A-LEVEL COMPUTER SCIENCE • EXERCISE SHEET

1.3 Compression

Name: _____ Class: _____ Date: _____ **Total: 35 marks**

KEY WORDS compression 压缩 • lossy 有损 • lossless 无损 • run-length encoding 行程编码 • bitmap 位图

Objective

Build the skills to answer exam questions on **compression** 压缩 — why it is used, the difference between lossless and lossy, and how text, images and sound are compressed.

You must be able to:

- explain the **need** for compression (storage and **bandwidth** 带宽)
- explain the difference between **lossless** 无损 and **lossy** 有损 compression, and justify which to use
- describe how a text file, bitmap image, vector graphic and sound file can be compressed

1 Worked examples

Study these first. Each one shows the method for a question type used later — follow the steps and you can do the Practice and Exam-style questions yourself.

■ Lossless or lossy?

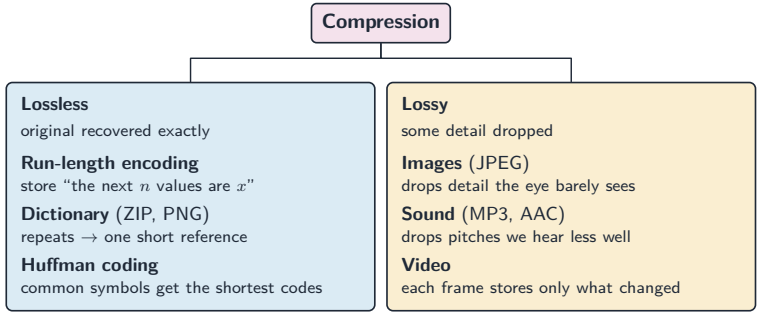
Compression makes a file smaller, so it needs less storage and less bandwidth to send. There are two kinds — the key question in the exam is *which one is suitable and why.*

	keeps every bit?	smaller file?	use for	examples
lossless	yes — exact original back	smaller	text, code, data that must stay exact	ZIP, PNG
lossy	no — drops fine detail	much smaller	photos, music, video	JPEG, MP3

So: choose **lossless** when the data must be recovered exactly; choose **lossy** when a much smaller file matters more than perfect detail (e.g. streaming).

■ Run-length encoding (a lossless method)

Run-length encoding 行程编码 (RLE) stores each *run* of identical values as a (count, value) pair, instead of repeating the value. It shrinks large flat areas a lot, but does nothing for noisy data (where runs are length 1).



Lossless versus lossy compression methods

■ Compressing text and sound

- **Text (lossless):** a **dictionary** method (ZIP) replaces repeated sequences of characters with a short reference; **Huffman coding** gives the most frequent characters the shortest codes.
- **Sound (lossy, e.g. MP3):** drop pitches the ear barely hears, and quiet sounds masked by louder ones — much smaller, slight quality loss.
- **Video (lossy): spatial** compression shrinks each frame (like JPEG); **temporal** compression stores only what *changed* from the previous frame.

2 Practice

Now apply the methods above.

2.1 State the difference between **lossless** and **lossy** compression. [2]

2.2 Give one situation that needs **lossless** compression and one suited to **lossy** compression, with a reason for each. [2]

2.3 Write the run-length encoding of this pixel row: **W W W W W B B B W** (W = white, B = black). [2]

2.4 State why run-length encoding gives almost no saving on a detailed photograph. [1]

3 Exam-style questions

3.1 A website streams live video. Explain why **lossy** compression is used rather than lossless. [3]

3.2 (a) Describe how run-length encoding compresses a black-and-white bitmap image. [2]

(b) State one kind of image for which run-length encoding gives a large saving. [1]

3.3 Explain how a text file can be compressed **without losing any data**. [2]

3.4 A music app streams songs to phones.

(a) Justify the use of lossy compression for the songs. [2]

(b) State one drawback of using lossy compression here. [1]

3.5 Describe the difference between **spatial** and **temporal** compression in a video file. [3]

3.6 A messaging app compresses the media its users send, and checks every message for transmission errors.

(a) A sound clip is compressed by **reducing the sampling rate**. **Explain** the effect on the file size and on the quality, and **state** which type of compression this is. [3]

(b) A photograph is compressed with a **lossless** method instead. **Explain** why the app might choose lossless for a scanned document but lossy for a holiday photograph. [3]

(c) The system uses **even parity**. **Complete** each byte by writing the missing parity bit in the space provided. [2]

seven data bits	parity bit
`1011010`	_____
`1110111`	_____

(d) The app also uses a **parity block check** with even parity. **Complete** the missing row and column. [3]

	b1	b2	b3	b4	parity
byte 1	1	0	1	1	_____
byte 2	0	1	1	0	_____
byte 3	1	1	0	1	_____
parity byte	_____	_____	_____	_____	

(e) **Explain** why a parity **block** check can locate a single flipped bit, while a parity **byte** check can only detect that something is wrong. [3]

Solutions

Each answer shows the steps, then the **result**.

2.1

- lossless: the original data is recovered **exactly**
- lossy: some detail is **permanently removed** to make the file smaller

2.2

- lossless — e.g. a program / document / spreadsheet, because every bit must be exact
- lossy — e.g. a streamed photo / song / video, because a much smaller file matters more than perfect detail

2.3

- group identical pixels into runs
- **5W 3B 1W**

2.4

- a photograph has detail that changes pixel to pixel, so runs are only 1 long — storing (count, value) is no shorter

3.1

- video is a huge amount of data and must be sent in **real time** over limited **bandwidth**
- lossless would not shrink it enough, so it would keep buffering/freezing
- lossy makes it small enough to stream with only minor quality loss

3.2

(a) each row is split into runs of the same colour; each run is stored as a (count, colour) pair instead of every pixel

- long runs of one colour take far less space

(b) e.g. a simple logo / icon / large areas of one colour

3.3

- use a lossless method: a dictionary replaces repeated character sequences with a short reference, **or** Huffman coding gives frequent characters shorter codes
- the exact text can still be rebuilt

3.4

(a) songs are large and streamed over mobile data, so a much smaller file downloads faster and uses less data

- the listener barely notices the dropped detail

(b) e.g. some audio quality is lost permanently

3.5

- spatial compression reduces the data **within a single frame** (like a JPEG image)
- temporal compression stores only the **differences between frames**
- most of one frame is the same as the previous one

3.6

(a) fewer samples are taken each second, so there is less data and the file is **smaller**

- the higher frequencies can no longer be represented, so the sound is duller and less like the original
- data has been thrown away permanently, so this is **lossy** compression

(b) a scanned document must stay **exactly** readable — losing detail can turn one character into another

- the image is mostly flat white, which lossless methods compress very well anyway
- a photograph has continuous tones where small changes are invisible to the eye, so lossy compression saves far more space at no noticeable cost

(c) `1011010` has four 1s, already even, so the parity bit is `0`

- `1110111` has six 1s, also even, so the parity bit is `0`

(d) even parity by row: byte 1 has three 1s → `1`; byte 2 has two 1s → `0`; byte 3 has three 1s → `1`

- the parity byte holds the column parities: `1,0,1` → `0`; `0,1,1` → `0`; `1,1,0` → `0`; `1,0,1` → `0`

(e) a single byte's parity bit only says that the number of 1s is wrong — any one of the bits could be the culprit

- in a block check the flipped bit breaks the parity of **both** its row and its column
- the row and column intersect at exactly one position, so the faulty bit can be found and corrected