Fundamentals of Information Theory

# Data Compression

Yayu Gao
School of Electronic Information and Communications
Huazhong University of Science and Technology
Email: yayugao@hust.edu.cn

# Outline

- Efficiency vs. Reliability
- Three key questions about data compression
- What is source coding?
- Get to know some codes
- What do we want from a source code?
- Kraft inequality——constraints on prefix codes
- How to find the optimal code?
- Shannon’s first theorem——Zero-error source coding theorem
- From Theory to Applications: source coding algorithms

# Outline

- Efficiency vs. Reliability
- Three key questions about data compression
- What is source coding?
- Get to know some codes
- What do we want from a source code?
- Kraft inequality——constraints on prefix codes
- How to find the optimal code?
- Shannon’s first theorem——Zero-error source coding theorem
- From Theory to Applications: source coding algorithms

# 本节学习目标

1. 理解效率与可靠性之间的折衷关系
2. 说出信源编码器与信源译码器各自的目标
3. 写出信源编码效率的评价指标
4. 说出信源编码优化问题
5. 说出什么是non-singular code
6. 说出什么是Uniquely decodable code
7. 说出什么是prefix code
8. 说出以上三种code的优缺点
9. 说出对信源编码的三个要求

**重难点：**

- 信源编码优化问题
- 认识几种编码类型

# 01

# Efficiency vs. Reliability

## Core Problem of Communication Systems

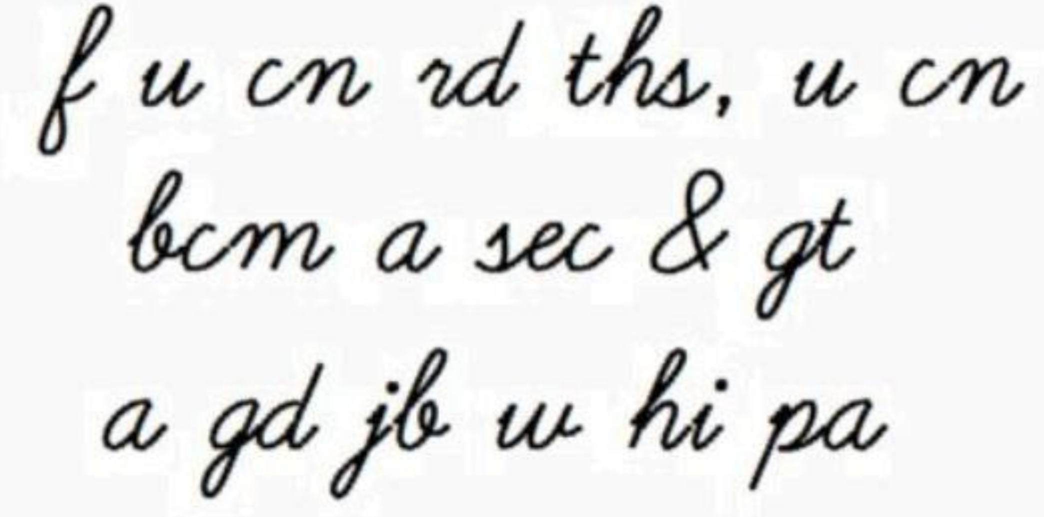
# Major challenges for establishing a communication theory

- How much information is transmitted?
  - **Quantify** information carried by symbols

- How to evaluate the performance of communication systems?
  - Transmission **efficiency** in communication systems
  - **Accuracy** of information transmission

- Nature of real-world scenarios?
  - Noise interference
- Core problems: **efficiency vs. reliability**
- Pioneering work by Claude E. Shannon and Norbert Wiener

# Efficiency vs. Reliability


- f u cn rd ths, u cn bcm a sec & gt a gd jb w hi pa!

- If you can read this, you can become a secretary and get a good job with high pay!

# Efficiency vs. Reliability

- The eternal issues of information theory
  - Lose reliability to achieve higher efficiency
  - Lose efficiency to achieve higher reliability

- Balance between efficiency and reliability
  - **Efficiency**
    - digital case : send as few symbols as possible
    - analog case : reduce the time that the channel or the bandwidth is used
  - **Reliability**
    - digital case : reduce the error probability as small as possible
    - analog case : reduce the noise as much as possible

# Source coding vs. Channel coding

- **Source Coding**
  - Core problem: **efficiency**
  - Efficiency: having an average code length that is as small as possible
  - Example: to use shorter code for the English letters which appear frequently, so as to reduce the average code length

- **Channel Coding**
  - Core problem: **reliability**
  - Reliability: to cope with the errors in the transmission
  - Example: to send the same sequence multiple times, so as to recover from the errors in channel

