**Comparison Between Different Compression Techniques**

*A*

***Project Report***

*submitted in partial fulfillment of the*

*requirements for the award of the degree of*

**BACHELOR OF TECHNOLOGY**

in

**COMPUTER SCIENCE & ENGINEERING**

Specialization

**BUSINESS ANALYTICS AND OPTIMISATION**

by

|  |  |
| --- | --- |
| **Student Name** | **Roll Number** |
| **TUSHAR GOYAL** | **R103218163** |
| **VINAY** | **R103218167** |
| **SOURABH VERMA** | **R103218156** |
| **SHIVAM SAHU** | **R103218140** |

*under the guidance*

*of*

**Dr. Shashi Bhushan**



**Department of Informatics**

**School of Computer Science**

**University of Petroleum & Energy Studies**

**Bidholi, Via Prem Nagar, Dehradun, UK**

**November – 2020**



**DECLARATION**

We hereby certify that the project work entitled **“Comparison Between Different Compression Techniques ”**in partial fulfilment of the requirements for the award of the Degree of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING specialization (BUSINESS ANALYTICS AND OPTIMIZATION) and submitted to the Department of Informatics, School of Computer Science, University of Petroleum & Energy Studies, Dehradun, is an authentic record of  our work carried out during a period from **Aug., 2020**to **Dec., 2020** under  the  supervision of  **Shashi Bhushan.**

The matter presented in this project has not been submitted by me/ us for the award of any other degree of this or any other University.

**Tushar Goyal Vinay Sourabh Verma Shivam Sahu**

**R103218163 R103218167 R103218156 R103218140**

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

**Date: \_\_\_\_\_\_\_\_2020 Shashi Bhushan**

**(Project mentor)**

**Dr. Thipendra Pal Singh**

Head**,**Department of Informatics

School of Computer Science

University of Petroleum & Energy Studies

Dehradun – 248 001 (Uttarakhand)



**ACKNOWLEDGEMENT**

We wish to express our deep gratitude to Shashi Bhushan, for all advice, encouragement and

constant support he has given us throughout our project work. This work would not have been

possible without his support and valuable suggestions.

We sincerely thank to our respected Head of Department, Informatics, **Dr. Thipendra Pal**

**Singh,** for his guidance and support as and when required.

We are also grateful to **Dr. Manish Prateek, Dean SCS**, for providing the necessary facilities

to carry out our project work successfully.

We would like to thank all our friends for their help and constructive criticism during us

project work. Finally, we have no words to express our sincere gratitude to our **parents** who

have shown us this world and for everything they have given to us.

Tushar Goyal Vinay Sourabh Verma Shivam Sahu

R103218163 R103218167 R103218156 R103218140

**ABSTRACT**

Data compression is a common requirement for most of the computerized applications. Data

compression is known for reducing storage and communication costs, with data compression can

shorten the time of data exchange. There are number of data compression algorithms, which are

dedicated to compress different data formats. Even for a single data type there are number of

different compression algorithms, which use different approaches. This project examines lossless

data compression algorithms and compares their performance. A set of selected algorithms are

examined and implemented to evaluate the performance in compressing text data. A data

compression algorithm is to be developed which consumes less time while provides more

compression ratio as compared to existing techniques.

Keywords: Text Data Compression, Huffman Algorithm, Shannon Fano Algorithm, Run Length Algorithm, lossless data compression.

**TABLE OF CONTENTS**

**S.No. Contents Page No.**

**1)** **Introduction**

* 1. Problem statement
  2. Requirements
  3. Objective

**2) Related Work**

2.1) Pros & Cons of Huffman,

Shannon-Fano, & Run Length Encoding.

2.2) Literature review

**3) System analysis**

3.1) Existing Analysis

3.2) Motivation

**4) Design**

4.1) Methodology

4.2) Algorithm for Huffman Encoding

4.3) Algorithm for Run Length Encoding

4.4) Algorithm for Shannon-Fano Encoding

4.5) Flow Charts

4.5.1) Flow Chart of Huffman Encoding.

4.5.2) Flow Chart of Run Length Encoding.

4.5.3) Flow Chart of Shannon-Fano Encoding.

**5) Implementation**

5.1) Pseudocode

5.2) Result Analysis

**6) Conclusion and Future Scope**

**7) References**

**INTRODUCTION**

In this modern era of technology, everything is comprised of data. So, the problem in hand is to make the use of an adaptive compression technique, which can improve the compression ratio of existing compression techniques and could enhance the storage capacity of various storage devices by compressing the size of the original text file in a lossless manner. Data Compression Algorithms aim at minimizing the size of the data so that it occupies less amount of disk space and even help in reducing the network congestion since compressed data would use less bandwidth while being transmitted over a data communication channel.

Text compression aims at substituting the original symbol with a shorter symbol in the source code, which contains the same information but with a smaller system. In other words, using the data compression, the size of a particular file can be reduced. This is very useful when processing, storing or transferring a huge file, which needs lots of resources. If the algorithms used to encrypt works properly, there should be a significant difference between the original file and the compressed file. When data compression is used in a data transmission application, speed is the primary goal. Speed of transmission depends upon the number of bits sent, the time required for the encoder to generate the coded message and the time required for the decoder to recover the original ensemble. In a data storage application, the degree of compression is the primary concern. Compression can be classified as either lossy or lossless. Lossless compression techniques reconstruct the original data from the compressed file without any loss of data. Thus, the information does not change during the compression and decompression processes. These kinds of compression algorithms are called reversible compressions since the original message is reconstructed by the decompression process. Lossless compression techniques are used to compress medical images, text and images preserved for legal reasons, computer executable file and so on. The decompression process results an approximate reconstruction. It may be desirable, when data of some ranges which could not recognized by the human brain can be neglected.

* 1. **Problem statement**

1. Data compression systems include as an integral part a certain classifier that allows you to select the most efficient compression algorithm.
2. A compression problem from the algorithmic point of view is to find an effective and efficient algorithm to remove various redundancy from certain types of data by using lossless data compression algorithm.
3. It is used to examines the performance of the Run Length Encoding Algorithm, Huffman Encoding Algorithm, Shannon Fano Algorithm.
4. In order to examine the performance of these algorithms in compressing text data is evaluated and compared.
   1. **Objective**
5. To compare & evaluate the performance of the algorithms.
6. To implement and compare the compressed file sizes using Huffman Encoding, Run Length Encoding, and Shannon Fano Algorithm.
7. To reduce the amount of redundant information in the stored or communicated data.
8. To compress the text data and also allows the original data to be reconstructed from the compressed data.
   1. Requirements

1) Software requirements

1.1) Language: C

1.2) Operating system: Windows 7,8,10 or Mac

1.3) Compiler: GCC

2) Hardware requirements

2.1) RAM: 512MB or above

2.2) Processor: Intel platinum

2.3) Hard Disk Size: 16GB for 32-bit OS or 20GB for 64-bit OS

**2)** **Related work**

2.1) **Pros and Cons**

1) Pros and Cons of Huffman Encoding:

|  |  |
| --- | --- |
| Pros | Cons |
| 1. This encoding scheme results in saving lot of storage space, since the binary codes generated are variable in length. 2. It generates shorter binary codes for encoding symbols/characters that appear more frequently in the input string. 3. The binary codes enervated are prefix-free. 4. Lossless techniques like Huffman encoding are suitable only for encoding text and program files | 1. Lossless data encoding schemes, like Huffman encoding, achieve a lower compression ratio compared to lossy encoding techniques. 2. Since length of all the binary codes is different, it becomes difficult for the decoding software to detect whether the encoded data is corrupt. 3. The lossless techniques that use Huffman encoding are considerably slower than others. 4. lossless techniques like Huffman   encoding is not suitable for  encoding digital images. |

