DATA COMPRESSION ALGORITHMS AND THEIR ANALYSIS

Srujana B and Hema Varshita M

Contents

***Abstract*—Ever since wireless communications took over the world, data and it's efficient transmission have been playing a vital role. Our main goal has become to send maximum amount of data in less time, that is to decrease the latency of data. This paper provides the analysis of DEFLATE data compression algorithm and it's comparison with the compression method that uses Burrows Wheeler Transform tested on sample dataset.**

1 DATA COMPRESSION

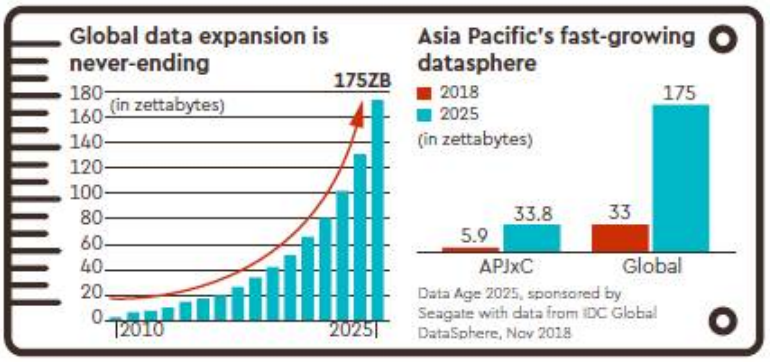


Fig. 1.0: Data Growth

- The graph above depicts how global data is constantly expanding through years. It is important to note the role it plays in our daily life.
- Data Compression is used to save cost of processing and latency time in transmitting huge zeta bytes of data
- It is the process of modifying, encoding or converting the bits structure of data in such a way that it consumes less memory.
- Also known as source coding or bit-rate reduction.
- Data compression has wide implementation in data communications(SMS,MMS)
- A common data compression technique removes and replaces repetitive data elements and symbols to reduce the data size
- Data compression algorithms are mainly classified into Lossy and Lossless categories
  – Lossy algorithms delete all repeated data
  – Lossless algorithms save all repetitive data
- Hence compression ratio is higher in lossy data algorithms. When decompressed, some data is lost when lossy algorithms are used. So, lossless algorithms are used for compressing text data where as lossy algorithms are used for audio, image and video files.
- Here are some lossless data compression algorithms – Huffman
  – LZ78 (and LZW variation)
  – LZ77
  – Arithmetic coding
  – Sequitur
  – Prediction With Partial Matching (ppm)
- We mainly focus on LZ77 + Huffman which is collectively called the DEFLATE Algorithm. This algorithm is used in gzip file

compression format.
- Compression ratio = $\frac{OriginalFilesize}{CompressedFilesize}$

2 HUFFMAN ALGORITHM

- Variable length codes, called prefix codes based on their frequencies are assigned to the input characters
- Greater the frequency, smaller is the length of code that is assigned to it.
- These codes are binary strings that are used for transmission of message
- A code with prefix character $0, 11$ has the prefix property whereas a code with prefix character $0, 1, 11$ doesn't. It is because here '1' is already a prefix of '11'. This method enables us to ensure that our data is uniquely decodable.
- A huffman tree is constructed using Min heaps and priority queues
- As soon as the tree is constructed, its receiver scans the incoming bit stream and reads 0 as left traversal, and 1 as right traversal until the leaf node is reached
- The bit stream of an alphabet will be obtained after one traversal to the leaf node. The first 8 bits (= 1 byte) will be used to compress the data

3 LZ77 ALGORITHM

- It is abbreviated as Lempel-Zeiv 77 algorithm
- It is also a reversible algorithm. We can fully recover the original string without any loss of data
- In this algorithm we use a bit to distinguish literal characters from pairs
- The text goes in a sliding window
- It consists of a Search buffer(SB) and a look ahead buffer(LAB) inside a sliding window. The encoder checks for the symbol in look ahead buffer considering search buffer as the dictionary
- Encoded string using this algorithm consists of tuples of size 3, (offset, length, symbol)
- Offset is the count from diving line between search and look ahead buffer where the longest match occurred.
- Length is number of characters in LAB , which found match in SB.
- Symbol is the immediate character which did not find a match
- Text is read from right to left in the buffer.
- Although there are a lot of variants of LZ77, the main concept for each of them remains the same.
- The default zip algorithm uses 32Kb sliding window
  https://www.zlib.net/

