DATA COMPRESSION

Project report submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering/Information Technology

By

Akshat Gupta (141241)

Under the supervision of

(**Prof. Dr. Satya Prakash Ghrera**)

to

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

Candidate's Declaration

I hereby declare that the work presented in this report entitled **"DATA COMPRESSION"** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from August 2016 to December 2016 under the supervision of **Dr. Satya Prakash Ghrera** (**Professor, Brig (Retd.) and Head, Dept. of CSE and IT**).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Akshat Gupta, 141241

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Dr. Satya Prakash Ghrera

Professor, Brig (Retd.) and Head, Dept. of CSE and IT

Dated: 27-11-2017

Acknowledgement

It is our privilege to express our sincerest regards to our project supervisor **Prof. Dr. Satya Prakash Ghrera**, for their valuable inputs, able guidance, encouragement, whole-hearted cooperation and direction throughout the duration of our project.

We deeply express our sincere thanks to our Head of Department **Prof. Dr. Satya Prakash Ghrera** for encouraging and allowing us to present the project on the topic "DATA COMPRESSION" at our department premises for the partial fulfillment of the requirements leading to the award of B-Tech degree.

At the end I would like to express my sincere thanks to all my friends and others who helped me directly or indirectly during this project work.

Date: 27-11-2017 Akshat Gupta(141241)

Table of Contents

List of Figures

List of Graphs:

Chapter-1

INTRODUCTION

1.1 Introduction

Data compression is a process that reduces the dimensions of the statistics or the number of bits by removing immoderate facts and redundancy. Now, the query arises as to why a shorter records collection is extra suitable?

The answer is easy; it lessens the storage and transmission. Data compression is a commonplace compulsion for most laptop applications. Data compression has an important standing in the discipline of report garage and the distributed gadget. Data compression is used in the multimedia discipline, textual content documents and database tables.

Data compression techniques may be classified in numerous approaches. One of the most crucial class standards is whether the compression algorithms cast off a number of the statistics that cannot be retrieved all through decompression. The algorithm that eliminates some of the data is called lossy data compression. And the algorithm that plays the same element that we compressed after decompression is known as lossless records compression. The lossy data compression algorithm is generally used while best consistency with the authentic records is not wished after decompression.

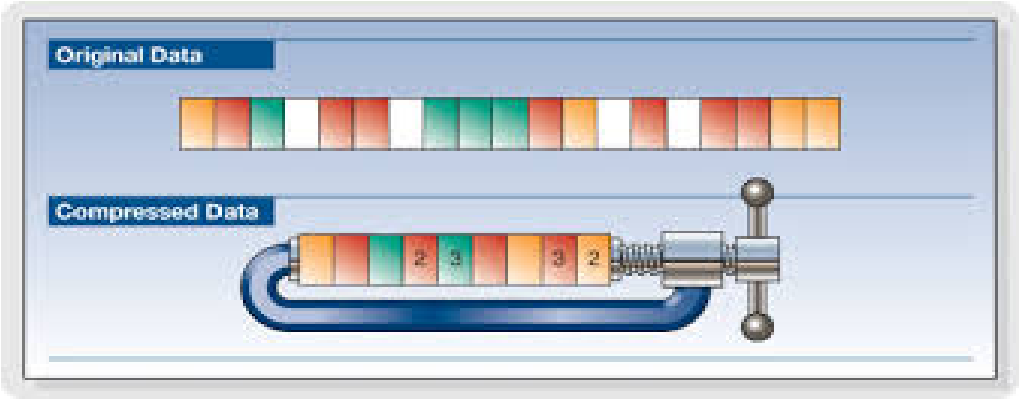


Fig 1.1: Data compression

Instance of lossy records compression is compression of video or picture statistics. Lossless records compression is used in textual content report, database tables and in medical photo due to the fact regulation of policies.various lossless facts compression set of rules have been proposed and used. Some of predominant techniques are HuffmanCoding, Run-Len Encoding, Arithmetic Encoding and Dictionary primarily based Encoding. on this document we take a look at distinct algorithms and supply contrast among them in line with their performances.

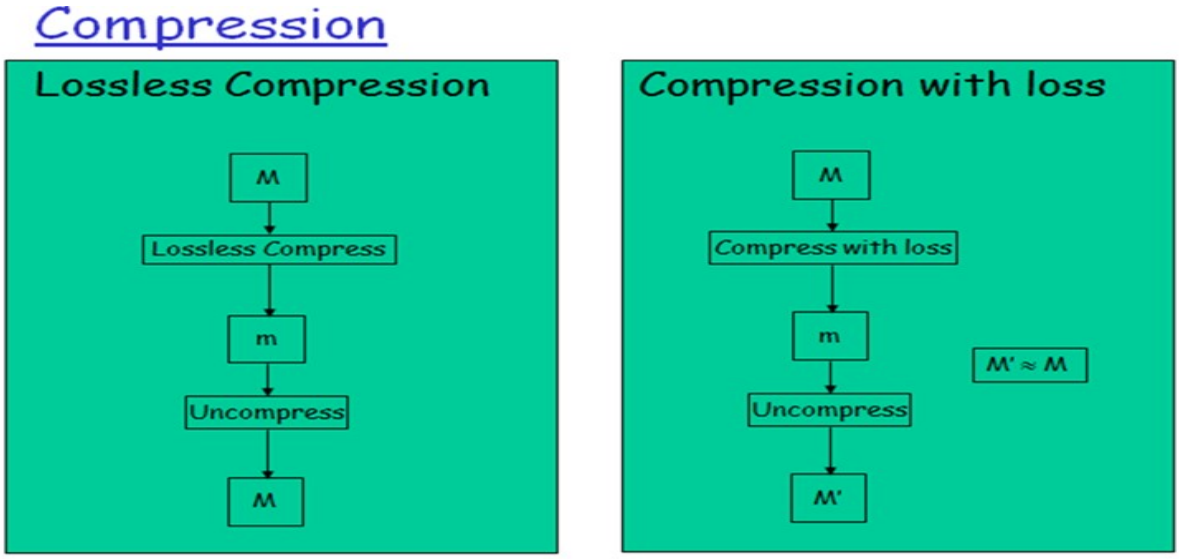


Fig 1.2: Data Compression types

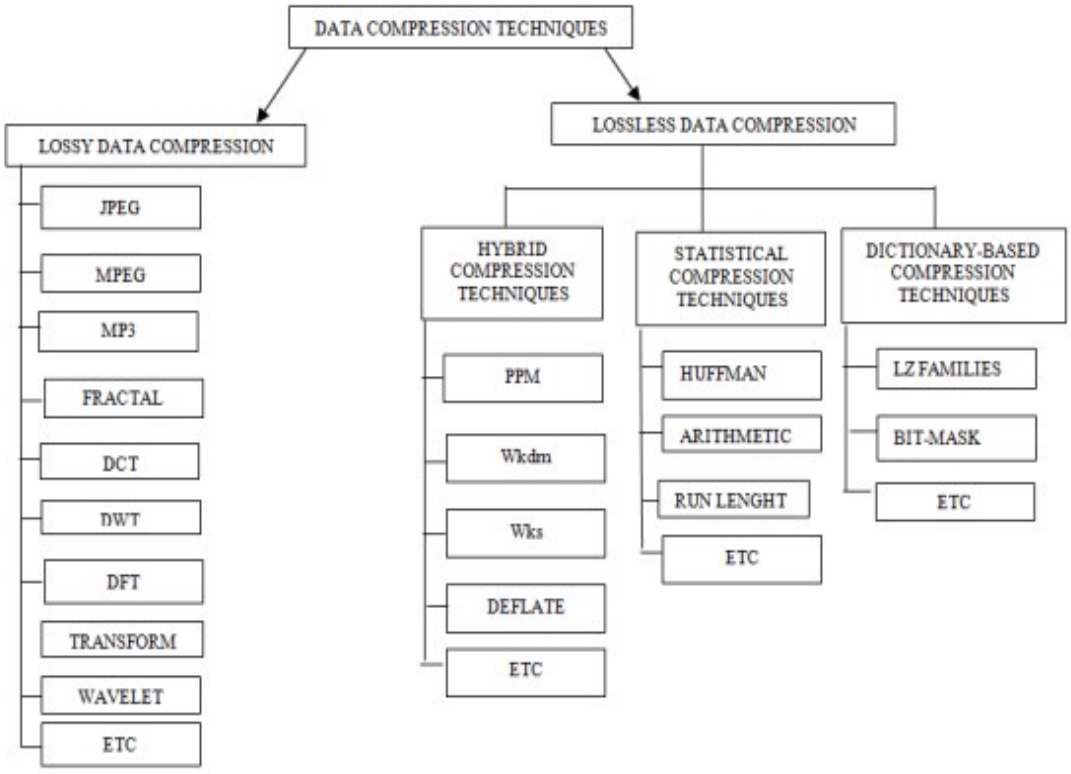


Fig 1.3: Data Compression techniques

Compression is used just about everywhere. all the pictures you get at the internet are compressed, usually in the JPEG/GIF codecs, mostly modems usually use compression, HDTV can be compressed using MPEG-2, and several report structures automatically compress files whilst saved and the rest folks do it by way of hand. The neat factor about compression, as with the other subjects we can cowl in this route, is that the algorithms used in the actual international make heavy use of a huge set of algorithmic equipment, along with sorting, hash tables, attempts, and FFTs. furthermore, algorithms with sturdy theoretical foundations play a important function in actual-global packages.

we will use the universal term message for the items we need to compress, which may be both files or messages. The mission of compression includes two components- an encoding algorithm, and a deciphering set of rules for reconstruction. those two additives are typically intricately tied collectively considering that they each must recognize the shared compressed representation. We distinguish among lossless algorithms, which can re-create the authentic message precisely.

Lossless algorithms are usually used for textual content, and lossy for pictures and sound wherever a little bit of loss in decision is often undetectable. Lossy problems in an abstract sense, but, and does now not mean random lost pixels, however instead way lack of a quantity which includes a frequency element. As an instance, one may suppose that lossy textual content compression could be unacceptable because they're imagining missing or switched characters. Don’t forget rather a device that reworded sentences into a greater fashionable form, or replaced words with synonyms in order that the document can be higher compressed. Technically the compression would be lossy since the textual content has modified, however the “that means” and clarity of the message is probably absolutely maintained, or even stepped forward.

Is there a lossless algorithm that can compress all messages? There was at least one patent application that claimed with a view to compress all documents (messages)—Patent five,533,051 which has the title “Methods for Data Compression”. The patent application claimed that if it changed into implemented recursively, a file will be decreased to almost not anything. With a bit concept you ought to persuade yourself that this is not possible, at the least if the source messages can incorporate any bit-series. we are able to see this by means of a easy counting argument. let’s bear in mind all 1000 bit messages, as an example. There are 21000 exceptional messages we can send, each which desires to be enormously identified with the aid of the decoder. It ought to be clear we can’t represent that many special messages via sending 999 or fewer bits for all the messages — 999 bits could simplest permit us to send 2999 distinct messages. The fact is if any person message is shortened by way of an set of rules, then some different message wishes to be lengthened.

It is, in truth, feasible to go in addition and display that for a set of input messages of constant period, if one message is compressed, then the average length of the compressed messages over all feasible inputs is usually going to be longer than the authentic enter messages. Bear in mind, as an instance, the 8 viable 3 bit messages. If one is compressed to 2 bits, it isn't difficult to convince your self that two messages will need to amplify to 4 bits, giving a mean of 31/8 bits. Lamentably, the patent become granted.

As referred to inside the advent, coding is the task of taking chances for messages and producing bit strings primarily based on these possibilities. How the possibilities are generated is part of the model thing of the set of rules, that's mentioned in later.
In practice we generally use chances for components of a larger message as opposed to for the whole message, e.g., every individual or phrase in a textual content. To be regular with the terminology in the previous phase, we can consider every of these components a message on its very own, and we can use the time period message sequence for the larger message made up of these additives. In trendy every little message may be of a extraordinary type and are available from its very own possibility distribution. as an example, when sending an photo we'd ship a message specifying a shade observed by messages specifying a frequency component of that colour.

Even the messages specifying the coloration might come from one-of-a-kind possibility distributions for the reason that chance of precise colors mightdepend at the context.We distinguish among algorithms that assign a completely unique code (bit-string) for every message, andones that “combination” the codes collectively from multiple message in a row. within the first class we can take into account Huffman codes, that are a form of prefix code. within the later class, we recall mathematics codes and dictionary primarily based methods. AC can yield better compression, but can require the encoder to increase in time in sending the messages. Dictionary based totally compression works exceptional for text and monochrome pictures but the problem is the files which can be compressed however that do not incorporate any repetitive information at all may even grow bigger. Dictionary primarily based compression is a fairly antique compression approach. All latest computer structures have the horsepower to use greater efficient algorithms but is still used.