# Efficiency vs. Reliability: Redundancy

- 信源有冗余，可进行压缩
- **信源编码：**
- 信源编码是通过尽可能压缩信源冗余度的手段，实现提高通信有效性的目的。

例：中华人民共和国 $\xrightarrow{\text{压缩}}$ 中国　效率最高

显然，压缩后信源的冗余度越低，通信的**有效性**越好。但另一方面，信源冗余度过低，甚至没有冗余度，又会带来通信**可靠性**方面的问题。

# Efficiency vs. Reliability: Redundancy

**若通信过程中出现错误：**

**1. 当信源无冗余度**

中国 → ×国　美国 法国 德国…？

中国 → 中×　中国 中央 中间…？

**2. 当信源存在一定冗余度**

中华人民共和国 → ×华人民×和国 $\xrightarrow{\text{恢复}}$ 中华人民共和国

**结论**：通信有效性（信源编码）与可靠性（信道编码）

往往是一对矛盾。

# 02

# Three key questions about data compression

# Question 1: Why do we want to compress data?


- Save storage space

- Improve communication efficiency

# Question 2: Can we compress the data?

→ 鸟唱着歌儿，
花飘着清香，
春光明媚，
真是美啊！

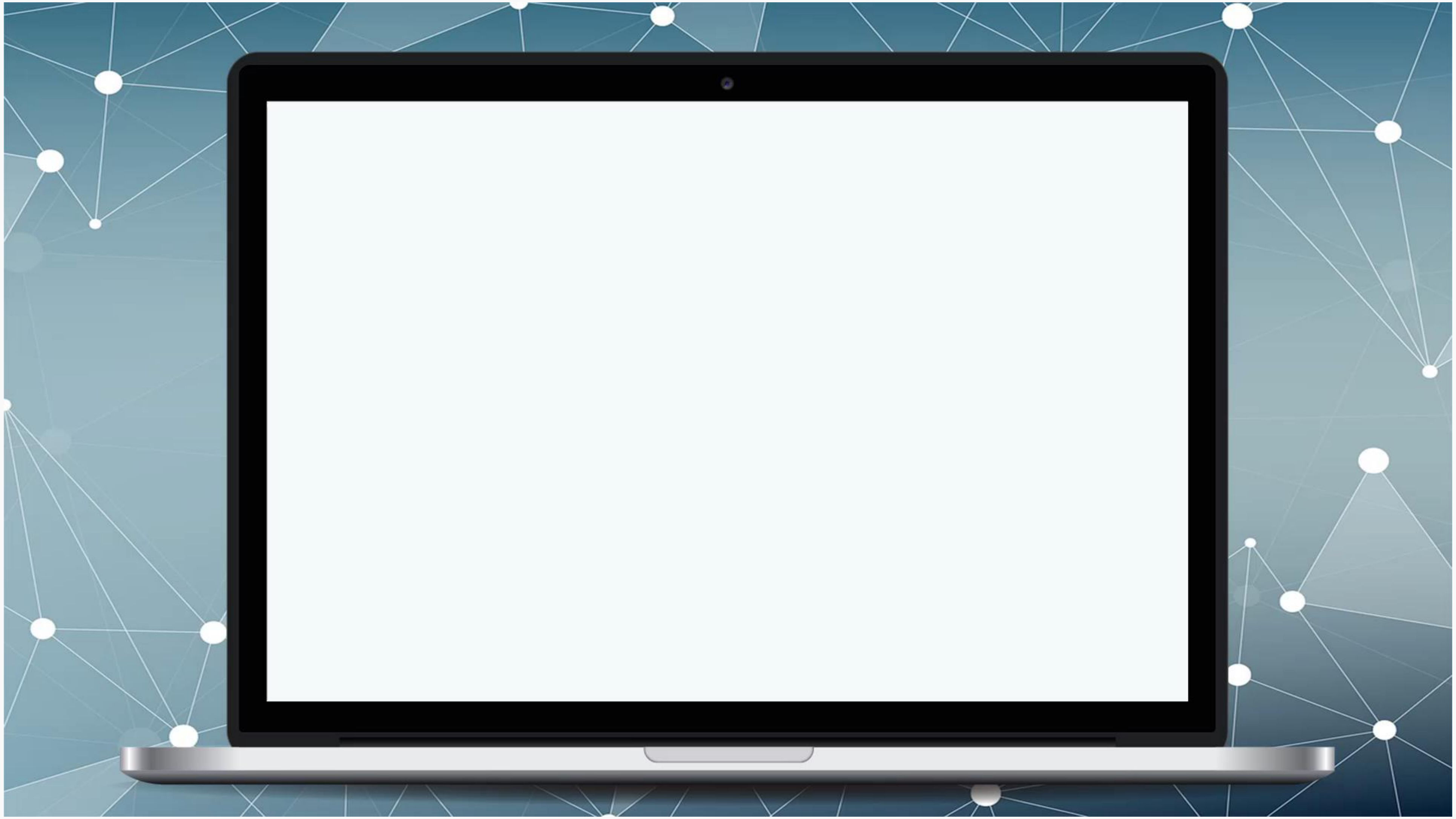
→ 鸟语花香

- There exists redundancy in the messages we transmit.

Eliminate redundancy → Compress data

- **Basic idea**: keep the same meaning and present it in the shortest manner.

# Question 3: Can we compress the data unlimitedly?



# Key question of this chapter

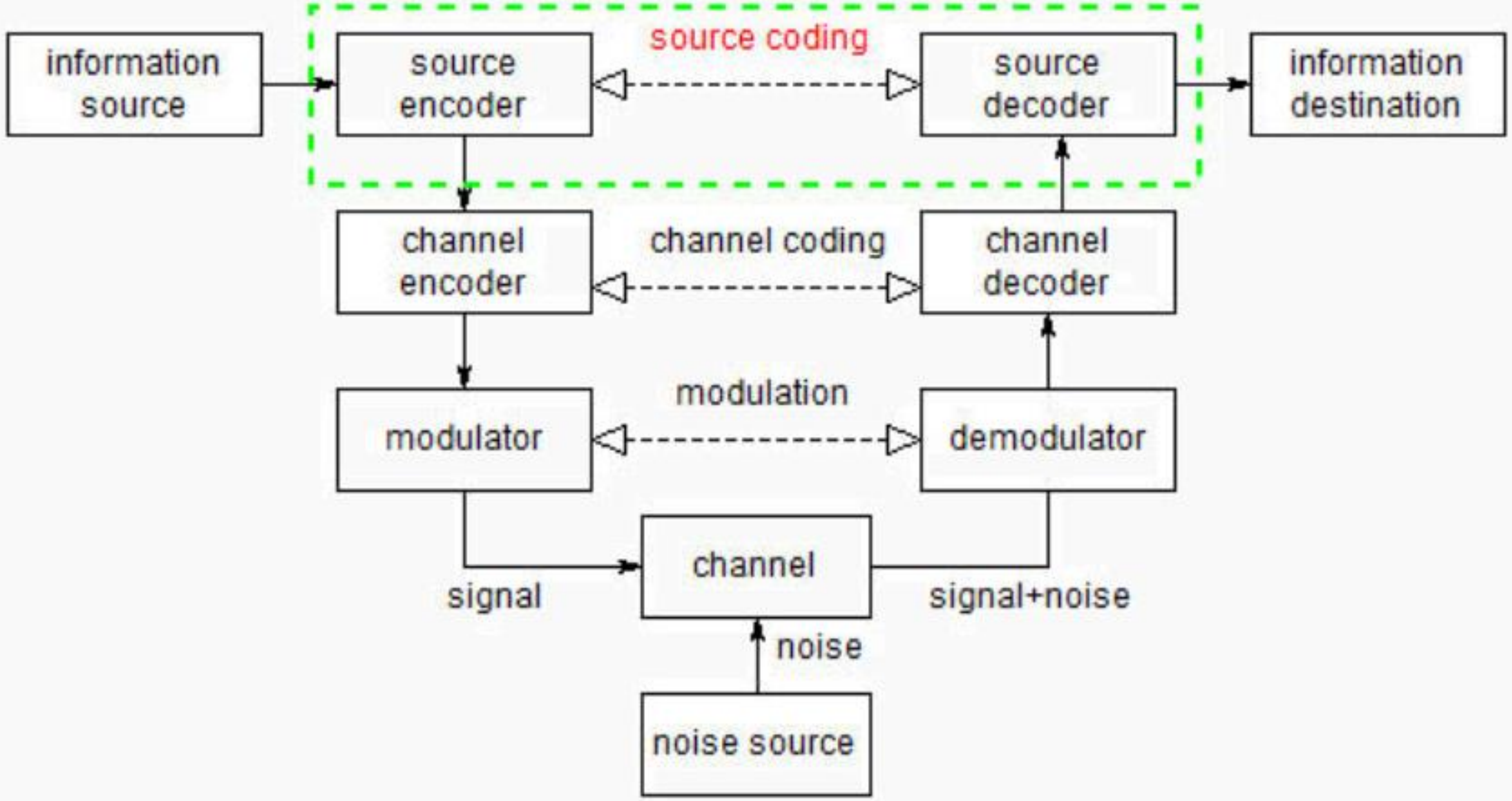
- Data compression has a limit?
- What is the limit?

