An Alacritous Compression by Amalgamating Modified Adaptive Huffman and Lempel-Ziv-Welch Algorithm

Suman Laha
Guru Ghasidas Vishwavidyalaya
Koni, Bilaspur, CG-495009.

Abstract: Compression refers to reducing the amount of data bits which used to represent store and/or transmit file content, without excessively reducing the amount of the input data. Data Compression technique can enables exact reproduction of the input data on decomposition which leads to loss in compressed data. The Adaptive Huffman Algorithm with FGK process used to compress the particular set of files without loss of data but the expected result of compression is very poor. To overcome the poor compression, it will be modified along with Vitter algorithm. Vitter Algorithm helps to compress the total number of data files and to produces the better efficiency but the drawback was the execution time of data file. Since it requires more execution time, more space, results are in in efficient of the compressed data. So to rectify the arising problems, this paper introduced Modified Adaptive Huffman algorithm along with Lempel-Ziv-Welch (LZW) Algorithm. Lempel-Ziv-Welch is a data based technical algorithm which compresses the repetitive sequences of data very well. LZW requires no earlier data about the information stream and it can pack the info stream in one single go with its straight forwardness and enabling quick execution to bring the first content information document with highest compression ratio.

Keywords: Data Compression, FGK, Vitter Algorithm, High compression ratio, Data File.

1. INTRODUCTION

Data compression is the way toward altering, encoding or changing over the bits structure of information such that it devours less space on disk. It supports dropping the storage size of one or more data occasions or elements. Data compression is also known as source coding or bit-rate reduction. Data compression empowers sending a data protest or file rapidly finished a network or the Internet and in upgrading physical capacity assets. Data compression has wide implementation in computing services and solutions, specifically data communications. Data compression works through a few packing systems and programming arrangements that use information compression algorithms to lessen the information measure. A typical information compression strategy evacuates and replaces dull information components and images to lessen the information estimate.

The viewed picture compression utilizing basic coding procedures called Huffman .Discrete Wavelet Transform (DWT) coding and fractal algorithm is done. These systems are straightforward in usage and use less memory. Huffman coding system includes in lessening the repetitive information in input pictures.DWT can have the capacity to enhance the nature of compressed picture [1].The paper gives a philosophy to loss information pressure in keen appropriation frameworks utilizing the particular esteem decay strategy. The clarified strategy is prepared to do fundamentally lessening the volume of information to be transmitted through the correspondences organize and precisely reproducing the first information [2]. There are two classifications of pressure systems utilized with computerized illustrations, loss and lossless. While every utilization distinctive procedures to pack records, both have a similar point. To search for copy information in the realistic and utilize a considerably more conservative information portrayal. Using this approach, the signal is partitioned into a set of 8 samples and each set is DCT-transformed. The least-significant transform coefficients are removed before transmission and are filled with zeros before an inverse transform [3]. Loss and Lossless each have different strategies which are utilized by various record organizes and accomplish distinctive outcomes. Accordingly not all loss or lossless organizations will utilize similar techniques. It is past the extent of this Unit to take a gander at these techniques in detail so you won't be surveyed on them. The Unit entitled Digital Imaging: Bitmaps covers compression strategies in more detail. If you are a not unclear about this, the following may help:

- Loss compression methods include DCT (Discreet Cosine Transform), Vector Quantization and Huffman coding
- Lossless compression methods include RLE (Run Length Encoding), string-table compression, LZW (Lempel Ziff Welch) and zlib.

In this it is to analyze the challenges and openings in the arrangement of adaptable remote structures to get a handle on the colossal data time frame. On one hand, study the front line arranging models and banner handling frameworks adaptable for managing tremendous data development in remote frameworks [4]. Compression strategies are also called algorithms, which are figuring's that are utilized to pack files. Associations that make record positions make their own specific algorithms and battle with each other to make the best game plan.

The work concerns the ensured organization of 3D remedial pictures, with the essential point that such organization must occur in an absolutely clear manner for the end-customer, paying little regard to the computational and frameworks organization capacities may use [5]. Algorithms look for monotonous data, i.e. repeat regards that aren't required, in the reasonable and make another depiction of the data. This depiction influences a more diminutive archive to appraise yet keeps the vital data for demonstrating the sensible. The compression procedure starts by linearizing multi-dimensional preview data.

The key thought is to fit/foresee the progressive data focuses with the best fit determination of bend fitting models [6]. Basically when you take a gander at a compacted document you should see no or little distinction in the realistic. The paper surveys the improvement and pattern of data stream bunching and examines run of the mill data stream grouping algorithms clarified as of late, for example, Birch algorithm, Local Search algorithm, Stream algorithm and CluStream algorithm [7].

The target of this examination is to outline computationally proficient designs for executing discrete wavelet change based ultrasonic three-dimensional (3D) information compression algorithm on a reconfigurable ultrasonic framework on-chip (SoC) equipment stage [8]. Utilizing exact perceptions that tactile information have solid spatiotemporal compressibility, this paper clarifies a novel compressive information accumulation plot for remote sensor systems [9]. A CODEC (compacted/decompressed) is utilized complete the algorithm to spare a record in a packed arrangement and open a compacted document. CODECs can be executed in either equipment or programming. This paper presents flywheel, a HTTP proxy service that expands the life of versatile information designs by compacting reactions in-flight between starting point servers and customer programs [10].

The development of the Lempel-Ziv-Welch (LZW) [11] Coded Probabilistic Finite State Automata (LZW-Coded PFSA) to arrange exercises, for example, strolling, hopping, running, midriff pivots, and shoulder turns. The PFSA uncover the hidden design of a given action and characterize it without making any an earlier presumptions by gathering designs from the sensor estimations. LZW-Coded PFSA select the ideal variable length state from the time-arrangement information and pack with effectiveness.

In the modified Adaptive Huffman algorithm the repeated data should be coded with another symbol. So it takes more codal symbols ,occupies more space and increases the compression and decompression timings. Thus to code the repeated data's and to classify the similar data's, this paper introduced LZW algorithm in Adaptive Huffmon algorithm. The rest of the paper is organized as related researches in the section 2,the section 3 comprises the implementation of Adaptive Huffman algorithm with LZW, section 4 comprises of comparison results and analysis followed by references.

2. Related Research

Tooth Zhang et.al,[12] explained a constant data compression and adjusted convention method for wide-zone estimation frameworks (WAMS). The compression algorithm joins special case compression (EC) with swing entryway inclining (SDT) compression. The compression rationale is intended to play out this algorithm progressively. Choice of compression parameters and data remaking is displayed. An adjusted convention is presented by enhancing the organization of data outlines characterized by IEEE standard C37.118 for compacted data parcels. The clarified compression system and convention were connected to the phasor estimation units (PMUs) of a hydropower plant in Guizhou Power Grid in Southwest China. A low-recurrence wavering occurrence was recorded by this system. The crude, packed and remade data were broke down to check the compression and decide the precision of the clarified method. Additionally, the wavelet-based data compression, the independent EC and SDT are contrasted and the clarified compression procedure. The compression proportions of the ESDC will reach in the scope of 6 to 11, and declines the correspondence data by roughly 75% just with not more precision progressively for both the relentless and dynamic states.