Records stress is a device for encoding makes a decision that lets in extensive lower inside the mixture quantity of bits to store or transmit a file. The extra facts being managed, the greater it fees concerning stockpiling and transmission charges. to put it it appears that evidently, statistics Compression is the technique of encoding facts to much less bits than the primary illustration so it consumes less garage space and less transmission time even as conveying greater than a system.statistics compression algorithms are classified in approaches i.e. lossy and lossless records compression algorithm. A compression set of rules is applied to alternate over facts from a easy to make use of arrangement to at least one superior for smallness. In like manner, an uncompressing device offers lower back the records to its specific shape.

1.2 Problem Statement

The fundamental issue/problem of lossless compression is to decompose a records set (as an instance, a textual content file or an image) into a sequence of events, then to encode the activities the usage of as few bits as feasible. The idea is to assign quick codewords to extra probably events and longer code words to much less probable activities. Data can be compressed whenever some activities are much more likely than others. Statistical coding techniques use estimates of the possibilities of the events to assign the codewords. Given a set of mutually distinct events e_1, e_2, e_3, _ _ _ _ , e_n, and an correct assessment of the probability distribution P of the occasions, Shannon proved that the the smallest possible predicted number of bits had to encode an occasion is the entropy of P, denoted with the aid of

$$H(P) = \sum_{i=1}^{n} -p\{e_i\} \log_2 p\{e_i\}$$

where $p\{e_i\}$ is the probability that e_i event occurs. An optimal code outputs $\log_2 p$ bits to encode an event whose probability of occurrence is p. Pure arithmetic codes supplied with accurate probabilities provide optimal compression. In theory, arithmetic codes assign one "codeword" to each possible data set. Shorter codes means larger subintervals and thus highly probable input data sets.

In practice, the sub-interval is refined one by one using the probablity of the events, with bits being output. Arithmetic codes almost always give better compression than prefix codes, but they lack the direct correspondence between the events in the input data set and bits or groups of bits in the coded output file.

A statistical coder should work in co-occurrence with a modeler that estimates the chance of every feasible event at every factor in the coding. The probability model need not describe the process that generates the data; it merely has to provide a PD for the data items. The probabilities do not even have to be particularly accurate, but the more accurate they are, the better the compression will be.

If the calculations are made wrong, the file may even be expanded rather than compressed, but the authentic datum can still be recreated. To obtain maximum compression of a file, we need both a good probability model and an efficient way of representing (or learning) the probability model.

Lossless compression is applied while it's miles vital that the primary statistics and the decompressed facts be indistinguishable. Lossless content material information compression calculations generally abuse factual extra in this kind of direction so as to speak to the sender's statistics all the greater in brief without a blunder or any form of lack of vital records contained interior of the content material records information. when you consider that most of the people of this present fact records has actual excess, thusly lossless facts compression is attainable.

Living proof, In English content, the letter "an" is a notable deal greater basic than the letter 'z',and the likelihood that the letter "t" might be trailed with the aid of the letter "z" is little. So this kind of repetition can be evacuated utilising lossless compression. Lossless compression techniques can be labeled by type of statistics they're meant to percent. Compression calculations are basically utilized for the compression of content material, photographs and sound.

most lossless compression initiatives utilize various kinds of calculations: one that creates a authentic model for the information statistics and every other which maps the records information to bit strings making use of this version as part of such a route, to the point that as frequently as possible skilled information will supply shorter yield than unbelievable(much less non-stop) information. The upside of lossless strategies over lossy structures is that Lossless compression consequences are in a closer illustration of the first information data. The execution of calculations may be thought about making use of the parameters, as an instance, Compression Ratio and Saving percentage.

1.3 Objective

Our objective is to put into effect diverse information compression algorithms and examine the consequences acquired to maximize the compression ratio and reduce the compression time.

1.4 Methodology

so as to check the performance of lossless compression algorithms, the mathematics Encoding algorithm, Run period Encoding set of rules, Huffman Encoding algorithm and Lempel Zev Welch algorithm are carried out and examined with a set of textual content files. Performances are evaluated by way of computing the special measuring factors:

Measuring the performance of RLE set of rules:
since the Run period Encoding set of rules does now not use any statistical technique for the compression system, the Compression and Decompression times, document Sizes, Compression Ratio and Saving percent are calculated. several files with special document sizes and text styles are used for computation.

Measuring the overall performance of Huffman approach:
Adaptive Huffman Encoding is likewise applied on the way to compare with other compression and decompression algorithms. record sizes, compression and uncompression times, code performance and entropy are calculated for Huffman algorithm.

Measuring the performance of LZW set of rules:
in view that this algorithm is not based totally on a statistical version, entropy and code efficiency are not calculated. Compression and decompression method, report sizes, compression ratio and saving possibilities are calculated.

Measuring the overall performance of arithmetic Encoding set of rules:
The compression and decompression times and record sizes are calculated for this algorithm. because of the underflow problem, the unique document can't be generated after the decompression procedure.

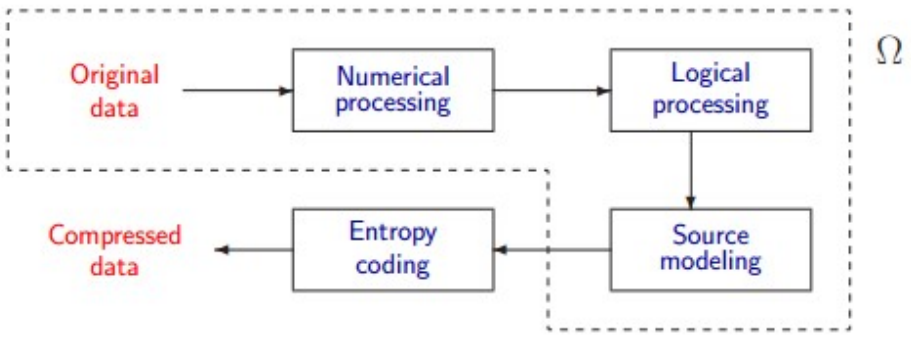
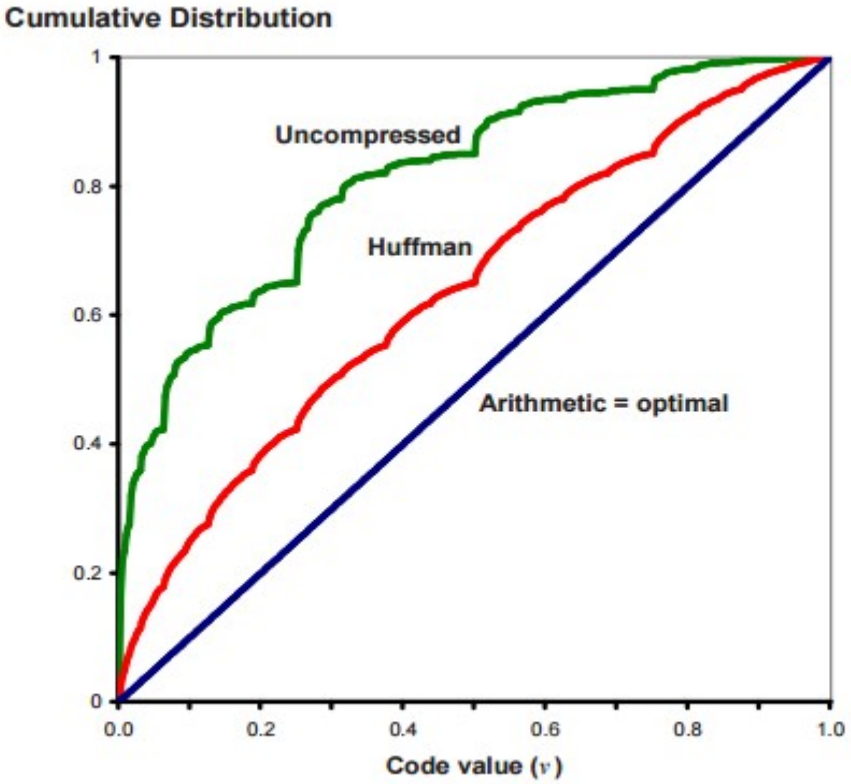


Fig 1.4: System with typical processes for data compression (single data source Ω).



Graph 1: Cumulative distribution of code values generated by different coding methods.

Chapter#2

LITERATURE-SURVEY

2.1 Arithmetic Coding:

Data compression techniques are classified in keeping with lack of facts into businesses, namely lossless data compression strategies and lossy data compression techniques. Lossless algorithms reconstruct the authentic message precisely from the compressed message, and lossy algorithms only remake an approximate instance of the message. Lossless algorithms are usually used for text, for photographs and sound in which a little bit of loss in resolution pitch or other is frequently undetectable, or as a minimum customary. Lossy is utilized in an summary feel, however, and does not mean random misplaced pixels, but instead way loss of a amount which includes frequency aspect, or perhaps loss of noise Lossless compression techniques are used to compress, of necessity, scientific photographs, text and images preserved for legal reasons, machine executable documents, amongst others. Lossy compression techniques reconstruct the unique message with loss of a few records. It isn't always possible to re-make the original message using the deciphering method, and is called irreversible compression. The decompression method produces a probable reconstruction, which can be desirable where information of some degrees that could not be identified with the aid of the human brain help can be ignored. Such techniques may want to be used for multimedia pictures, video and audio to attain extra compact data compression.

Lossless data compression is used to compact documents or data right into a smaller manner. It is regularly used to package up software program before it's miles dispatched over the net or downloaded from a web site to lessen the quantity of time and bandwidth required to transmit the facts. Lossless data compression has the constraint that once records are uncompressed, it must be identical to the unique information that turned into compressed. pix, audio, and video compression which include JPG, MP3, and MPEG alternatively use lossy compression schemes which throw away a number of the authentic facts to compress the documents even in addition. We can be focusing at the lossless kind. There are usually classes of lossless compressors: dictionary compressors and statistical compressors. Dictionary compressors (together with Lempel-Ziv based totally algorithms) build dictionaries of strings and replace whole companies of symbols. The statistical compressors expand fashions of the data of the enter information and use those fashions to control the final output.

Arithmetic coding is a statistical compression technique that uses estimates of the chances of activities to assign code phrases. ideally, brief code phrases are assigned to greater

probably occasions and longer code phrases are assigned to less in all likelihood occasions. Theoretically, arithmetic codes assign one ”code phrase” to every possible statistics set. The mathematics coder need to work collectively with a modeler that estimates the possibilities of the activities in the coding. To gain right compression, a very good chance version and green manner of representing the chance model are required. The fashions can be adaptive, semi-adaptive, or non-adaptive. Adaptive models dynamically estimate the in all likelihood of each event based on previous occasions. Semi-adaptive models use a preliminary bypass of the statistics to collect some data, and non-adaptive models use constant probabilities for all facts. a bonus of mathematics coding is the separation of coding and modeling because it permits the complexity of the modeler to alternate while not having to adjust the coder. The drawback is this is runs more slowly and is greater complex to put in force than LZ primarily based algorithms.

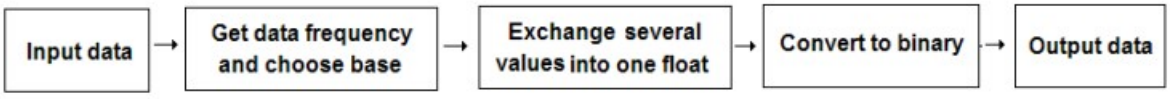
There are three foremost classes of lossless information compression techniques: the ones using statistical fashions, those that require the use of a dictionary, and those that use both statistical and dictionary-based totally techniques. Dictionary-based compression schemes tend to be used greater for archiving packages (on occasion in conjunction with different techniques), at the same time as actual-time situations generally require statistical compression schemes. this is because dictionary-based totally algorithms have a tendency to have slow compression speeds and fast decompression speeds even as statistical algorithms tend to be similarly speedy all through compression and uncompression. Statistical compression schemes determine the output primarily based on the opportunity of prevalence of the input symbols and are usually utilized in real-time packages. The algorithms have a tendency to be symmetric (the decoder mirrors the steps of the encoder); therefore, compression and decompression commonly require the identical quantity of time to finish.. Dictionary-based compression techniques are typically utilized in archiving applications including compress and gzip because the interpreting process has a tendency to be faster than encoding. Hybrid compression methods share traits with both statistical and dictionary-primarily based compression techniques. these algorithms normally contain a dictionary scheme in a scenario in which simplifying assumptions may be made about the enter facts. Lossless facts compression set of rules consists of: Limphel Ziv own family (Dictionary-based totally encoding), Run-period Encoding compression (Statistical coding), Huffman Encoding (Statistical coding), arithmetic Encoding (Statistical coding), Bitmask coding (Dictionary-primarily based). determine 1-1 depicts information compression strategies.