# 03

# What is Source Coding?

# Who will do the job of data compression?

- In communication systems, data compression is done by the **source coding module**.



# Overview of source coding

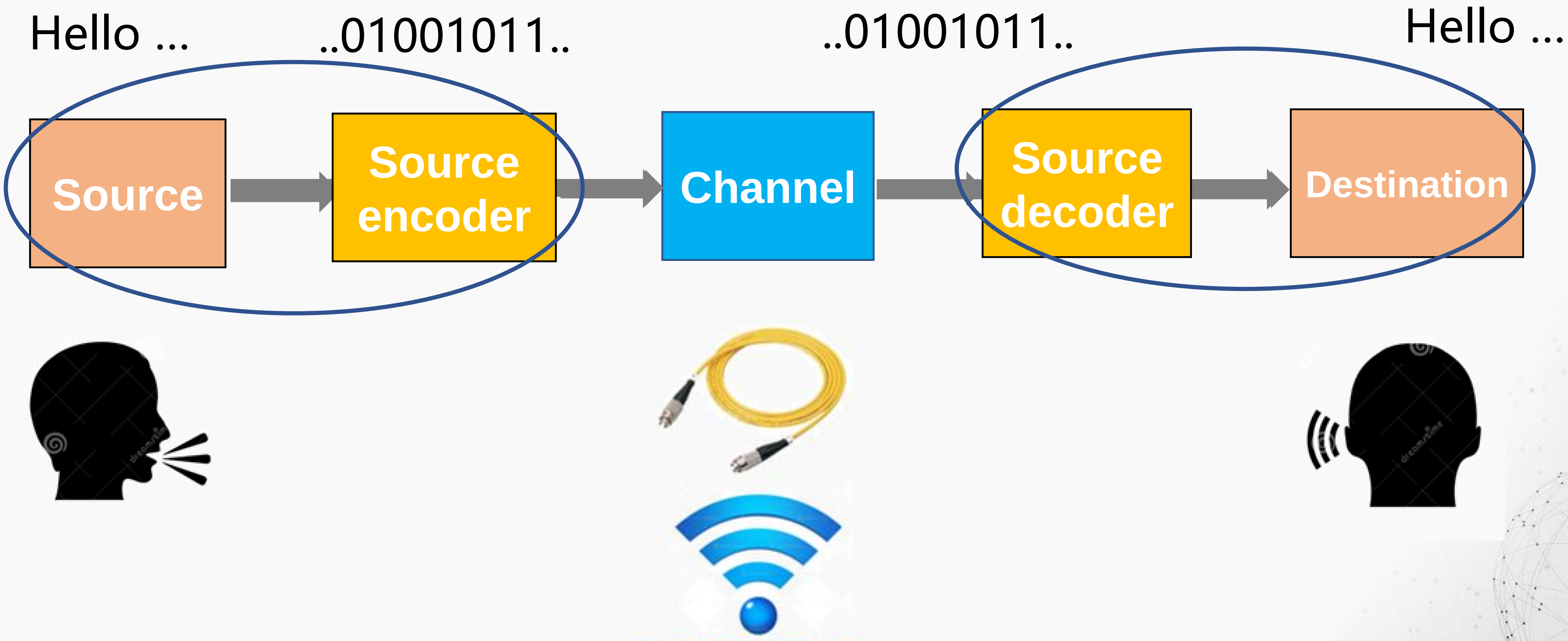
- Theoretical foundation of source codes
  - Zero-error source coding theorem
  - Rate-distortion source coding theorem

- Classification of source codes
  - Discrete source coding: zero-error coding
    - Example: Text compression
  - Continuous source coding: rate-distortion coding
    - Example: video compression

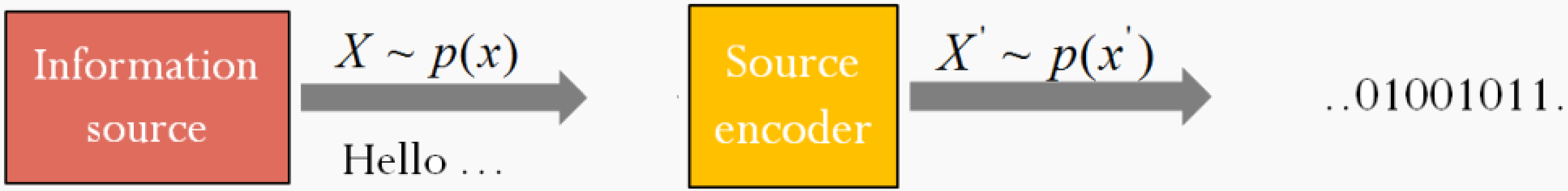
- In this chapter, we focus on the **zero-error coding for discrete information sources**.

# What is source coding?

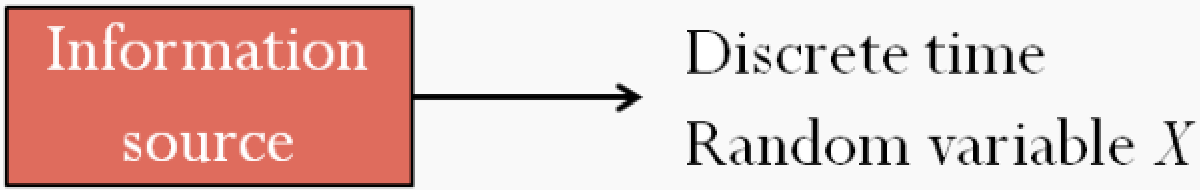
- In communication systems, data compression is done by the **source coding module**.



# Source **encoder** side...

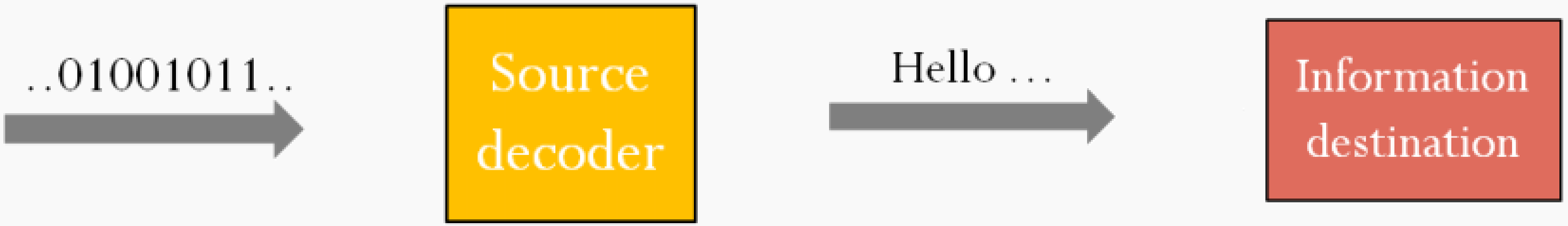



- How to characterize the information source?