Nandita Vijaykumar et.al,[13] presented the Core-Assisted Bottleneck Acceleration (CABA) structure that utilizes sit without moving on-chip assets to lighten diverse bottlenecks in GPU execution. CABA gives adaptable systems to consequently produce "help twists "that execute on GPU centers to perform particular errands that can enhance GPU execution and efficiency. CABA empowers the utilization of sit out of gear computational units and pipelines to reduce the memory transmission capacity bottleneck, e.g. By utilizing help twists to perform data compression to exchange less data from memory. On the other hand, a similar structure can be utilized to deal with situations where the GPU is bottlenecked by the accessible computational units, in which case the memory pipelines are sit and can be utilized by CABA to accelerate calculation, e.g., by performing remembrance utilizing help warps. CABA is a general substrate that can lighten the memory transmission capacity bottleneck in present day GPU frameworks by empowering adaptable executions of data compression algorithms. To alleviate a wide range of framework bottlenecks in throughput-arranged designs, and there is not any more effective usage.

Cristian Perra et.al,[14] demonstrated the Plenoptic images are obtained from the projection of the light intersection a framework of miniaturized scale focal point clusters which reproduces the scene from various bearing into a camera gadget sensor. Plenoptic images have an alternate structure regarding standard computerized images, and novel algorithms for information compression. It clarifies an algorithm for the compression of plenoptic images. The smaller scale images creating a plenoptic picture are prepared by a versatile expectation device, going for lessening information relationship before entropy coding happens. The initial assumption provides a better understanding of the plenoptic signal structure but have to leverage the development of novel compression algorithms with limited compression ratio

Yafei Xing et.al, [15] explained a Holographic data assume a vital part in late three-dimensional imaging and additionally minuscule applications. Colossal measures of capacity limit will be included for this sort of data. In this way, it winds up important to create productive multi dimensional image compression plans for capacity and transmission purposes. It center around the moved separation data, acquired by the stage moving algorithm, where two arrangements of distinction data should be encoded. All the more definitely, a divisible vector lifting plan is researched so as to abuse the two-dimensional attributes of the holographic substance. In

addition, the clarified deterioration has been adjusted to the data substance. Because of the specific structures of such data, the expanding expectation channel length increases just regarding bitrate sparing and visual nature of reproduction.

Chunsheng Zhu et.al,[16] explained a preparing of the tangible data in WSN– MCC incorporation, by distinguishing the basic issues concerning WSN– MCC coordination and clarifying a novel tactile data handling system, which goes for transmitted attractive tangible data to the portable clients in a quick, solid, and secure way. The clarified structure could drag out the WSN lifetime, diminish the capacity necessities of the sensors and the WSN entryway, and lessen the movement load and transfer speed prerequisite of tactile data transmissions. Furthermore, the structure is fit for checking and anticipating the future pattern of the tangible data movement, and in addition enhancing its security. It gave system equipped for upgrading the system lifetime, the capacity prerequisite, the security and observing execution of WSNs, and the security of the transmitted tactile data and of decreasing the movement and transfer speed required for tangible data transmissions and the distributed storage and handling overhead. The execution of the system empowers the portable clients to safely acquire their coveted tangible data speedier however it neglects to retrieve.

3. An Innovation of Efficient Lossless Text Data Compression Techniques:

The improved Huffman Coding algorithm works in two phases to compress the text data specially in the numeric data and symbolic data. This paper presents different data compression methodologies performing the lossless data to compress the data further and to produce the final output. The original text data file is compressed by Adaptive Huffman Algorithm along with the FGK process only with the particular set of files. To compress large number of files the FGK process along with the Vitter algorithm is introduced to compress the total number of data files. This algorithm was named as Modified Adaptive Huffman Algorithm. In modified Adaptive Huffman algorithm each numeric code and the corresponding binary codes will be generated dynamically to obtain the compressed binary output. But the generated binary code forgets to identify the repeated data's, it codes the similar data's separately. Hence it occupies more space, more time and efficiency to bring the original output data's. So modified Adaptive Huffman coding techniques has a need for Efficient Lempel-Ziv-Welch (LZW) to identify the repeated datas.LZW algorithm gets the modified input data's from the Adaptive Huffman Algorithm identifies the repeated data's and symbols the repeated data in a single code with the highest compression ratio. Then the data's will be decoded within LZW provides the output stream in one single pass with simplicity and fast execution without loss in data. The following figure 1 explains the process of lossless data compression.

