

AQA Computer Science A-Level
**4.5.6 Representing images,
sound and other data**
Intermediate Notes



Specification:

4.5.6.1 Bit patterns, images, sound and other data:

Describe how bit patterns may represent other forms of data, including graphics and sound.

4.5.6.2 Analogue and digital:

Understand the difference between analogue and digital:

- data
- signals

4.5.6.3 Analogue/digital conversion:

Describe the principles of operation of:

- an analogue to digital converter (ADC)
- a digital to analogue converter (DAC)

Know that ADCs are used with analogue sensors.

Know that the most common use for a DAC is to convert a digital audio signal to an analogue signal.

4.5.6.4 Bitmapped graphics:

Explain how bitmaps are represented.

Explain the following for bitmaps:

- resolution
- colour depth
- size in pixels

Calculate storage requirements for bitmapped images and be aware that bitmap image files may also contain metadata.

Be familiar with typical metadata.

4.5.6.5 Vector graphics:

Explain how vector graphics represents images using lists of objects.

Give examples of typical properties of objects.

Use vector graphic primitives to create a simple vector graphic.



4.5.6.6 Vector graphics versus bitmapped graphics:

Compare the vector graphics approach with the bitmapped graphics approach and understand the advantages and disadvantages of each.

Be aware of appropriate uses of each approach.

4.5.6.7 Digital representation of sound:

Describe the digital representation of sound in terms of:

- sample resolution
- sampling rate and the Nyquist theorem

Calculate sound sample sizes in bytes.

4.5.6.8 Musical Instrument Digital Interface (MIDI):

Describe the purpose of MIDI and the use of event messages in MIDI.

Describe the advantages of using MIDI files for representing music.

4.5.6.9 Data compression:

Know why images and sound files are often compressed and that other files, such as text files, can also be compressed.

Understand the difference between lossless and lossy compression and explain the advantages and disadvantages of each.

Explain the principles behind the following techniques for lossless compression:

- run length encoding (RLE)
- dictionary-based methods

4.5.6.10 Encryption:

Understand what is meant by encryption and be able to define it.

Be familiar with Caesar cipher and be able to apply it to encrypt a plaintext message and decrypt a ciphertext. Be able to explain why it is easily cracked.

Be familiar with Vernam cipher or one-time pad and be able to apply it to encrypt a plaintext message and decrypt a ciphertext. Explain why Vernam cipher is considered as a cypher with perfect security.

Compare Vernam cipher with ciphers that depend on computational security.



Bit patterns, images, sound and other data

So far, we've only seen bit patterns used to represent **numbers**. However, computers also use bit patterns to represent **all other forms of data**, including **pictures** and **sound**.

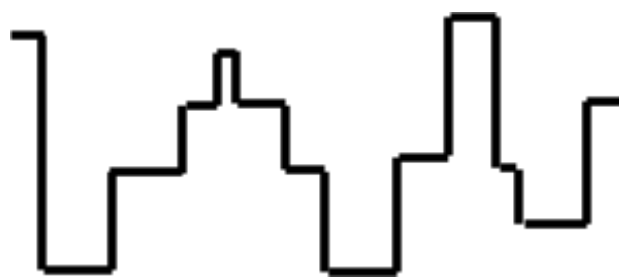
Analogue and digital

Analogue data has **no limits** to the values that it can take. In contrast, digital data can only take **particular values**.

Analogue and digital signals vary in a similar way. An analogue signal can take **any values** and can change **as much as required** whereas a digital signal must always take **one of a specified range of values** and can only change value **at specified intervals**.



Analogue signal



Digital signal

Analogue/digital conversion

Digital to analogue conversion

When converting from digital to analogue, a device called a **digital to analogue converter** (or **DAC** for short) is used. The device reads a bit pattern representing an analogue signal and outputs an analogue electrical **current**.

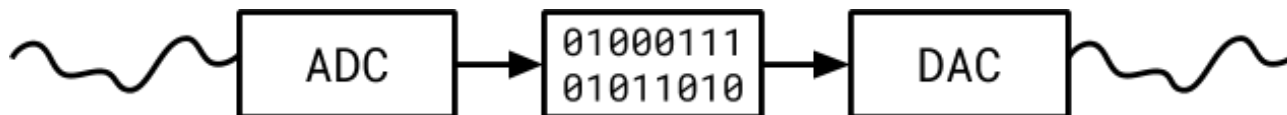
Analogue to digital conversion

When a computer needs to make use of analogue sensors, they use an **analogue to digital converter** (**ADC** for short) to convert the analogue signal to a digital bit pattern. The device works by taking a **reading** of an analogue signal at **regular intervals** and recording the value in a process called **sampling**.

Samples are taken at a specific **frequency**, which determines the **number of samples taken per second**. This is usually a **high number**.