**2**) Pros and Cons of Run Length Encoding:

|  |  |
| --- | --- |
| Pros | Cons |
| 1. This algorithm is very easy to implement and does not require much CPU horsepower. 2. The consecutive sequences of symbols are identified as runs and the others are identified as non runs in this algorithm. Thus, this algorithm deals with some sort of redundancy. 3. RLE algorithm is very useful for data that has a lot of data with the same value in sequence such as file icons, line drawings, and animation. | 1. **RLE** compression is only efficient with files that contain lots of repetitive data. 2. Does not work well for English text however, is almost always a part of a larger compression system. 3. RLE is not very suitable applied to data that has sentences that has a meaning, because it will result in increasing the size of data compression than the initial data. |

3) Pros and Cons of Shannon-Fano Encoding:

|  |  |
| --- | --- |
| Pros | Cons |
| 1) For Shannon Fano coding procedure we do not need to build the entire codebook instead, we simply obtain the code for the tag corresponding to a given sequence.  2) It is entirely feasible to code sequenced of length 20 or much more. | 1)  In Shannon-Fano coding, we cannot be sure about the codes generated. There may be two different codes for the same symbol depending on the way we build our tree.  2) This algorithm is not able to achieve the code as efficiently as Huffman's algorithm. |

2.2) **Literature review**

The problem of data compression is one of the important aspects in the development of information technology. Data compression is a process of resizing a file or document to be smaller in size. Along with the development of hardware and software technology is increasingly sophisticated and complex that demands the efficiency in terms of data storage and memory [1].

Communication is the exchange of thoughts, messages, or information, as by speech, visuals, signals, writing or behaviour. When two entities are communicating and do not want a third party to listen in or know, they need to communicate in such a way that it doesn’t get intercepted. Secure communication is needed and there exists many tools for this. With secure communication, if we use compressed form of the messages which are being sent, this will make an effective and powerful system. Compressed message will be smaller in size than the original and less bits will be needed to make it confidential [2].

This is very useful when processing, storing or transferring a huge file, which needs lots of resources. If the algorithms used to encrypt works properly, there should be a significant difference between the original file and the compressed file. When data compression is used in a data transmission application, speed is the primary goal. Speed of transmission depends upon the number of bits sent, the time required for the encoder to generate the coded message and the time required for the decoder to recover the original ensemble. In a data storage application, the degree of compression is the primary concern. Compression can be classified as either lossy or lossless. Lossless compression techniques reconstruct the original data from the compressed file without any loss of data. Thus, the information does not change during the compression and decompression processes [3].

We require the ability to reconstitute the original file from the compressed version at any time. Data compression is a method of encoding rules that allows substantial reduction in the total number of bits to store or transmit a file. The more information being dealt with, the more it costs in terms of storage and transmission costs [4].

We check the performance of these three lossless data compression algorithms for compressing text data and then the compressed text data will be evaluated and compared.

**3) System Analysis**

**3.1) Existing Analysis**

Detailed analysis shows that the lossy techniques performs better quantitatively whereas lossless is better qualitatively. However, lossless techniques are more effective as there is no data loss during the process. Initially, the dearth of quantitative analysis of these compression methodologies was prevalent as such information were not easily accessible or even available to pursue further researches in it. This research has implemented and analysed most prevalent techniques and quantified results on most critical parameters.

**3.2) Motivation**

 We will broaden knowledge of compression techniques as well as the mathematical foundations of data compression, become aware of existing compression standards and some compression utilities available. You will also benefit from the development of transferable skills such as problem analysis and problem solving. Data compression has wide application in terms of information storage, including representation of the abstract data type string and file compression. Huffman coding technique is the basis for the compact command of the UNIX operating system. The primary difficulty associated with variable-length codewords is that the rate at which bits are presented to the transmission channel will fluctuate, depending on the relative frequencies of the source messages. This requires buffering between the source and the channel. Advances in technology have both overcome this difficulty and contributed to the appeal of variable-length codes. Current data networks allocate communication resources to sources on the basis of need and provide buffering as part of the system. The driving motivation of our technique is to perform significantly faster lookups without the need to decompress the compressed data object, particularly with increasing data object sizes that are being encountered.

**4) Design**

4.1) Methodology

There are number of different data compression methodologies, which are used to compress different data formats like text, video, audio, image files. Data compression techniques can be broadly classified into two major categories, “lossy” and “lossless” data compression techniques.

Currently there are many methods that can be used to compress data. And each method has different results and ways. In this report we will discuss the comparison of data compression using three different algorithms, there are using Shannon-Fano Algorithm, Huffman Algorithm, and Run Length Encoding Algorithm.

These are the following comparisons factors that have been selected:

1) **Compression Ratio** is the ratio between the size of the compressed file and the size of the source file.

Compression Ratio= size after compression/size before compression.

2) **Compression Factor** is the inverse of the compression ratio. That is the ratio between the size of the source file and the size of the compressed file.

Compression Ratio= size before compression/size after compression.

In order to test the performance of lossless compression algorithms, the Run Length Encoding Algorithm, Huffman Encoding Algorithm, Shannon Fano Algorithm are implemented and tested with a set of text files. Performances are evaluated by computing the above-mentioned factors.

1) Compression ratio using Huffman encoding.

2) Compression ratio using Run Length encoding.

3) Compression ratio using Shannon-Fano encoding.

4.2) **Algorithm for Huffman Encoding**

Step1: Compute the probability of each character.

Step2: Sort the set of data in ASCENDING order.

Step3: Create a new node where the left child is the lowest in the sorted list and the right is the second lowest in the sorted list.

Step4: Chop-off those elements in the sorted list as they are now part of one node and add the probabilities. The result is the probabilities for the new node.

Step5: Perform insertion sort on the list with the new node. Step6: Repeat steps 3, 4, 5 until, only have one node left.

4.3) **Algorithm for Run Length Encoding**

step1: Pick the character from source string.

step2: Append the picked character to the destination string.

step3: Count the number of subsequent occurrences of the picked character and

append the count to destination string.

step4: Pick the next character and repeat steps 2, 3 and 4 if end of string is NOT

reached.

4.4) **Algorithm for Shannon-Fano Encoding**

Step1: Create table providing frequencies / counts.

Step2: sort symbols according to their frequencies/ Probabilities in descending order.

Step3: Recursively divided into two parts, each with approx. (binary) same number

of counts.

Step4: Add a binary 0 to the code words of the upper part and a binary 1 to the

lower part.

Step5: Search for the next part containing more than two symbols, repeat the step3

and step4.