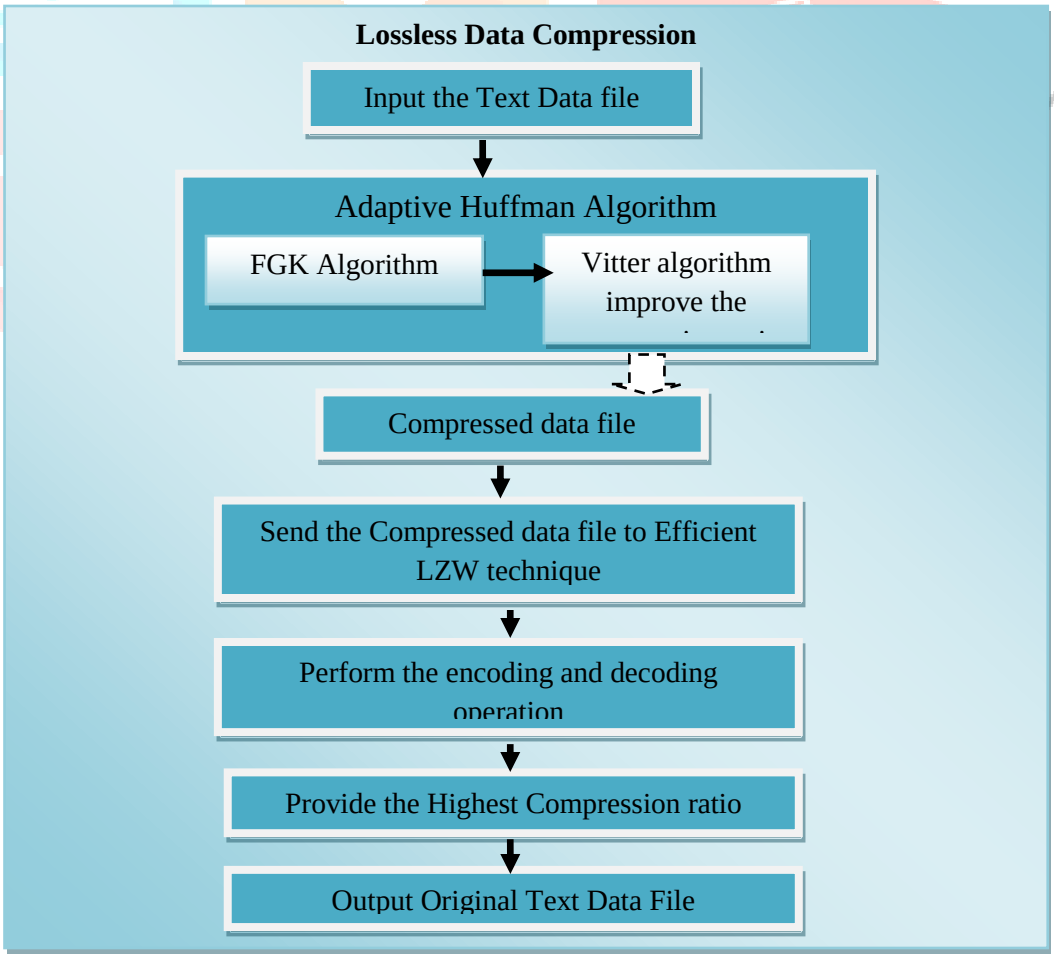


Figure 1: Architecture for Lossless Data Compression Technique

3.1 Compression of Input Data by Adaptive Huffman coding with FGK and Vitter Algorithm

Adaptive Huffman coding is likewise called Dynamic Huffman coding. It is an adaptive coding system in view of Huffman coding. It grants assembling the code as the images are being transmitted, having no underlying information of source circulation, that permits one-pass encoding and adjustment to changing conditions in data. The advantage of one-pass strategy is that the source can be encoded progressively, however it turns out to be more touchy to transmission blunders, since only a solitary misfortune ruins the entire code. There are various usage of this strategy, the most remarkable are FGK (Faller-Gallager-Knuth) and Vitter algorithm.

3.1.1 FGK (Faller-Gallager-Knuth) Algorithm:

The reason for algorithm FGK is the Sibling Property. A double code tree with non negative weights has the sibling property if every hub (aside from the root) has a sibling and if the hubs can be numbered arranged by no diminishing weight with every hub contiguous its sibling. Moreover the parent of a node is higher in the numbering. A binary prefix code is a Huffman code if and only if the code tree has the sibling property. In this sibling model is described in Fig 2.

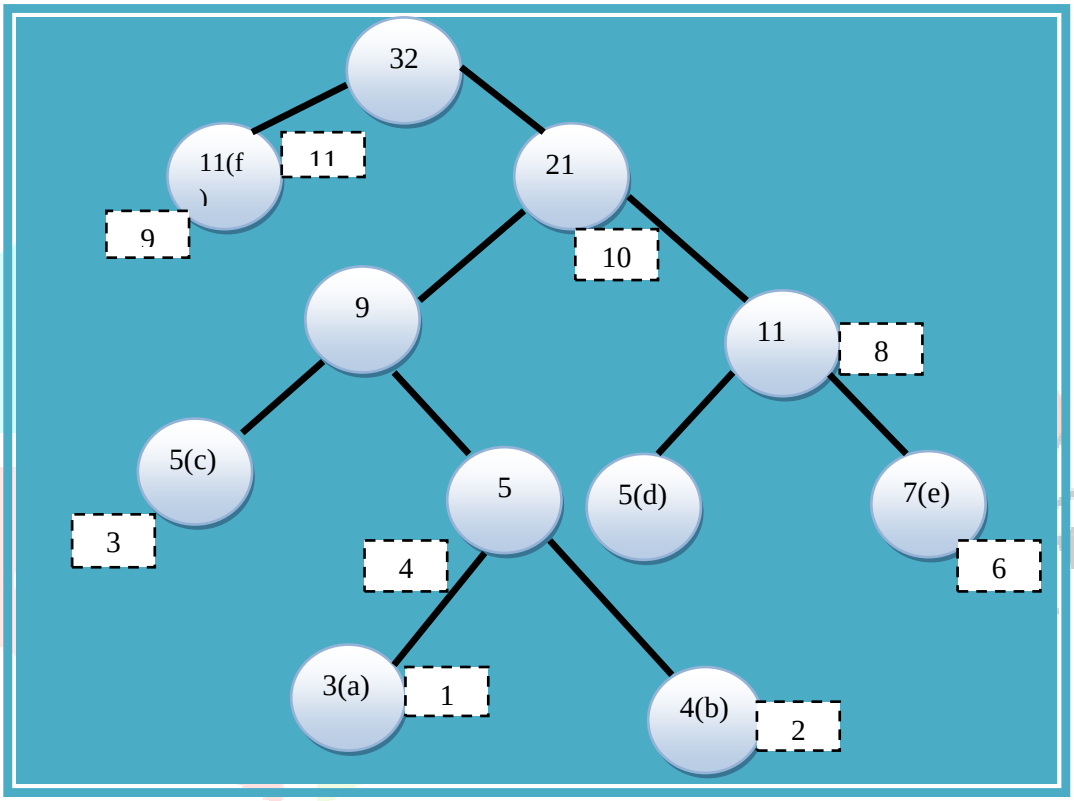


Figure 2: Process of FGK Algorithm

Note that node numbering corresponds to the order in which the nodes are combined by Huffman's algorithm, first nodes 1 and 2, then nodes 3 and 4.

Algorithm FGK compares favorably with static Huffman code, if we consider also overhead costs. If T=total number of bits transmitted by algorithm FGK for a message of length t containing n distinct symbols, then the equation is,

$$S = n+1 \le T \le 2S + t - 4n + 2 \quad (1)$$

Where S is the execution of the static Huffman, t is the length and n is an image. So the execution of algorithm FGK is never much more regrettable than twice ideal. Since the FGK Algorithm has a few downsides about the hub or-sub tree swapping, so here utilizing the Vitter disclosed algorithm to enhance it.

3.2.3 Vitter Algorithm:

Vitter Algorithm presents two changes over algorithm FGK, calling the new plan algorithm V. As a tribute to his work, the algorithm is turned out to be well known with the letter flipped topsy turvy algorithm V. Swapping of hubs amid encoding and interpreting is difficult. In FGK algorithm the quantity of changing (considering a double cost for the updates that move a changed node two levels higher) is bounded by $[d_t / 2]$, where d_t is the length of the added symbol in the old tree (this bound require some effort to be proved and is due to the work of Vitter). In algorithm V, the number of changing is bounded by 1. Moreover algorithm V,

not only minimize $\sum w_i l_i$ as Huffman and FGK, but also $Max l_i$, i.e. the height of the tree, and $\sum l_i$, i.e. is better suited to code next symbol, given it could be represented by any of the leaves of the tree. When transmit an NYT symbol and have to transmit code for the NYT node, then for its generic code. For every symbol that is already in the tree, only have to transmit code for its leaf node. In this operation is performed by Figure 3.