Once the value of the analogue signal has been recorded, it can be stored **digitally** as a bit pattern.

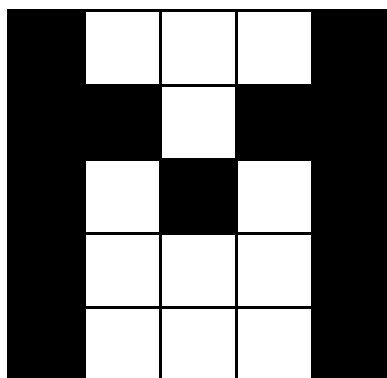


Bitmapped graphics

Computers represent **images** in two different ways, one of which is by using **bitmap graphics**. In bitmap graphics, an image is broken down into **pixels**, each of which has a **binary** value assigned to it.

The **resolution** of an image refers to the **number of pixels** in an image, for example, the image below could be said to have a resolution of **5 × 5** pixels.

The **value assigned** to a pixel **determines the colour** of the pixel. The example below shows the **binary representation** of a simple bitmap image in which a 1 represents a black pixel and a 0 represents a white pixel.

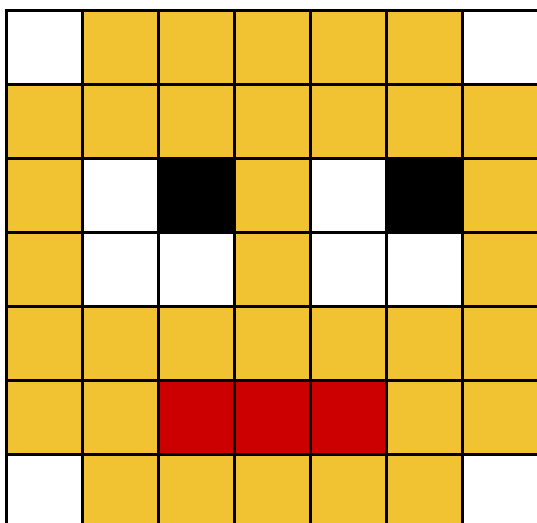


```

  1 0 0 0 1
  1 1 0 1 1
  1 0 1 0 1
  1 0 0 0 1
  1 0 0 0 1
  
```

The **number of bits** assigned to a pixel in an image is called its **colour depth**. In the example above, each pixel has been assigned **one bit**, allowing for 2 different colours to be represented.





```

00 11 11 11 11 11 00
11 11 11 11 11 11 11
11 00 01 11 00 01 11
11 00 00 11 00 00 11
11 11 11 11 11 11 11
11 11 10 10 10 11 11
00 11 11 11 11 11 00
  
```

In order to calculate the **storage required** to represent a bitmap image, multiply the **number of pixels** (width × height) by the **bit depth**.

The picture of the face has $7 \times 7 = 49$ pixels, each of which is assigned **two bits**, so it requires **98 bits** to be represented.

$$7 \times 7 \times 2 = 98 \text{ bits}$$

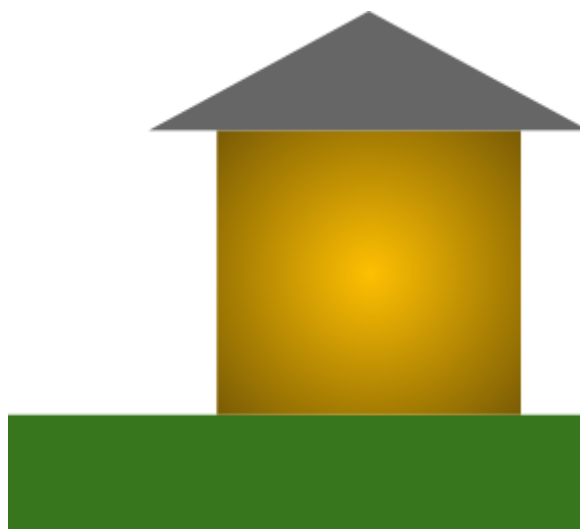
This method of calculating the storage requirements for bitmapped images produces a **minimum value**. This is because bitmap image files may also contain **metadata**, typical examples of which include the image's **width**, **height**, **date created** and **colour depth**.



