COURSE PROJECT REPORT

LEMPEL-ZIV-WELCH (LZW) ALGORITHM FOR COMPRESSION AND DECOMPRESSION

(180030037 & 180030001)

Introduction

There are two categories of compression techniques, lossy and lossless.

Whilst each uses different techniques to compress files, both have the same aim:
To look for duplicate data in the graphic (GIF for LZW) and use a much more compact data representation. Lossless compression reduces bits by identifying and eliminating statistical redundancy.

No information is lost in lossless compression. On the other hand, Lossy compression reduces bits by removing unnecessary or less important information.

The LZW algorithm is a very common compression technique. This algorithm is typically used in GIF and optionally in PDF and TIFF. Unix's 'compress' command, among other uses. It is lossless, meaning no data is lost when compressing.

The algorithm is simple to implement and has the potential for very high throughput in hardware implementations. It is the algorithm of the widely used Unix file compression utility compress, and is used in the GIF image format.

The Idea relies on recurring patterns to save data space. LZW is the foremost technique for general purpose data compression due to its simplicity and versatility.

Procedure

LZW compression works by reading a sequence of symbols, grouping the symbols into strings, and converting the strings into codes.

Because the codes take up less space than the strings they replace, we get compression.

1. LZW compression uses a code table, with 4096 as a common choice for the number of table entries. Codes 0-255 in the code table are always assigned to represent single bytes from the input file.
2. When encoding begins the code table contains only the first 256 entries, with the remainder of the table being blanks. Compression is achieved by using codes 256 through 4095 to represent sequences of bytes.
3. As the encoding continues, LZW identifies repeated sequences in the data, and adds them to the code table.
4. Decoding is achieved by taking each code from the compressed file and translating it through the code table to find what character or characters it represents.

Scope and Limitations

Typically, every character is stored with 8 binary bits, allowing up to 256 unique symbols for the data. This algorithm tries to extend the library to 9 to 12 bits per character.

The new unique symbols are made up of combinations of symbols that occurred previously in the string. It does not always compress well, especially with short, diverse strings.

But is good for compressing redundant data, and does not have to save the new dictionary with the data: this method can both compress and uncompress data.

Implementation

This program requires user input as a python file name, an input text file name and bit length through command line.

Encoding

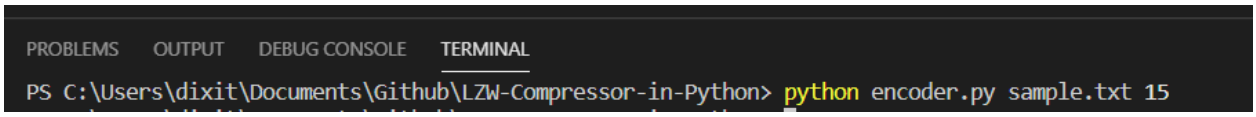
The idea of the compression algorithm is the following: as the input data is being processed, a dictionary keeps a correspondence between the longest encountered words and a list of code values.

The words are replaced by their corresponding codes and so the input file is compressed. Therefore, the efficiency of the algorithm increases as the number of long, repetitive words in the input data increases.

1. The input data is encoded using the encoder.py file.

2. The dictionary of size 256 is built and initialized, using the python dictionary data structure in the dictionary, key are characters and values are the ascii values the lzw compression algorithm is applied and we get the compressed data.
3. The program outputs the compressed data and stores it to an output file named inputFileName.lzw
4. The compressed data is of 2 bytes.

Usage :



Decoding

The LZW decompressor creates the same string table during decompression.

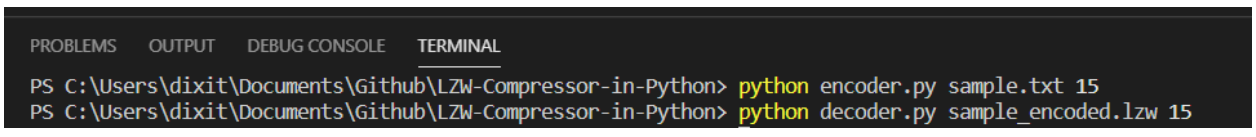
It starts with the first 256 table entries initialized to single characters.

The string table is updated for each character in the input stream, except the first one.

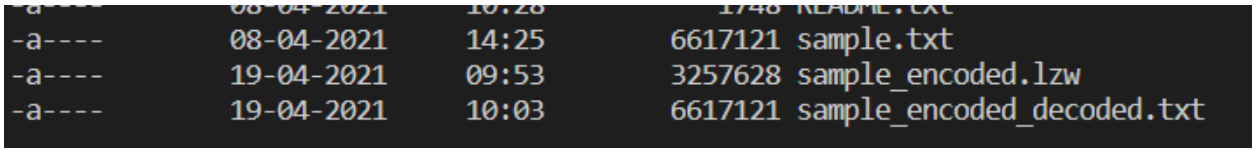
Decoding achieved by reading codes and translating them through the code table being built.

1. The compressed data is decompressed using the decoder.py file.
2. The dictionary of size 256 is built and initialized, using the python dictionary data structure in the dictionary, key are characters and values are the ascii values the lzw decompression algorithm is applied and we get the decompressed data.
3. The program outputs the decompressed data and stores it to an output file named inputFileName_decoded.txt

Usage:



Performance



Original File Size = 6617121 bytes
Compressed File Size = 3257628 bytes
Decoded File Size = 6617121 bytes

Compression Ratio achieved = 3257628/6617121 * 100 = **49.23 %**
