Unit 3: Fundamentals of Data Representation

Units for Measuring of Data

A bit is the most basic unit of data measurement. B represents a byte, and b represents a bit	A byte is 8 bits. A bit can be 0 or 1.

Converting Between Units of Measurement

Size	Unit	Examples of Converting
1,000 bytes	1 kilobyte - KB	80 bits = 80/8 = 10 bytes
1,000 kilobytes - 1,000 KB	1 megabyte - MB	8,000 Bytes = 8,0000 / 1,000 = 8 KB
1,000 megabytes - 1,000 MB	1 gigabyte - GB	2,400 KB = 2,400 / 1,000 = 2.4 MB
1,000 gigabytes - 1,000 GB	1 terabyte - TB	3,500 MB = 3,400 / 1,000 = 3.4 GB

Decimal (Base 10)	Binary (Base 2)	Hexadecimal (Base 16)
This is the number system we are most used to.	Used by computers to represent all data.	Used to represent larger numbers.
10 characters, 0-9	2 characters, 0 and 1	16 characters, 0-9 and A - F
1, 2, 20, 40, 999	0, 000, 01010, 00101010	1, 6, A, AF, 1A, AB1E8

Creating a Hoffman Tree

1. Count how many times each character appears in the string.
   a. Write the list in order with the most common letter.
   b. Characters with the same value can be placed in any order.
2. Add together the number of the two least common characters in a new block and label it with the total.
3. Move the new block into the list based on its number.
4. Repeat steps 2 and 3 until only one block remains.
5. Label the "branches" in the tree, working from the top down.
   a. Label the branches going in one direction 1 and the other 0

Encoding the binary stream

The tree can be compressed into a single string.

1. Start at the top node.
2. Encode each letter using the path of 1s and 0s.
3. Join the stream together

Calculating the Bits Needed to Store Compressed Strings

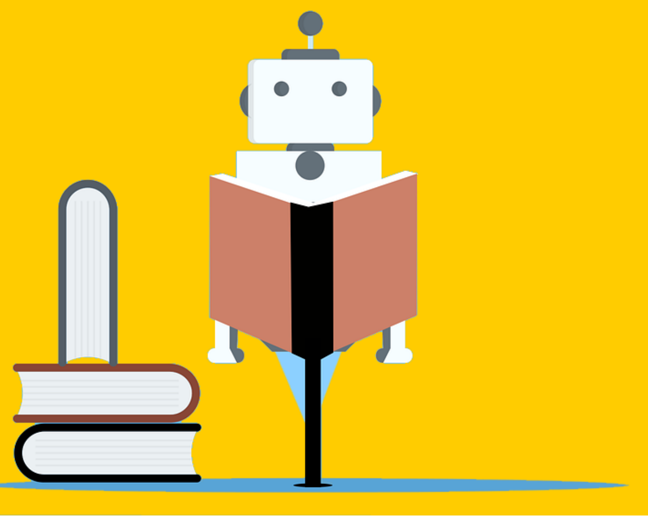
1. Take the length of the bit pattern for each character.
2. Multiply it by the times the pattern is used.
3. Add this together for each character.

Calculating the Bits Needed to Store Uncompressed ASCII

1. Number of characters x 7

Hoffman Coding

- A way to reduce the number of bits needed to send or store a message
- A lossless compression method.
- Looks at how often a data item, for example a character in a string, occurs.
- Tries to use fewer bits to store common data which frequently occurs.
- Comprised of:
  - A Huffman tree, giving each character a unique code
  - A binary stream of the character sequence



Maths in Binary

Addition

Always follow these for rules:

- 0 + 0 = 0
- 1 + 0 = 1
- 1 + 1 = 10 (binary for decimal 2)
- 1 + 1 + 1 = 11 (binary for decimal 3)

Multiplication

To multiply move the digits to the left and fill the gaps after the shift with 0:

- To multiply by two, all digits shift one place to the left
- To multiply by four, all digits shift two places to the left, etc., etc.

Division

To divide move the digits to the right and fill the gaps after the shift with 0:

- To divide by two, all digits shift one place to the right
- To divide by four, all digits shift two places to the right, etc., etc.

Images in Binary

- Images are broken down using a grid.
- Each square in the grid is known as a pixel.
- Pixel is short for Picture Element.

Colour Depth

- The more colours an image uses, the more bits per pixel are used. This is called the Colour Depth.
- The greater the colour depth, the larger the image file will be.

Image Size

- The size of an image is measured in the number of pixels used.
- This is written as pixels wide x pixels high.
- For example, 5x5, 20x40, or 1024x768.

