Web Science Compression

Objectives

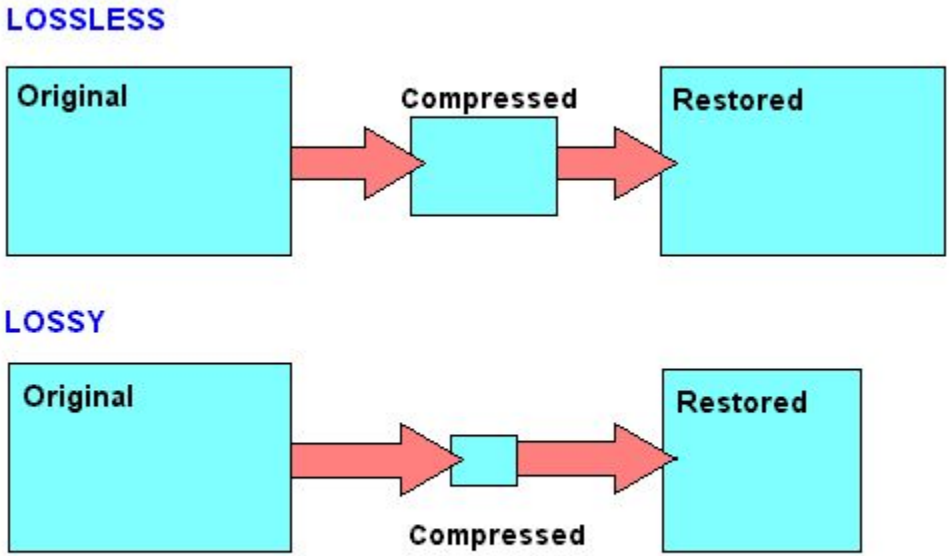
Understand the purpose of compression (advanatges) .

Understand how the 2 Compression techniques work (Lossless and Lossy.

Be able to select the most appropriate method under given circumstances

Understand any possible downsides to using compression

lossless vs Lossy Compression



Why Compress Data?

What type of files take up most disk space?

What problems would we encounter when we try and transfer large files?

What are the solutions?

How Compress - By using Redundancy or Patterns

One technique to use our storage more optimally is to compress the files. By taking advantage of redundancy or patterns, we may be able to "abbreviate" the contents in such a way to take up less space yet

maintain the ability to reconstruct a full version of the original when needed.

2 Advantages :

Store more stuff on disk

Shorten the time needed to copy/send a file over a network.

If the document / message is mega important

If it is urgent that the receiver get the message / document super quick should you compress?

Example only of a possible downside.

Compression Formats

MP3

JPEG

These tend to take advantage of known features of that type of data (such as the propensity for pixels in an image to be same or similar colors to their neighbors) to compress it.

Run-length encoding (RLE) is a form of lossless compression

The standard ASCII character encoding uses the same amount of space (one byte or eight bits, where each bit is either a 0 or a 1) to store each character. Common characters don't get any special treatment; they require the same 8 bits that are used for much rarer characters such as "#" or "&"

A file of 1000 characters encoded using the ASCII scheme will take 1000 bytes (8000 bits); no more, no less

In practice, it is not the case that all 256 characters in the ASCII set occur with equal frequency

Run-length encoding (RLE) is a form of lossless compression

RLE is a simple method of compressing data by specifying the number of times a character or pixel colour repeats followed by the value of the character or pixel. The aim is to reduce the number of bits used to represent a set of data. Reducing the number of bits used means that it will take up less storage space and be quicker to transfer.

For a text file it is critical that no data is lost and therefore a lossless

compression method must be used.

Run-length encoding (RLE) is a form of lossless compression

The process involves going through the text and counting the number of consecutive occurrences of each character (called “a run”). The number of occurrences of the character and the character code are then stored in pairs. The first part of the pair is the count and the second part is the character.

Run-length encoding (RLE) is a form of lossless compression

Example: aaaabbbbbbcdddddd

There are 16 characters in the example so 16 bytes (assuming ASCII is being used) are needed to store these characters in an uncompressed format:

Example as ASCII: 97 97 97 97 98 98 98 98 98 98 99 100 100 100 100 100

Run-length encoding (RLE) is a form of lossless compression

RLE can be used to store that same data using fewer bytes.