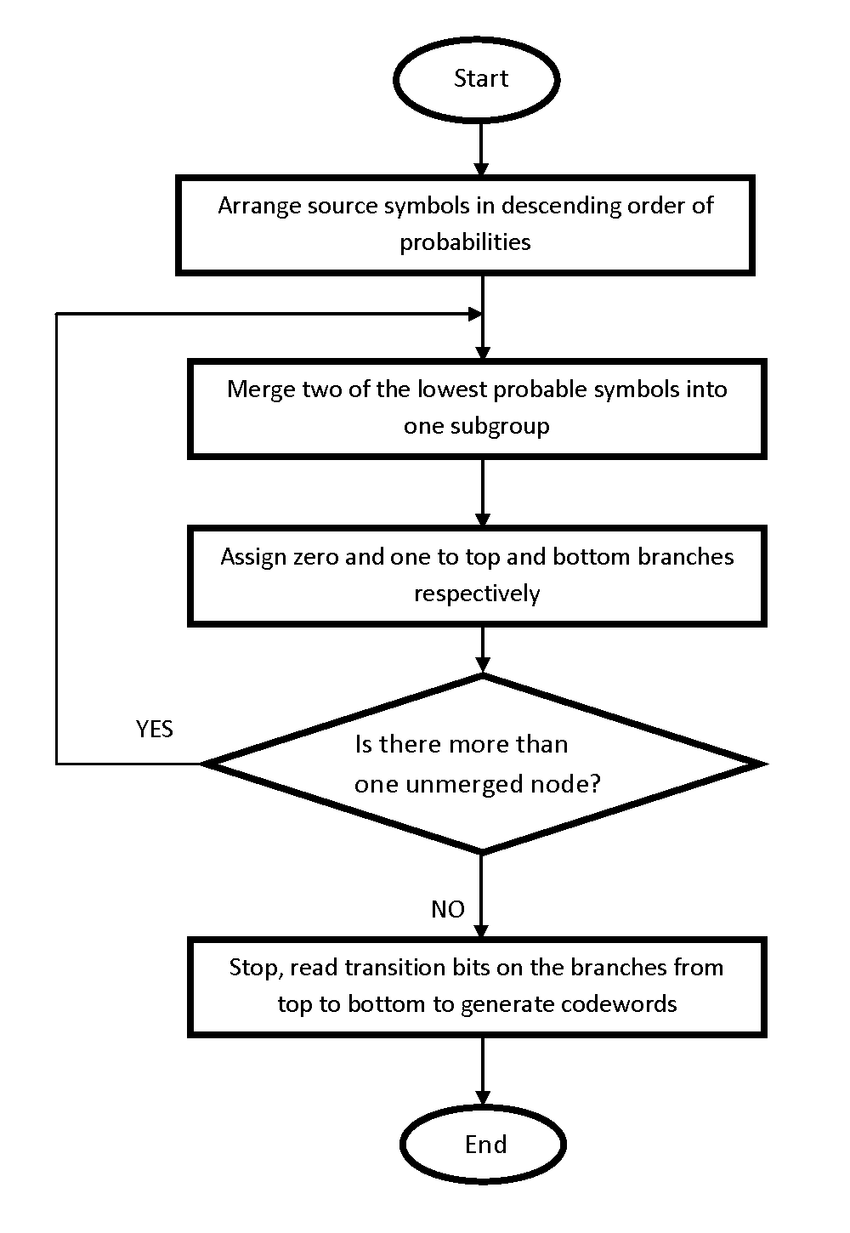
Step6: Coding of the origination data according to code words.

Step7: Create the coding tree

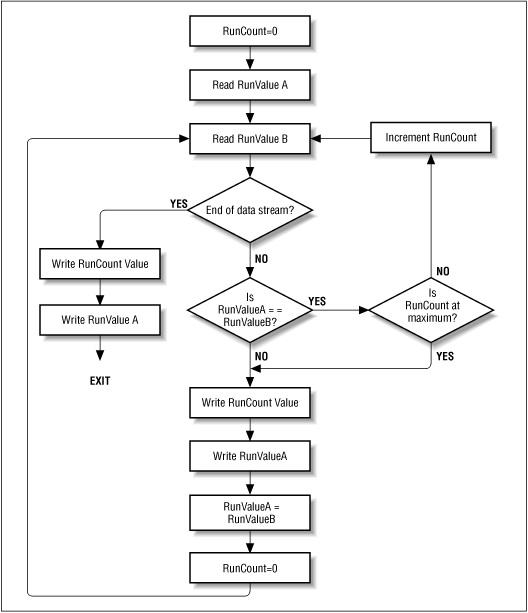
Step8: Transmit Codes instead of Tokens.

**4.5) Flow Charts**

4.5.1) Huffman Coding Algorithm [6]



4.5.2) Run Length Encoding



4.5) Shanno-FanoAlgorithm 

**5) Implementation**

## 5.1) Pseudocode (Huffman):

## 1. Find frequencies of all the characters in the string

## 2. Make a node corresponding to every (character, frequency)

## 3. Heapify the nodes

## 4. While heap has 2 or more nodes

## 5. Pop 2 nodes say first and second

## 6. Make a new node x, with frequency = first.freq + second.freq

## 7. Make first node as left child of x

## 8. Make second node as right child of x

## 9. Insert x into the heap

## 10. Extract the last node which is the root of the tree

## /\*For assigning codes to each character, \*/

## 11. Traverse the tree

## 12. If you move left, assign the path 0

## 13. If you move right, assign the path 1

## 14. If you reach the leaf node:

## /\* Code for the character of the leaf node is the path from the root to that node. \*/

## 

## 

## 5.2) Pseudocode (Run Length):

## 

## Loop: count = 0

REPEAT

get next symbol

count = count + 1

UNTIL (symbol unequal to next one)

output symbol

IF count > 1

output count

GOTO Loop

## 5.3) Pseudocode (Shannon-Fano):

1. Find the probability of occurrence of each character in the string

2. Sort the array of characters in the increasing order of their probabilities, say A

3. fanoShannon(A):

4. If (size(A)==1)

5. return

6. Divide A into left and right such that the difference between the sum of probabilities of left half and right half is minimum