Vector graphics

Vector graphics represent images using **objects** and **shapes** such as rectangles, circles and lines. The **properties** (such as fill colour, fill style and dimensions) of each geometric object or shape in the image are stored in a **list**.

shape	properties
rectangle	fill-colour: green fill-style: solid height: 2 width: 10 start-position: (0, 0)
square	fill-colour: yellow fill-style: vignette width: 6 start-position: (4, 2)
triangle	fill-colour: grey fill-style: solid width: 7 start-position: (3, 8)



Vector graphics versus bitmapped graphics

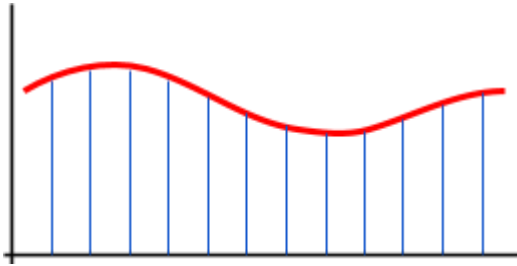
Because vector graphics use shapes rather than pixels, they can be enlarged **without losing quality**. Enlarging a bitmap image results in a **blurry** or even **pixelated** image whereas enlarging a vector graphic results in **no loss of clarity**.

Vector graphics frequently use **less storage space** than bitmapped graphics, as information is stored for each shape, rather than for every single pixel in an image.

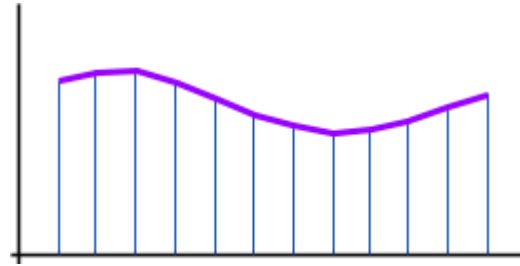


Digital representation of sound

Computers represent sound as a **sequence of samples**, each of which takes a **digital value**. The number of samples per second is called the **sampling rate**.



Analogue signal sampled



Samples used to recreate signal digitally

The **number of bits** allocated to each sample is referred to as the **sample resolution**. Higher sample resolutions result in **greater audio quality** but also **increased file size**.

The size of a sound sample can be calculated by **multiplying together** the **duration of the sample in seconds**, the **sampling rate in Hertz** and the **sample resolution**.

For example, a **45 second long** audio file sampled at **500 Hz** with a sample resolution of **16 bits** would require **45000 bytes** of storage.

$$45 \times 500 \times 16 = 360000 \text{ bits}$$
$$360000 \div 8 = 45000 \text{ bytes}$$

The Nyquist Theorem

The Nyquist theorem states that the sampling rate of a digital audio file must be **at least twice** the frequency of the sound. If the sampling rate is below this, the sound may not be accurately represented.



Musical Instrument Digital Interface (MIDI)

Musical instrument digital interface, or **MIDI**, is used with **electronic musical instruments** which can be **connected to computers**. MIDI stores sound as a series of **event messages**, each of which represents an **event** in a piece of music. These can be thought of as a **series of instructions** which could be used to recreate a piece of music.

Event messages could contain information such as:

- The **duration** of a note
- The **instrument** with which a note is played
- How **loud** a note is (its **volume**)

There are numerous advantages to using MIDI over a sampled recording of a piece of music. Using MIDI allows **easy manipulation** of music **without loss of quality**. The instruments on which notes sound can be changed, notes can be **changed** and the duration of notes can be altered.

Furthermore, MIDI files are often **smaller in size** than sampled audio files.

However, MIDI **can't be used for storing speech** and sometimes results in a **less realistic sound** than sampled recordings.

Data compression

Files are compressed in order to **reduce their size**. Smaller files can be **transferred faster** between storage devices.

Images are often compressed, but sound files and text files can also be compressed.

There are two categories of compression, **lossy** and **lossless**.

Lossy compression

When using lossy compression, **some information is lost** in the process of reducing the file's size. This could be **reducing the resolution** of an image or **lowering the sample resolution** of an audio file.



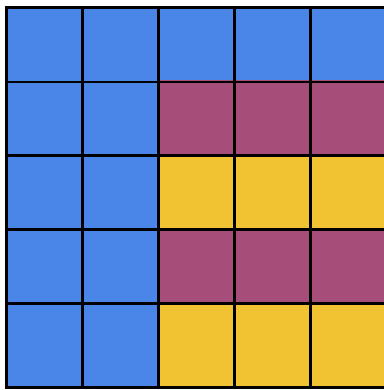
Lossless compression

In contrast to lossy compression, there is **no loss of information** when using lossless compression. The size of a file can be reduced **without decreasing its quality**.

Two methods of lossless compression are **run length encoding** and **dictionary-based methods**.

Run length encoding (RLE)

Run length encoding (**RLE** for short) **reduces the size** of a file by removing **repeated information** and replacing it with **one occurrence** of the repeated information followed by the **number of times** it is to be repeated.



