2 + 2x$	210

How about if we want the generator matrix for a specific encoding map?

Exercise

Find the generator matrix G that can be used as the encoder according to the given table.

$\vec{u}$	$P_{\vec{u}}(x)$	$\vec{c}$
00	0	000
01	x	012
02	$2x$	021
10	1	111
11	$1+x$	120
12	$1+2x$	102
20	2	222
21	$2+x$	201
22	$2+2x$	210

Solution

We want

- $(1,0)G = (1,1,1)$, so the first row of G is $(1,1,1)$.
- $(0,1)G = (0,1,2)$, so the second row of G is $(0,1,2)$.

Therefore

$$G = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \end{pmatrix}$$

More generally:

- We want a generator matrix for a given encoding map.
- It suffices to find the codewords that correspond to the following length-k information words:

$$(1,0,0,\ldots,0,0)$$
$$(0,1,0,\ldots,0,0)$$
$$(0,0,1,\ldots,0,0)$$
$$\vdots$$
$$(0,0,0,\ldots,1,0)$$
$$(0,0,0,\ldots,0,1)$$

- The corresponding codewords are linearly independent (proof below) and are k in number. Hence they form a basis.

We prove that a linear encoding map sends linearly independent information vectors to linearly independent codewords.

Proof:

Let $\vec{u}_i \in \mathbb{F}^k$, $i = 1, \ldots, k$, be a collection of linearly independent information vectors and let $\vec{c}_i \in \mathbb{F}^n$ be the corresponding codewords.

We prove that the the codewords are linearly independent.

Suppose, to the contrary, that the codewords are linearly dependent, i.e.,

$$\sum_{i=1}^{k} \lambda_i \vec{c}_i = 0,$$

with some of the λ_i non-zero.

Then

$$\sum_{i=1}^{k} \lambda_i \vec{u}_i = 0,$$

which contradicts the assumption that the information vectors are linearly independent. □

In particular, the map that we used to define the code sends $\vec{u} \in \mathbb{F}^k$ to

$$\vec{c} = \big(P_{\vec{u}}(a_1), \ldots, P_{\vec{u}}(a_n)\big).$$

If $\vec{u}$ has 1 at position i and 0 elsewhere, then $P_{\vec{u}}(x) = x^{i-1}$.

Hence, the map that we have used to defined RS codes has the generator matrix

$$G = \begin{pmatrix} 1 & 1 & 1 & \ldots & 1 \\ a_1 & a_2 & a_3 & \ldots & a_n \\ (a_1)^2 & (a_2)^2 & (a_3)^2 & \ldots & (a_n)^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ (a_1)^{k-1} & (a_2)^{k-1} & (a_3)^{k-1} & \ldots & (a_n)^{k-1} \end{pmatrix}.$$

Example (CD-ROM)

CDs use Cross-Interleaved Reed-Solomon Codes (CIRC) as follows:

- Each byte of the source is seen as an element of $\mathbb{F} = \mathbb{F}_{2^8}$.
- Source symbols, are encoded using a $(28, 24)$ RS code over $\mathbb{F}$.
- The elements of a sequence of many codewords are interleaved.
- The result is encoded using a $(32, 28)$ RS code over $\mathbb{F}$.

Notice that both codes have $d_{min} = 5$, but this small distance is compensated by the interleaver which distributes strings of errors among many codewords.

The end result is that error bursts up to 4000 bits can be corrected. We have roughly that many bits in a track segment of length 2.5mm.

Source: Standard ECMA-130, "Data interchange on read-only 120mm optical data disks (CD-ROM)", 2nd Edition, 1996.

Outline

Summary of Chapter 3

Basic Concepts of Error Detection and Correction:

- Two basic channel models: Erasure channel and Error channel.
- Code $=$ subset of all possible sequences of length n (over D-ary alphabet).
  - Convenient to talk about the number of codewords via the parameter $k = \log_D |\mathcal{C}|$.
- Minimum Distance Decoding
- Minimum Distance of a code
- Singleton's bound: $d_{min} \leq n - k + 1$.
  - A code satisfying this bound with equality is called MDS code.

Summary of Chapter 3

Linear Codes: Basic Properties, Design, Decoding

- Finite field: A finite set with two operations ("sum" and "product") satisfying the "natural" properties (as you know them over the reals).
  - Only exists if the cardinality of the finite set is a prime power, $\mathsf{card}(\mathbb{F}) = p^m$.
- Vector spaces over finite fields. Subspaces. Basis.
- Linear Code $=$ Subspace of a vector space over a finite field
- Generator matrix ($=$ collection of vectors spanning the subspace)
  - Particularly desirable form: Systematic generator matrix.
- Parity check matrix
- Key example: Hamming codes

Reed-Solomon Codes

- A very popular class of linear codes over a finite field $\mathbb{F}$. They are MDS!
- Block length has to be $n \leq |\mathbb{F}|$ (so we need large finite fields).
- They can be neatly described via *polynomials.*