ECM1416 DATA STRUCTURES AND ALGORITHMS

PEDRO PEREIRA 700044334

# 1. Index

[1. Index 2](#_Toc68774509)

[2. Literature review on compression algorithms 3](#_Toc68774510)

[3. List of all Data Structures and algorithms 7](#_Toc68774511)

[3.1 Data Structures 7](#_Toc68774512)

[4. Performance analysis 8](#_Toc68774513)

[5. Weekly progress 9](#_Toc68774514)

[6. References 9](#_Toc68774515)

# 2. Literature review on compression algorithms

Data compression is the procedure which converts a data set into a code to reduce the size of the data set so it is easier to store and transmit, as can be seen in the figure below.

Data set before compressionon

Compression method

Data set after compression

Figure 1 – Data compression procedure

This procedure can be done through many different algorithm techniques such as the Huffman, Shannon – Fano, Lempel Ziv Welch, Run Length Encoding among others and these methods can be of two different types: lossless compression and lossy compression.

In lossless compression techniques there is no loss of data, meaning that after the information has been losslessly compressed, the original data can be recovered exactly from the compressed data.

The use for this compression technique can be clearly seen in the compression of text as small differences between the original data and the reconstructed data can alter the meaning of the text or even make it void of meaning, for example: Consider the original sentence: “Could you please hand me the bread?” is altered to “Could you please *hang* me the bread?”, despite only one letter being altered the phrase has lost grammatical cohesion and no longer makes sense.

Lossless compression is also used in data of any that may need to be processed at a later time for more information, such as, a radiological image that may not present any significant differences after being reconstructed from a lossy compression, but if the image was to be later enhanced it could show bigger differences that could compromise the image and make it unusable or even mislead the radiologist which is far from desired since this mistake may cost a human life so lossless compression is used.

In lossy compression techniques there is some loss of data, meaning that after the original information is lossyly compressed it cannot be recovered or reconstructed exactly. However, for allowing this loss of data that will not permit the reconstructed data to be exact, lossy compression has, in general, much higher compression ratios than its counterpart, lossless compression.

This compression technique can be of use in audio. An original audio file that contains inaudible sounds, these are sounds that are outside human hearing capacity, is able to be compressed in a way where these sounds are discarded. This makes it so the audio data in the file after it has been compressed cannot be reconstructed to its original state, but it is not important since the data lost was not relevant, making lossy compression extremely useful.

Lossy compression also sees general use in video where the reconstruction of a video sequence does not need to be exactly like the original video since this normally does not lead to any notable differences.

A compression algorithm’s performance can be measured in many different ways, such as: the amount of compression, the compression quality, the relative complexity of the algorithm, the memory that the algorithm requires to implement it and the time it takes an algorithm to perform the compression. More often than not, the most important evaluators of performance are the first two, so those are the ones I will explain more in depth here.

As it was mentioned before, in lossy compression, the data after compression will not be able to be reconstructed exactly to the original data, sometimes 70%, 80% can be reconstructed depending on the compression algorithm. This is what was earlier referred to as compression quality, if the quality is high the reconstruction and the original data have very small differences, if it is low the reconstruction and the original will have bigger differences.

After this, it might look like the compression algorithm that provides the highest quality would always be preferred, but this is not the case since here the compression quality links with the amount of compression. By amount of compression, what is being discussed is the number of bits required to represent the data before compression to the number of bits required to represent the data after compression which is normally referred to as *compression ratio.* A compression algorithm that reduces a text file from 100 000 bits to 10 00 bits has a compression ratio of 10:1 or, in percentage, 90%.

For example: when compressing an image there is some compression quality that can be lost without it becoming too noticeable to the human eye, so normally a compression quality of 100% is not preferred and instead a smaller compression quality that brings with it a better compression ratio is what is utilized.

Two examples of lossless compression algorithms are: Huffman coding and Shannon-Fano coding

Huffman coding is a compression algorithm, it was developed by David A. Huffman and published in a paper in 1952. It is a data compression technique that uses the number of occurrences of a data item, i.e. how often a character exists in a text file, to reduce the number of bits it requires to compress it by assigning smaller bits to the data items with the highest frequency.

This is a data compression method that belongs to the class of lossless compression given that the original data can be retrieved from the compressed data in full.

In detail, Huffman coding works in the following way:

* Firstly, it finds the frequency of all the characters in a string and stores them in nodes, like this: A,5;
* Secondly, it finds the two least frequent nodes and creates a new node with the two characters and their frequency combined: AB, 10, that will be their parent node. We will be using binary encoding 0 and 1 will be assigned to the child nodes.
* Thirdly, repeat the previous step until there is only one node left with all the characters in the string and the frequency is equal to how many characters the string has.