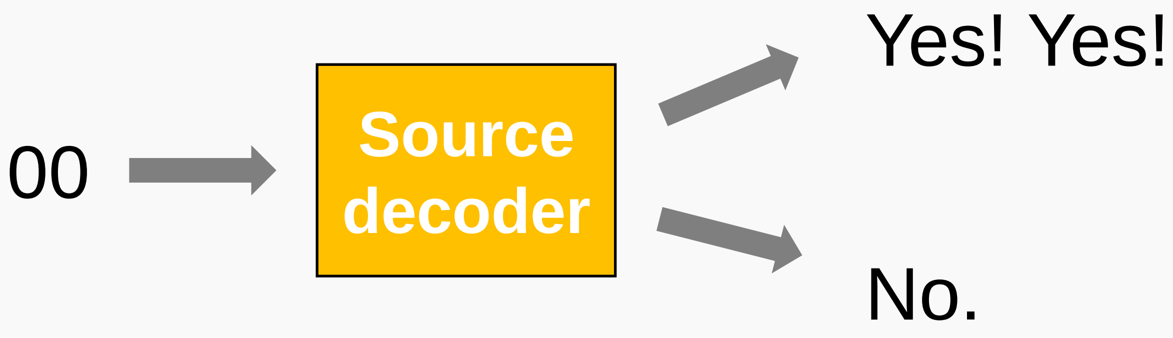


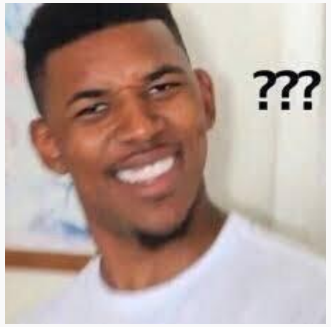

- How to represent the information of the source **efficiently**?

# Source **decoder** side…




- **Computational complexity** to recover the original sequence.
- Whether **uniquely** recover the original sequence?
  - e.g. “Yes” is coded as “0”, “No” is coded as “00”.



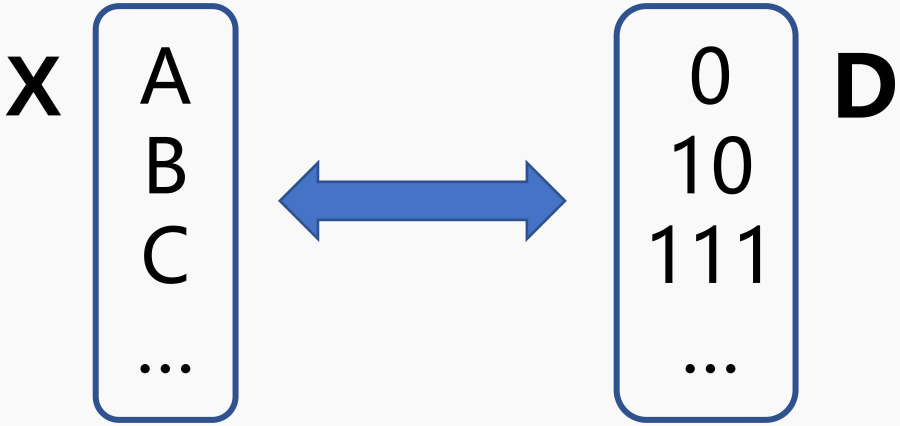
# Source code: definition

- A source code ***C*** for a random variable $X$ is a mapping between the space of $X$ to the space of code $D$.

$$\mathbf{C}:\mathbf{X}\rightarrow\mathbf{D}:\ \mathbf{C}(\mathbf{x}),$$

where $\mathbf{D}$ is the set of finite length strings of symbols from a $D$-ary alphabet[1].

- Let $C(x)$ denote the codeword corresponding to $x$.
- Let $l(x)$ denote the length of $C(x)$.




**How to design an efficient code?**

# Which code is more efficient?

- Source X={a, b, c}
- Codebook D={0,10,111}
- Output string: aabaacabaa

| | | |
|---|---|---|
| A<br>B<br>C | ⟷ | 0<br>10<br>111 |

Pr(X=a)=0.7
Pr(X=b)=0.2
Pr(X=c)=0.1

Source Code #1:  C(a)=111, C(b)=10, C(c)=0
Bit String #1: 1111111011111101111011111

Source Code #2: C(a)=0, C(b)=10, C(c)=111
Bit String#2: 0010001110100

- What is the total code length? $l(a) \times n(a) + l(b) \times n(b) + l(c) \times n(c)$
- What is the average code length per symbol?

$$\frac{l(a) \times n(a) + l(b) \times n(b) + l(c) \times n(c)}{n(a) + n(b) + n(c)} = \sum_{x \in \mathcal{X}} p(x) l(x)$$

# Coding efficiency: Expected length of a source code

- Definition: The expected length $L(C)$ of a source code $C(x)$ for a random variable $X$ with p.m.f. $p(x)$ is given by

$$\boxed{L(C) = \sum_{x \in \mathcal{X}} p(x) l(x)}$$

where $l(x)$ is the length of the codeword associated with $x$.

- Shorter average code length $\Rightarrow$ Higher efficiency $\Rightarrow$ Better compression

# Source code: example #1

- $r.v. X$

$$\begin{aligned} \Pr(X = a) &= 1/2, \\ \Pr(X = b) &= 1/4, \\ \Pr(X = c) &= 1/8, \\ \Pr(X = d) &= 1/8. \end{aligned}$$

$$\begin{aligned} C(a) &= 0, \\ C(b) &= 10, \\ C(c) &= 110, \\ C(d) &= 111. \end{aligned}$$

$$\boxed{H(X) = -\sum_{x \in \mathcal{X}} p(x) \log p(x) = 1.75 \text{ bits}}$$

$$\boxed{L(C) = \sum_{x \in \mathcal{X}} p(x) l(x) = 1.75}$$

- Observation:

$$H(X) = L(C)$$

- Example bit string: 0110111100110
- Decoded symbol: acdbac

# Source code: example #2

For $r.v. X$,

$$\boxed{\begin{aligned} \Pr(X = a) &= 1/3, \\ \Pr(X = b) &= 1/3, \\ \Pr(X = c) &= 1/3. \end{aligned}}$$

$$\boxed{\begin{aligned} C(a) &= 0, \\ C(b) &= 10, \\ C(c) &= 11. \end{aligned}}$$

$$\boxed{H(X) = -\sum_{x \in \mathcal{X}} p(x) \log p(x) = 1.58 \text{ bits}}$$

$$\boxed{L(C) = \sum_{x \in \mathcal{X}} p(x) l(x) = 1.66}$$

Observation:

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log p(x) = 3 \cdot \frac{1}{3} \cdot \log_2 3 = 1.58 \text{ bits}.$$