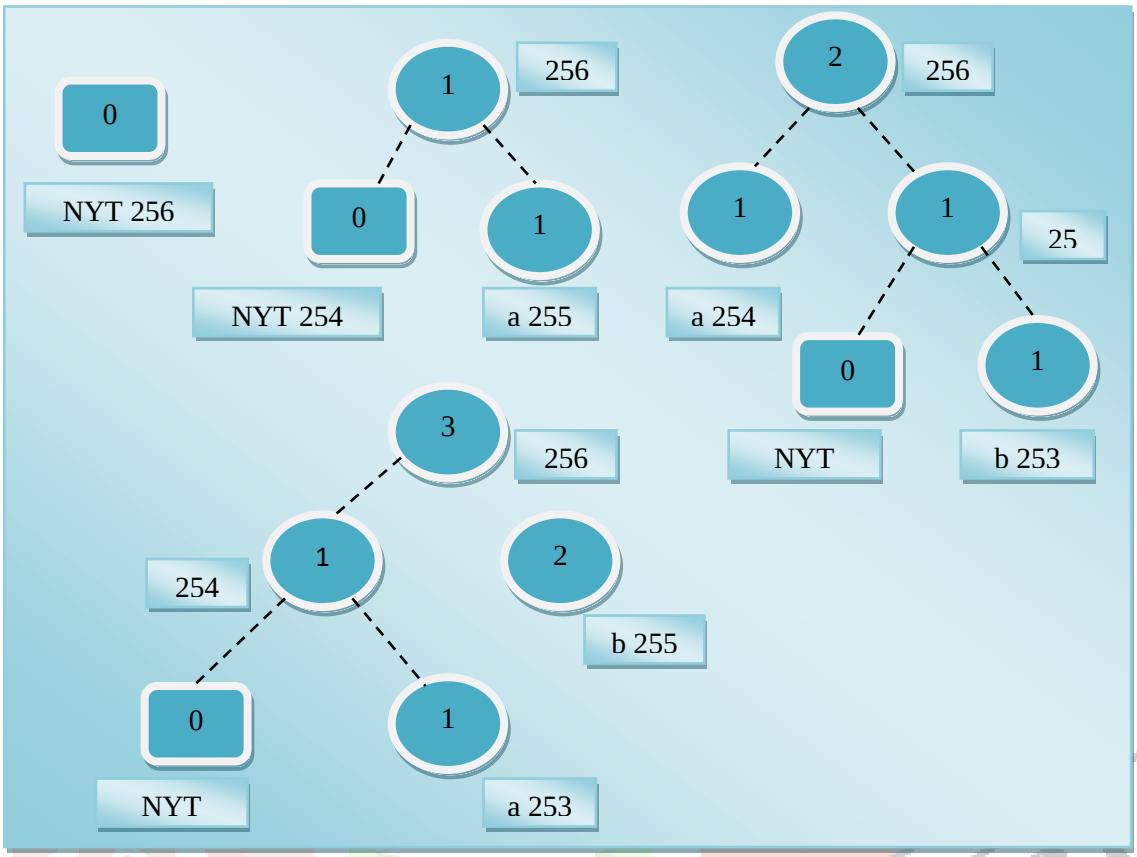


Figure 3: Performance of Vitter Algorithm

The output of the "abb" gives 01100001 001100010 11.The execution of the NYT image transmission is at first with an exhaust tree. For "a" transmit its binary code. Then NYT brings forth two tyke hubs: 254 and 255, both with weight 0. Increment weight for root and 255. Code for "a", related with hub 255, is 1.For "b" transmit 0 (for NYT hub) at that point its binary code. And the NYT brings forth two kid hubs: 252 for NYT and 253 for leaf hub, both with weight 0. Increment weights for 253, 254, and root. To keep up Vitter's invariant that all leaves of weight w continue (in the certain numbering) every interior hub of weight w, the branch beginning with hub 254 ought to be swop (regarding images and weights, yet not number requesting) with hub 255. Code for "b" is 11.For the second "b" transmit 11. At that point the comfort of clarification this progression doesn't precisely take after Vitter's algorithm, however the impacts are identical. At last go to leaf hub 253. Notice they have two squares with weight 1. Hub 253 and 254 is one square (comprising of leaves), hub 255 is another piece (comprising of inward hubs). For hub 253, the greatest number in its piece is 254, so swap the weights and images of hubs 253 and 254. Presently hub 254 and the branch beginning from hub 255 fulfill the Slide And Increment condition and consequently should be exchange. Finally increment hub 255 and 256's weight. Future code for "b" is 1, and for "an" is currently 01, which mirrors their recurrence. The accompanying
figure is clarified algorithm of including an image.

```
leaf_ to_ increment = NULL
S = pointer to the leaf node containing the next symbol
IF (S is NYT) THEN
Extend P by adding two children
Left child becomes new NYT and right child is the new symbol leaf
node
S = parent of new symbol leaf node
leaf_ to_ increment = Right Child of p
ELSE
Swap S with leader of its block
IF (new p is sibling to NYT) THEN
leaf_ to_ increment = S
S = parent of p
WHILE (S != NULL)
Slide_ And_ Increment(p)
IF (leaf_ to_ increment != NULL)
Slide_ And_ Increment(leaf_ to_ increment)
```

Figure 4: Algorithm for adding a symbol

These two targets are come to through another numbering plan, called implicit numbering. The nodes of the tree are numbered in expanding request by level; nodes on one level are numbered lower than the nodes on the following more elevated amount. Nodes on a similar level are numbered in expanding request from left to right. On the off chance that this numbering is satisfied (and in FGK it isn't for the most part satisfied), certain sorts of updates can't occur.

The way to limit the other sort of trades is to keep up the accompanying invariant. For each weight w, all leaves of weight w go before (in the verifiable numbering) every inside hub of weight w. The exchanges, in the algorithm V, are intended to reestablish certain numbering, when another image is perused, and to safeguard the invariant. In the event that T=total number of bits transmitted by algorithm V for a message of length t containing n particular images, at that point the condition is,

$$V = S - n + 1 \le T \le 2S + t - 2n + 1 \quad (2)$$

Even from a pessimistic standpoint at that point, Vitter's versatile technique may transmit one more piece for each code word than the static Huffman strategy. Observationally, algorithm V marginally outflanks algorithm. In this execution isn't fulfilled, so the Huffman coding methods were offered for Lempel-Ziv-Welch (LZW) for information based procedure. This strategy will enhance the execution of compression.3.3 Innovation on Efficient LZW (Lempel–Ziv–Welch) Compression technique

The LZW algorithm is an exceptionally normal compression strategy. This algorithm is ordinarily utilized as a part of GIF and alternatively in PDF and TIFF. Unix's pack charge, among different employments. It is lossless, which means no data is lost when compacting. The algorithm is easy to execute and has the potential for high throughput in equipment usage. It is the algorithm of the generally utilized Unix file compression utility pack, and is utilized as a part of the Zip file design. The Idea depends on reoccurring examples to spare data space. LZW is the chief strategy for broadly useful data compression because of its straightforwardness and adaptability. It is the premise of numerous PC utilities that claim to twofold the limit of your hard drive.

3.3.1 Working Principle of LZW

LZW compression works by perusing an arrangement of images, gathering the images into strings, and changing over the strings into codes. Since the codes consume up less space than the strings they supplant, to get compression. Trademark highlights of LZW incorporates,

- LZW compression utilizes a code table, with 4096 as a typical decision for the quantity of table passages. Codes 0-255 in the code table are constantly appointed to speak to single bytes from the information document.
- When encoding starts the code table contains just the initial 256 sections, with the rest of the table being spaces. Compression is accomplished by utilizing codes 256 through 4095 to speak to groupings of bytes.
- As the encoding proceeds with, LZW distinguishes rehashed arrangements in the information, and adds them to the code table.
- 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.

Example: ASCII code. Regularly, every character is put away with 8 binary bits, permitting up to 256 one of a kind images for the information. This algorithm tries to stretch out the library to 9 to 12 bits for every character. The new interesting images are comprised of blends of images that happened beforehand in the string. It doesn't generally pack well, particularly with short, various strings. However, is useful for compacting repetitive information, and does not need to spare the new lexicon with the information, this technique can both pack and uncompressed information.

