Compression

Packing data into a smaller space

Alwin Tareen

Compression Algorithms

- Digital data representations often involve tradeoffs between quality and file size.
- Many storage formats use compression techniques that store patterns of bits, rather than an exact representation of the bits.
- Data compression is a set of steps for packing data into a smaller space, while allowing for the original data to be recreated.

Compression Algorithms

Compression is a two-way process

- A compression algorithm can be used to make a data file smaller.
- However, the compression algorithm can be run in the other direction, to decompress the file into its original form(this only applies to lossless compression).

Compression Algorithms

The Compression Ratio

- A good measure for comparing the effectiveness of compression algorithms is to compute the following compression ratio:

$$\frac{(\text{original file size} - \text{compressed file size})}{\text{original file size}} \times 100$$

- For example, let the original file size = 240 bytes, and the compressed file size = 177 bytes.

$$\begin{aligned}\text{compression ratio} &= \frac{(240 - 177)}{240} \times 100 \\ &= 26.25\%\end{aligned}$$

Compression Algorithms

- Note that a better compression ratio does not guarantee that one compression algorithm is more effective than another.
- Some compression algorithms are tuned to a specific type of data, for example: text, music, images, video, etc.

Lossless Compression(Zip, GIF, PNG)

- Lossless compression means that compression has occurred with zero loss of information.
- Lossless compression packs data in such a way that the compressed package can be decompressed, and the data can be pulled out exactly the same as it went in.
- This is very important for computer programs and archives, since even a small alteration in a computer program's file will make it completely unusable.

Lossy Compression(JPEG, MP3, MPEG)

- Lossy compression indicates that there has been some data lost through the compression process.
- In other words, lossy compression throws out some of the data, so that there's less information to store.
- Lossy compressions work well with media files, such as images or music, because the human eye and ear has limits on the level of detail that it can detect.
- Lossy compression can never be undone, because the original information can never be reconstructed, once it has been lost.
- Therefore, you can't go from a lossy-compressed image back to the original image.

Run-length Encoding(Lossless)

- This is where you consider a piece of text, and indicate repeated instances of a character.
- This type of compression works by reducing how much wasted space exists in a piece of text.
- For example, if the text sample is: `AAAAABBBB`
- It can be compressed into the following: `5A4B`
- This indicates that there are two runs of text: a run of five `A`'s and another of four `B`'s.

Run-length Encoding(Lossless)

- The problem with run-length encoding is that it doesn't work with certain patterns of data.
- Consider the following text sample: `ABBAABAAB`
- This would be compressed as: `1A2B2A1B2A1B`
- Note that the compressed version is longer than the original sample of text.

Huffman Encoding(lossless)

- This is a type of frequency compression, that overcomes the problems with run-length encoding.
- Each distinct value in a piece of data is given a code.
- Values that occur often are assigned shorter codes.
- Values that occur infrequently are assigned longer codes.

Huffman's Algorithm

- Build a subtree using the two symbols with the lowest probability.
- At each step, choose the two symbols or subtrees with the lowest probability, and combine them to form a new subtree.
- Continue in this manner until all the symbols in the set have been exhausted.

Compression: End of Notes