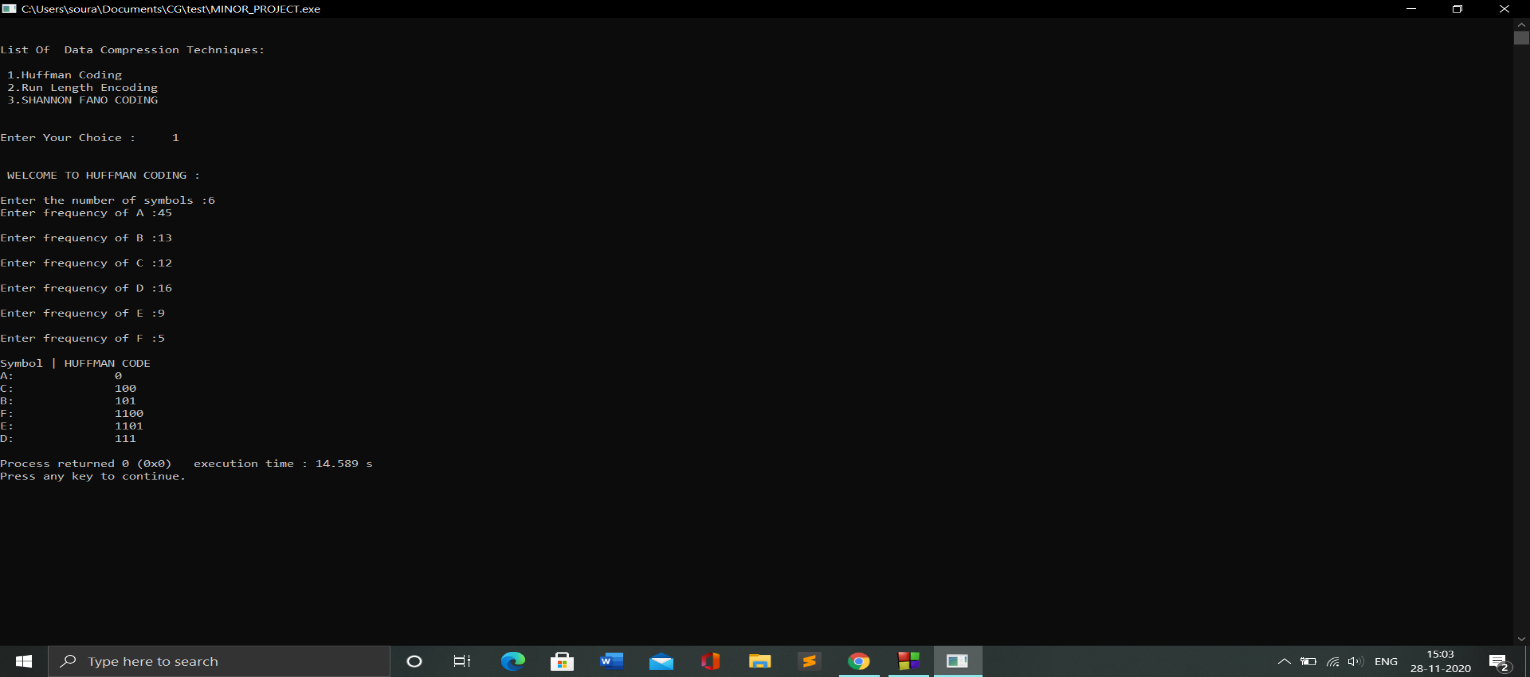
7. Append 0 in the codes of the left half

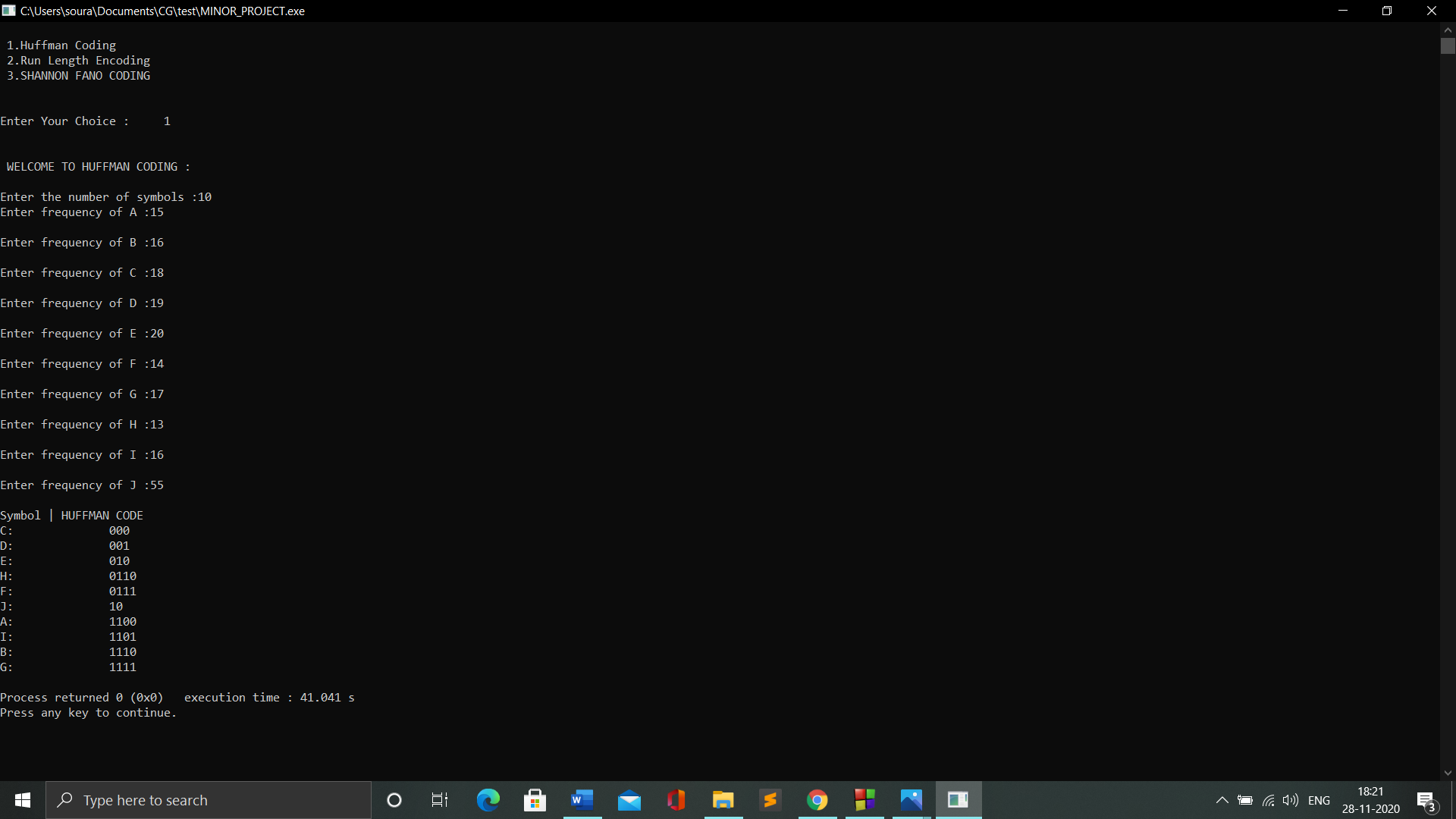
8. Append 1 in the codes of the right half

9. fanoShannon(left)

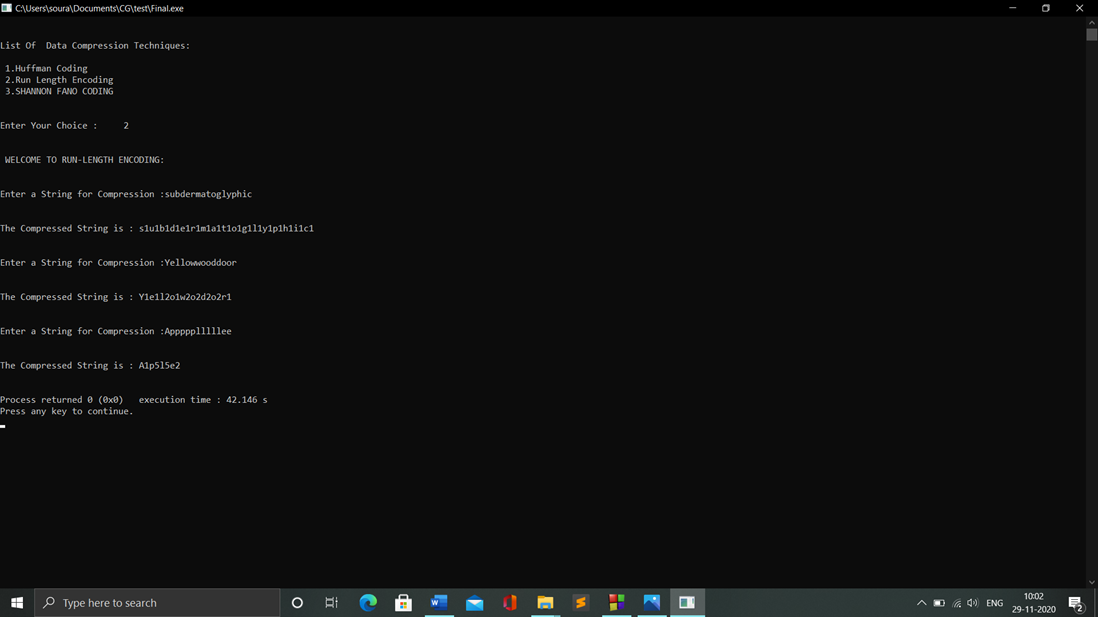
10. fanoShannon(right)