The Huffman method assigns an vital quantity of bits to each symbol, whilst arithmetic coding assigns one lengthy code to the complete input string. mathematics coding has the ability to compress facts to its theoretical restriction. arithmetic coding combines a statistical model with an encoding step, which includes a few mathematics operations. The maximum simple statistical version could have a linear time complexity of N[log(n)+a] +Sn where N is the overall variety of enter symbols, n is the current number of specific symbols, a is the mathematics to be executed, and S is the time required, if important, to maintain internal facts shape. This converts the complete enter statistics right into a unmarried floating factor number. A floating factor variety is much like quite a number with a decimal factor, like 4.five in preference to 41/2. however, in mathematics coding we are not dealing with decimal quantity so we name it a floating point rather than decimal factor. the premise for records compression is the mathematical price of information. facts contained in a image x is given through

$L(x) = \text{Log}_2 \frac{1}{p(x)}$

This cost also describes the range of bits vital to encode the symbol. This definition reinforces the perception of data. First, the extra probably the occurrences of a image, the fewer bits are used to symbolize it. Conversely, the least frequent symbols provide extra data by using their occurrence. Secondly, if there are n equally possibly messages, log2n bits may be required to encode every message. this is the statistics value of each message

$L(x) = \text{Log}_2 \frac{1}{p(x)} = \log_2 n$



Graph 2: Arithmetic Data compression Flow Graph

Let's take an example we have string:

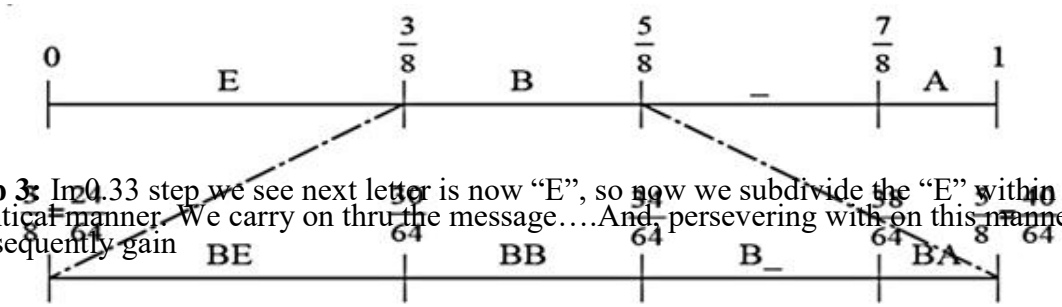
BE_A_BEE

And now use the AC algorithm.

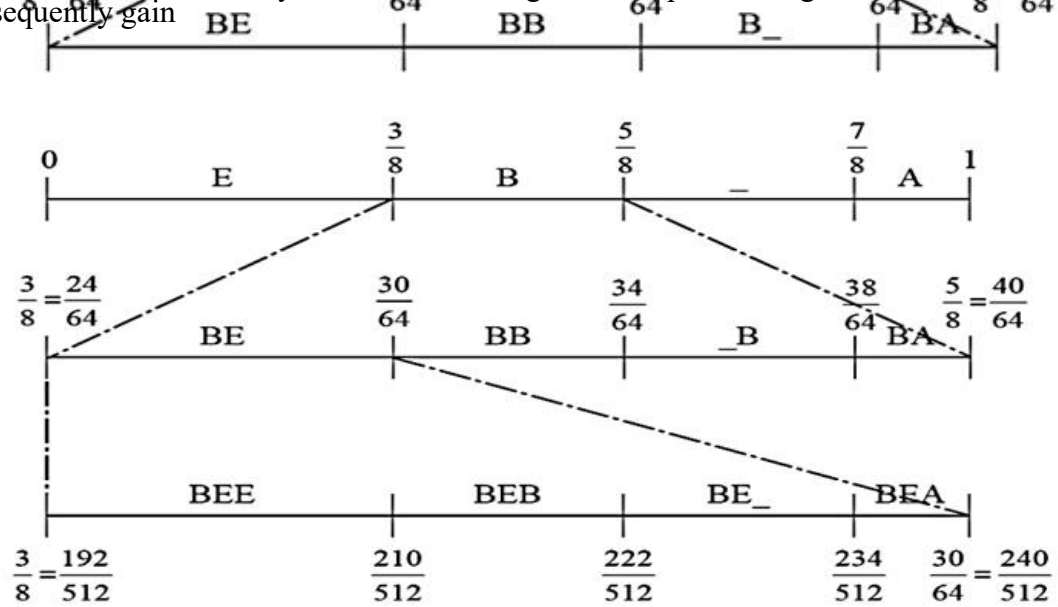
Step 1: in the first step we do is look at the frequency count for the different letters:

E	B	_	A
3	2	2	1

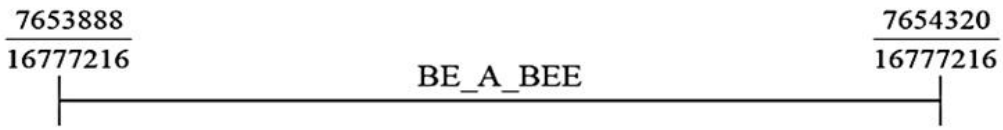
Step 2: In second step we encode the string through dividing up the period [0, 1] and allocate every letter an intervalwhose length depends on how regularly it matter in the string. Our string begin with a "B", so we take the „B" c language and divide it up again inside the identical way:



Step 3: In 0.33 step we see next letter is now "E", so now we subdivide the "E" within the identical manner. We carry on thru the message….And persevering with on this manner, we subsequently gain



And so on, we obtain:



So we represent the message as any number in the interval.

[7653888/16777216,7654320/16777216]

$7653888 = (7*10^6) + (6*10^5) + (5*10^4) + (3*10^3) + (8*10^2) + (8*10) + 8$

Binary numbers are almost precisely the equal, only we address powers of two in preference to energy of 10. The rightmost digit of binary wide variety is unit (as before) the one to its left offers the variety of 2s, the next one the number of 4s, and shortly.So

110100111 = $(1*2^8) + (1*2^7) + (0*2^6) + (1*2^5) + (0*2^4) + (0*2^3) + (1*2^2) + (1*2^1) + 1 = 256 + 128 + 32 + 4 + 2 + 1 = 423$ in denary (i.e. base 10).

New Character	Low value	High Value
	0.0	1.0
B	0.2	0.3
I	0.25	0.26
L	0.256	0.258
L	0.2572	0.2576
SPACE	0.25720	0.25724
G	0.257216	0.257220
A	0.2572164	0.2572168
T	0.25721676	0.2572168
E	0.257216772	0.257216776
S	0.2572167752	0.2572167756

(a)

Encoded Number	Output Symbol	Low	High	Range
0.2572167752	B	0.2	0.3	0.1
0.572167752	I	0.5	0.6	0.1
0.72167752	L	0.6	0.8	0.2
0.6083876	L	0.6	0.8	0.2
0.041938	SPACE	0.0	0.1	0.1
0.41938	G	0.4	0.5	0.1
0.1938	A	0.2	0.3	0.1
0.938	T	0.9	1.0	0.1
0.38	E	0.3	0.4	0.1
0.8	S	0.8	0.9	0.1
0.0				

(b)

Fig 2.1 (a) Encoding of “BILL GATES” (b) Decoding of “BILL GATES

2.1.1 Practical Implementation:

The procedure of encoding and decoding a circulation of symbols using mathematics coding isn't always too complex. but before everything glance, it seems completely impractical. most computers support floating point numbers of up to eighty bits or so. Does this suggest you need to begin over whenever you finish encoding 10 or 15 symbols? Do you want a floating point processor? Can machines with specific floating point codecs speak using mathematics coding?

As it seems, arithmetic coding is quality done the use of widespread 16-bit and 32-bit integer math. No floating point math is required, nor wouldn't it assist to use it. rather, we use an incremental transmission scheme wherein constant-length integer-nation variables get hold of new bits at the low stop and shift them out the high give up, forming a unmarried quantity that can be as long as the range of bits to be had on the laptop's garage medium.

Within the preceding section, I showed how the algorithm works by retaining music of a excessive and coffee quantity that bracket the variety of the possible output wide variety. while the algorithm first starts up, the low range is about to 0.0, and the high quantity to one. To paintings with integer math, first change the 1.0 to zero.999...., or.111 ... in binary.

My implementation makes use of 16-bit unsigned math, so the preliminary value of high is 0xFFFF, and occasional is 0. We recognize that the excessive cost maintains with FFs forever, and coffee keeps with 0s for all time, so we can shift the ones more bits in with impunity while they're wanted.

if you believe our bill GATES instance in a 5-digit register, the decimal equivalent of our setup might look like discern 7(a). To locate our new variety numbers, we need to apply the encoding set of rules from the preceding phase. We first calculate the range between the low and excessive values. The difference between the 2 registers could be a hundred thousand, now not 99999, due to the fact assuming the excessive sign in has an infinite range of 9s delivered on to it, we want to increment the calculated distinction. We then compute the new excessive value the usage of the components from the preceding phase: high = low + high range (image).

In this example the excessive variety became .30, which gives a new value for excessive of 30000. Before storing the new cost of high, we want to decrement it, once again because of the implied digits appended to the integer price. So the new fee of excessive is 29999.

The calculation of low follows the same path, with a resulting new price of 20000. So now excessive and coffee appear like this:

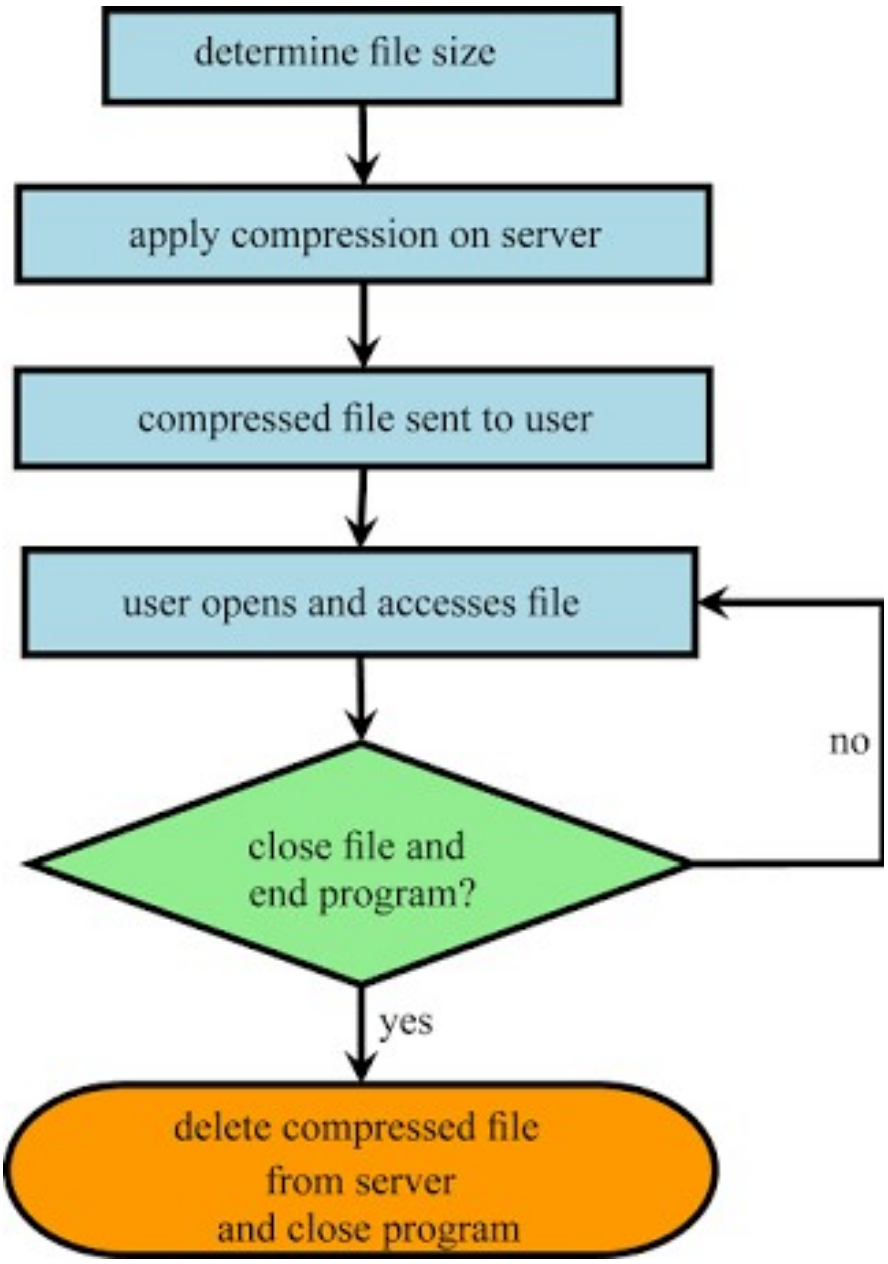
high: 29999 (999...)

LOW: 20000 (000...)