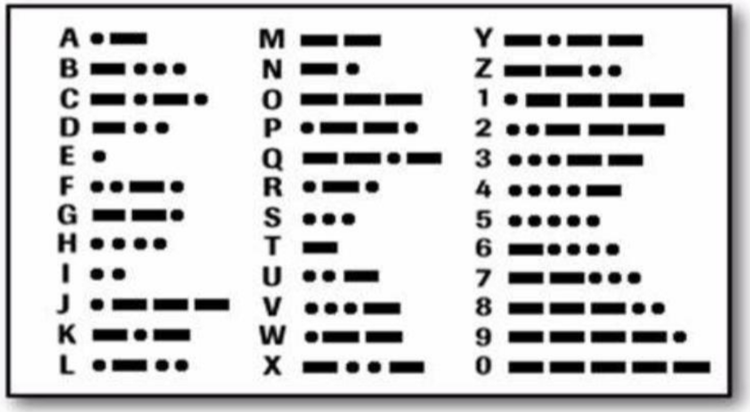
$$L(C) = \sum_{x \in \mathcal{X}} p(x) l(x) = \frac{1}{3} \cdot 1 + \frac{1}{3} \cdot 2 + \frac{1}{3} \cdot 2 = 1.66.$$

$$\therefore H(X) < L(C)$$

# Source coding problem: intuitive idea

Information source → Discrete time Random variable $X$

- Problem: design a source code to minimize the average codeword length
  - Also referred to as data compression problem
- Intuitive idea
  - Allocate the shortest code words to the most probable outcomes;
  - Allocate the inevitably longer ones to less likely outcomes
- Well-known example: Morse code



# 04

# Get to know some codes!

# Non-singular code

$$\mathbf{C} : \mathbf{X} \rightarrow \mathbf{D}: \mathbf{C}(\mathbf{x})$$

- Non-singular code:
  - Every element of the range of $X$ maps into a different string in **D**, i.e.

$$\boxed{x_i \neq x_j \Longrightarrow C(x_i) \neq C(x_j)}$$

- Good enough?
  - {A, B, C, D}={0, 1, 10, 110}
  - Received code string: 110
  - Unambiguous for single symbol, for a stream needs comma.

# Uniquely decodable code

- Extension $C*$ of a code $C$
  - Mapping from finite length strings of $X$ to finite length strings of $\mathbf{D}$, defined by

$$\boxed{C(x_1 x_2 \ldots x_n) = C(x_1)\, C(x_2) \ldots C(x_n)}$$

  - Example:
    If $C(x_1) = 00$, $C(x_2) = 11$, then $C(x_1 x_2) = 0011$.

- Uniquely Decodable Code
  - A code is called uniquely decodable if its extension is non-singular.
  - Only one possible source string

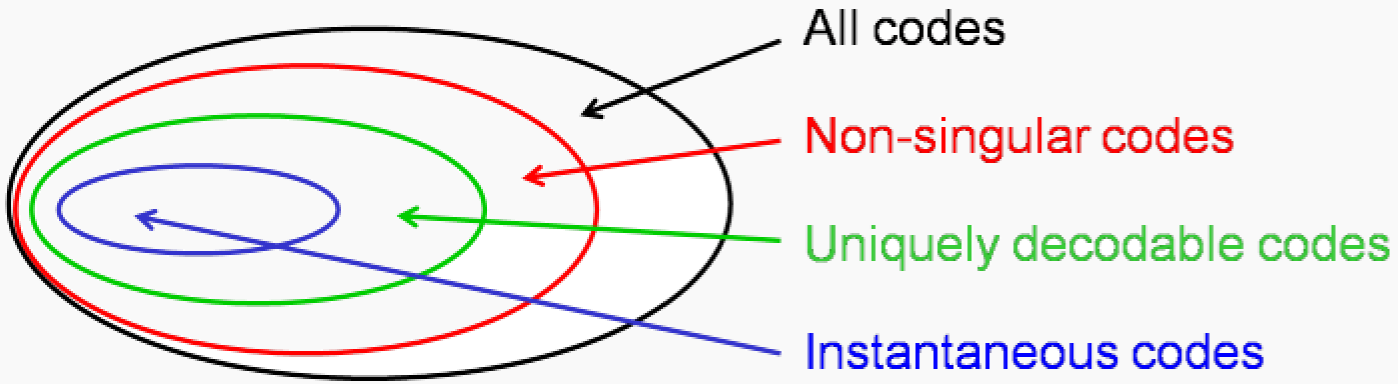
- Good enough?
  - {A, B, C, D}={10, 00, 11, 110}
  - Received code string: 00110001011
  - Need to wait until the entire string is received to decode it.

# Prefix/Instantaneous code

- Prefix Code = Instantaneous Code
  - A code is called a prefix code or an instantaneous code if no codeword is a prefix of any other codeword.
  - Can be decoded without reference to the future codewords

- Good enough?
  - {A, B, C, D}={1, 01, 001, 0001}
  - Received code string: 10010100011
  - **Desirable design goal**