3.3.2 Implementation of Efficient LZW

The possibility of the compression algorithm is the accompanying: as the information is being handled, a content information keeps a correspondence between the longest experienced words and a rundown of code esteems. The words are supplanted by their relating codes thus the info record is compacted. Consequently, the effectiveness of the algorithm increments as the quantity of long, tedious words in the info information increments. Compression procedure of LZW is to Adding the static Huffman execution and Vitter execution. Therfore the String Table is,

$$ST = S + V \tag{3}$$

Where ST is the string table, S is the Static Huffman performance and V is the Vitter. This equation is to perform the encoding and decoding performance.

Encoding LZW

Use the LZW algorithm to compress the string: BABAABAAA.The steps involved are systematically shown in the diagram below.

Table 1: LZW Compression Step 1

Encoder Output		String Table	
Output Code	Representation	Code word	String
66	B	256	BA

Table 2: LZW Compression Step

Encoder Output		String Table	
Output Code	Representation	Code word	String
66	B	256	BA
65	A	257	AB

Table 3: LZW Compression Step 3

Encoder Output		String Table	
Output Code	Representation	Code word	String
66	B	256	BA
65	A	257	AB
256	BA	258	BAA

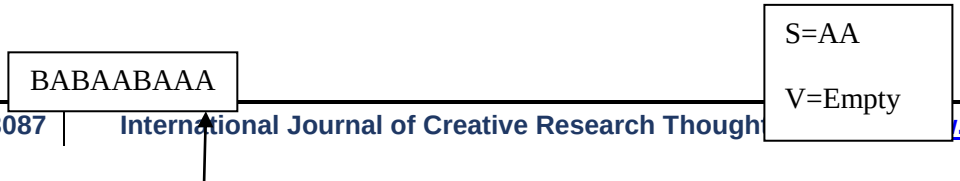
Table 4: LZW Compression Step 4

Encoder Output		String Table	
Output Code	Representation	Code word	String
66	B	256	BA
65	A	257	AB
256	BA	258	BAA
257	AB	259	ABA

Table 5: LZW Compression Step 5

Encoder Output		String Table	
Output Code	Representation	Code word	String
66	B	256	BA
65	A	257	AB
256	BA	258	BAA
257	AB	259	ABA
65	A	260	AA

Table 6: LZW Compression Step 6



Encoder Output		String Table	
Output Code	Representation	Code word	String
66	B	256	BA
65	A	257	AB
256	BA	258	BAA
257	AB	259	ABA
65	A	260	AA
260	AA		

```
Initialize table with single character strings
S= first input character
WHILE not end of input stream
V = next input character
IF S + V is in the string table
ST = S + V
ELSE
output the code for S
add S + V to the string table
S = V
END WHILE
output code for S
```

Figure 5: Pseudo code for LZW Encoding

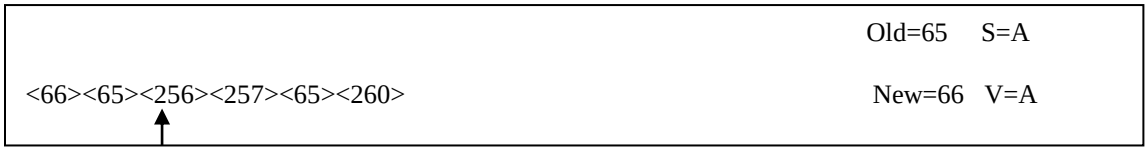
The compressed values are <66><65><256><257><65><260>.In this values to decompressed by using LZW technique. The following figures are explaining the decompression process.

LZW Decoding:

Use LZW to decompress the output sequence of : <66><65><256><257><65><260>

The steps involved are systematically shown in the diagram below.

Table 7: LZW decompression step 1



Decoder Output	String Table	
String	Code word	String
B		
A	256	BA

Table 8: LZW decompression step 2

Old=256 S=BA

<66><65><256><257><65><260> New=256 V=B

Decoder Output	String Table	
String	Code word	String
B		
A	256	BA
BA	257	AB

Table 9: LZW decompression step 3

Old=257 S=AB

<66><65><256><257><65><260> New=257 V=A

Decoder Output	String Table	
String	Code word	String
B		
A	256	BA
BA	257	AB
AB	258	BAA

Table 10: LZW decompression step 4

Old=65 S=A

<66><65><256><257><65><260> New=66 V=A

Decoder Output	String Table	
String	Code word	String
B		
A	256	BA
BA	257	AB
AB	258	BAA
A	259	ABA

Table 11: LZW decompression step 5

<66><65><256><257><65><260>

Old=65 S=A
New=66 V=A

Decoder Output	String Table	
String	Code word	String
B		
A	256	BA
BA	257	AB
AB	258	BAA
A	259	ABA
AA	260	AA

```
Initialize table with single character strings
      OLD = first input code
        Output translation of OLD
 WHILE not end of input stream
          NEW = next input code
 IF NEW is not in the string table
          S = translation of OLD
          ST = S + C
 ELSE
          S = translation of NEW
          output T
          C = first character of S
          OLD + C to the string table
          OLD = NEW
 END WHILE
```

Figure 6: Pseudo Code for LZW Decompression Algorithm

Finally the LZW algorithm compresses repetitive sequences of data very well. Since the code words are 12 bits, any single encoded character will expand the data size rather than reduce it.

4. Results and Discussion

This segment clarified the execution of the system of Data Compression which plays out the compression, decompression and Compression proportion. Compression and decompression time is essentially with an adjustment in compression proportion. Encoding is the way toward putting a grouping of characters (letters, numbers, accentuation, and certain images) into a particular arrangement for productive transmission or capacity. Decoding is the contrary procedure - the change of an encoded organize once again into the first succession of characters. Encoding and decoding are utilized as a part of data interchanges, systems administration, and capacity. The clarified framework for the packed data it is utilized to actualize in the working stage of JAVA with the related framework arrangement.4.1 Performance Analysis

The compression time, decompression time and compression ratio is must always be low for obtaining efficient result. The compression time, decompression time and compression ratio is shown below tables.

Table 12: Compression time for Modified Adaptive Huffman with Efficient LZW

File Name	File Size(byte)	Compression Time(m sec)
F1	22,094	50236
F2	44,355	150731
F3	11,252	14356

This table shows the compression time of Lossless process.Here the file F1 having the file size 22,094,which has the compression time is 50236.The file F2 having the file size 44355,which has the time of compression is 150731.Finally the file F3 having the file size 11,252 which has the compression time is 14356 and the performance of the result is shown in below graph.