**Output Screen**

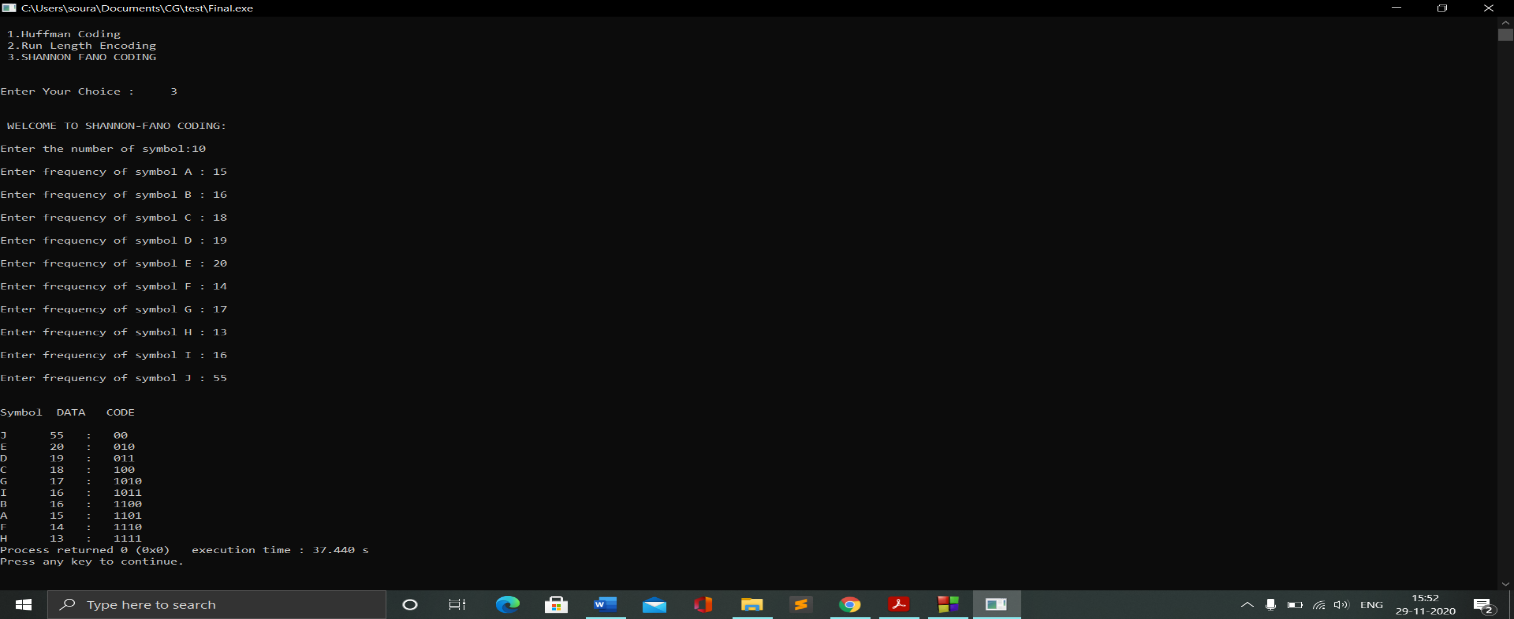
1.  OUTPUT for HUFFMAN CODING

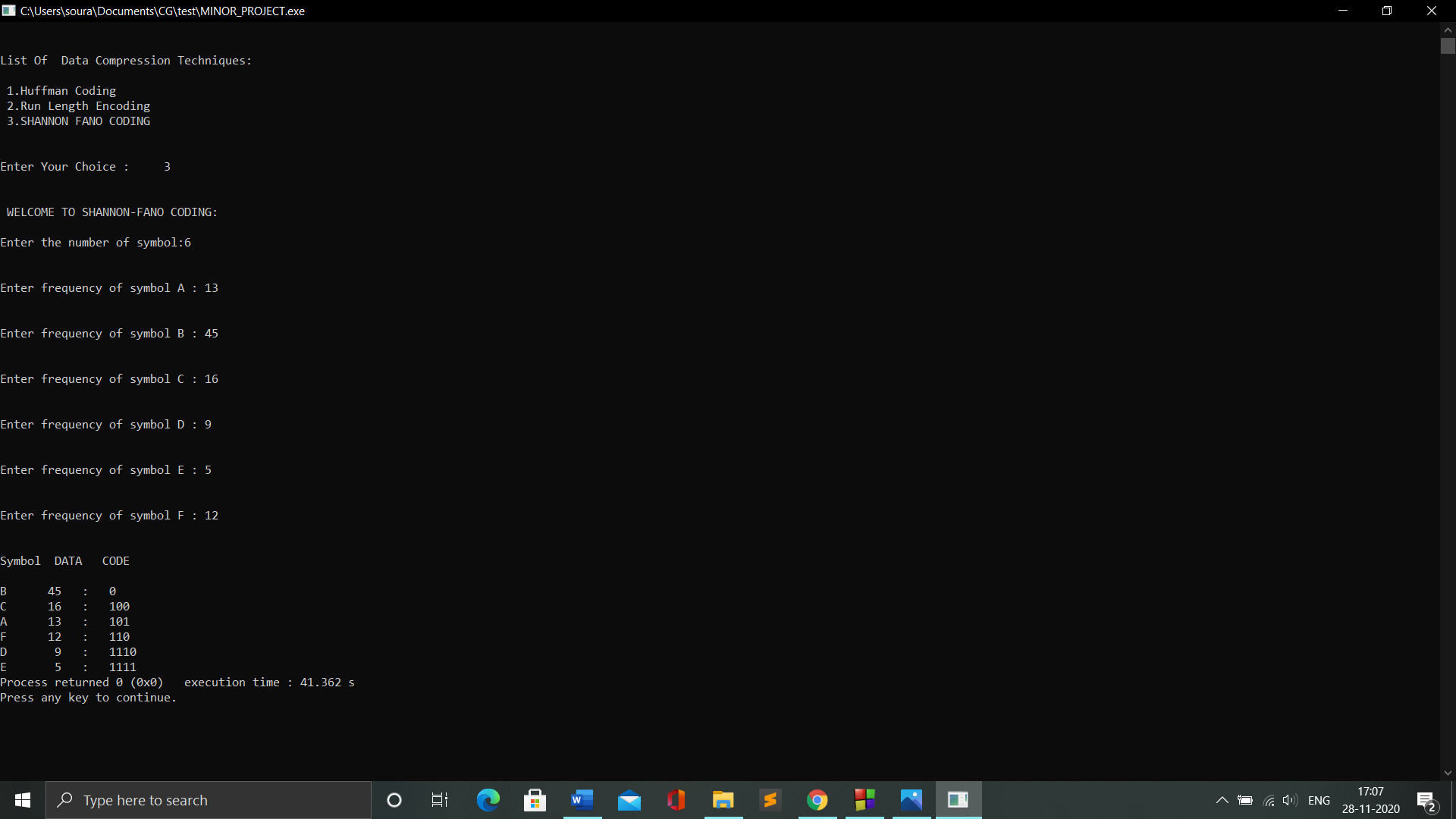


1. OUTPUT for RUN-LENGTH ENCODING



1. OUTPUT for SHANNON-FANO





**5.2) Result Analysis**

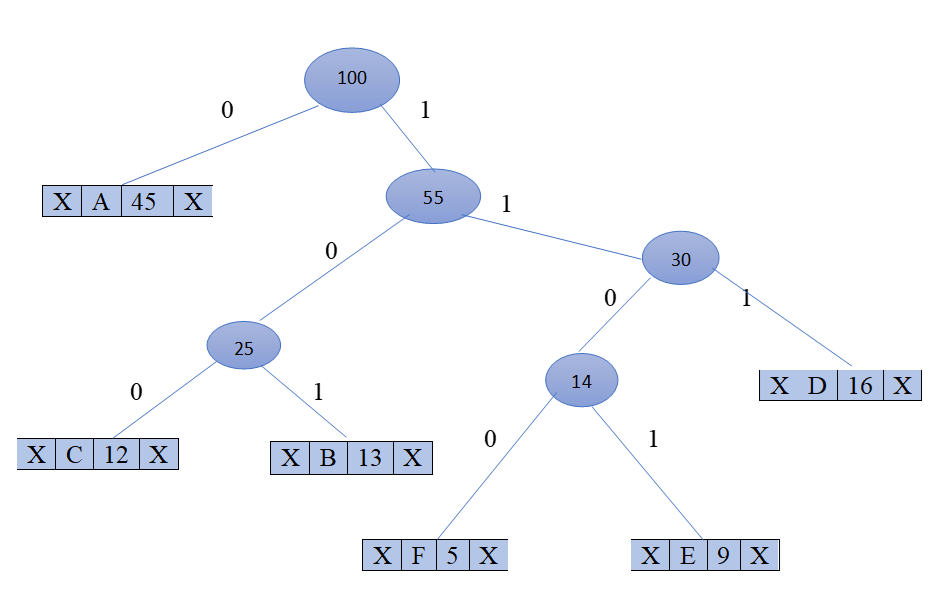
A.) HUFFMAN CODING

For the First Output of Huffman:

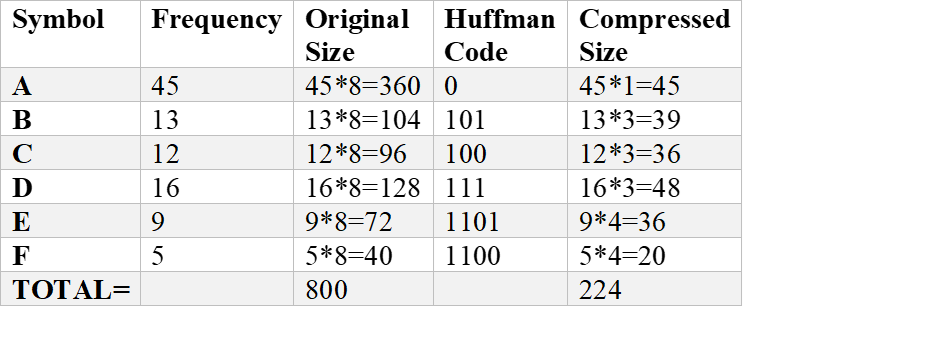
INPUT:

Symbols= {A, B, C, D, E, F}

Frequency= {45,13,12,16,9,5}

HUFFMAN TREE for the Following Data:

**TABLE FOR HUFFMAN ENCODING:**



Original File Size=800 bit=100 byte

Compressed File Size=224bit=28 bytes

&, COMPRESSTION RATIO= 28/100 =0.28

Similarly, for Second Output:

Symbol= {A, B, C, D, E, F, G, H, I, J}

Frequency= {15, 16 ,18 ,19,20, 14, 17 ,13 ,16 ,55}

Original File Size=1628 bits=203.5 bytes

Compressed File Size=645 bits=80.625 bytes

&, Compression Ratio =80.625/203.5=0.396

1. RUN-LENGTH ENCODING

**For the first OUTPUT:**

STRING=” subdermatoglyphic”

COMPRESSED STRING=” s1u1b1d1e1r1m1a1t1o1g1l1y1p1h1i1c1”

Original File size= 17\*8=136 bits= 17bytes

Compressed File Size=34\*8=272 bits=34 bytes

Compression Ratio=34/17=2.00

**For the second OUTPUT:**

STRING=” Yellowwooddoor”

COMPRESSED STRING=” Y1e1l2o1w2o2d2o2r1”

Original File size= 14\*8=112 bits= 14 bytes

Compressed File Size=18\*8=144 bits=18bytes

Compression Ratio=18/14=1.286

**For the third OUTPUT:**

STRING=” Appppplllllee”

COMPRESSED STRING=” A1p5l5e2”