At this factor, the maximum substantial digits of high and low fit. due to the nature of our set of rules, high and coffee can continue to grow in the direction of one another without pretty ever matching. because of this after they match inside the maximum giant digit, that digit will by no means change. So we will now output that digit as the first digit of our encoded wide variety. this is completed by way of shifting each excessive and coffee left through one digit, and shifting in a nine inside the least considerable digit of high. The equivalent operations are executed in binary inside the C implementation of this algorithm.

As this manner keeps, high and occasional are continually growing closer together, then transferring digits out into the coded word. The manner for our "invoice GATES" message looks as if this:

	HIGH	LOW	RANGE	CUMULATIVE OUTPUT
Initial state	99999	00000	100000	
Encode B (0.2-0.3)	29999	20000		
Shift out 2	99999	00000	100000	.2
Encode I (0.5-0.6)	59999	50000		.2
Shift out 5	99999	00000	100000	.25
Encode L (0.6-0.8)	79999	60000	20000	.25
Encode L (0.6-0.8)	75999	72000		.25
Shift out 7	59999	20000	40000	.257
Encode SPACE (0.0-0.1)	23999	20000		.257
Shift out 2	39999	00000	40000	.2572
Encode G (0.4-0.5)	19999	16000		.2572
Shift out 1	99999	60000	40000	.25721
Encode A (0.1-0.2)	67999	64000		.25721
Shift out 6	79999	40000	40000	.257216
Encode T (0.9-1.0)	79999	76000		.257216
Shift out 7	99999	60000	40000	.2572167
Encode E (0.3-0.4)	75999	72000		.2572167
Shift out 7	59999	20000	40000	.25721677
Encode S (0.8-0.9)	55999	52000		.25721677
Shift out 5	59999	20000		.257216775
Shift out 2				.2572167752
Shift out 0				.25721677520



Data Compression at a glance

2.1.2 Underflow:

This scheme works properly for incrementally encoding a message. there's enough accuracy retained in the course of the double precision integer calculations to make certain that the message is accurately encoded. However, there's potential for a lack of precision beneath certain occasions.

In the occasion that the encoded phrase has a string of 0s or 9s in it, the excessive and coffee values will slowly converge on a cost, however may not see their most huge digits in shape without delay. For example, excessive and coffee may additionally appear to be this:

Excessive: 700004
Low: 699995

At this factor, the calculated variety goes to be best a digit lengthy, this means that the output phrase will no longer have enough precision to be as it should be encoded. Even worse, after some more iterations, high and low may want to appear like this:

High: 70000
Low: 69999

At this factor, the values are permanently caught. The variety among excessive and occasional has come to be so small that any calculation will continually return the identical values. but, because the maximum full-size digits of both phrases are not equal, the algorithm cannot output the digit and shift. It looks as if a deadlock.

The way to defeat this underflow problem is to save you matters from ever getting this terrible. The original algorithm stated some thing like "If the maximum vast digit of high and occasional fit shifts it out". If the two digits do not fit, but at the moment are on adjacent numbers, a 2nd takes a look at wishes to be implemented. If high and occasional are one aside, we then check to peer if the 2d maximum sizable digit in excessive is a zero, and the 2nd digit in low is a nine. in that case, it manner we're on the road to underflow, and need to do so.

When underflow rears its ugly head, we head it off with a slightly one of a kind shift operation. in preference to moving the maximum great digit out of the phrase, we just delete the 2nd digits from excessive and coffee, and shift the rest of the digits left to refill the space. The most giant digit stays in area. We then must set an underflow counter to remember that we threw away a digit, and we aren't pretty positive whether or not it was going to turn out to be as a zero or a 9. The operation looks like this:

	Before	After
High:	40344	43449
Low:	39810	38100
Underflow:	0	1

After every recalculation operation, if the maximum good sized digits do not fit up, we are able to test for underflow digits again. If they may be gift, we shift them out and increment the counter.

While the MSD do eventually converge to a single cost, we first output that value. Then, we output all of the "underflow" digits which have been previously discarded. The underflow digits may be all 9s or 0s, relying on whether excessive and espresso converged to the higher or decrease cost. in the C implementation of this set of rules, the underflow counter keeps song of how many ones or zeros to put out.

Mathematics coding is an foremost entropy coding technique because it gives quality compression ratio and commonly achieves better effects than Huffman Coding. it is quite complicated as compared to the alternative coding techniques. while a string is converted in to mathematics encoding, the characters having most probability of occurrence will be stored with fewer bits and the characters that do not occur so frequently might be stored with extra bits, ensuing in fewer bits used standard. Arithmetic coding converts the flow of enter symbols into a unmarried floating point number as output. unlike Huffman coding, mathematics coding does no longer code every symbol one by one. each symbol is rather coded by way of considering all prior data. as a result a records flow encoded on this fashion need to usually be study from the start. therefore, random get entry to isn't feasible.

The primary implementation of arithmetic coding described above has important difficulties: the shrinking contemporary c program language period calls for using excessive-precision mathematics, and no output is produced till the entire report has been study. The maximum clear-cut method to both of those troubles is to output every leading bit as soon as it's miles recognised, and then to double the length of the contemporary c program language period in order that it reflects most effective the unknown part of the final c program language period. Witten, Neal, and Cleary add a clever mechanism for stopping the contemporary c program language period from shrinking too much when the endpoints are close to half however straddle half of. if so we do now not but know the subsequent output bit, however we do know that something it is, the subsequent bit could have the opposite price; we simply preserve song of that fact, and enlarge the modern interval symmetrically approximately half. This observe-on procedure may be repeated any range of times, so the cutting-edge period length is continually strictly longer than 1/four.

Mechanisms for incremental transmission and glued precision arithmetic have been advanced by way of Pasco , Rissanen , Rubin , Rissanen and Langdon , Guazzo , and Witten, Neal, and Cleary. The bit-stuffing idea of Langdon and others at IBM that limits the propagation of contains in the additions serves a function just like that of the follow-on system defined above.

COMPRESSION METHOD	ARITHMETIC	HUFFMAN
Compression ratio	Very good	Poor
Compression speed	Slow	Fast
Decompression speed	Slow	Fast
Memory space	Very low	Low
Compressed pattern matching	No	Yes
Permits Random access	No	Yes

Fig 2.2: Comparison between arithmetic and Huffman coding methodologies

2.2 Huffman Encoding :

Huffman coding is an entropy encoding algorithm for lossless statistics compression. In this set of rules fixed length codes are replaced with the aid of variable length codes. while the use of variable-length code phrases, it is suited to create a prefix code, fending off the need for a separator to determine codeword barriers. Huffman Coding uses such prefix code.

Huffman process works as we observe:

1. Symbols with excessive frequency are expressed using shorter encodings than symbols which arise less regularly.

2. The two symbols that arise least often will have the same length. The Huffman algorithm makes use of the greedy approach i.e. at each step the set of rules chooses the pleasant to be had alternative. A binary tree is constructed up from the bottom up. to peer how Huffman Coding works, allow's take an instance. expect that the characters in a report to be compressed have the subsequent frequencies:

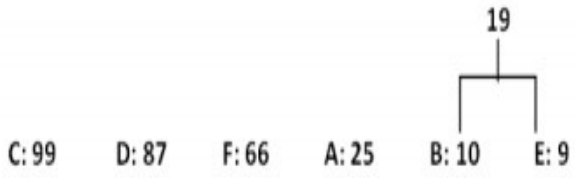
A: 25 B: 10 C: 99 D: 87 E: 9 F: 66

The processing of constructing this tree is:

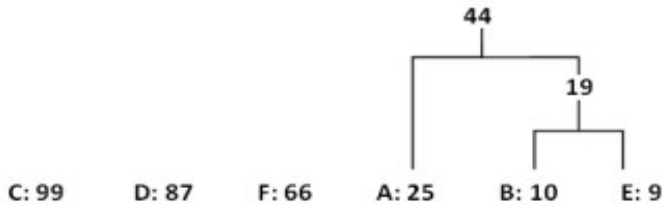
1. Make a list of leaf nodes for every image and make up the nodes in the order from in the order of descend.

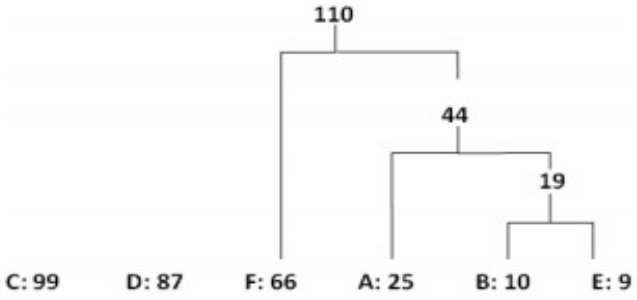
C:99 D:87 F:66 A:25 B:10 E:9

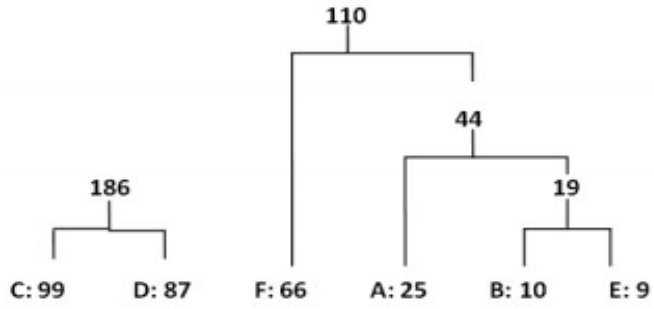
2. Pick out leaf nodes with the lowest frequency. Create a parent node with those two nodes and assign the frequency same to the sum of the frequencies of two infant nodes.

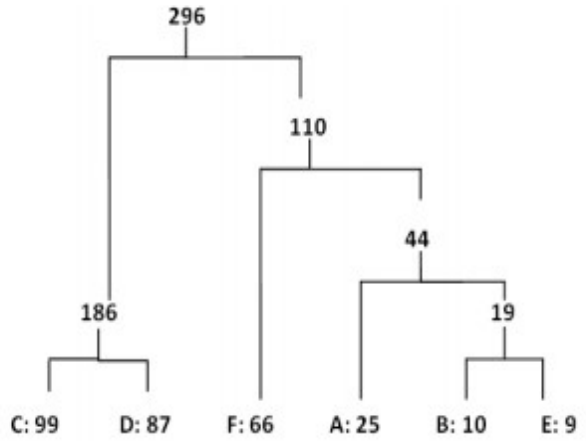


Now upload the determine node inside the list and dispose of the two baby nodes from the listing. And repeat this step until you have most effective one node left









3. Now label every aspect. The left child of every parent is categorized with the digit 0 and right toddler with 1. The code phrase for each source letter is the sequence of labels alongside the direction from root to the leaf node representing the letter.

Huffman Codes are proven beneath within the table:

C	00
D	01
F	10
A	110
B	1110
E	1111

2.3 Run Length Encoding Algorithm:

RLE is a statistics compression set of rules that is supported with the aid of maximum bitmap document codecs, consisting of TIFF, BMP, and PCX. RLE is desirable for compressing any form of statistics irrespective of its data content, however the content of the information will affect the compression ratio executed by using RLE. although maximum RLE algorithms cannot reap the excessive compression ratios of the greater superior compression strategies, RLE is each smooth to implement and brief to execute, making it a terrific alternative to either using a complicated compression set of rules or leaving your photo records uncompressed.

In practice, an encoded run may additionally contain 1 to 128 or 256 characters; the run be counted generally carries because the wide variety of characters minus one (a fee in the variety of zero to 127 or 255). the second one byte is the cost of the man or woman within the run, that is inside the variety of zero to 255, and is referred to as the run value.

Uncompressed, a char-run of 15A chars would normally be requiring 15 bytes to store:
AAAAAAAAAAAAAAA

The same string after RLE shall need only two bytes:
15A

The 15A code created to show the character group is known as an RLE packet. right here, the first byte, 15, is the run rely and carries the wide variety of repetitions. the second byte, A, is the run price and contains the real repeated cost in the run.

RLE schemes are easy and speedy, but their compression performance depends at the sort of image facts being encoded. A black-and-white image this is mainly white, along with the page of a ebook, will encode thoroughly, because of the big quantity of contiguous statistics this is all the same colour. An photograph with many colours this is very busy in appearance, however, along with a photograph, will not encode very well. that is due to the fact the complexity of the picture is expressed as a huge quantity of different colors. And due to this complexity there may be notably few runs of the identical coloration.

2.4 LZW Data Compression:

If you had been to take a look at nearly any information record on a pc by individual, you will observe that there are numerous recurring styles. LZW is a statistics compression method that takes gain of this occurrence-loop. Like any adaptive/dynamic compression method, the concept is to

(1) Begin with an initial version,

(2) examine statistics piece by way of piece,

(3) and update the version and encode the facts as you move along.