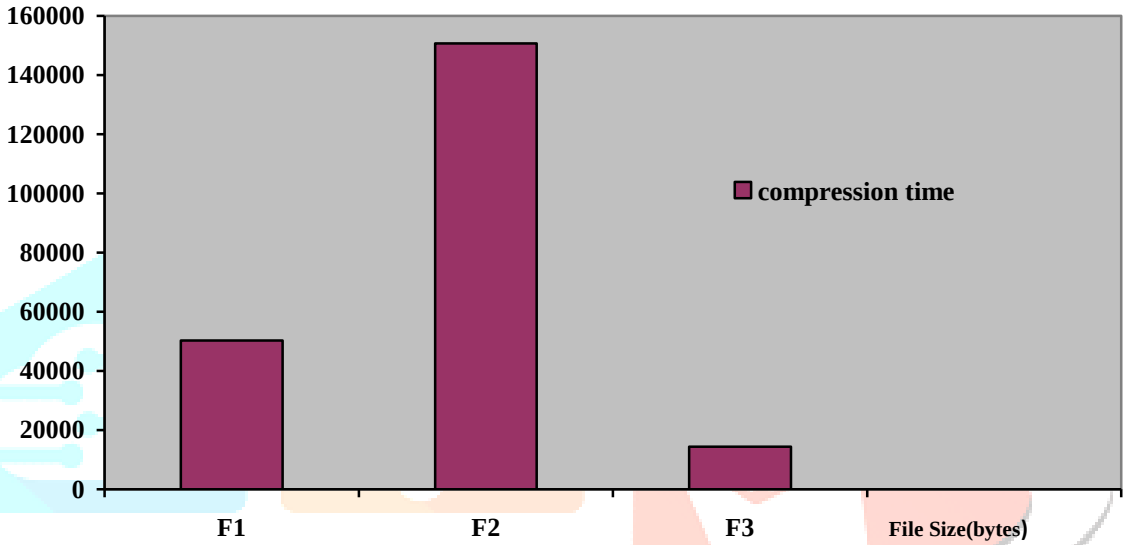


Figure 7: Compression time for Modified Adaptive Huffman with Efficient LZW

Table 13: Decompression time for Modified Adaptive Huffman with Efficient LZW

File Name	File Size(byte)	Decompression Time(m sec)
F1	22,094	6142
F2	44,355	6883
F3	11,252	2317

This table shows the decompression time of Lossless process.Here the file F1 having the file size 22,094,which has the decompression time is 6142.The file F2 having the file size 44,355,which has the time of decompression is 6883.Finally the file F3 having the file size 44,355 which has the decompression time is 2317 and the performance of the result is shown in below graph.

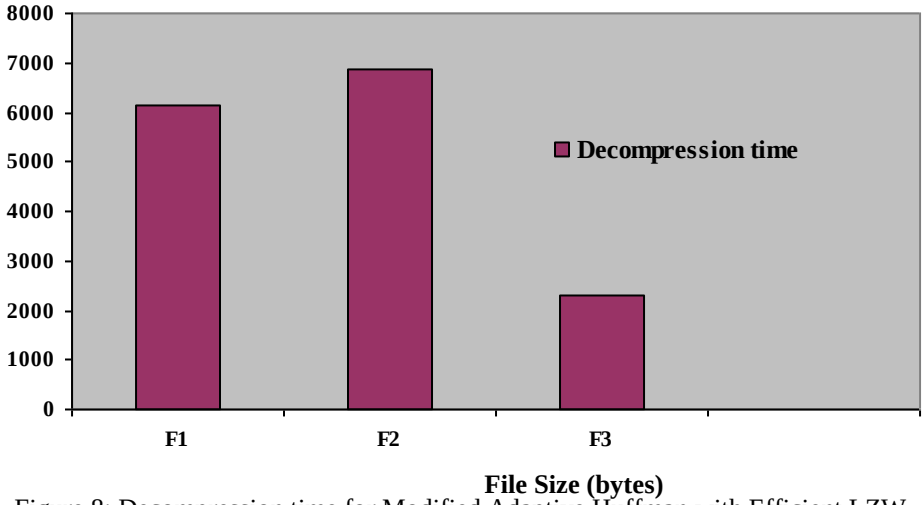


Figure 8: Decompression time for Modified Adaptive Huffman with Efficient LZW

Table 14: Compression Ratio for Modified Adaptive Huffman with Efficient LZW

File Name	File Size	Compression Ratio
F1	22,094	62.7633
F2	44,355	58.2336
F3	11,252	70.30235

This table shows the compression ratio of Lossless process.Here the file F1 having the file size 62,7633,which has the compression ratio is 62.7633.The file F2 having the file size 44,355 which has the compression ratio is 58.2336.Finally the file F3 having the file size 11,252,which has the compression ratio is 70.30235 and the performance of the result is shown in below graph.

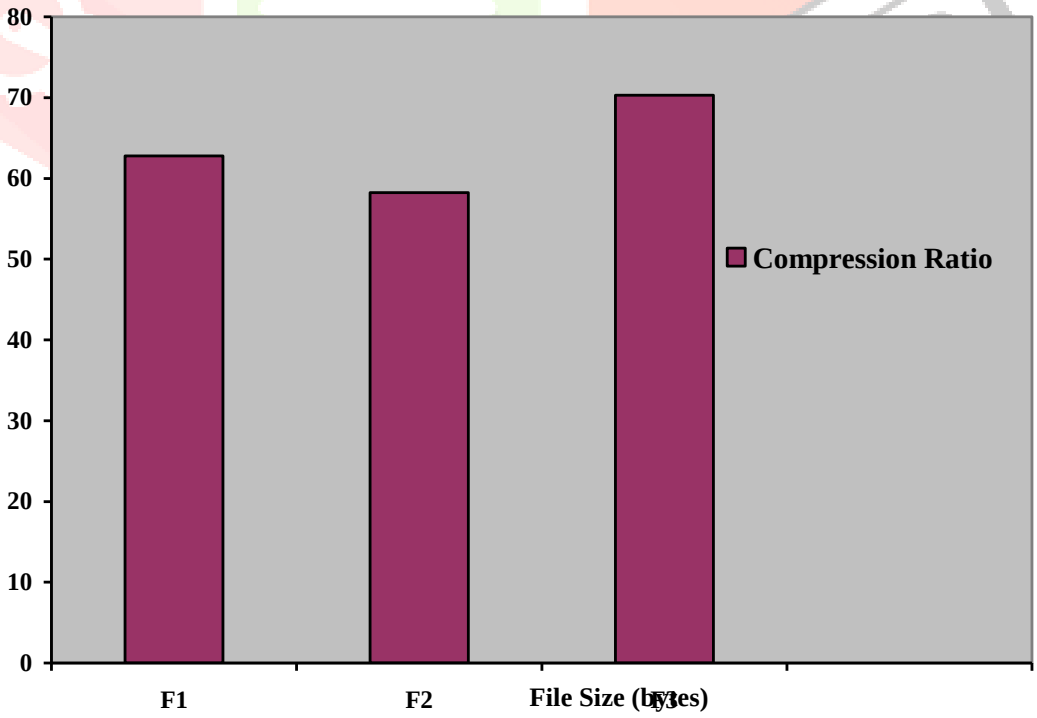


Figure 9: Compression Ratio for Modified Adaptive Huffman with Efficient LZW

4.2. Comparison Results

The data size utilized for compression time is 50236, 150731 and 14356.In this section perform the Comparison of Adaptive Huffman, Efficient LZW and Modified Adaptive Huffman with Efficient LZW. In that data values are generated on Fig 4.4.