Original File size= 13\*8=104 bits= 13 bytes

Compressed File Size=8\*8=64 bits=8 bytes

Compression Ratio=8/13=0.615

1. SHANNON-FANO ENCODING

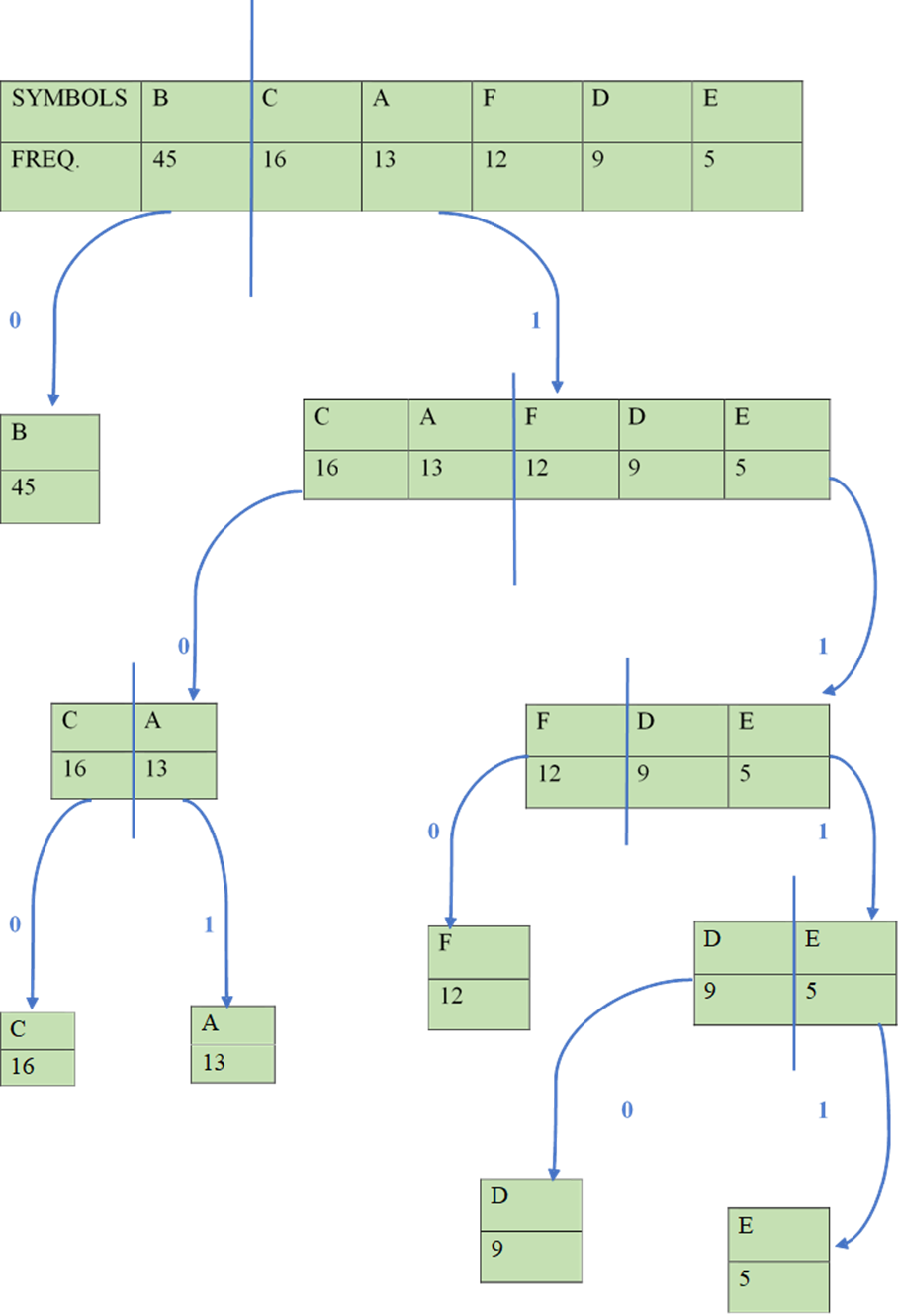
**For the First OUTPUT:**

Symbol= {A, B, C, D, E, F}

Frequency= {13, 45 ,16 ,9,5, 12}

**SORTED TABLE:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| SYMBOLS | B | C | A | F | D | E |
| FREQ. | 45 | 16 | 13 | 12 | 9 | 5 |



**TABLE FOR SHANNON-FANO ENCODING:**



Original File Size=800 bit=100 byte

Compressed File Size=224bit=28 bytes

&, COMPRESSTION RATIO= 28/100 =0.28

Similarly, for Second Output:

Symbol= {A, B, C, D, E, F, G, H, I, J}

Frequency= {15, 16 ,18 ,19,20, 14, 17 ,13 ,16 ,55}

Original File Size=1628 bits=203.5 bytes

Compressed File Size=650 bits=81.25 bytes

&, Compression Ratio =81.25/203.5=0.4

Table 1.1. Huffman Encoding results

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| File  No. | Original File | Compressed File | Compression  Ratio | Compression Factor |
| 1 | 800 | 224 | 0.28 | 3.571 |
| 2 | 1628 | 645 | 0.396 | 2.525 |
| 3 | 2679 | 659 | 0.246 | 4.065 |
| 4 | 3074 | 1792 | 0.583 | 1.715 |
| 5 | 8144 | 5361 | 0.658 | 1.52 |

Table 1.2. Run-Length Encoding results

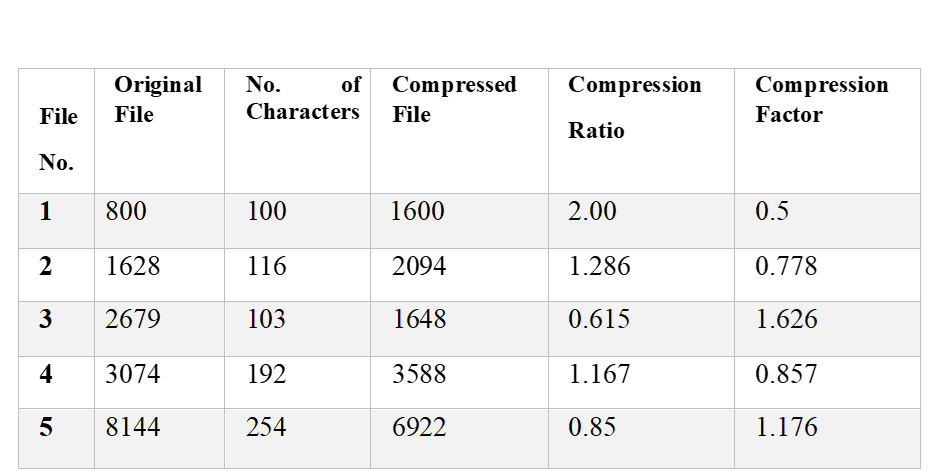
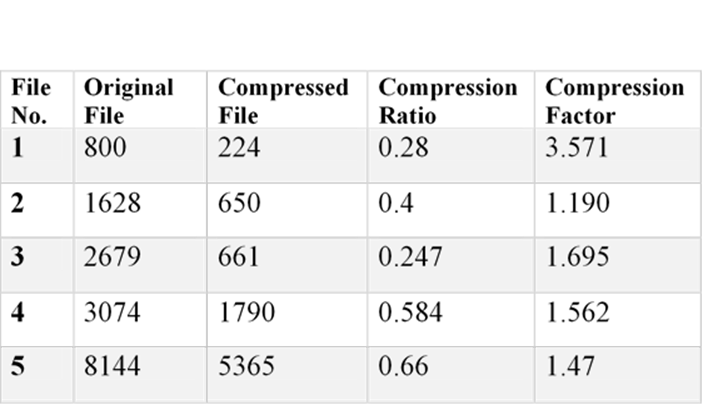
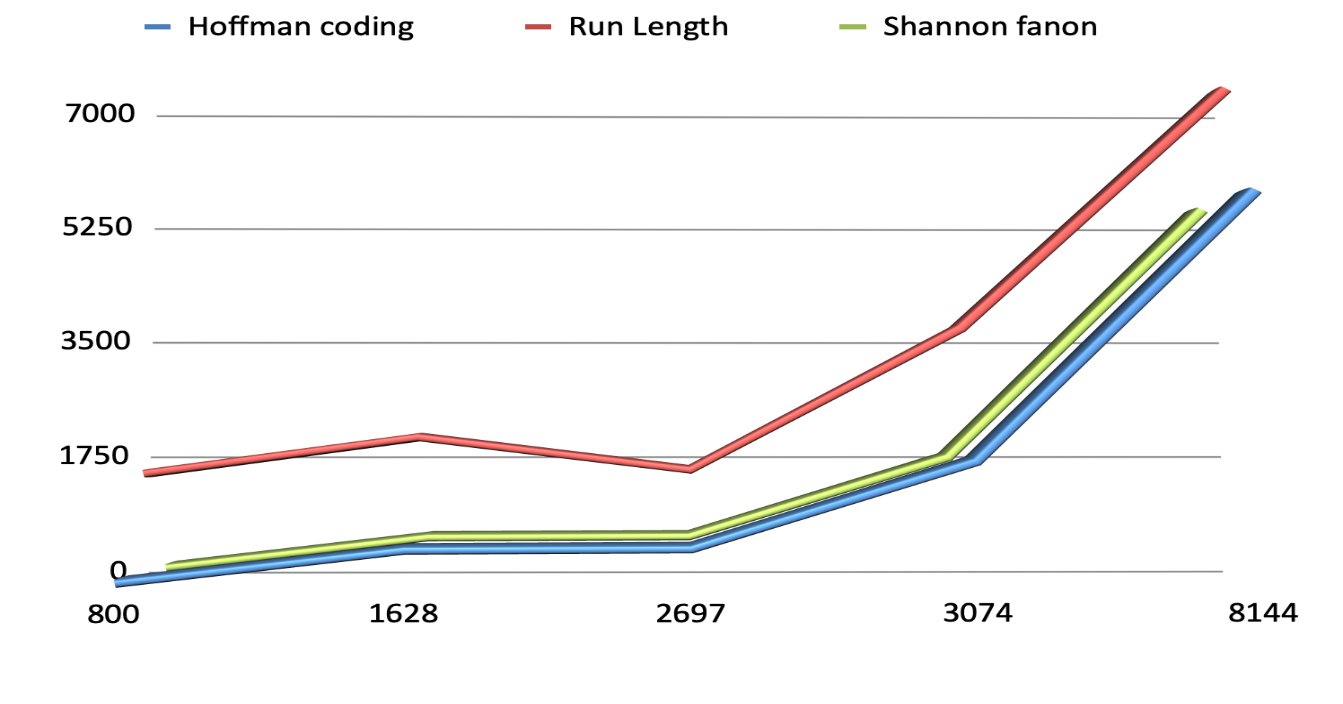