File Size

- height in pixels X width in pixels X colour depth per pixel
- Dividing by 8 gives the value in bytes

Run Length Encoding (RLE)

- A lossless compression method.
- Relies on the original data having repeating digits.
- Finds patterns in the original data to save space.
- Runs of data are sequenced within the original data which have the same value.
- These runs are stored in frequency/data pairs.

Character Encoding

- Because computers work in binary, all characters must be stored in binary.
- The characters which a computer can use are called a Character Set.
- A character code is the number assigned to a character within a character set.
- When storing and transmitting characters, the computer will use the character code.
- Character codes are grouped together, and run in order within that group.
- For example, in ASCII A is 65, B is 66 and so on.
- This makes it easy to calculate different character codes.

ASCII (American Standard Code for Information Interchange)

- Ascii uses 7 bits, giving a character set of 128 characters.
- These are represented the ASCII Table.
- Each character has its own assigned numbers, some examples are below.
- Included in the table are:
  - 32 control codes - mainly to do with printing
  - 32 punctuation codes, symbols, and space
  - 26 upper case letters
  - 26 lower case letters
  - Numbers 0-9

Unicode

- The ASCII Character set is too small to hold every character and symbol in English and other languages such as Chinese and Arabic.
- Unicode uses 16 bits, giving a character set of 65,536 characters.
- Unicode uses the same character codes as ASCII up to 127.
- Unicode also includes additional symbols and characters such as emojis.

Sound in Binary

- Sound is analogue, so must be converted to binary for computers to understand it.
- The amplitude (volume) of the sound is measured at a point in time, this is called sampling.
- Many samples are put together to represent the sound.

Amplitude - The height of the sound wave at the time it was sampled. The higher the amplitude the louder the sound.

Sample Rate

- The sample rate is the number of samples taken in one second.
- It is measured in Hertz (Hz).
- 1 Hertz = 1 sample per second.
- Small sampling interval = high sample rate = better quality sound file = larger file.

Sample Resolution

- The sample resolution is the number of bits used to store each sample.
- The higher the sample rate the more accurate the representation is, but the more space needed.

Calculating Sound File Sizes

- The elements above can be used to calculate the file size of a sound file.
- Larger files will give a more accurate representation of the original sound.

Compression

- Ways to reduce the amount of storage space required for data
- Large files are difficult and expensive to store and transmit.
- Compression techniques reduce file sizes.
- There are many different ways to compress data, each with their own pros and cons.
- It is important to balance the reduction in file size with any reduction in quality.
- Different compression techniques will work best in different scenarios.
- Lossy compression means that data is lost and can not be recovered once the file is compressed.
- Lossless means that no data is lost and the original contents of the file can be completely recovered.

Converting Between Bases

Binary to Denary

1) Draw your conversion table.

128	64	32	16	8	4	2	1
1	1	0	0	1	1	0	0

2) Write the binary number in the conversion table.
3) Add together all numbers with a 1 beneath them

$$\begin{array}{r} 128 \\ 64 \\ +\quad 8 \\ 4 \\ \hline 204 \\ \cancel{1}\cancel{1} \end{array}$$

11001100 in binary is 204 in denary

Denary to Binary

1) Draw your conversion table.
2) Is the number higher than the first column in the table?
   a) If so, put a 1 in that column and work out the difference.
   b) If not, put a 0 in that column.
3) Repeat the step above with the difference.
4) Keep going until the difference is 0, put a 0 in any empty columns.
5) Read the number from the bottom row of the table.

128	64	32	16	8	4	2	1
1	1	0	0	1	0	0	0

200 – 128 = **72**

72 – 64 = **8**

8 – 8 = **0**

200 in denary is 11001000 in binary

Denary to Hexadecimal

1) Divide the denary number by 16 and write down both the answer and the remainder. 62 ÷ 16 = 3 R 14
2) Divide the answer by 16 again. Write down both the answer and the remainder. 3 ÷ 16 = 0 R 3
3) Keep going until you reach an answer of 0.
4) Read the remainders from bottom to top. 3 14
5) Convert each remainder to hex. 3E

62 in binary is 3E in hexadecimal

0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F

Binary to Hexadecimal
3

1) Draw two separate conversion tables.
2) Write the binary number across both tables.
3) For each table, add up the numbers which have a 1 beneath them.
4) Convert each number to hexadecimal.

8	4	2	1
0	1	1	0

8	4	2	1
1	1	0	1

4 + 2 = 6   8 + 4 +1 = 13

6   D

01101101 in binary is 6D in hexadecimal

0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F