LZW is a "dictionary"-based totally compression set of rules. Which means in place of tabulating individual counts and constructing bushes (as for Huffman encoding), LZW encodes statistics by way of referencing a dictionary. for this reason, to encode a substring, simplest a unmarried code wide variety, similar to that substring's index inside the dictionary, needs to be written to the output record. even though LZW is regularly explained within the context of squeezing textual content files, it could be used on any sort of report. however, it generally performs first-rate on documents with repeated substrings, inclusive of textual content documents.

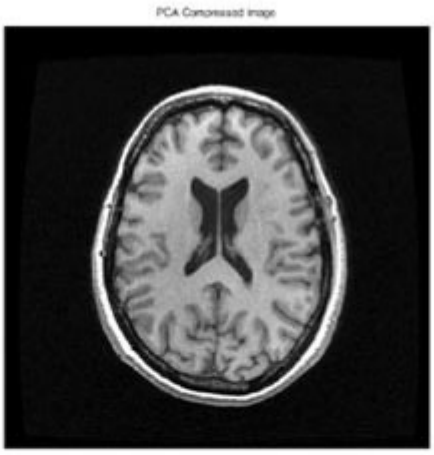
Image Compression with Principal Component Analysis

The increase in picture transmission extended throughout the arena. therefore, the scale of image has an essential role so that you can transmit the picture in lesser time and within the allotted bandwidth. This results in the requirement of better approach for lowering the picture size. this can be completed by way of the usage of the photograph compression technique which focuses to get rid of the redundancy occurs within the photograph in a manner that it ought to not have an effect on the photograph reconstruction. there are numerous researches suggests their own photo compression method to meet the wishes. All method has its own advantages and disadvantages. This paper focuses on the usage of the most important element analysis (PCA) method which is widely recognized for its better capability for dimensionality reduction. To deal with the trouble of large covariance matrix in PCA, 2-Dimensional (2DPCA) is used in this paper. 2DPCA without delay calculates the eigenvectors of the covariance matrix without matrix-to-vector conversion. This photograph compression approach is built using VLSI structure. The simulation result suggests that the proposed technique results.

Image Compression with principal component Analysis is a regularly occurring utility of the measurement discount technique. bear in mind from a preceding publish that hired singular cost decomposition to compress an photo, that an photo is a matrix of pixels represented by RGB color values. consequently, primary thing evaluation may be used to reduce the dimensions of the matrix (image) and venture the ones new dimensions to reform the photo that keeps its features while it is lesser in size in k-weight. We can use PCA to encode the image of a anything behind it. because the quantity of essential additives used to assignment the new information increases, the first-class and representation compared to the unique image enhance.

Image compression with PCA is a beneficial and comparatively straightforward utility of the technique by imaging an image as a (n instances p) or (n times n) matrix manufactured from pixel shade values. there are many other real-world applications of PCA, inclusive of face and handwriting recognition, and other situations when managing many variables .

Original image: TIFF with 512x512 pixels*

Recovered image (512x512 pixels) from 512 principal components


Memory necessary (final data)=512x512=262144 units of memory
Compression factor (ρ)=262144/262144=1
Compression rate (1-ρ)=1-1=0
Mean squared error (MSE)=0

Original image: TIFF with 512x512 pixels*

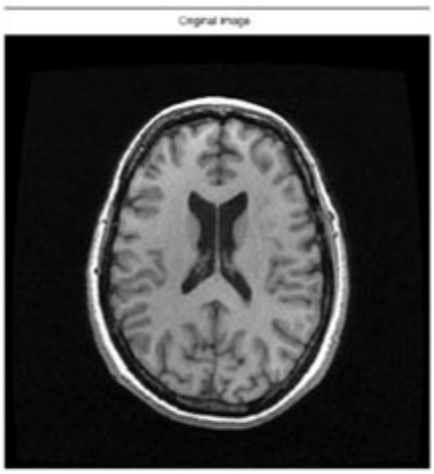
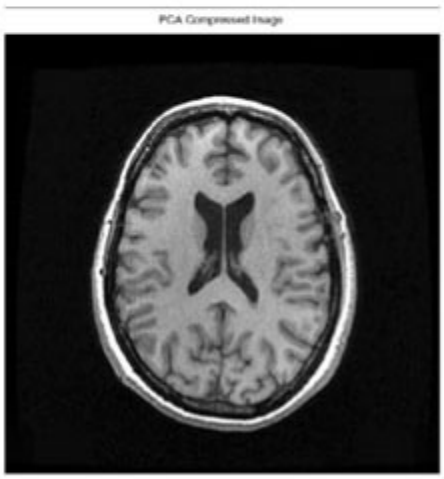


Image recovered (512x512 pixels) from 112 principal components



Memory necessary (final data)=112x512=57344 units of memory
Compression factor (ρ)=57344/262144=0.219
Compression rate (1-ρ)=1-0.219=0.781
Mean squared error (MSE)=0213

Original image: TIFF with 512x512 pixels*

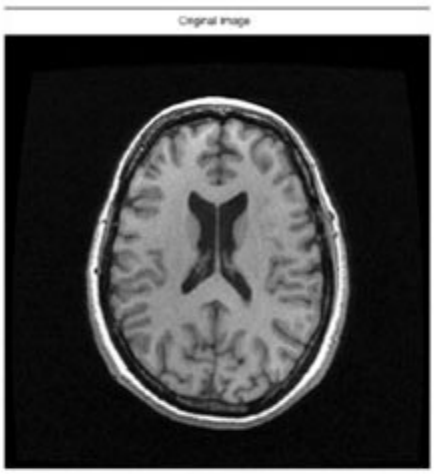
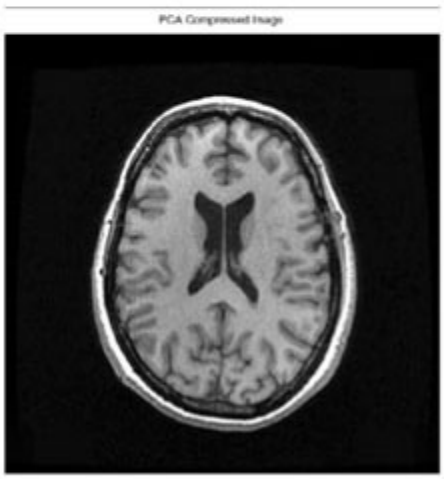



Image recovered (512x512 pixels) from 112 principal components


Memory necessary (final data)=112x512=57344 units of memory
Compression factor (ρ)=57344/262144=0.219
Compression rate (1-ρ)=1-0.219=0.781
Mean squared error (MSE)=0213

The number of components in compression affects the remaking of the original image from the compressed image. This technique allows savings of storage space at a high level, which can be important in healthcare applications and in processing huge chunks of data.

Facts contained in a set of records is saved in a computational structure with reduced dimensions primarily based on the quintessential projection of the records set onto a subspace generated through a device of orthogonal axes(three). The premiere machine of axes can be received the use of the Singular Values Decomposition (SVD) technique(4). The decreased dimension computational structure is selected in order that relevant data characteristics are identified with little lack of records(three). this type of discount is superb in numerous instances: for picture compression, information illustration, calculation reduction important in subsequent processing, and goes on.

Use of the PCA approach in facts measurement reduction is justified with the aid of the easy representation of multidimensional statistics, using the records contained inside the facts covariance matrix, ideas of linear algebra(three) and simple statistics. The studies finished by using Mashal et al.(five) followed the PCA method inside the choices of pix from a multimedia database. in line with Smith(6), PCA is an actual photo compression set of rules with minimum lack of information.

The PCA approach permits the identification of standards in records and their expression in one of these way that their similarities and variations are emphasised. as soon as styles are discovered, they can be compressed, i.e., their dimensions may be reduced without a whole lot loss of data. In summary, the PCA method can be used as a virtual picture compression algorithm with a low stage of loss.

Table 1. Format of data for a Principal Component Analysis from n observations of variables X_1 and X_p

Case	X_1	X_2		X_p
1	a_{11}	a_{12}	...	a_{1p}
2:	a_{21}	a_{22}	...	a_{2p}
...	...	...	...	...
...	...	...	...	...
n	a_{n1}	a_{n2}	...	a_{np}

Discussion at a glance

In some programs, including brain feature pictures, the important precept is the variation of the resonance signal over time. In those situations, the spatial records may be maintained in a reference document, making it viable to compress next photos with no loss. however, it is nevertheless essential to evaluate the pertinence of the application of excessive compression charges whilst an evaluation of systems of reduced dimensions relative to the scale of the voxels is wanted.

Furthermore, the remark of the outcomes from the software of the PCA technique in scientific photos can be taken into consideration a complexity measure. In different words, pictures with dense texture patterns generally tend to supply exclusive results with the use of the approach defined.

New secondary programs (primarily based on the outcomes right here described) may additionally encompass diverse situations within the medical habitual. these packages enjoy the methods described earlier. on this way, the comprehension of the principles right here provided is essential for the better use of clinical applications based totally on those foundations.

```
library(jpeg)
cut <- readJPEG('cut.jpg')
ncol(cut)
## [1] 300
nrow(cut)
## [1] 497

a <- cut[,,1]
s <- cut[,,2]
d <- cut[,,3]

cut.a.pca <- prcomp(a, center = FALSE)
cut.s.pca <- prcomp(s, center = FALSE)
cut.d.pca <- prcomp(d, center = FALSE)

asd.pca <- list(cut.a.pca, cut.s.pca, cut.d.pca)

for (j in seq.int(3, round(nrow(cut) - 10), length.out =
11)) {
  pca.img <- sapply(asd.pca, function(j) {
    compressed.img <- j$x[,1:j] %*% t(j$rotation[,1:i])
  }, simplify = 'arrray')
```

```
  writeJPEG1(pca.img,
paste01('compressed/cut_compressed_', round(i,0),
'_components.jpg', sep = ''))
}
```

Chapter-3

SYSTEM DEVELOPMENT

3.1 Arithmetic Coding

3.1.1 Compression algorithm:

```
begin
count frequency of input symbols
output(symbol's frequency)
interval I := new interval 0..9999
split I according frequency of symbols
readSymbol(X)
while (X!=EOF) do
begin
if (MSB(Low) == MSB(High) then
begin
output(MSB)
in case of need output discarded digits
shift left High and Low
end
else
begin
if (underflow danger) then
begin
shift left High and Low from second position
end
end
new I := interval I accordant with X
split I according frequency of symbols
readSymbol(X)
end
output(remainder)
end
```

3.1.2 Decompression algorithm

- Range = (high - low) + 1 See where the number lands
- Temp = ((code - low) + 1) * scale) - 1) / range)
- See what symbols corresponds to temp.
- Range = (high - low) + 1 Extract the symbol code
- High = low + ((range * high_values [symbol]) / scale) - 1
- Low = low + (range * high_values [symbol - 1]) /
- Loop.
- Msb of high = msb of low?
- Yes
    - Go to shift
- No
    - Second msb of low = 1 and Second msb of high = 0 ?
    - Yes
        - Code = code ^ 4000h
        - Low = low & 3FFFh
        - High = high | 4000h
        - go to shift
    - No
        - The routine for decoding a symbol ends here.

Shift:

- Shift low to the left one time. Now we have to put in low, high and code new bits
- Shift high to the left one time, and or the lsb with the value 1
- Shift code to the left one time, and or it the next bit in the input
- Repeat to the first loop. 3.1.3 Model development:

The want to correctly expect the probability of symbols inside the enter statistics is inherent to the nature of arithmetic coding. The precept of this form of coding is to lessen the quantity of bits had to encode a character as its possibility of appearance increases. So if the letter "e" represents 25 percentage of the input information, it might handiest take 2 bits to code. If the letter "z" represents most effective zero.1 percent of the enter records, it might take 10 bits to code.

If the model isn't always producing chances accurately, it'd take 10 bits to symbolize "e" and a couple of bits to symbolize "z," causing data expansion instead of compression.
the second one situation is that the model wishes to make predictions that deviate from a uniform distribution. The more the model is well built at predicting, the better the CR can be.

