



Mark Exhaust Systems Limited

Storage and Transfer of Zipped files

Ministry/ Organization name: Mark Exhaust Systems Limited

Problem Statement: Storage and Transfer of Zipped Files

Team Name: Gravity

Team Leader Name: Dhairya

College Code: U-0564

Idea/Solution

File compression is used to reduce the file size of one or more files. When a file or a group of files is compressed, the resulting "archive" often takes up 50% to 90% less disk space than the original file. We are using the zlib, gzip and bz2 modules in python which provide essential data and file compression tools.

For example, a .tar file is often compressed using GZip to create .tar.gz file, sometimes called a .tgz file. In the case of the ZIP file archive where we are focusing, the compression algorithms are already part of the zipfile module provided by the python.

ZIP archives use a combination of Huffman coding and LZ77 to give fast compression and decompression times and reasonably good compression ratios. LZ77 is pretty much a generalized form of RLE and it will often yield much better results.

Both the algorithms used in the ZIP archives are lossless algorithms which promises complete recovery of the original data. We have used Huffman coding internally for compression of our data.

There are quite a lot of real-world applications of Huffman Encoding. ZIP is perhaps the most widely used compression tool that uses Huffman Encoding as its basis. The latest of the most efficient lossless compression algorithms, Brotli Compression, released by Google last month also uses Huffman Coding.

Huffman is really, really good at some things. Most notably with data that repeats order a lot and contains a sub-set of the character space. For example English language text files. The English language tends to have the same letters followed by the same other letters.

Newer arithmetic and range coding schemes are often avoided because of patent issues, meaning Huffman remains the work-horse of the compression industry.

Huffman algorithm allows the most repeating bytes to represent the least number of bits.

For instance:

We have a data represented in the form of 12 bytes

ppppppppqqrr

Bits Character

0 p

10 q

110 r

00000001 01010110 11000000

Here 5 padding bits are used

This is a small simulation of how our data represented in the form of bytes
is going to be archived and how we have implemented huffman algorithm

This algorithm works faster with bigger data

Technology Stack

- We have used Python as our software development language as Python is well-known for getting your code quickly developed than other languages.
- We have used Anaconda, a powerful and wide functional toolset for using a lot of python related packages.
- We chose a PyQt5 as GUI toolkit for our desktop application as it is most useful for desktop creation python bindings for the Qt (C++ based) software application development framework.

Use Cases

Compression optimizes the use of storage capacity. To take full advantage of this capability, it is important to understand which use cases and workloads benefit most from compression and which benefits less from compression because of computational overhead.

Compressing data is computationally expensive, while decompressing data is less so. From this fact it follows that workloads where data is written once and read frequently, such as user data storage, are most suitable for compression. Examples of such workloads are file servers, archiving, and backup.

Workloads that write data sequentially-such as Hadoop, data analytics, and data warehousing-also benefit from compression.

While compressing initial write operations is computationally expensive, compressing rewrites is more expensive. Each rewrite comprises a read decompress operation followed by a write compress operation.

Huffman coding can be used with data that repeats order a lot and contains a subset of the character space. For example English language text files. The English language tends to have the same letters followed by the same other letters.

Almost all communications with and from the internet are at some point Huffman encoded (A number of communication protocols use it). Most image files (jpegs) are Huffman encoded. Most music files (mp3s) are Huffman encoded.

One reason Huffman is used is because it can be discovered via a slightly different algorithm called adaptive Huffman.