4 DEFLATE ALGORITHM

- A significantly better compression rate is achieved by combining LZ77 with an additional entropy coding algorithm. DEFLATE uses Huffman codes for that purpose
- Raw data ——> LZ77 ——> Huffman ——> Compressed data
- The LZ77 algorithm finds repeated substrings and replaces them with backward references
- The second stage, i.e. Huffman algorithm replaces commonly used symbols with shorter representations and less commonly used symbols with longer representations
- DEFLATE is a pretty straight forward algorithm which is modified according to the compressor needs. For eg : Zlib uses a 32kB sliding window
- A deflate complaint compressor streams data as blocks, thus applying different modes of compression on different blocks based on its properties.
- Usually, it has the following modes: (However, user can choose any number of modes)
  1) If the data is already compressed. Leave it, else deflate since compressed data might expand a little on deflation
  2) Compression, first with LZ77 and then with Huffman coding. Here, no extra space is needed to store the trees. (i.e
  3) Compression, first with LZ77 and then with Huffman coding with trees that the compressor creates and stores along with the data.

5 BWT TRANSFORM

- Burrows Wheelers Transform doesn't compress data but rearranges the character string into runs of similar characters.

- The resulting string has less number of runs leading to easy compression of the data by techniques like : Run length encoding, move to front transform.

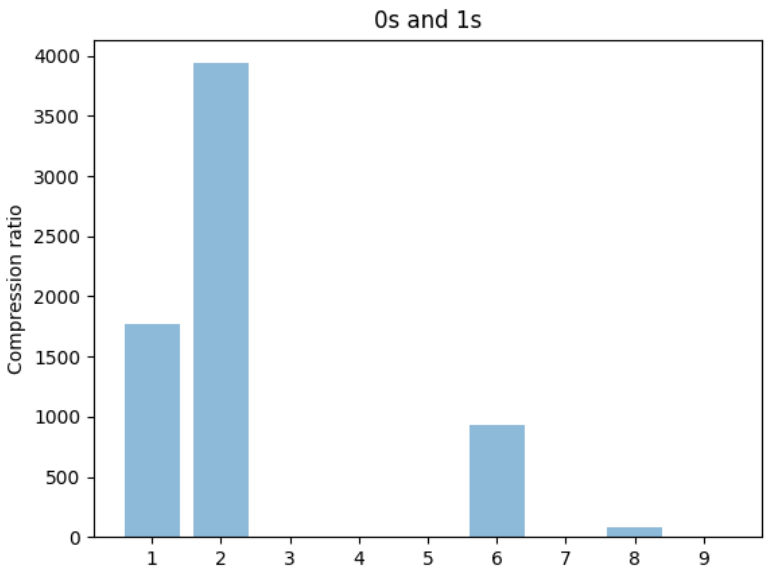
6 RLE

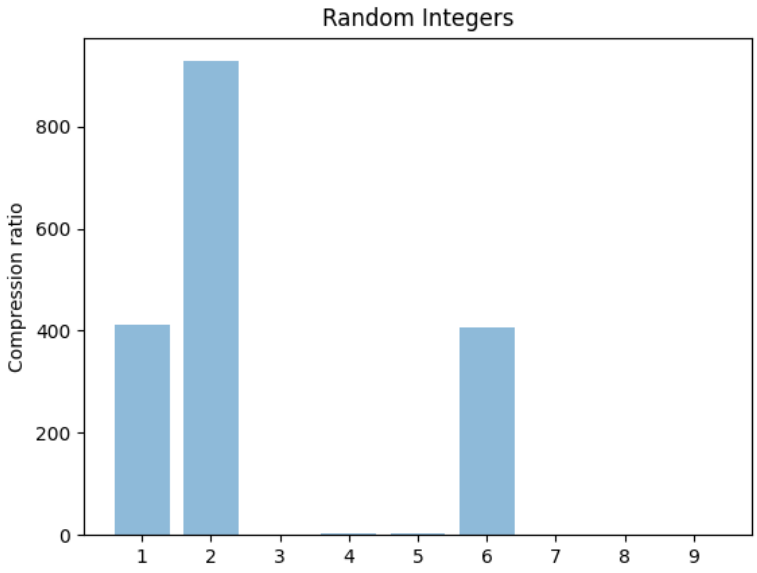
- Run Length Encoding (RLE) is a form of lossless data compression which runs on sequences having same value occurring consecutive times.
- It encodes data such that only a single value with it's count is stored.
- For example : AAABCAAAAABBCD is encoded as 3A1B1C5A2B1C1D
- Worst case comparison : Sequence - ABCD Encoded as - 1A1B1C1D. The output size is double of the input size.
- RLE is usually applied on the transformed data obtained from BWT.