Table 1.3. Shannon-Fano Encoding results

****

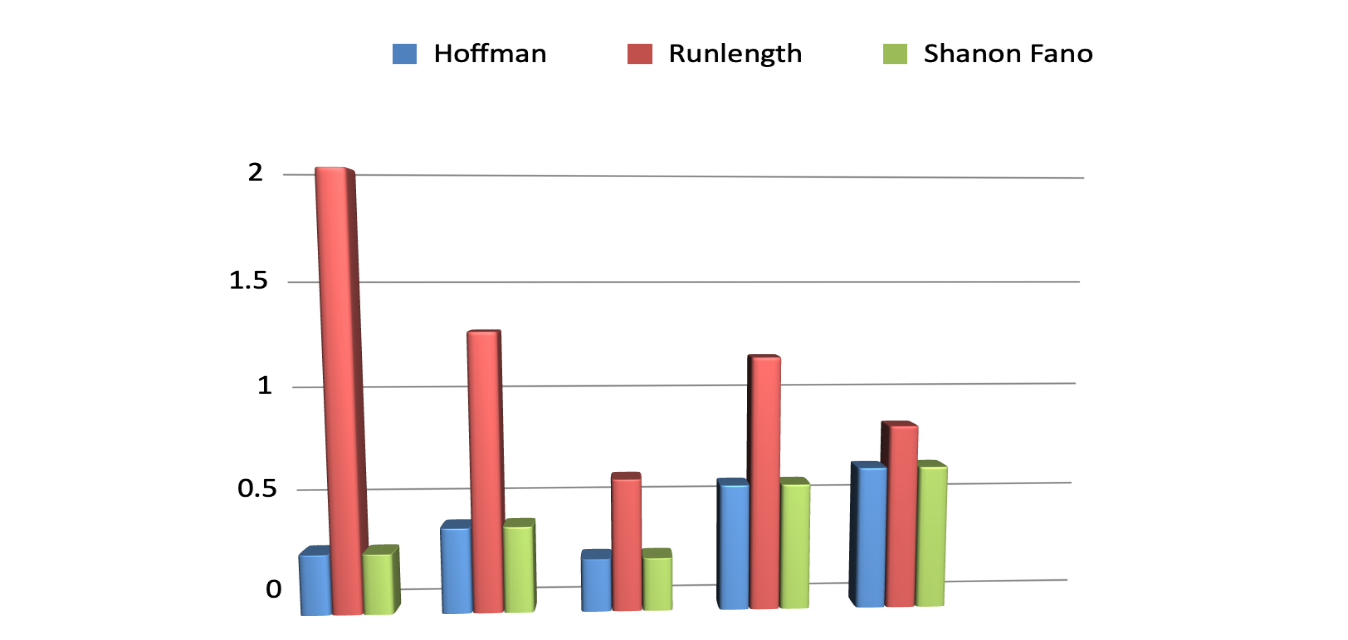
**Comparison on the basis of compressed file size:**



**Compressed File Size (in bytes)**

**(in bytes )**

**Original File Size (in bytes )**

 **Compression Ratio:**

**Compression Ratio**

**6) Conclusion and Future Scope**

Data compression is most consideration thing of the recent world. We have to compress a huge amount of data so as to carry from one place to other or in a storage format. And after doing 3 experiments above, it can be concluded that, in Shannon-Fano,and Huffman always get smaller result than previous bit. As for the Run Length Encoding algorithm, it depends on the sentence used. If the sentence used is a sentence that has a meaning, usually the algorithm is always obtained results greater than the previous bits. Whereas if the word used is a word that has a loop, it can produce a smaller end result than the previous bit. So, it all depends on the data used. These proposed compression techniques are improved the efficiency of compression on text. Huffman encoding Algorithm is suitable for the given text.

**7) References**

[1] Dea Ayu Rachesti, Tito Waluyo Purboyo, and Anggunmeka Luhur Prasasti, “of Text Data Compression Using Huffman, Shannon-Fano, Run Length Encoding, and Tunstall Methods” ,available at <http://www.ripublication.com>. International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 23 (2017) pp. 13618-13622.

[2] Nigam Sangwan, ” Text Encryption with Huffman Compression”, International Journal of Computer Applications (0975 – 8887) Volume 54– No.6, September 2012, Institute of Technology and Management University, Gurgaon, Haryana, India.

[3] S.R. KODITUWAKKU, & U. S.AMARASINGHE , ” COMPARISON OF LOSSLESS DATA COMPRESSION ALGORITHMS FOR TEXT DATA”, Indian Journal of Computer Science and Engineering Vol 1 No 4 416-426.

[4] Amandeep Singh Sidhu, & Er. Meenakshi Garg, “Research Paper on Text Data Compression Algorithm using Hybrid Approach”, International Journal of Computer Science and Mobile Computing, Vol.3 Issue.12, December- 2014, pg. 01-10.

[5] T.SubhamastanRao , M.Soujanya, T.Hemalatha, &T.Revathi, ” Simultaneous Data Compression and Encryption”, ) International Journal of Computer Science and Information Technologies, Vol. 2 (5) , 2011, 2369-2374.

# [6] Tito Waluyo Purboyo, & Anggunmeka Luhur Prasasti, “ Comparison of Data Compression in Text Using Huffman, Shannon-Fano, Run Length Encoding, and Tunstall Method” , International Journal of Applied Engineering Research · January 2017.

[7] P.Yellamma, and Dr.Narasimham Challa, “Performance Analysis Of Different Data Compression Techniques On Text File”, International Journal of Engineering Research & Technology (IJERT) Vol. 1 Issue 8, October – 2012.

[8] Shrusti Porwal, Yashi Chaudhary, Jitendra Joshi, Manish Jain, “Data Compression Methodologies for Lossless Data and Comparison between Algorithms”, International Journal of Engineering Science and Innovative Technology (IJESIT) Volume 2, Issue 2, March 2013.

process.