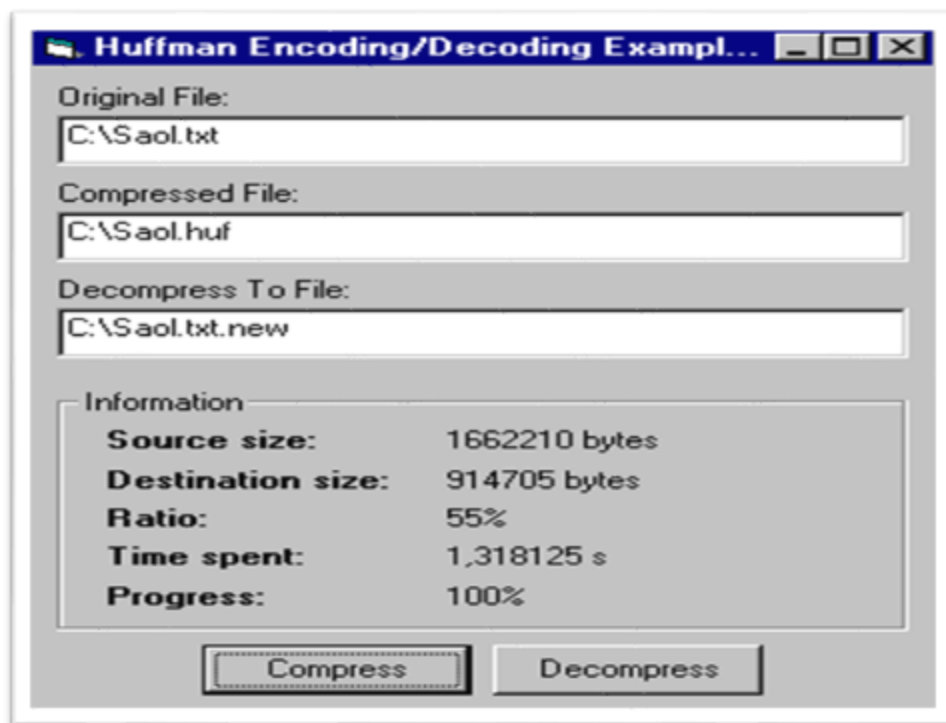
To solve the use the best algorithm for the situation problem, zip files allow a number of different compressions to be used depending on what the best one is for a given file.

Used Cases	Real Life Instances	Suggestions
User data	File system, User data v-Disk	Post-process compression with 3-5 hour delay
Virtual Desktop Infrastructure	VMware View, Citrix Xen Desktop	VCAI snapshots, linked clones, or full clones with Fingerprint on Write rather than container compression
Data processing	Hadoop, Data Analytics, data Warehousing	Inline compression
Transactional applications(Airline reservations, payroll, employee records, manufacturing, and shipping)	Exchange, Active Directory, SQL Server, Oracle	Native application compression where available; otherwise inline compression
Archival or Backup	Handy Backup, Backblaze,SugarSync	Inline compression unless the data is already compressed through other mechanisms

We have created a simulation of our working model. We have implemented huffman coding for file compression. We have also tested it using python.


For GUI we have used Pyqt which is a powerful tool for converting your scripts into a desktop application.

We have also tested zip module of python which internally operates on the huffman algorithm and LZ77 to give us the fast compression and decompression of our data without data loss.





This is how our GUI will look like

Huffman algorithm is implemented using python and a executable file is made which compresses the given file into zip format

Name	Date modified	Type	Size
 sample	13-01-2020 22:27	Text Document	4,086 KB



before implementing huffman algorithm

A sample.txt file is there which we need to compress

Name	Date modified	Type	Size
 huffman	13-01-2020 22:16	Python File	5 KB
 usehuffman	13-01-2020 22:23	Python File	1 KB

executable python compression files

The usehuffman.py is a executable file on executing it will convert sample.txt file into its zip file.

Name	Date modified	Type
 sample	13-01-2020 22:27	Text Document
 sample	14-01-2020 14:55	Compressed (zipp...

FUTURE ENHANCEMENT

The speech output produced in the Huffman coding is without any emotion. The future coders can be used to predict the emotion of the speech and they can be coded along with the speech. The recognition engine accuracy for large vocabulary can be still improved by using some good recognition engine. The safety marking code for unlisted word has been inserted into the Huffman tree randomly. Instead, the probability for custom coding can be generated in the real time. The text prediction and correction system might be embedded with recognition engine so that the user can feel more free and convenient.