Table 15: Compression time for Adaptive Huffman, Efficient LZW and Modified Adaptive Huffman with Efficient LZW

Algorithms	Compression Time
Adaptive Huffman	111646.00
Efficient LZW	235375.00
Modified Adaptive Huffman with Efficient LZW	71774.33

From this table shows the compression time of Adaptive Huffman, Efficient LZW,and Modified Adaptive Huffman with Efficient LZW.Here Adaptive Huffman algorithm having the compression time is 111646,Efficient LZW algorithm having the compression time is 235375.Finally the Modified Adaptive Huffman with Efficient LZW algorithm having the compression time 71774.33 and the performance of the result is shown in below graph.

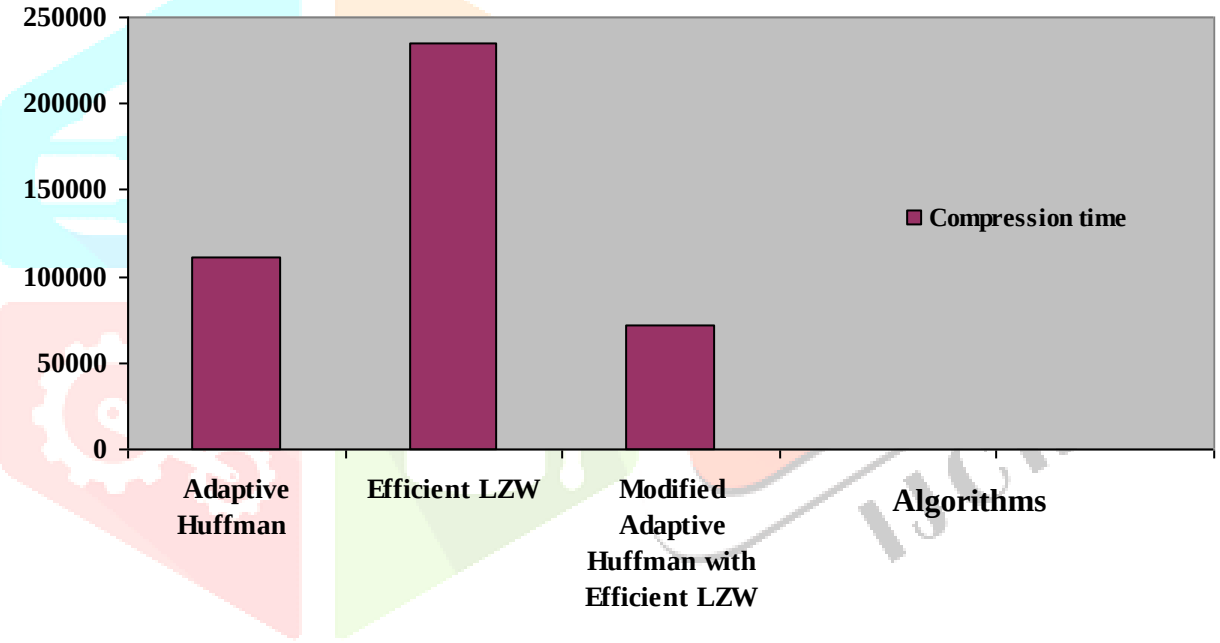


Figure 10: Compression time for Adaptive Huffman, Efficient LZW, Modified Adaptive Huffman with Efficient LZW

The data size utilized for decompression time is 6142,6883and 2317 In this section perform the Comparison of Adaptive Huffman, Arithmetic Efficient LZW and Modified Adaptive Huffman with Efficient LZW.In that data values are generated on Fig 4.5.

Table 16: Decompression time for Adaptive Huffman, Efficient LZW and Modified Adaptive Huffman with Efficient LZW

Algorithms	Decompression Time
Adaptive Huffman	8351.333
Efficient LZW	5906.333
Modified Adaptive Huffman with Efficient LZW	4911.00

From this table demonstrates the decompression time of Adaptive Huffman, Efficient LZW,and Modified Adaptive Huffman with Efficient LZW.Here Adaptive Huffman algorithm having the decompression time is835130.33, Efficient LZW algorithm having the decompression time is 59063.333.Finally the Modified Adaptive Huffman with Efficient LZW algorithm having the decompression time is 49114.00 and the execution of the outcome is appeared in underneath diagram.

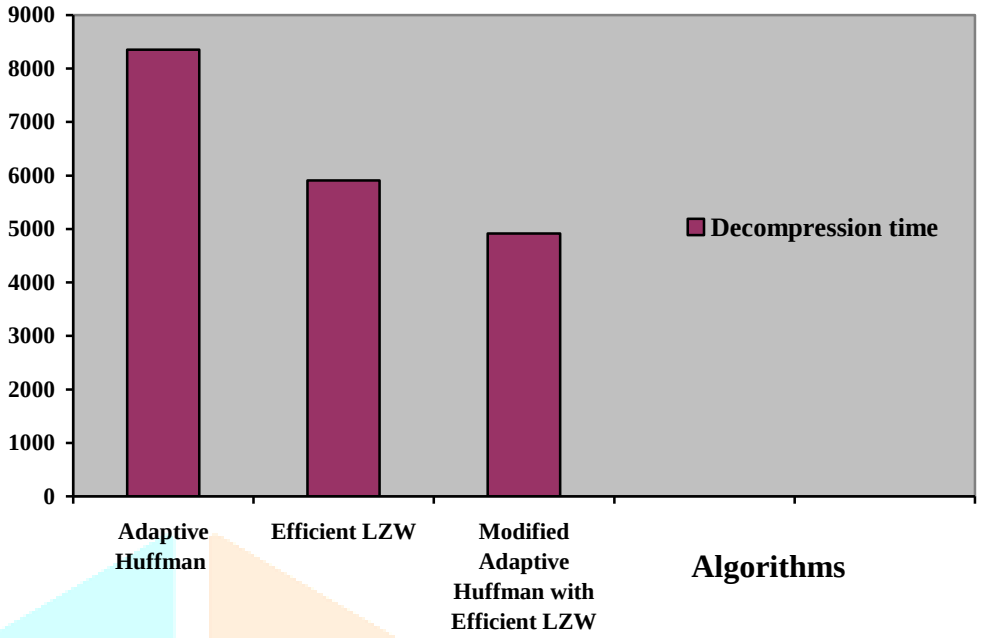


Figure 11: Decompression time for Adaptive Huffman, Efficient LZW and Modified Adaptive Huffman with Efficient LZW

The data size utilized compression ratio is 62.7633, 58.2336 and 70.3025. In this section perform the Comparison of Adaptive Huffman, Efficient LZW and Modified Adaptive Huffman with Efficient LZW. In that data values are generated on Fig 4.6.

Table 17: Compression Ratio for Adaptive Huffman, Arithmetic and Efficient LZW

Algorithms	Compression Ratio
Adaptive Huffman	61.86367
Efficient LZW	62.4298
Modified Adaptive Huffman with Efficient LZW algorithm	66.763083

From this table shows the compression ratio of Adaptive Huffman,Efficient LZW and Modified Adaptive Huffman with Efficient LZW algorithm. Here Adaptive Huffman algorithm having the compression ratio is 61.86367, LZW algorithm having the compression ratio is 62.4298.Finally Modified Adaptive Huffman with Efficient LZW algorithm having the compression ratio of 66.763083 and the performance of the result is show below graph.