BLUE₅
 BLUE₂ PURPLE₃
 BLUE₂ YELLOW₃
 BLUE₂ PURPLE₃
 BLUE₂ YELLOW₃

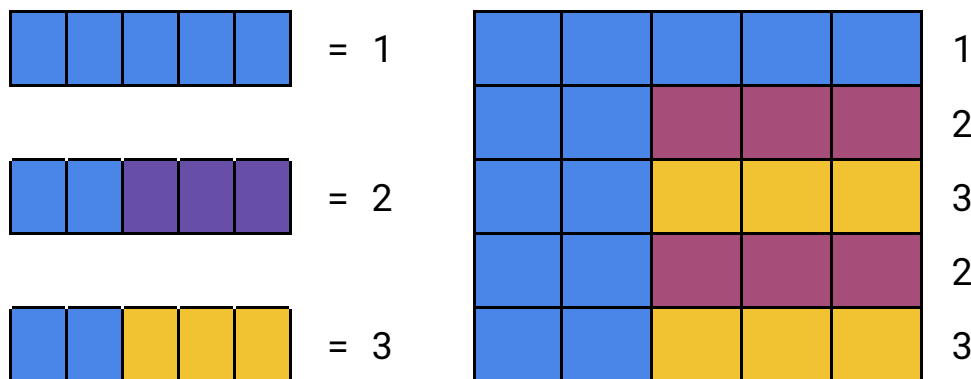
Using RLE to **replace repeated pixels** with one pixel colour and a **number or repetitions** reduces the storage space required to represent the image.



Dictionary-based methods

When a file is compressed with a dictionary-based method, a **dictionary** containing **repeated data** is **appended** to the file.

For the picture above, the dictionary on the left could be used.



Using the dictionary, the file could be represented using just the data 12323, as shown on the right.

This method results in a **significant reduction** in size, but don't forget that the dictionary used to compress the data **has to be present in the file** in order for the image to be reproduced. This will **increase the size** of the file.

Lossy Compression	Lossless Compression
Some information is lost in the compression process	No loss of information
Quality of file is reduced	No loss of quality

Encryption

Encryption is the process of **scrambling data** so that it **cannot be understood if intercepted** in order to **keep it secure during transmission**. Unencrypted information is referred to as **plaintext** and encrypted information is called **ciphertext**. A **cipher** is a type of encryption method.

In order to decrypt ciphertext, you must know the **encryption method** used and the **key** used to encrypt the information.




Caesar ciphers

Caesar ciphers encrypt information by **replacing characters**. One character is **always** replaced by the **same character**. There are two types of Caesar cipher that you need to be aware of. **Shift ciphers** and **substitution ciphers**.

Shift ciphers

When encrypting using a shift cipher, all of the letters in the alphabet are **shifted by the same amount**. The amount by which characters are shifted forms the key.

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

The example above uses a **shift** of three characters, so the key is **three**. Using the key three, the plaintext “BAT” could be encrypted as the ciphertext “YXQ”.

Substitution ciphers

Substitution ciphers are a type of Caesar cipher in which letters are **randomly replaced**.

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

Using the cipher in the example, the plaintext “DOG” would be encrypted as “DUB”.

Caesar ciphers can be **easily cracked**. For example, the most frequently occurring letter in an encrypted message is likely to be an E.



Vernam ciphers

The Vernam cipher is an example of a one-time pad cipher. This means that **each key should only ever be used once**. Additionally, the Vernam cipher requires the key to be **random** and **at least as long as the plaintext** that is to be encrypted.

The Vernam cipher works by:

1. **Aligning** the characters of the **plaintext** and the **key**
2. Converting each character to **binary** (using an **information coding system**)
3. Applying a logical **XOR** operation to the two bit patterns
4. Converting the result back to a character

Example. encrypting:

Plaintext	H	I
Plaintext binary	01001000	01001001
Key	u	r
Key binary	01110101	01110010
Plaintext binary XOR key binary	00111101	00111011
Ciphertext	=	;

In the example above, each of the characters in the plaintext and the key are converted to binary, then XORed before being converted back to characters.

As the example shows, the plaintext HI is encrypted by a Vernam cipher with the key ur as the ciphertext = ;.

Synoptic Link

Computers use **information coding systems** to represent characters.

Information coding systems are covered under **Fundamentals of Data Representation**.

XOR

Logical operation which outputs true when exactly one of its inputs is true.



Example, decrypting:

When decrypting using a Vernam cipher, the key that was used to encrypt the plaintext is used again.

Ciphertext	=	;
Ciphertext binary	00111101	00111011
Key	u	r
Key binary	01110101	01110010
Ciphertext binary XOR key binary	01001000	01001001
Plaintext	H	I

The Vernam cipher is the only cipher mathematically proven to be completely secure.

Computational security

All ciphers other than the Vernam cipher are, in theory, crackable, but not within a reasonable timeframe given current computing power. Ciphers that use this form of security are said to rely on [computational security](#).