# Classes of codes



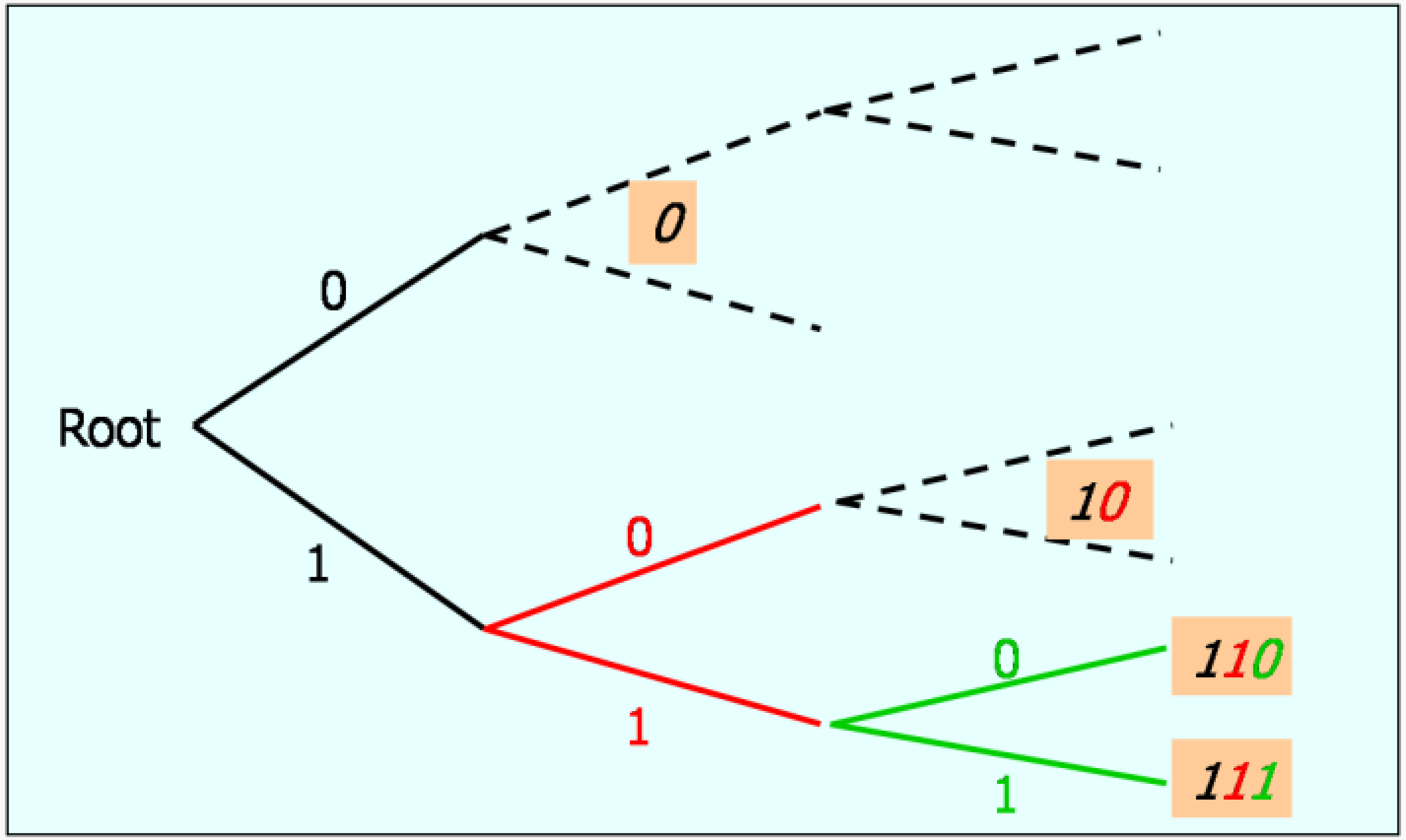

| X | Singular | Non-singular | Uniquely decodable | Prefix |
|---|---|---|---|---|
| A | 0 | 0 | 10 | 0 |
| B | 1 | 010 | 00 | 10 |
| C | 0 | 01 | 11 | 110 |
| D | 0 | 10 | 110 | 111 |

- The shortest codeword cannot be assigned for all symbols in a prefix code.

# Code tree

- We can always construct the code tree of a prefix code.

| X | Prefix |
|---|---|
| A | 0 |
| B | 10 |
| C | 110 |
| D | 111 |




- Can you tell what is the signature of a prefix code?

# 05

# What do we want from a source code?

# What do we want from a source code?

- **Efficiency**
  - Find codes with the minimum average code length.
- **Reversibility**
  - The code must be uniquely decodable
- **Instantaneous code**
  - Detect where the code for one input symbol ends and the next begins.
- **Easy implementation** of the code
  - From algorithm design's point of view

**Compression**

**Zero-error**

**Engineering**

# Which one is the champion code?

For the discrete information source $S$, the output symbols are $A$, $B$, $C$, $D$. The probability of each symbol is given as $P$, $C_1 \sim C_6$ are possible source codes.

| Source Symbol | $P$ | $C_1$ | $C_2$ | $C_3$ | $C_4$ | $C_5$ | $C_6$ |
|---|---|---|---|---|---|---|---|
| $A$ | 0.6 | 00 | 0 | 0 | 0 | 0 | 0 |
| $B$ | 0.25 | 01 | 10 | 10 | 01 | 10 | 10 |
| $C$ | 0.1 | 10 | 110 | 110 | 011 | 11 | 11 |
| $D$ | 0.05 | 11 | 1110 | 111 | 111 | 01 | 0 |
| | | | | | | | |

# Requirement #1: Efficiency

- Average code length of the other codes

$$
\begin{aligned}
C_1: &\quad 2 \times 0.6 + 2 \times 0.25 + 2 \times 0.1 + 2 \times 0.05 = 2 \\
C_2: &\quad 1 \times 0.6 + 2 \times 0.25 + 3 \times 0.1 + 4 \times 0.05 = 1.60 \\
C_3: &\quad 1 \times 0.6 + 2 \times 0.25 + 3 \times 0.1 + 3 \times 0.05 = 1.55 \\
C_4: &\quad 1 \times 0.6 + 2 \times 0.25 + 3 \times 0.1 + 3 \times 0.05 = 1.55 \\
C_5: &\quad 1 \times 0.6 + 2 \times 0.25 + 2 \times 0.1 + 2 \times 0.05 = 1.40 \\
C_6: &\quad 1 \times 0.6 + 2 \times 0.25 + 2 \times 0.1 + 1 \times 0.05 = 1.35
\end{aligned}
$$

- $C_6$ is the most efficient code

# Requirement #2: Reversibility

- Reversibility
  - Possible to decode the code words.
- Singular codes
  - Multiple source symbols have the same code
  - $C_6$ is a singular code, as $A$ and $D$ have the same code.
- Not all non-singular codes are reversible
  - $C_5$ is non-singular, but non-reversible

$$\underset{A}{\underline{0}}\ \underset{C}{\underline{11}}\ \underset{A}{\underline{0}} \qquad \underset{D}{\underline{01}}\ \underset{B}{\underline{10}}$$

- Reversible codes are also called uniquely decodable codes
  - $C_1 \sim C_4$ are uniquely decodable

# Requirement #3: Instantaneous Code

- Instantaneous code
  - It is possible to decode each word without reference to succeeding code symbols
  - No need to buffer the source sequence

- Example
  - $C_3$ and $C_4$ are uniquely decodable and equally efficient
  - $C_4$

| | | | |
|---|---|---|---|
| 0 111 111 0$\cdots$ | $\rightarrow$ | $A\,D\,D\,\cdots$ | *OK!* |
| 0111111 | $\rightarrow$ | $B\,D\cdots$*or* $C\,D\cdots$ | *Which one?* |

  - $C_3$

| | | |
|---|---|---|
| 0111111 | $\rightarrow$ | $A\,D\,D$ |

  - $C_3$ is instantaneous, but $C_4$ isn't

# Which one is the champion code?

| Source Symbol | $P$ | $C_1$ | $C_2$ | $C_3$ | $C_4$ | $C_5$ | $C_6$ |
|---|---|---|---|---|---|---|---|
| $A$ | 0.6 | 00 | 0 | 0 | 0 | 0 | 0 |
| $B$ | 0.25 | 01 | 10 | 10 | 01 | 10 | 10 |
| $C$ | 0.1 | 10 | 110 | 110 | 011 | 11 | 11 |
| $D$ | 0.05 | 11 | 1110 | 111 | 111 | 01 | 0 |
| $\bar{L}$ | | 2 | 1.60 | 1.55 | 1.55 | 1.40 | 1.35 |

- C1 and C2 are also instantaneous codes
- Design goal:
  - We want a uniquely decodable, instantaneous code with the shortest average code length.
  - C3 is the best candidate

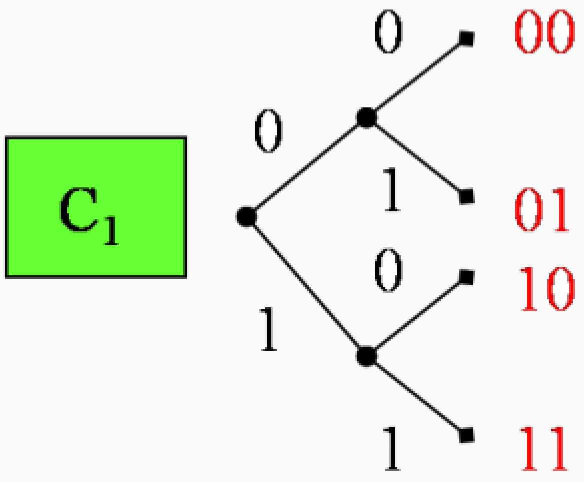
# How to recognize an instantaneous code?