Simplest by using correctly locating chances that deviate from a uniform distribution can the variety of bits be decreased, leading to compression. Of direction, the elevated possibilities should accurately mirror reality, as prescribed by using the primary situation.
it could appear that the chance of a given image going on in a information movement is fixed, however this is not quite actual. depending at the version getting used, the opportunity of the individual can trade quite a chunk. for example, whilst compressing a C software, the opportunity of a newline individual within the text is probably 1/40. This probability might be decided through scanning the complete textual content and dividing the quantity of the character's occurrences by way of the total range of characters. however if we use a modeling method that looks at a unmarried preceding person, the possibilities alternate. if so, if the preceding man or woman changed into a "", the opportunity of a newline character is going as much as half. This advanced modeling technique leads to better compression, despite the fact that each fashions were producing correct chances.
The maximum green technique for computing distributions relies upon at the statistics type. when we are coping with completely unknown statistics we may want model to work in a totally automatic manner. In other instances, we will use some information of the facts houses to reduce or eliminate the version effort. underneath we give an explanation for the capabilities of a number of the maximum not unusual strategies for estimating distributions.
• Use a constant distribution this is available earlier than encoding and decoding, generally anticipated by way of amassing statistics in a large wide variety of ordinary samples. This approach may be used for sources which include English textual content, or climate records, however it not often yields the great outcomes because few records resources are so simple as to be modeled with the aid of a unmarried distribution. furthermore, there is very little flexibility (e.g., data for English text do now not match well Spanish textual content). on the other hand, it can paintings properly if the source version is very particular, and in reality it's miles the most effective alternative in some very complex models in which significant records can best be collected from a totally massive quantity of records.
• Use pre-described distributions with adaptive parameter estimation. as an instance, we will count on that the facts has Gaussian distribution, and estimate only the imply and variance of every image. If we allow just a few values for the distribution parameters, then the encoder and decoder can create numerous vectors with all the distribution values, and use them consistent with their commonplace parameter estimation.
• Use -skip encoding. a first bypass gathers the data of the supply, and the second one bypass codes the records with the collected statistics. For interpreting, a scaled version of vectors p or c should be covered at the start of the compressed facts. as an instance, a ebook can be archived (compressed) collectively with its unique symbol information. it's miles feasible to lessen the computational overhead via sharing approaches between passes. for example, the primary bypass can simultaneously accumulate information and convert the records to run-lengths.

• Use a distribution primarily based on the incidence of symbols previously coded, updating c with each image encoded. we can begin with a very approximate distribution (e.g., uniform), and if the chances alternate often, we can reset the estimates periodically. This approach, defined within the next segment, is quite powerful and the maximum convenient and flexible. however, the constant update of the cumulative distribution can boom the computational complexity drastically. An opportunity is to replace handiest the possibility vector p after every encoded symbol, and update the cumulative distribution c less regularly.

3.1.4 Computational analysis:

Arithmetic coding is widely known for its optimality, and the fact that it is able to be a totally flexible and powerful device for coding complex information sources [1, 2, 4, 6, 10]. at the equal time, practitioners additionally realize that it had now not been more typically used due to its excessive computational complexity. If we don't forget the many years of studies on techniques for decreasing its complexity, it may appear that there is little wish for brand new breakthroughs and for its tremendous adoption. however, new outcomes display that, in reality, the evolution of arithmetic coding is following an unusual course, and the maximum promising opportunity is to transport to the only purpose. This takes place because most of the price-reduction strategies for arithmetic coding had been evolved for the hardware that turned into available 10 or greater years in the past, whilst multiplications and divisions had been too gradual for coding functions.

Currently even cheaper processors can carry out unique arithmetic very speedy. Taking the fact that arithmetic is any such essential venture of any processor; we can anticipate even more blessings within the future. for this reason, there may be a want to identify what are the mathematics coding responsibilities so that it will remain virtually important in determining the computational complexity. using this information we need to be able to find out a way to higher exploit the processor's mathematics abilities for quicker coding. The sources of mathematics coding computational complexity encompass [1, 2]:

• period replace and arithmetic operations

• symbol interpreting (interval seek)

• chance estimation (supply modeling)

Our approach is to degree the performance of several implementations, changing one parameter at time, or at the least as few as viable. This way we will examine the significance of every of the tasks referred to above, and additionally select the first-rate techniques. at the same time as the main objective of this paper is to assess the difference in complexity of a variety of duties and techniques, in some graphs we upload effects from a few well-known implementations, due to the fact they could provide an absolute reference (we explain while the comparisons aren't fair). To avoid redundancy, in this file we do no longer present all of the details of our implementation and analysis, considering that maximum of it may be discovered in references [1, 2] (which also affords a far extra entire presentation, and set of references on arithmetic coding concept and practice). however, the reader have to be conscious that there's a considerable amount of programming for each test: we had to write specific applications to check all the vital duties, in more than 10 distinctive complete implementations of arithmetic coding.

because these types of moves can be tightly incorporated in a single implementation, it's been tough to honestly identify the overall performance bottlenecks. on this work we tackle this trouble by means of doing an extensive comparative evaluation. Our strategy is to degree the overall performance of several implementations, changing one parameter at time, or at least as few as viable. This way we are able to evaluate the importance of each of the tasks stated above, and also pick the high-quality strategies.

even as the main objective of this paper is to assess the difference in complexity of a diffusion of responsibilities and techniques, in a few graphs we upload outcomes from a few well-known implementations, due to the fact they could provide an absolute reference (we explain whilst the comparisons are not truthful). To avoid redundancy, in this document we do not gift all the info of our implementation and evaluation, considering that maximum of it is able to be located in references [1, 2] (which additionally affords a much greater whole presentation, and set of references on arithmetic coding idea and practice). however, the reader need to be aware that there's a large amount of programming for every test: we needed to write particular packages to test all of the essential tasks, in not less than 10 unique complete implementations of the rules of arithmetic coding. Even though some strategies for complexity reduction we present here are not new, we agree with that is the primary time that their use for mathematics coding is reported in this form of complexity comparisons. in addition, we believe that the sequential separation of the analysis of the exclusive duties, with the identification of the maximum applicable implementation to be used in other exams enabled a much better expertise of the complexity issues.

For instance, we affirm that in modern-day processors the bit-based totally renormalization is an awful lot slower than byte-primarily based renormalization [16]. In exercise, this means that no meaningful analysis of other much less important resources of complexity can be finished using bit-primarily based renormalization, considering that its results are so large as to mask the whole lot else. After replacing it with byte-based renormalization we observed that we should without delay become aware of the subsequent maximum essential supply of complexity, and surely show the blessings of using specific techniques.

In truth, this overlaying impact is another crucial motivation for this work. There are some previous examinations of mathematics coding complexity that may be out of date now not most effective because of new processors. For instance, because the exams finished by way of Moffat et al. [6, 8] used bit-based renormalization; we bear in mind that they needed to be revisited and the conclusions up to date. Our experimental consequences affirm our expectation that the techniques that yield the satisfactory effects are the ones which can fully take advantage of the processor's hardware abilities. In truth, the concept of changing one complex mission with numerous easy ones is shown to be counterproductive in numerous instances. This actually demonstrates, for example, the sturdy obstacles on throughput of mathematics coding techniques that work simplest on binary symbols. similarly, we display that we will have full-size pace profits for the renormalization, decoder image seek, and code edition obligations. It became discovered that our quickest adaptive version has speeds on larger alphabets which might be approaching Huffman coding, with the benefit of being less difficult to implement, and being plenty more versatile.

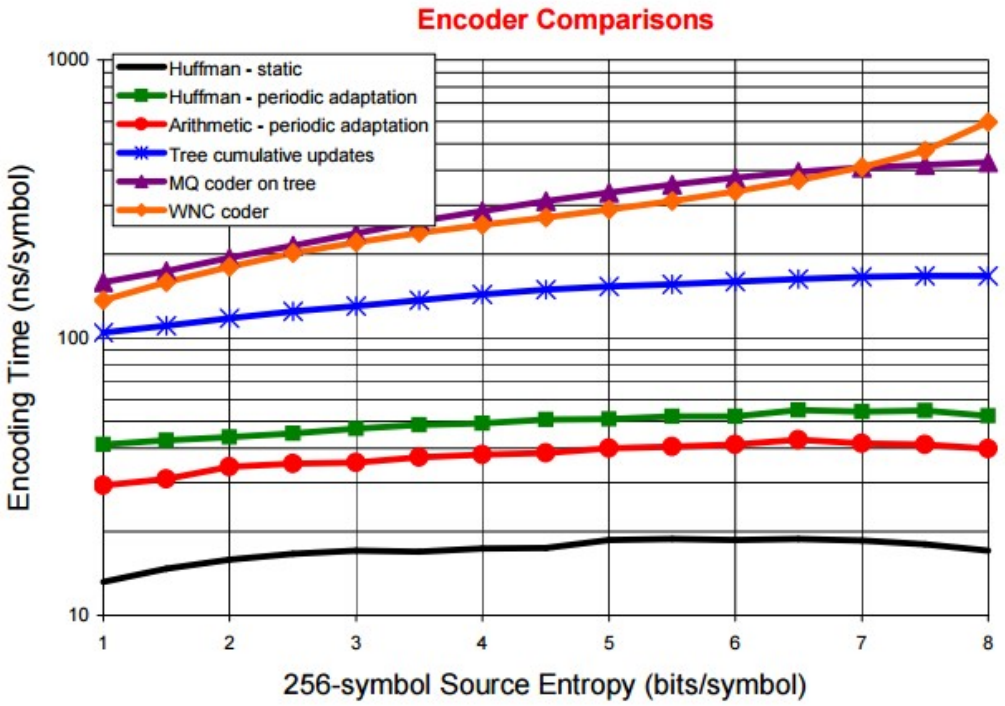
Conversely, the strategies that try to decompose the coding process in many simple steps are irrevocably out of date. In fact, we had been surprised to peer that bit-primarily based techniques are already one or two orders of magnitude slower, and we count on that the distinction in speed will maintain growing swiftly.

determine 3-1 shows the assessment of the encoder consequences. be aware that the vertical axis right here has a logarithmic scale. the primary two graphs display that static Huffman coding continues to be honestly the quickest entropy-coding set of rules, however, at the identical time, despite periodic version, the adaptive version of Huffman coding may be considerably slower. The encoding instances of our fastest implementation are proven subsequent. We are able to see that it's far truly slower than static Huffman, but corresponding to Huffman with periodic updates. The current velocity may be ideal considering the optimum compression and more convenience for modeling unknown resources.

For Instance: a easy, static version of input data:

- 60% probability of symbol 'a' -> the set would become [0, 0.6)
- 20% probability of symbol 'b' -> the set would become [0.6, 0.8)
- 10% probability of symbol 'c' -> the set would become [0.8, 0.9)

10% probability of symbol END-OF-DATA. -> the set would become [0.9, 1)



Graph 3: Encoding time of different algorithms compared

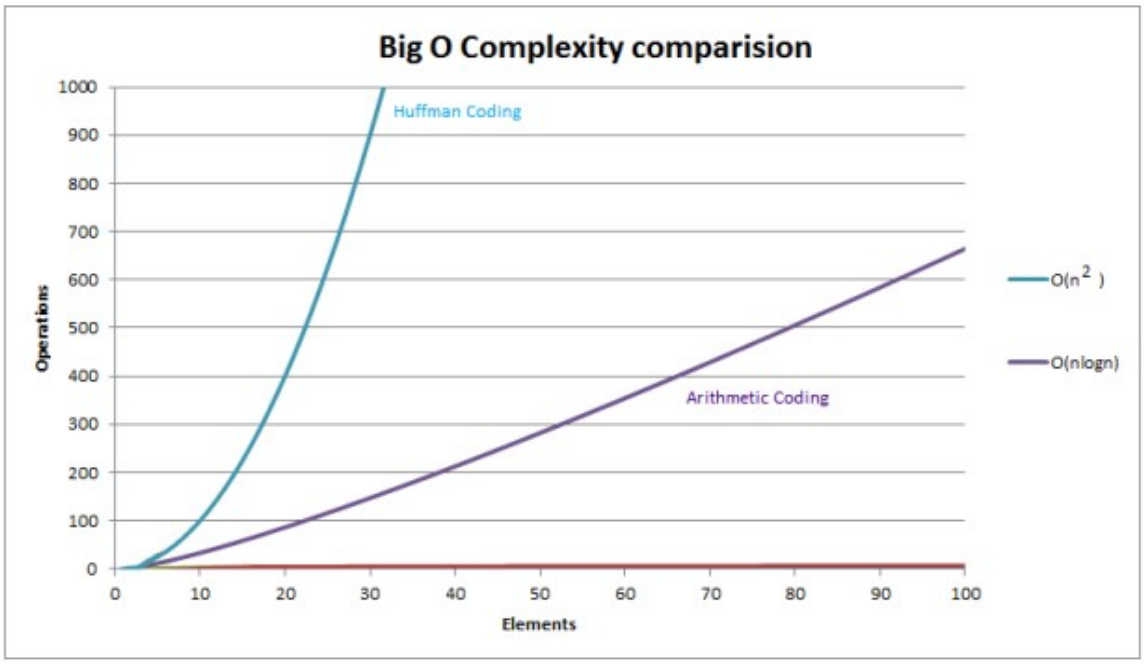
The presence of EOD image way that the flow could be 'internally terminated', as is fairly common in data compression; the primary and most effective time this symbol appears within the data movement, the decoder will understand that the entire circulate has been decoded.

The encoder has essentially just three pieces of records to keep in mind: the following image that needs to be encoded, the cutting-edge period, the probabilities of symbols. due to this, is pretty clean to modify the algorithm to adaptive model.

The encoder divides the cutting-edge c language into sub-periods, every representing a fraction of the contemporary period proportional to the probability of that picture. Whichever c language corresponding to the actual symbol this is subsequent to be encoded will become the c program language period used inside the next step.