Example: aaaabbbbbbcdddddd

This could be written as the following sequence of occurrence-character pairs:

Run-length encoding for the above Example: (4,a) (6,b) (1,c) (5,d)

If this text is then compressed using RLE we would end up with: 04 97 06 98

01 99 05 100

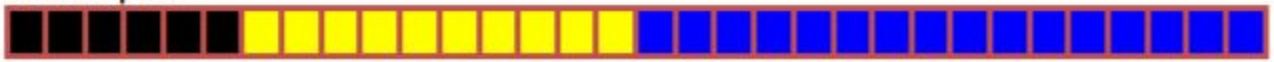
compressed version only requires 8 bytes - a reduction from the original 16 bytes

RLE for bitmapped image data

RLE for bitmapped image compression works in a similar way to text compression. The effectiveness depends on the image being compressed; images with long runs of pixels of the same colour will provide a higher compression ratio than images that have frequently changing colours.

Byte level RLE - (for 256-colour images)

Example:



6 black, 10 yellow, 16 blue

If the colour table for the example above specified black as 16, yellow as 126 and blue as 20 then the corresponding bytes would be:

00000110 00010000 00001010 01111110 00010000 00010100

Variable-length encoding

Other methods of encoding exist , variable length works on the premise why use 8 bits to encode the letter “z” which is seldom used. Instead use fewer bits for more common characters. For further information look up “Huffman coding”.

Huffman Coding

The encoded phrase requires a total of 34 bits, shaving a few more bits from the fixed-length version. What is tricky about a variable-length code is that we no longer can easily determine the boundaries between characters in the encoded stream of bits when decoding. I boxed every other character's bit pattern above to help you visualize the encoding, but without this aid, you might wonder how you will know whether the first character is encoded with the two bits 01 or the three bits 010 or perhaps just the first bit 0?

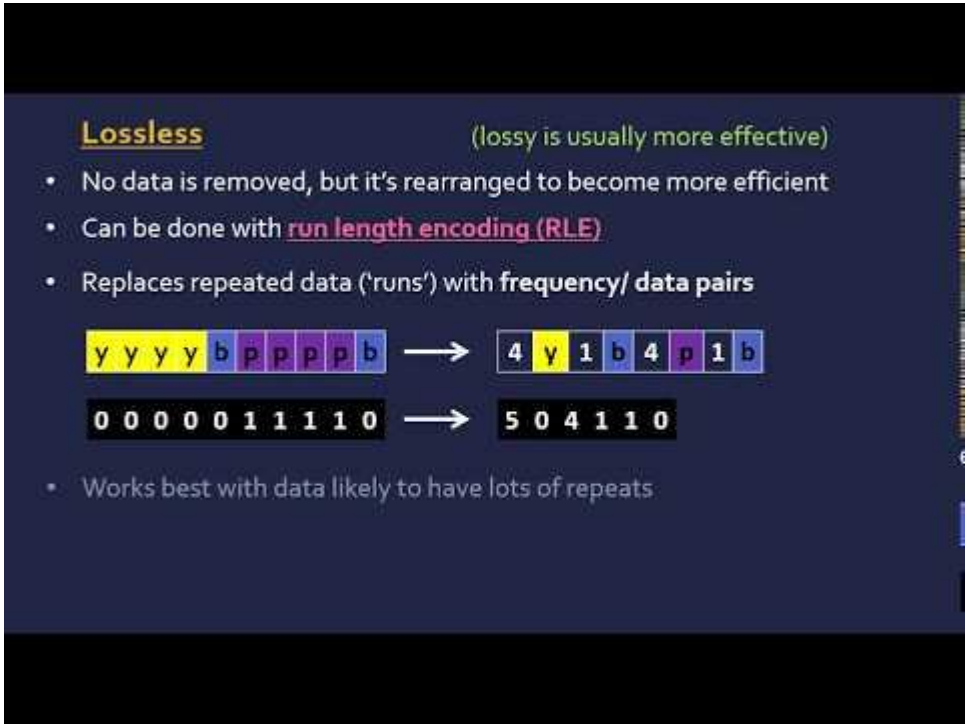
Past paper Questions / Similar Questions

Identify two different types of file which, if compressed, could make use of a lossy compression algorithm.

Evaluate lossy compression and lossless compression when used to download files. [4]

Discuss two factors that would affect the decision to use either lossless or lossy compression when transferring across the Internet.

Video Time


Lossless
(lossy is usually more effective)
• No data is removed, but it's rearranged to become more efficient
• Can be done with run length encoding (RLE)
• Replaces repeated data ('runs') with frequency/ data pairs
y y y y b p p p p b
4 y 1 b 4 p 1 b
0 0 0 0 0 1 1 1 1 0
5 0 4 1 1 0
• Works best with data likely to have lots of repeats

Past paper Questions / Similar Questions

When a user requests a file from a particular website, the website uses lossy compression to send the file to the user over the internet.(a) Discuss how this use of lossy compression might affect the user’s experience.

Past paper Questions / Similar Questions

A large file is being downloaded from a website using TCP/IP protocols.

(e) Evaluate the choice of compression software in the transfer of the file. [5]