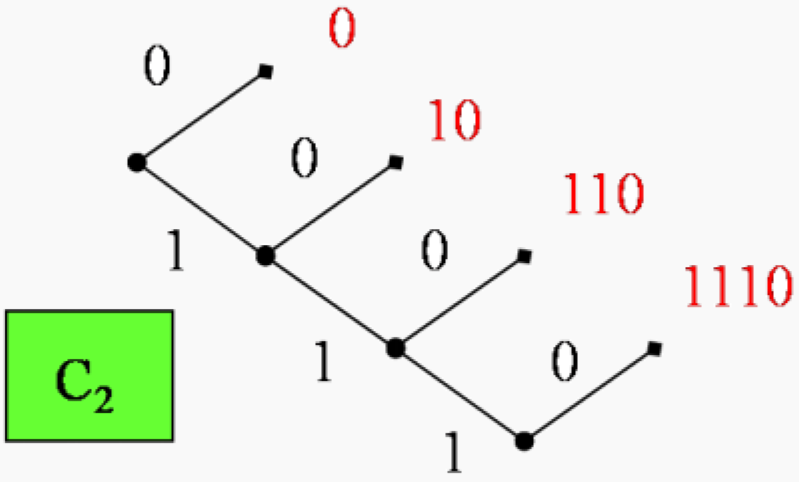
- **Prefix condition**
  - A necessary and sufficient condition for a code to be instantaneous is that no complete word of the code is a prefix of some other code word

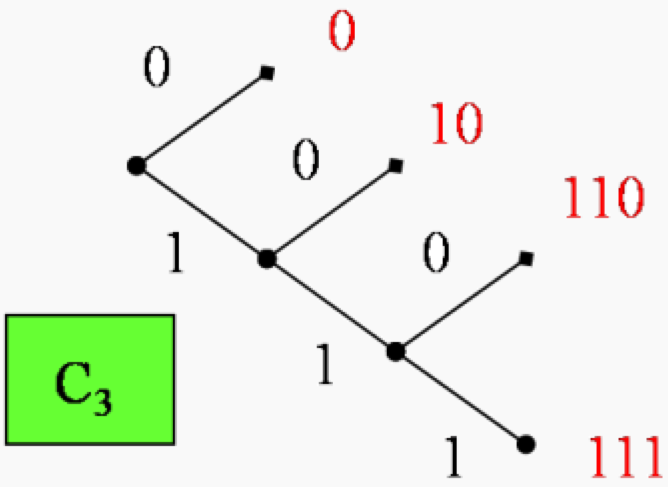
| Source Symbol | $C_4$ |
|---|---|
| $A$ | 0 |
| $B$ | 01 |
| $C$ | 011 |
| $D$ | 111 |

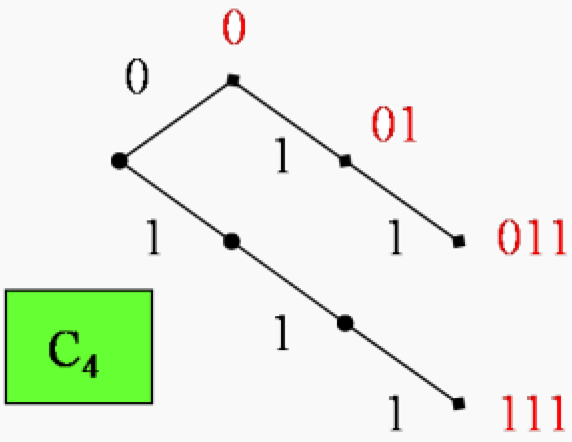
- Why is C4 not an instantaneous code?
  - In case of a 0 in the sequence, we can not distinguish between the code words 0, 01 or 011.
  - 0 is a prefix of different code words.

