Data

AP Computer Science Principles

Binary Numbers

Computers store everything as **bits** 位–each a `0` or `1`. A group of 8 bits is a **byte** 字节. Numbers are stored in **binary** 二进制 (base 2), where each place is a power of two $(1, 2, 4, 8, 16, \dots)$ instead of the powers of ten in **decimal** 十进制. For example, binary `1011` is $8 + 2 + 1 = 11$.

place value	128	64	32	16	8	4	2	1
bits	**1**	0	0	**1**	0	**1**	**1**	0

$150 = 128 + 16 + 4 + 2 \Rightarrow$ `10010110`

An 8-bit place-value chart: the 1s sit under the values that add to the number

Worked example. To convert binary `1101` to decimal, write the place values `8 4 2 1` under the bits `1 1 0 1` and add the ones that have a `1`: $8 + 4 + 0 + 1 = 13$. Going the other way, convert `19` to binary by subtracting the largest power of two that fits: $19 - 16 = 3$, then $3 - 2 = 1$, then $1 - 1 = 0$, so the bits sit at the `16`, `2`, and `1` places $\rightarrow$ `10011` (check: $16 + 2 + 1 = 19$).

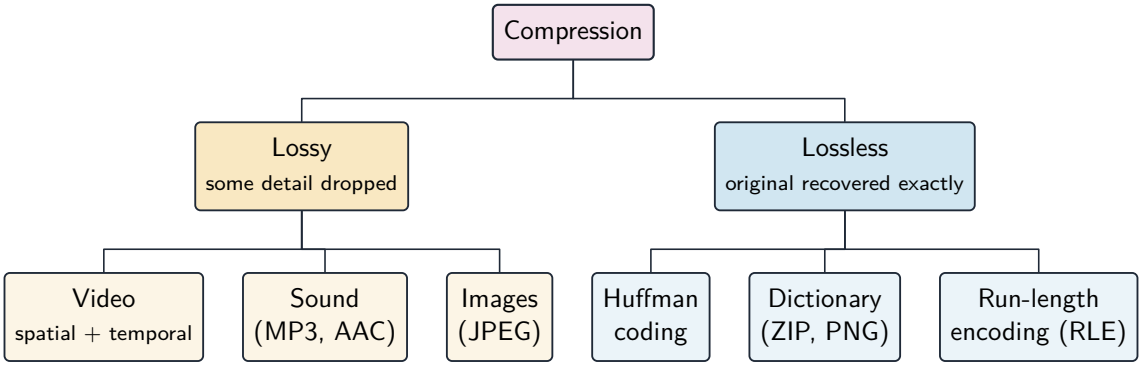
Because a computer has a **finite** number of bits, it can represent only a limited range of values. This causes two effects tested on the exam:

- **Overflow error** 溢出错误: a number too large for the available bits cannot be stored correctly.
- **Round-off (rounding) error** 舍入错误: numbers with **decimals** (real numbers) can only be **approximated**, because infinitely many real values must map onto finitely many bit patterns.

All data –text, images, sound –is ultimately encoded as binary. An image is a grid of **pixels** 像素, each stored as numbers for its colors; sound is stored as numbers sampled many times per second.

Data Compression

Compression 压缩 reduces the number of bits needed to store or send data. Two kinds:



Compression methods: lossless versus lossy, with common examples

- **Lossless compression** 无损压缩 lets you restore the **exact** original data (used for text and programs, where every bit matters).
- **Lossy compression** 有损压缩 throws away some data to shrink the size **further** (used for photos, music, video, where a small quality loss is acceptable).

Choosing between them trades **size against fidelity**: lossless keeps everything but saves less; lossy saves more but loses detail permanently. Prefer lossless when the data must be exact.

Extracting Information from Data

Data 数据 becomes useful when we extract **information** 信息 from it –patterns, trends, and answers to questions. Large data sets can reveal correlations a small one cannot, but data must be **cleaned** (fixing errors and inconsistencies) and often **transformed** or **filtered** first. A **correlation** 相关性 between two things does not prove that one **causes** the other –a key caution. **Metadata** 元数据 (data about data, like a photo's date and location) helps organize and search large collections.

Using Programs with Data

Programs process data at scales humans cannot. Common operations are **filtering** 过滤 (keeping only rows that meet a condition), **cleaning** (removing errors), and **visualizing** 可视化 (charts and graphs that make patterns visible). Combining data from multiple sources can reveal more, but raises **privacy** 隐私 concerns. Interactive tools and visualizations let people explore data and draw their own conclusions.

Exam skill: be able to explain how a program helps find information in a large data set, and why correlation shown in the data does not establish causation.

Exam tips

- Convert confidently between **binary, decimal**, and (where asked) hexadecimal — practise until it is quick.

- Remember a bit is one binary digit and a byte is 8 bits; n bits represent 2^n values.
- Explain that all data —numbers, text, images, sound —is stored as binary, and that finite bits cause **overflow** and **round-off**.
- Distinguish **lossless** from **lossy** compression and when each is appropriate.
- Show the analog-to-digital idea: **sampling** turns a continuous signal into discrete values.