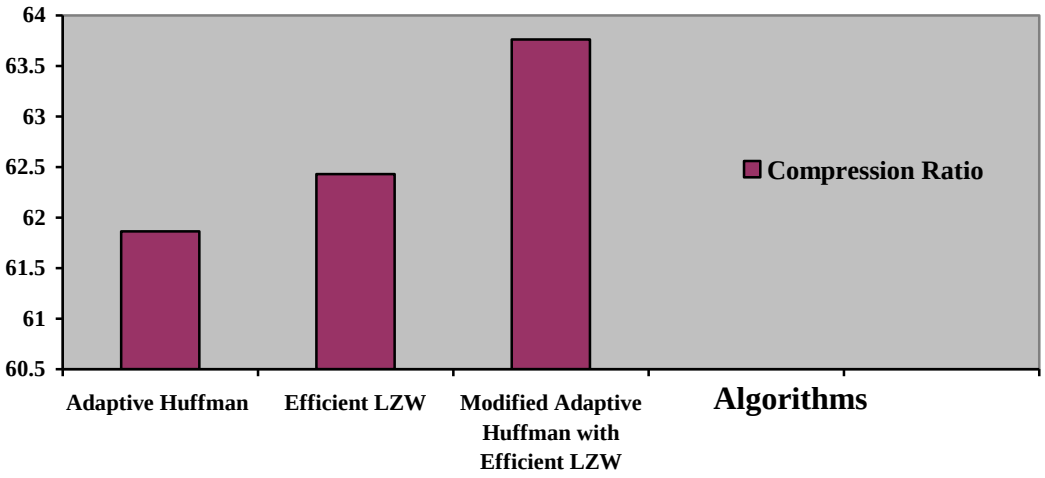


Figure 12: Compression Ratio for Adaptive Huffman, Efficient LZW and Modified Adaptive Huffman with Efficient LZW

In the table 4.4, 4.5 and 4.6 demonstrates the effectiveness of the clarified strategy is contrasted and the diverse strategies. The Adaptive Huffman and Efficient LZW are utilized to give the compression and decompression time in inefficient manner. The Compression Time of Adaptive Huffman has 111646, LZW contains 235375.00 and proposed algorithm contains 71774.33. The Decompression Time of Adaptive Huffman has 8351.333,LZW contains 5906.333 and proposed algorithm contains 4911.00, The Compression ratio of Adaptive Huffman has 61.86367, Efficient LZW has 62.4298 and proposed algorithm contains 66.763083.The Explained procedure effectively enhances the productivity contrast with alternate systems. Thus it explains the proposed Adaptive Huffman with LZW performs well compared to Adaptive Huffman and efficient LZW algorithms.

5. Conclusion

To store large number of files in limited space, the demand for compressing the data will be increased in day to day life. Hence the essential need is to compress the data without any loss. So the paper explained the lossless compression of data using Adaptive Huffman with LZW. The explained algorithms were implemented in Java, valuating the compression ratio, compression time and decompression time using text files without loss. The comparisons were made with Adaptive Huffman, LZW and Arithmatic Coding. The results showed that Huffman and Arithmetic coding techniques were offered for Lempel-Ziv-Welch (LZW) indicated the best results with the highest compression ratio of 66.763083, followed by Adaptive Huffman and Efficient LZW with compression ratio of 61.8636 and 62.4298. It was noted that the performance of the data compression algorithms on compression time and decompression time depend on the characteristics of the files, the different symbols and symbol frequencies. The Adaptive Hoffman with LZW Algorithm reads the attributes of the documents, symbols, frequencies and provides the output data file as same as input data file with efficiently on its time and ratio without the loss of any data.

References

1. Jasmi, R., Praisline, B., Perumal, and Pallikonda Rajasekaran, M. 2015. Comparison of image compression techniques using huffman coding, DWT and fractal algorithm. Computer Communication and Informatics (ICCCI), 2015 International Conference on. IEEE.
2. de Souza, Julio Cesar Stacchini, Tatiana Mariano Lessa Assis, and Bikash Chandra Pal. 2017. Data compression in smart distribution systems via singular value decomposition. IEEE Transactions on Smart Grid, 8(1): 275-284.
3. Birvinskas, Darius, et al. 2015. Fast DCT algorithms for EEG data compression in embedded systems. Computer Science and Information Systems, 12(1): 49-62.
4. Bi, Suzhi, et al. 2015. Wireless communications in the era of big data. IEEE communications magazine, 53(10): 190-199.
5. Castiglione, Arcangelo, et al. 2015. Cloud-based adaptive compression and secure management services for 3D healthcare data. Future Generation Computer Systems, 43: 120-134.
6. Di, Sheng, and Franck Cappello. 2016. Fast error-bounded lossy HPC data compression with SZ. Parallel and Distributed Processing Symposium, 2016 IEEE International. IEEE.
7. Ding, Shifei, et al. 2015. Research on data stream clustering algorithms. Artificial Intelligence Review, 43(4): 593-600.
8. Govindan, Pramod, et al. 2016. Hardware and software architectures for computationally efficient three-dimensional ultrasonic data compression. IET Circuits, Devices & Systems, 10(1): 54-61.
9. Liu, Xiao-Yang, et al. 2015. CDC: Compressive data collection for wireless sensor networks. IEEE Transactions on Parallel and Distributed Systems, 26(8): 2188-2197.
10. Agababov, V., Buettner, M., Chudnovsky, V., Cogan, M., Greenstein, B., McDaniel, S., Piatek, M., Scott, C., Welsh, M., Yin, B. 2015. Flywheel: Google's Data Compression Proxy for the Mobile Web. InNSDI, 15: 367-380.
11 .Wilson, J., Najjar, N., Hare, J., & Gupta, S. 2015. Human activity recognition using LZW-coded probabilistic finite state automata. In Robotics and Automation (ICRA), 2015 IEEE International Conference on IEEE, 3018-3023.
12. Zhang, Fang, Lin Cheng, Xiong Li, Yuanzhang Sun, Wenzhong Gao, and Weixing Zhao. 2015. Application of a real-time data compression and adapted protocol technique for WAMS. IEEE Transactions on Power Systems, 30(2): 653-662.
13. Vijaykumar, N., Pekhimenko, G., Jog, A., Bhowmick, A., Ausavarungnirun, R., Das, C., & Mutlu, O. 2015. A case for core-assisted bottleneck acceleration in GPUs: enabling flexible data compression with assist warps. In ACM SIGARCH Computer Architecture News. 43(3): 41-53.
14. Perra, C. 2015. Lossless plenoptic image compression using adaptive block differential prediction. In Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on IEEE, 1231-1234.
15. Xing, Yafei, et al. 2015. Adaptive nonseparable vector lifting scheme for digital holographic data compression. Applied optics, 54(1): A98-A109.
16. Zhu, Chunsheng, Hai Wang, Xiulong Liu, Lei Shu, Laurence Yang, T. and Victor Leung, C.M. 2016. A novel sensory data processing framework to integrate sensor networks with mobile cloud. IEEE Systems Journal, 10(3): 1125-1136.