# How to recognize an instantaneous code? Code tree

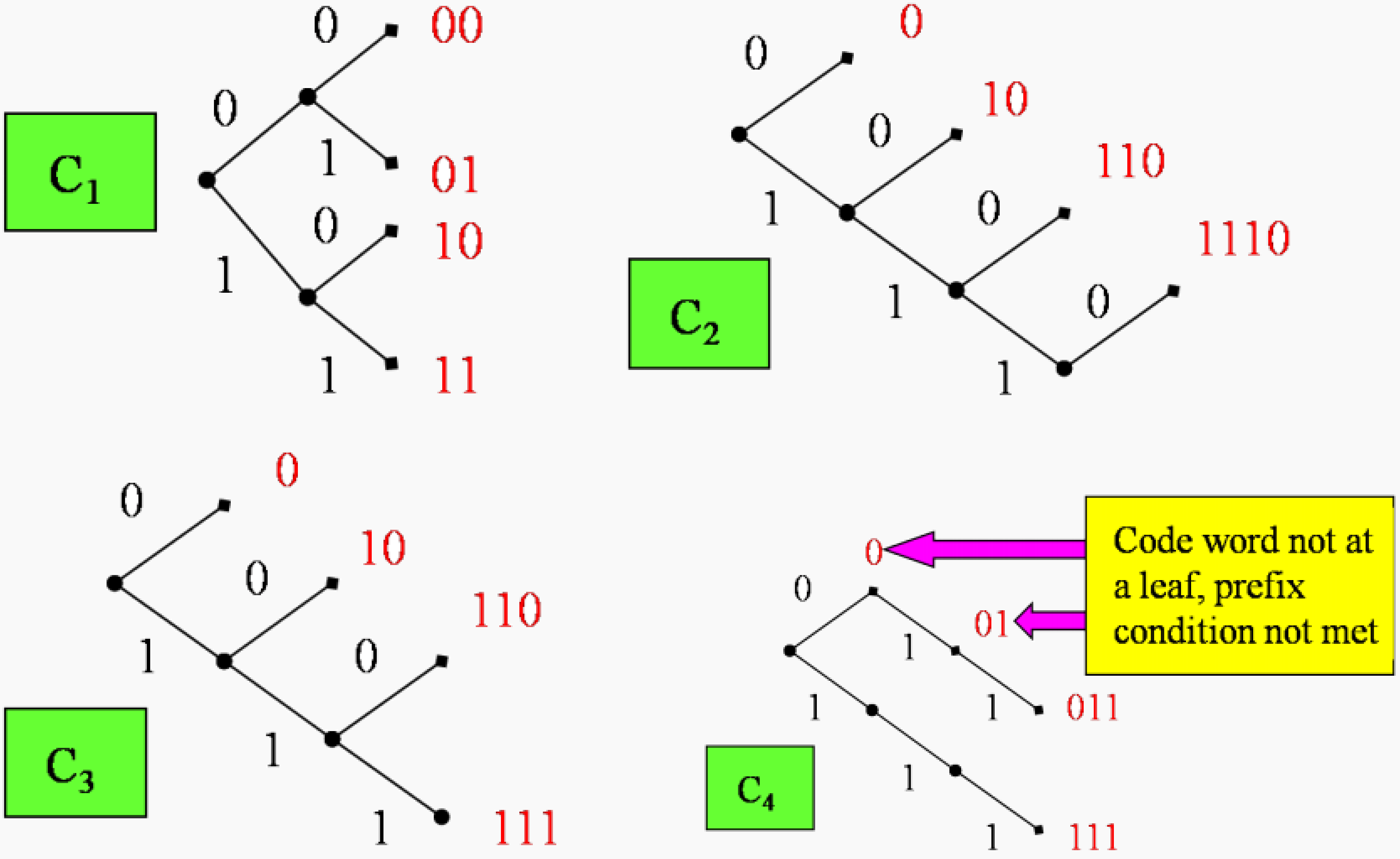


# How to recognize an instantaneous code? Code tree

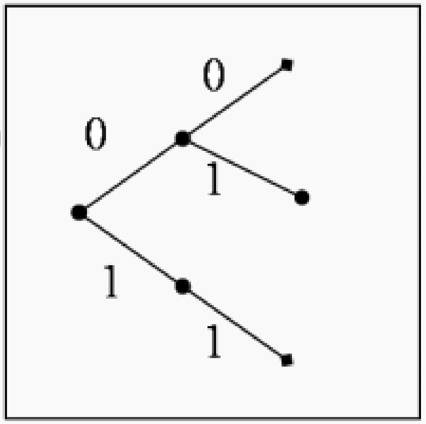


# Code tree: example

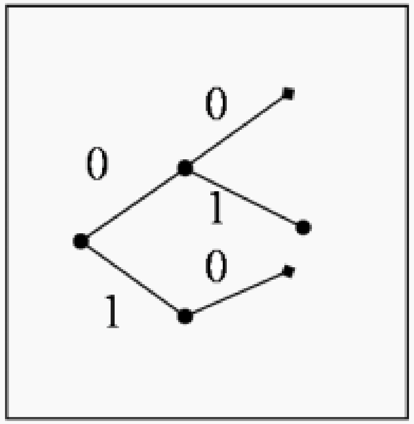
| Source Symbol | $P$ | $C_1$ | $C_2$ | $C_3$ | $C_4$ | $C_5$ |
|---|---|---|---|---|---|---|
| $S_1$ | 0.5 | 0 | 0 | 1 | 1 | 01 |
| $S_2$ | 0.1 | 11 | 11 | 01 | 01 | 001 |
| $S_3$ | 0.02 | 00 | 00 | 00 | 001 | 0001 |
| $S_4$ | 0.38 | 11 | 010 | 10 | 0001 | 00001 |

- What is the code tree of $C_3$?

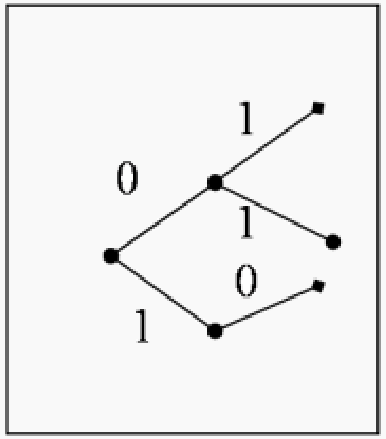
A)




B)




C)



# Summary: What do we want from a source code?

- **Efficiency**
  - Find codes with the minimum average code length.
- **Reversibility**
  - The code must be uniquely decodable
- **Instantaneous code**
  - Detect where the code for one input symbol ends and the next begins.
- **Easy implementation** of the code
  - From algorithm design's point of view

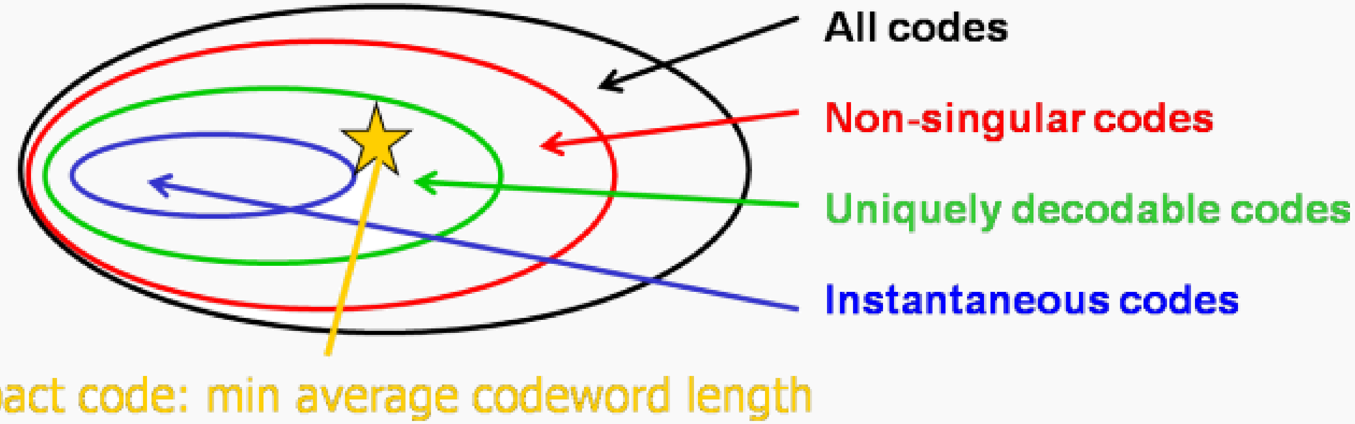
Compression

Zero-error

Engineering

# Summary: What do we want from a source code?

- In general, the optimal zero-error source coding problem is equivalent to find the optimal (shortest average length) uniquely decodable codes.
- Such a targeted code is called a compact code.
  - The uniquely decodable code with the smallest average code length for an information source S.
  - **How short can it be?**
    - **Shannon's first theorem**



# 本节学习目标

1. 理解效率与可靠性之间的折衷关系
2. 说出信源编码器与信源译码器各自的目标
3. 写出信源编码效率的评价指标
4. 说出信源编码优化问题
5. 说出什么是non-singular code
6. 说出什么是Uniquely decodable code
7. 说出什么是prefix code
8. 说出以上三种code的优缺点
9. 说出对信源编码的三个要求

**重难点：**

- 信源编码优化问题
- 认识几种编码类型

My Homepage

# Thank you!

**Yayu Gao**

**School of Electronic Information and Communications**

**Huazhong University of Science and Technology**

**Email: yayugao@hust.edu.cn**