After this, you should have a tree of nodes similar to the following:

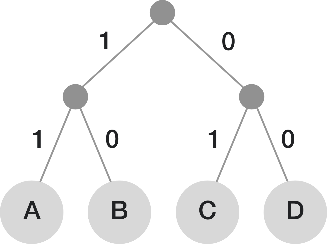


Figure 2 – Huffman Tree

Shannon-Fano coding receives its name from Claude Shannon and Robert Fano since these two both developed very methods in 1948 and 1949 respectively, that ended up being know by their combined name. It is considered inferior to Huffman coding since they have about the same level of complexity but the latter always achieves the lowest possible expected code word length.

The Shannon-Fano coding functions in the following way:

* Firstly, it finds the frequency of all the characters in a string and stores them in a list of character and frequency counts and sort it in a decreasing order of frequency counts;
* Secondly, split the list in two parts so that both parts have as close frequency counts as possible and assign 0 to the left part and 1 to the right part;
* Thirdly, repeat until all the characters are alone in a subpart.

After this, there should be a tree similar to the following:

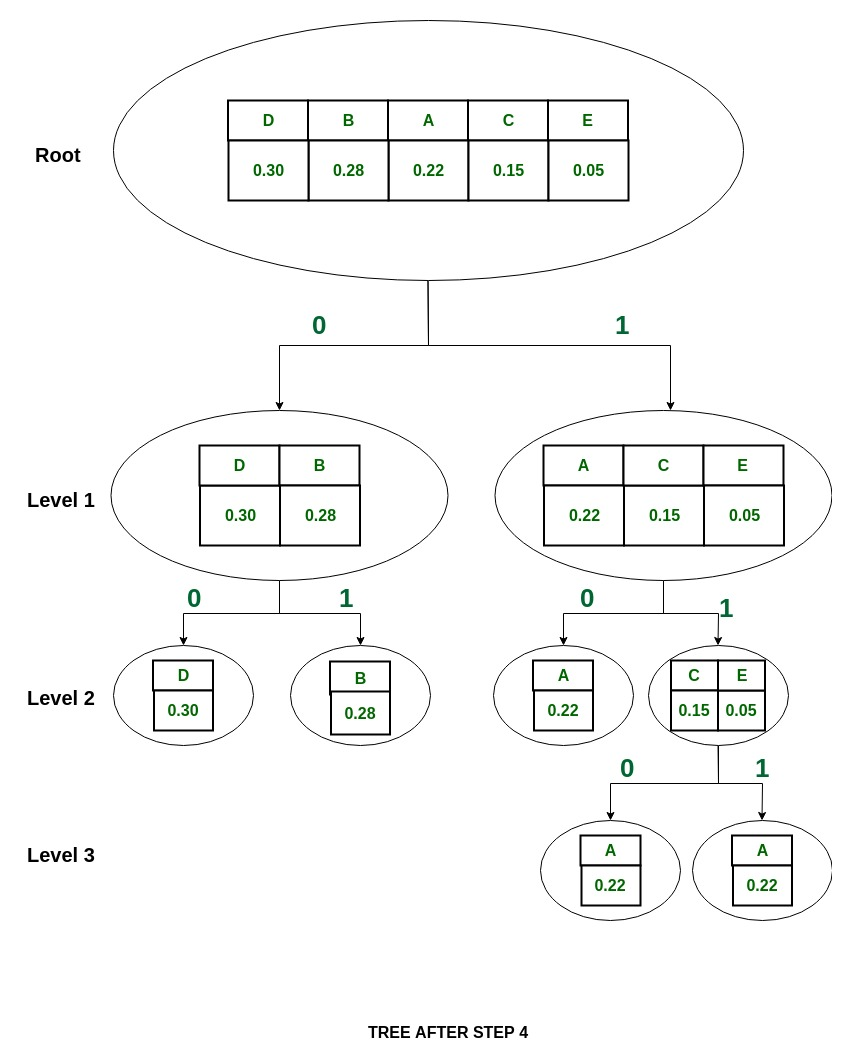


Figure 3 – Final tree for Shannon- Fano coding

*Note:* The two “A”s in the bottom right should be a C and an E respectively

An example of a lossy compression algorithm is JPEG. JPEG is not the .JPG file format, although all .JPG files are JPEG compressed, JPEG is a compression algorithm that was developed by the *Joint Photographic Experts Group*, hence the name, which is used to minimize the file size of photographic image files.

In order to discuss how it works, it is necessary to explain how computers represent photographic images. The way they do it is by dividing the file into a grid of pixels which each have their own colour and the bigger the image, the more pixels and the larger the file will be, so in order to show the image the computer needs to record the individual value of each pixel.