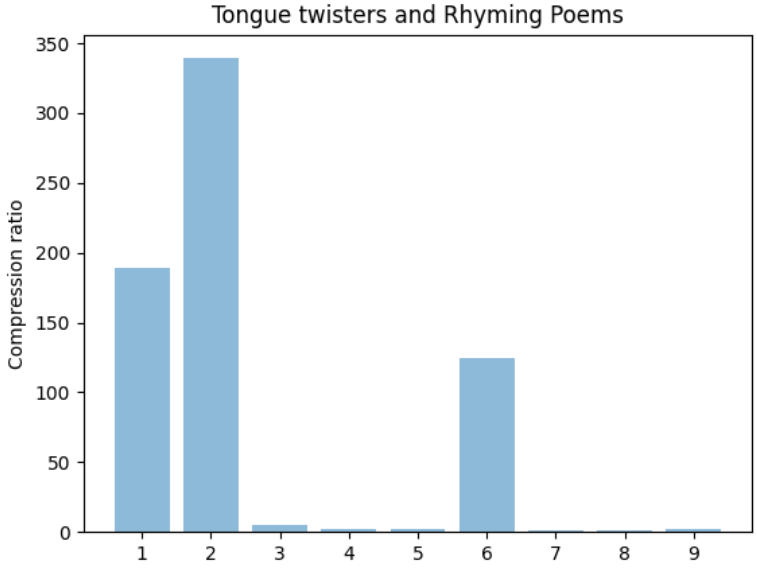
7 RESULTS

- We used 4 data sets to run a combination of the mentioned algorithms -
  1) 0s and 1s - It is a dataset consisting of random patterns of binary digits : Original Size 50 kB
  2) Random integers - This dataset contains random combinations of integers from 0 to 9 : Original Size 29.7 kB
  3) A data set consisting of tongue twisters and rhyming poems - Original size 54.4 kB
  4) A test data set consitsting of english text - Original size 73.4 kB
- Results of running this algorithm on the 9 algorithm combinations above have been tabulated below
- Algorithm / their combination in the plots and table is indicated by the number shown in the table below

1	BWT+RLE
2	BWT+RLE+HUFFMAN
3	BWT+LZ77+HUFFMAN
4	BWT+HUFFMAN
5	HUFFMAN
6	LZ77+BWT+RLE
7	LZ77+BWT+HUFFMAN
8	LZ77+HUFFMAN
9	BWT+LZ77







Algo	Test data	0s and 1s	0-9	T. Twisters
Original	73.4 kB	50 kB	29.7 kB	54.4 kB
1	374 B	29 B	72 B	303 B
2	215 B	13 B	32 B	169 B
3	41.8 kB	7.7 kB	32.6 kB	11.6 kB
4	44.8 kB	10.9 kB	13.6 kB	30.2 kB
5	44.8 kB	10.9 kB	13.6 kB	30.2 kB
6	341 B	55 B	73 B	435 B
7	50.2 kB	7.7 kB	29.1 kB	73.8 kB
8	50.2 kB	7.7 kB	29.1 kB	73.8 kB
9	93.8 kB	35.3 kB	108.8 kB	33.4 kB

