Character Codes and Compression

Teacher Notes to support Tenderfoot Unit 4: Bits And Bytes – The digital advantage

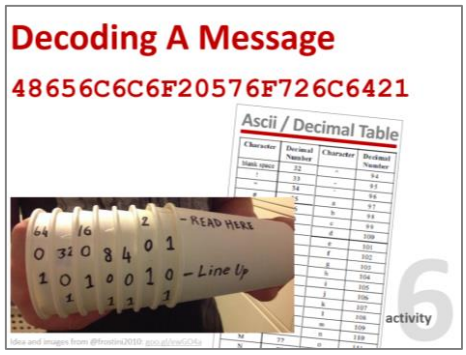
A brief introduction to ascii and Unicode character codes, and an online hex editor, via a practical exercise to convert a message in hexadecimal. The main practical exercise is in three parts and involves compressing text using the principles of Lempel–Ziv compression.

Preparation required:
Binary decoding wheels (or 8 paper cups, or asci / decimal table) per student.
Compression exercise sheets (3) for each student.

Decoding A Message

The session starts with a rather silly story as an introduction, using binary codes mapped to letters to reveal the message 'help im trapped'. Students are then challenged to come up with their own character coding scheme in small groups. The prompts can help steer the discussion if they are struggling:

- ***Are there existing codes?*** Morse Code is an obvious example of a character code that could translate directly to 1/0 representation. However, students should note that also requires spaces between letters and words to be translated. Without the 'pause' there is a problem with codes of different lengths. A fixed number of bits per character would solve this. A second point students might raise is the limited character set. Upper case letters and digits only.
- ***What symbols need encoding?*** This should generate a variety of suggestions. Build an ad-hoc list on the board. Ask if there is a more systematic way to think about this. A study of a computer keyboard might provide a good approach to ensure all punctuation symbols are accounted for.
- ***How many bits should be assigned to each character?*** Depending on the investigation so far, students should have around 100 symbols to encode. The largest 7 bit binary number is 127, so a Standard English alphabet character set should be able to be encoded in 7 bits. But what about supporting other alphabets? Ascii originated as the standard character set for those using the English alphabet using 7 bits, but as other character sets needed to be encoded, Unicode (using 16 bits) developed. Use the image for children to spot characters not in the English character set.



Through discussion children should become aware of the need for a common standard. The simple decoding activity introduces standard ascii, with each character taking 1 byte. For ease of translation, and as an extra challenge, each byte on the slide is expressed in hexadecimal, so each two hex digits are the equivalent of a binary byte. There are various ways to decode. The material from DJ Dates website: goo.gl/2r3QYe allows children to make their own decoding wheel. An alternative might be to look the values up using a decimal to ascii table, having decoded the values to decimal using the simple technique shown. Thanks to @frostini2010: goo.gl/ewGO4a for the idea. An ascii table with binary values could be used to look values up directly. Both tables are included to print. The easiest alternative for less able pupils might be to use an online hex to ascii convertor, such as that at goo.gl/h4ZlYo. The message is Hello World!

Online Ascii / Unicode tables can be quite confusing (hence the limited versions supplied). As noted earlier, Unicode (using 16 bits) allows for a far greater range of characters. A quick activity to highlight this on Windows machines is to hold the left Alt key and tap Unicode numbers on the number pad before releasing the Alt key (ensure that Num Lock is on). This sends the relevant character to the screen—try it in Word. For example Alt + 065 gives A and Alt+14789 gives 扅. The link gives examples of other widely used Alt – codes: goo.gl/viOId4.

Analysing A Message

Using Hello World as the example we can consider how much space a message takes. If children think it is 11 (or 10) bytes check they include a byte for both the space and exclamation mark. Once translated, type the message into Notepad and save the file. Look at the file properties to confirm the size of the message as 12 bytes.

Hex Editors, such as the one at en.webhex.net/ allow you to see the binary (expressed in hexadecimal) contents of a file. Spend a little time explaining what the Hex Editor is displaying. Point out the contents of the file are displayed on the right and the Hex value for each character displayed in sequence. Tools such as these allow us to look 'under the hood', rather like the Pixel Spreadsheet used earlier. Encourage students to upload short text files to confirm that they are all simply strings of binary / hex numbers. They can cross reference them to their asci tables.

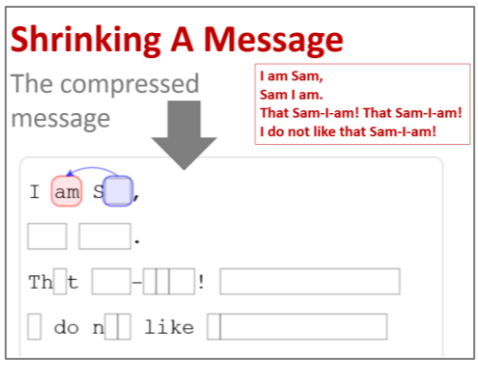
We can use the test file provided with the hex editor for a bit more analysis. The statistics option will show the frequency of each character. Clearly, some characters are more commonly used than others. The Table View is clearest. Identifying common occurrences of characters is the basis of Huffman compression – allocating shorter bit patterns to commonly used characters.

Shrinking A Message

Text files can be easily compressed. There are two approaches; Huffman compression uses shorter bit patterns for common characters whilst Lempel-Ziv (LZ) compression looks for repeating character sets. We look at the latter approach. Lempel Ziv compression is the basic technique behind many common compression methods, such as zip, named after the originators Abraham Lempel and Jacob Ziv.

The presentation displays the start of a child's rhyme (I am Sam). It is 75 bytes (not including new line characters). Clearly parts are repetition – so how might we shrink it? The slides that follow give an explanation, but allow a discussion first, and encourage children to be precise in their explanations.

The graphic is a visual representation of the compressed message. Provided children complete the links to the boxes in order, they should be able to decompress the message. The animated sequence gets them going, but once the first few links have been established this could be done as a class exercise on the board. Once the children get the idea, allow them to complete the sequence on the board, circling the source for each box contents. The method being used is the basis of Lempel-Ziv compression.



Once students have got the hang of it, issue the worksheet in the resources. This is the first of 3 exercises. The uncompressed rhyme is displayed on the following slide so students can check it. The second example shows a different representation. An animation demonstrates how numbers can be used to indicate the position and quantity of text to insert. This is how the compressed file would be stored, using pointers in place of repetitive text. Children usually find this more challenging.

However, for a computer following an algorithm, this would not present any difficulty. It can be tried individually, but might be a useful group activity. The final exercise provides a chance to compress, rather than decompress text. It follows a basic algorithm but is a challenging activity. The algorithm is explained in two stages. The exercise can be issued as a competitive challenge. How small can the example in the worksheet be compressed?