Whilst all the symbols are encoded, the output interval identifies, unambiguously, the collection of symbols that produced it. Everyone who has the very last interval and the model used can remake the image series that must were entered the encoder to result in that very last period.

It is not necessary to transmit the final period, however; it is handiest essential to transmit one fraction that lies inside that period. Specially, it is nice vital to transmit sufficient digits (in whatever base) of the fraction so that everyone fractions that begin with the ones digits fall into the very last length. Memory complexity depends on some of distinctive enter symbols, at maximum O(n), wherein n is period of a message.



Graph 4: Big O Complexity comparison of huffman and arithmetic Coding

3.2 Huffman set of rules:

3.2.1 Compression algorithm:

There are mainly essential elements in Huffman Coding

1) build a Huffman Tree from enter characters.

2) Move across the Huffman Tree and allocate codes to characters.

Steps to build Huffman Tree:

Input is array of unique characters in conjunction with their frequency of occurrences and output is Huffman Tree.

1. Create a (leaf) node for each specific character and create a min-heap of the leaf nodes

2. Take out nodes having the minimal occurrences from the min heap.

3. Create a new inner node with number of occurrences same as the sum of the other 2 nodes' frequencies. Make the first extracted node as its left toddler and the other one as its right infant. Add this node to the min-heap.

4. Repeat steps#2 and #3 until the heap contains simplest one node. The last node is the basis node and the tree is whole.

Steps to print codes from Huffman Tree:

Traverse the tree fashioned beginning from the foundation. hold an auxiliary array. while moving to the left child, write 0 to the array. even as shifting to the proper baby, write 1 to the array. Print the array when a leaf node is encountered.

3.2.1 Decompression algorithm:

To decode the encoded statistics we require the Huffman tree. We iterate thru the binary encoded information. To locate character similar to modern bits, we use following simple steps.

3.2.1 Decompression algorithm:

To decode the encoded data we require the Huffman tree. We iterate through the binary encoded data. To discover the character corresponding to the currently held bits, we use following simple steps.

1. We begin from root and do following till a leaf is discovered.
2. If cutting-edge bit is 0, we circulate to left node of the tree.
3. If the bit is 1, we pass to proper node of the tree.
4. If at some stage in traversal, we encounter a leaf node, we print individual of that specific leaf node and on the other hand continue the new release of the encoded records starting from step 1.

3.2.3 Model development:

Huffman's set of rules, expressed graphically, takes as input a list of nonnegative weights w(1), ... ,w(n) and creates a complete bin- tree [a bin tree is full whenever every single node has number of children 0 or 2] whose leaves are labeled with the weights. while the Huffman set of rules is used to assemble a code, the weights represent the possibilities associated with the supply letters. initially there is a set of singleton timber, one for every weight in the listing. At each step inside the set of rules the timber corresponding to the 2 smallest weights, w(i) and w(j), are merged into a brand new tree whose weight is w(i)+w(j) and whose root has two youngsters which can be the sub bushes represented via w(i) and w(j). The weights w(i) and w(j) are removed from the listing and w(i)+w(j) is inserted into the listing. This process continues till the burden list includes a unmarried value. If, at any time, there is more than one way to pick out a smallest pair of weights, the sort of pair can be selected. In Huffman's paper, the system starts offevolved with a non growing listing of weights. This detail is not vital to the correctness of the set of rules, however it does provide a extra green implementation [Huffman 1952].

3.2.4 Computational analysis:

Huffman coding is the maximum important competitor to mathematics coding. it's miles famous that static Huffman coding can be notably quicker than mathematics coding . then again, variations of Huffman coding that adapt for every coded image are known to be a good deal slower. considering the fact that we will use the equal periodic edition method for Huffman codes, it's far interesting to observe its coding overall performance earlier than evaluating it to mathematics coding.

Huffman encoding may be very rapid because it consists more often than not of a single table appearance-up, plus some instructions for aligning bits and extracting the code bytes. For Huffman coding even the periodic variation can growth complexity notably. This occurs because, despite the fact that the worst-case complexity of computing new Huffman codes is O(M log M), their computation is greater complex than simply sorting symbols and updating cumulative distributions. further, the tables required for instant decoding may be very large, and it might be impractical to update them even in a periodic fashion. Smaller tables can be used, however their computation isn't always as intuitive as for mathematics coding. A fundamental restriction in the software of Huffman coding is that can't be used effi- ciently at charges close to or beneath 1 bit according to symbol. As predicted, the redundancy of the Huffman code at 1 bit in step with symbol is pretty big, however it drops to smaller values moderately speedy. Huffman coding can produce near-most useful compression, but we may additionally need to combination statistics symbols to growth their mixed entropy. The sensible hassle is that it isn't very easy to attain this intention inside the situations wherein the source distribution is unknown, and it's far exactly in those cases that arithmetic coding will

The compression ratios for the selected documents are in the range from 55% to 65%. The compression ratio does not rely upon the file size but it relies upon on the structure of the file. A software source code, with better quantity of repeating words reasons the compression ratio to be too excessive. Compression ratios lay among 58% and 67.To compress a single character of 1 byte, this algorithm desires only four-five bits. Code efficiencies are greater than ninety eight% for all of the cases. for that reason this algorithm may be considered as an efficient set of rules.

3.3.

a) choose the primary person from source string.

b) Append the picked character to the blank spot char group.

c) count number the variation in subsequent occurrences of the chosen char and append the count number to string.

d) pick the next man or woman and repeat steps b) c) and d) if stop of string isn't reached.

3.3.2 version improvement:

RLE is perfect to compress any form of statistics no matter its records data, the information shall affect the CR executed by RLE. Saying along that most RLE rules are not able to reach very high compression ratios of the popular compression techniques, RLE is each clean to enforce and short to execute, making it an excellent opportunity to both the usage of a complicated compression algorithm or leaving your picture statistics uncompressed.

RLE gets in handy by lessening the physical length of a repeating group of characters. This occurrence, also referred to as a run, is generally stored in just two bytes. The primary byte represents the quantity of characters within the run and is referred to as the run be counted. In practice, an encoded run may also incorporate 1 to 128 or 256 characters; the run count number typically incorporates as the range of characters minus one (a value inside the variety of zero to 127 or 255). the second one byte is the fee of the man or woman in the run, that's within the variety of 0 to 255, and is referred to as the run price.

[33]

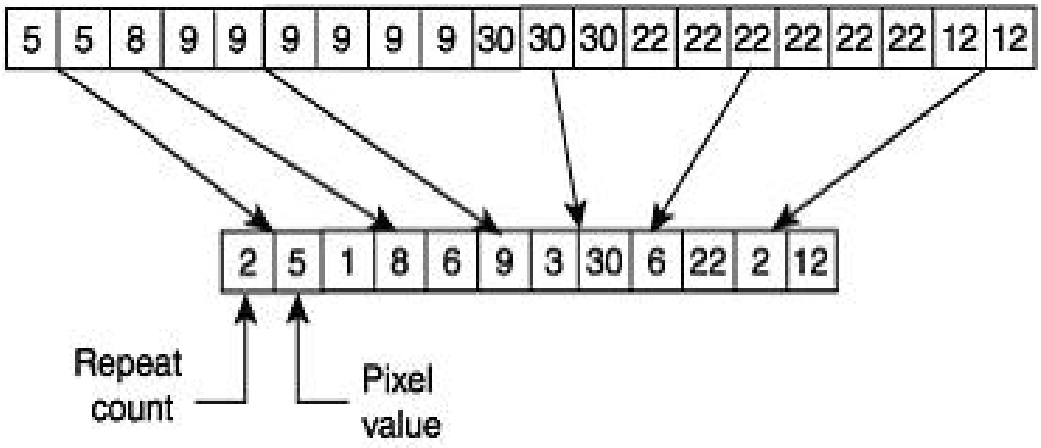
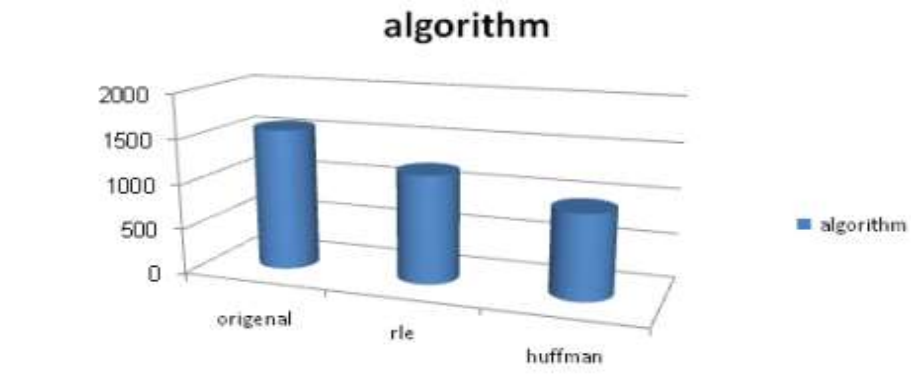


Fig 3.1:Simple RLE

3.3.3 Computational evaluation:

RLE schemes are simple and speedy, but their compression performance relies upon on the form of photo facts being encoded. A black-and-white picture this is in most cases white, which includes the web page of a ebook, will encode very well, because of the large amount of contiguous information that is all of the same shade. A picture with many colours that is very much busy in the 'looks', but, consisting of a image, will not encode very well. that is because the complexity of the photograph is expressed as a big number of different colorings. And due to this complexity there may be surprisingly few runs of the equal colour.

inside the worst case RLE generates the output records that's 2 instances more than the scale of enter information. this is because of the less quantity of runs inside the supply file. And the files which are compressed have very high values of compression ratio.



Comparative sizes of compressed data compared with original data.

[34]

3.4 LZW (Lempel–Ziv–Welch) Compression Technique (Dictionary Based):

3.4.1 Compression algorithm:

```
Start:
Initialize table with single character strings
        P = first input character
        WHILE not end of input stream
        C = next input character
        IF P + C is in the string table
        P = P + C
ELSE
        Output the code for P
        Add P + C to the string table
        P = C
END WHILE
        Output code for P
```

3.4.2 Uncompression algorithm:

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

S = S + C

ELSE

S = translation of NEW

output S

C = first character of S

OLD + C to the string table

OLD = NEW

END WHILE

3.4.3 version improvement:

LZW is a "dictionary"-primarily based compression set of rules. which means that instead of tabulating man or woman counts and constructing bushes (as for Huffman encoding), LZW encodes statistics through referencing a dictionary. hence, to encode a substring, best a single code wide variety, corresponding to

that substring's index within the dictionary, needs to be written to the output report. even though LZW is frequently defined within the context of compressing text documents, it is able to be used on any type of file. but, it generally performs first-rate on files with repeated substrings, such as textual content files.

Compression is performed in following manner: LZW begins out with a dictionary of 256 characters (within the case of 8 bits) and makes use of the ones as the "fashionable" man or woman set. It then reads information 8 bits at a time (e.g., 't', 'r', and so on.) and encodes the statistics because the quantity that represents its index inside the dictionary. whenever it comes throughout a new substring (say, "tr"), it provides it to the dictionary; whenever it comes across a substring it has already visible, it truly reads in a brand new character and concatenates it with the modern-day string to get a new substring. the following time LZW revisits a substring, it'll be encoded the use of a single variety.

The Uncompression system for LZW is also straightforward. similarly, it has a bonus over static compression strategies because of the truth no dictionary or other overhead facts is crucial for the interpreting set of regulations--a dictionary identical to the best created for the duration of compression is reconstructed for the duration of the machine. every encoding and interpreting programs have to start with the equal preliminary dictionary, in this case, all 256 ASCII characters.

3.four.4 Computational evaluation:

A dictionary is used by this set of rules and gives appropriate compression ratios for the deliver text files. The drawback is that the size of the dictionary had been given increased with the dimensions of the document considering the fact that more and more entries are added via the algorithm. It shows low efficiency, while a variety of sources is needed to gadget the dictionary. This set of regulations gives an great compression ratio which lies among 30% and 60%. that is an inexpensive price while it as compared with the opposite algorithms. The compression ratio decreases because the report length will growth, since the form of phrases can be represented with the resource of shorter dictionary entries.

LZW compression works great at the same time as applied on monochrome images and textual content documents that contain repetitive textual content/patterns. as an instance, the use of LZW compression, a checker board image along with repetitive black and white patterns can be compressed upto 70% of its real record length. for this reason an excessive compression ratio may be scored.

No. of Exp.	Original Data Size	LZW	Compression Ratio(Cr)
01	4.70 MB	1.87 MB	2.51
02	2.19 MB	933 KB	2.35
03	26 KB	12 K	2.16
04	71.6 KB	21.9 KB	3.26

Fig 3.2: Encoded data and compression ratio using LZW

Chapter-4
PERFORMANCE ANALYSIS

Performance assessment of compression algorithms may be achieved by different factors. But, the primary concern has constantly been the gap performance and time efficiency.we're the usage of different factors to research the set of rules.4.1 Compression Ratio

Compression ratio, aka compression power, is used to scale the reduction in facts-representation size produced through an information compression algorithm. The data compression ratio is similar to the bodily compression ratio used to measure bodily compression of materials.