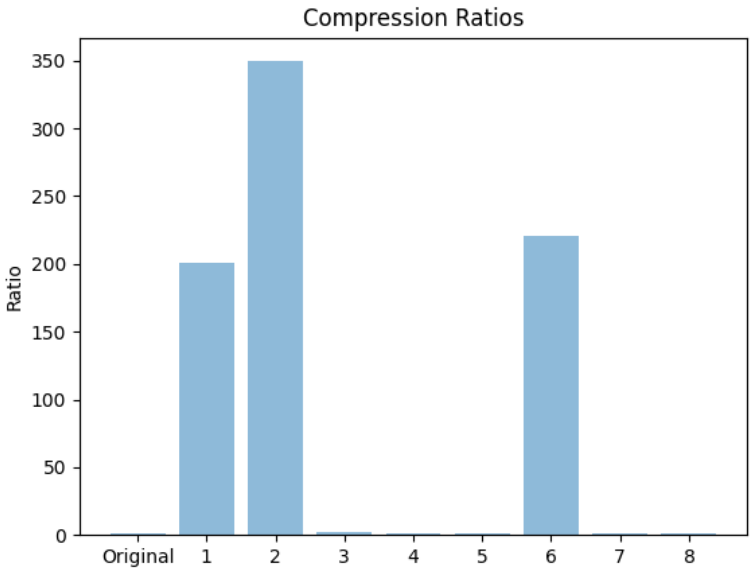
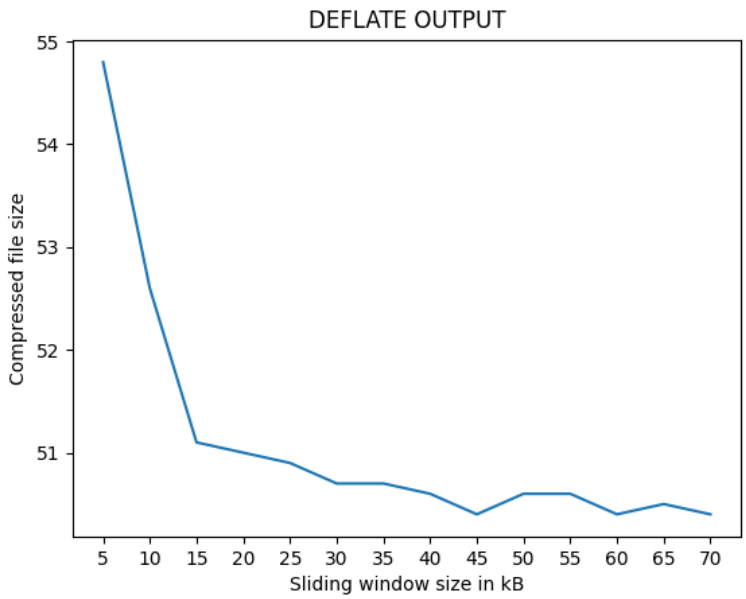


Fig. 4: Test data

8 CONCLUSION

- Results show that BWT+RLE with any other compression scheme gives the optimal compression ratio. At the same time it should be noted that it is computationally expensive.
- From the results, it is seen that there's no difference between BWT+HUFFMAN and HUFFMAN's compression ratio. This is because of the fact huffman only deals with the frequencies and bwt is just a rearrangement of character string.



- It must be noted that LZ77 algorithm achieves better results when there are repeating "sequences" of data.It also achieves a decent compression ratio, and has less computational overhead.
- DEFLATE OUTPUT (tested on test data set of size 73.4kB) plot shows the variation in output file size with change in Sliding Window size. If the size is too small, the algorithm may not be able to find repeated sequences efficiently.
- Beyond a certain size of sliding window, the file size would increase. This can be seen if we deal with larger datasets.

9 REFERENCES

- https://www.researchgate.net/publication/2485507_The_Burrows-Wheeler_Transform_Theory_and_Practice
- https://www.techiedelight.com/huffman-coding/
- https://zlib.net/feldspar.html
- https://en.wikipedia.org/wiki/LZ77_and_LZ78
- http://paper.ijcsns.org/07_book/201005/20100520.pdf

10 CODE

- https://github.com/nightKrwler/DataCompression