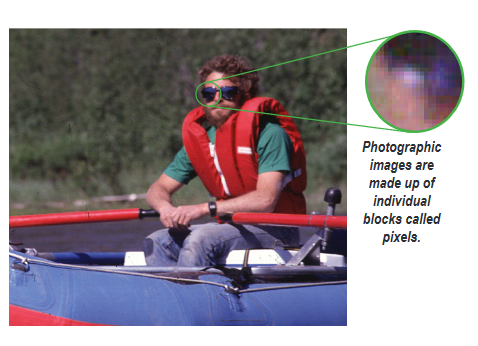


Figure 4 – How an image works

JPEG reduces the number of colour values that need to be saved by evening out transitional colours, the pixels in an image where the colour changes which reduces the quality of a photographic image but is often only done to the point where it is slightly noticeable and does not harm the overall quality of the image. As can be seen in the figure below:



Figure 5 – Effect of JPEG compression

# 3. List of all Data Structures and algorithms

## 3.1 Data Structures

In my Huffman coding, the following data structures were used:

* Files;
* Lists;
* Dictionaries;
* Classes;

Files

Text files of multiple sizes and languages were used to test my Huffman coding. They were the files I compressed into .bin files (binary files) to store the compressed data and they were used to store the decompressed data from the binary file.

Binary files were used to keep the information after this was compressed into bits.

Dictionaries

Dictionaries were one of the backbones of my program, since they were utilized to store the characters from the text files and their corresponding codes and to store the frequency count of each character which made the job of sorting by least frequent character and by smallest bit code as easy and simple as possible.

Lists

Lists were another one of the backbones of my program and are used all throughout for various purposes.

Classes

A class name Huffman tree was created and it was responsible for creating the nodes and assign them parent and children nodes so that the root node and the bit codes for each character can be found and used later on in the program.

# 4. Performance analysis

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| File Name | Original Size | Compressed size | Decompressed size | Compression ratio | Program run time |
| Little Folks (December 1884) A Magazine for the Young | 297KB | 172KB | 297KB | 42.35% | 1.15 seconds |
| Little Folks (December 1884) A Magazine for the Young (PT) | 292KB | 173KB | 298KB | 40.28% | 0.57 seconds |
| Little Folks (December 1884) A Magazine for the Young (FR) | 310KB | 183KB | 317KB | 41.09% | 0.62 seconds |
| The Choice of Life | 304KB | 167KB | 304KB | 45.12% | 0.54 seconds |
| The Choice of Life (PT) | 312KB | 171KB | 318KB | 45.26% | 0.60 seconds |
| The Choice of Life (FR) | 304KB | 180KB | 334KB | 45.07% | 0.59 seconds |
| Einsten.en | 456 667KB |  |  |  |  |
| tm29 | 262 144KB |  |  |  |  |
| Proteins | 102 400KB |  |  |  |  |

# 5. Weekly progress

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Week 1  07/03/2021 13/03/2021 | Week 2  14/03/2021 20/03/2021 | Week 3  21/03/2021 27/03/2021 | Week 4  28/03/2021 02/03/2021 | Week 5  03/03/2021 08/03/2021 |
| Noting weekly progress |  |  |  |  |  |
| Research on what a literature review is |  |  |  |  |  |
| Research on what Huffman coding is |  |  |  |  |  |
| Development of Huffman coding python code |  |  |  |  |  |
| Writing of the literature review |  |  |  |  |  |
| Listing of data structures and algorithms used in my Huffman coding |  |  |  |  |  |
| Testing my Huffman coding |  |  |  |  |  |
| Performance analysis of Huffman coding |  |  |  |  |  |
| Video recording |  |  |  |  |  |

# 6. References

<https://www.researchgate.net/publication/322557949_A_review_of_data_compression_techniques> Fitriya, L.A. & Purboyo, Tito & Prasasti, Anggunmeka. (2017). A review of data compression techniques. International Journal of Applied Engineering Research. 12. 8956-8963.

Khalid Sayood, Chapter 2 - Mathematical Preliminaries for Lossless Compression,Editor(s): Khalid Sayood,In The Morgan Kaufmann Series in Multimedia Information and Systems,Introduction to Data Compression (Fifth Edition),Morgan Kaufmann,2018,Pages 11-39,ISBN 9780128094747, <https://doi.org/10.1016/B978-0-12-809474-7.00002-1>. (<https://www.sciencedirect.com/science/article/pii/B9780128094747000021>)

Khalid Sayood, Chapter 3 - Huffman Coding, Editor(s): Khalid Sayood,In The Morgan Kaufmann Series in Multimedia Information and Systems, Introduction to Data Compression (Fifth Edition), Morgan Kaufmann, 2018, Pages 41-88, ISBN 9780128094747, <https://doi.org/10.1016/B978-0-12-809474-7.00003-3>. (<https://www.sciencedirect.com/science/article/pii/B9780128094747000033>)