Data compression ratio is given as the division between the uncompressed size and compressed size.

$$\text{Compression Ratio} = \frac{\text{Uncompressed Size}}{\text{Compressed Size}}$$

Be aware that this method applies in addition for compression, in which the uncompressed period is that of the unique; and for decompression, where the uncompressed size is that of the reproduction.

Another very popular is the space savings :

$$\text{Space Savings} = 1 - \frac{\text{Compressed Size}}{\text{Uncompressed Size}}$$

$$\text{Compression Ratio} = \frac{\text{Uncompressed Data Rate}}{\text{Compressed Data Rate}}$$

[38]

Instead of space savings, one speaks of data-rate savings:

$$\text{Data Rate Savings} = 1 - \frac{\text{Compressed Data Rate}}{\text{Uncompressed Data Rate}}$$

When the uncompressed information price is thought, the compression ratio can be calculated from the encoded information speed.

PRESENTS A SIMPLE COMPARISION BETWEEN THESE COMPRESSION METHODS

COMPRESSION METHOD	HUFFMAN CODING	ARITHMETIC CODING	LEMPEL ZIV WELCH(BIT REDUCTION METHOD)
Compression ratio	Poor	Very good	Very good
Compression speed	Fast	Slow	fast
Decompression speed	Fast	Slow	fast

4.2 Compression Speed:

Compression pace is associated with the facts format and the system type. the connection among application overall performance and host gadget parameters is a research situation count that is outdoor of the scope of this paper. At some point of the experiments, we hold using the equal machine for all the compressions, and make sure that our software is the only workload. This way, we are able to think about compression velocity as a feature of compression algorithm. The compression speed is likewise affected by compression buffer length, however we omit this element by means of using the same size of buffer, that's 16KB. while evaluating records compression algorithms, pace is constantly in terms of uncompressed facts handled in line with second.

a few programs use facts compression strategies even when they've a lot RAM and disk space that there may be no actual want to make files smaller.

File compression and delta compression are regularly used to speed up copying documents from one stop of a slow connection to another. Even on a single pc, some varieties of operations are drastically faster whilst achieved on compressed variations of statistics rather than immediately at the uncompressed records. Specifically, some compressed report formats are designed so that compressed sample matching -- trying to find a word in a compressed model of a textual content report -- is extensively quicker than trying to find that equal word in the unique uncompressed text report.

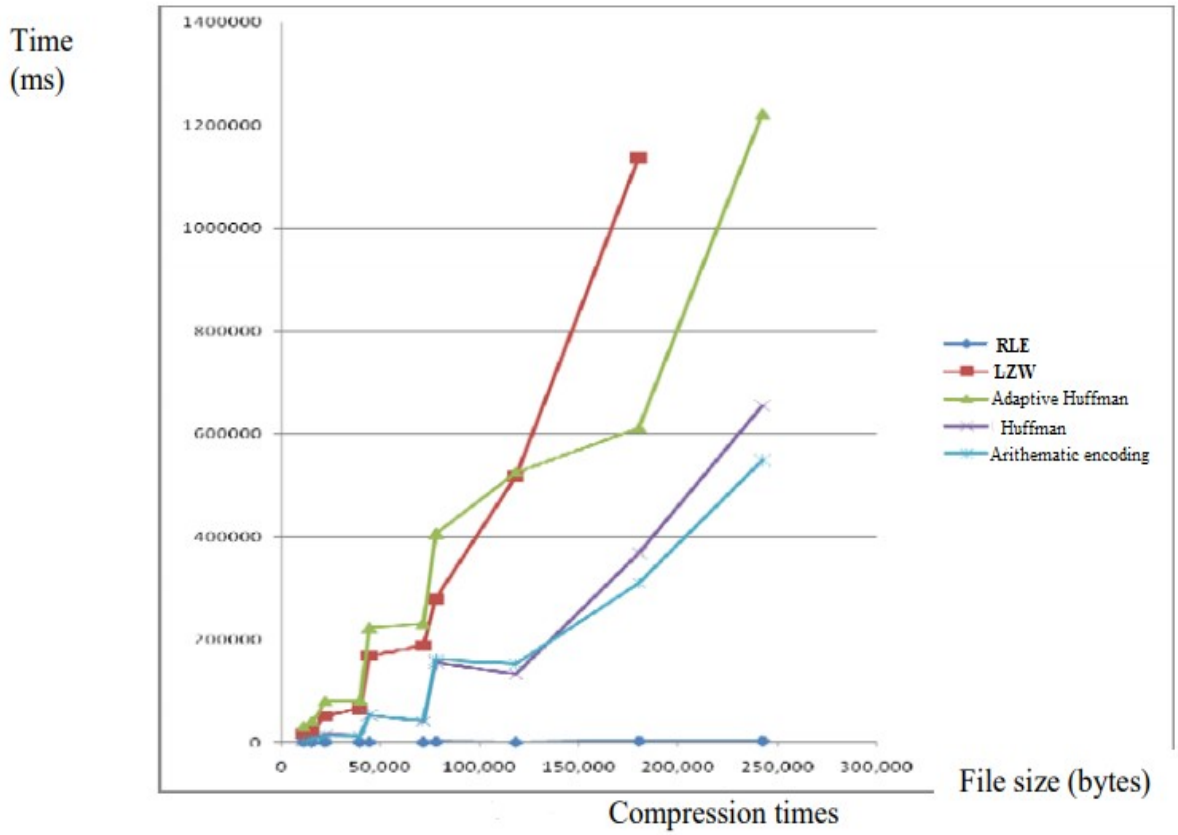
$$\text{speed} = \frac{\text{Uncompressed bits}}{\text{seconds to compress}}$$

The speed varies broadly from one machine to every other, from one implementation to some other. Even at the identical device and identical benchmark record and identical implementation source code, the usage of a different compiler may additionally make a decompressor run quicker.the speed of a compressor is nearly usually slower than the velocity of its corresponding decompressor.

Even with a fast modern CPU, compressed filesystem overall performance is often restrained by means of the speed of the compression algorithm. Many present day embedded systems -- in addition to some of the early computers that statistics compression algorithms were first advanced on -- are heavily constrained by using speed.

4.3 Compression Time:

The time taken with the aid of the set of rules to compress the document is calculated in milliseconds (ms).



Graph 5: Compression versus Time graph for different Algorithms

Chapter-5
CONCLUSION

5.1 Conclusion:

We implemented various algorithms to compression text and images.

Below we list the main conclusions resulting from our experiments.

• There's a substantial speed benefit as we circulate from renormalizations that keep one bit a time, to those that shop bits together in organizations of 1 or greater bytes.

• Byte-primarily based renormalizations need enough precision from the arithmetic operations (e.g., at least 16 or 32 bits) to assist a much wider range of period lengths. generally these can be pleasant supported by way of the native CPU operations, in place of approximations.

• Multiplications are actually sufficiently speedy, and their impact on the coding pace is small even for static binary coders.

• At the same time as binary coders perform all of the coding operations in the shortest time, their records throughput is limited to at most one bit in line with coded image. For fastest coding we ought to use methods that code symbols from larger alphabets because they can yield lots higher throughputs.• Arithmetic decoding can be significantly slower encoding, because of the search for the interval to which the coded symbol belongs. The best solution depends on the processor and data source.

• RLE is used only when series of characters is repeated in many instances. In worst case it generates output 2 instances greater than size of input facts. that is due to fewer amounts of run in source document. This set of rules does no longer provide considerable improvement.

• Huffman vs Arithematic coding: Huffman makes use of a static table for complete coding process, so it's far faster. however it does no longer produce green compression ratio. On opposite Arithematic algorithm can generate a high compression ratio, however its compression pace is sluggish.

• LEMPHEL ZIV WELCH is maximum efficient method among all other technique. It provide higher compression ratio. Its compression pace is fast. it really works nicely with strings and characters. it could compress as much as 50 percent of original size. It used with GIF snap shots and additionally TIFF files. This method also works well with extremely redundant files.

• PCA is not enough if the data is not linearly correlated. It doesn't have an effect on the size of your information. it is well worth to stated additionally that during PCA you don't normalize your information. that means that in case you exchange the dimensions of simply some of the variables on your information set, you may get special effects by using making use of PCA .

5.2 Future Scope:

From our evaluation it can be concluded that relying on the content of the authentic file, the performance of the algorithm varies. we have compared 4 lossless facts compression algorithm and our textual content bed changed into restricted in textual content facts. In future, extra compression algorithms(both lossless and lossy) may be implemented over a larger take a look at mattress which incorporates audio, video and photograph records.

In coming years, a system can be applied for you to use sensors to come across the file kind and then relying on that, it's going to select the ideal compression method for the report. each day the statistics to store and transmit is growing .that allows you to manage such large collection of statistics we must find extra efficient algorithms to compress it. additionally work may be performed on existing systems to enhance compression ratios.If the source entropy is small (e.g., below 3 bits/symbol), then it is probably best to use a search method that uses only multiplications, and try to optimize the search sequence .

Even though division is slower than the alternative operations, we are able to keep away from long searches via the use of one division in step with decoded image and a desk appearance-up for initializing the hunt. Small tables can drastically speed up the search.

REFERENCES:

[1] Khalid Sayood, "Introduction to Data Compression", Ed Fox (Editor), March 2000.

[2] Burrows M., and Wheeler, D. J. 1994," A Block-Sorting Lossless Data Compression Algorithm"SRC Research Report 124, Digital Systems Research Center.

[3] C.E. Shannon, "A mathematical theory of communication," Bell Syst. Tech. J.,vol. 27, pp. 398-403.

[4]Glen G. Langdon,Jr, "An Introduction to Arithmetic Coding." IBM Research Division, California.

[5]"Data Compression Methodologies for LossLess Data and Comparison between Algorithms",IJESIT Volume 2, Issue 2, March 2013.

[6] Amir Said, "Introduction to Arithmetic Coding - Theory and Practice",Imaging Systems Laboratory, 2004.

[7]Somefun, M. Adebayo &Adewale,"Evaluation of dominant text data compression techniques,"IJAIEM, 2014.

[8] I.H. Witten, R.M.Neal, and J.G. Cleary, "Arithmetic Coding for the data compression,"Commun. ACM, vol. 30, no. 6, pp. 520-540, June 1987.

[9] R. Pasco,"Source coding algorithms for fast data compression," Stanford Univ., Ph.D. dissertation, 1976.

[10] J.J. Rissanen, "Generalized Kraft inequality and arithmetic coding," IBM J. Res. Devel. , vol. 20, no.3, pp. 198-203, May 1976.

[11] F. Rubin, " Arithmetic stream coding using fixed precision registers," IEEE Trans. Information Theory, vol. IT-25, no. 6, pp. 520-540, June 1987.

[12] J.J. Rissanen and G.G. Langdon ," Arithmetic coding," IBM J. Res. Devel, vol. 23 no. 2, pp. 146-162, Mar. 1979.

[13] M. Guazoo, "A general minimum-redundancy source-coding algorithm," IEEE Trans. Information Theory, vol. IT-26, no. 1, pp. 15-25, Jan 1980.

[14] Manjeet Kaur, Er. Upasna Garg," Lossless Text Data Compression Algorithm Using ModifiedHuffman Algorithm," IJARCSSE, vol. 5, Issue. 7, 2015.

[15] A. Said, "Comparative Analysis of Arithmetic Coding Computational Complexity,"HewlettPackard Laboratories Report, HPL–2004–75, Palo Alto, CA, April 2004.

[16]M. Schindler, "A fast renormalization for arithmetic coding," Proc. IEEE Data Compression Conf., 1998.

[17] Texas Instruments Incorporated, "TMS320C6000 CPU and Instruction Set Reference Guide," Literature Number: SPRU189F, Dallas, TX, 2000.

[18] International Business Machines Corporation, "PowerPC 750CX/CXe RISC Microprocessor User's Manual," (preliminary edition), Hopewell Junction, NY, 2001.

[19] Intel Corporation, "Intel Pentium 4 Processor Optimization," Reference Manual 248966, Santa Clara, CA, 2001.

[20] Sun Microsystems Inc., "UltraSPARC III Technical Highlights," Palo Alto, CA, 2001.

[21] D.S. Taubman and M.W. Marcellin, "JPEG 2000: Image Compression Fundamentals," Standards and Practice, Kluwer Academic Publishers, Boston, MA, 2002.

http://www.stringology.org/DataCompression/ak-int/index_en.html

http://akbar.marlboro.edu/~mahoney/courses/Fall01/computation/compression/ac/ac_arithmetic.html

http://cotty.16x16.com/compress/nelson1.htm

http://www.drdobbs.com/parallel/arithmetic-coding-and-statistical-modeli/184408491