DATA COMPRESSION AND DECOMPRESSION USING LEMPEL-ZIV-WELCH TECHNIQUE

A
Project Report

submitted in partial fulfillment of the
requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE

Specialization in

Oil And Gas Informatics

By :

Name	Roll No
AMIT SINGH	R970216010
ANKIT SINHA	R970216014
RAHUL RAJ	R970216050

Under the guidance of

UPPARA RAJNIKANTH
INDUSTRY FELLOW
Department of Systemics

Department of Informatics
School of Computer Science
UNIVERSITY OF PETROLEUM AND ENERGY STUDIES
Bidholi, Via Prem Nagar, Dehradun, Uttarakhand

2018-19

CANDIDATES DECLARATION

We hereby certify that the project work entitled **DATA COMPRESSION USING LEMPEL-ZIV-WELCH TECHNIQUE** in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science And Engineering with Specialization in Oil and Gas Informatics and submitted to the Department of Informatics at School of Computer Science, University of Petroleum And Energy Studies, Dehradun, is an authentic record of our work carried out during a period from **August, 2018** to **December, 2018** under the supervision of **UPPARA RAJNIKANTH, Industry Fellow in Department of Systemics UNIVERSITY OF PETROLEUM AND ENERGY STUDIES** .

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.

NAME OF STUDENT :-(Amit Singh Ankit Sinha Rahul Raj)

Roll No.- R970216010 R970216014 R970216050

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

(Date: 17 Dec 2018)

(UPPARA RAJNIKANTH)
Project Guide

Dr. T.P SINGH

Head
Department of Informatics
School of Computer Science
University of Petroleum And Energy Studies
Dehradun - 248 001 (Uttarakhand)

ACKNOWLEDGEMENT

We wish to express our deep gratitude to our guide **UPPARA RAJNIKANTH**, for all advice, encouragement and constant support he has given us through out our project work. This work would not have been possible without his support and valuable suggestions.

We sincerely thank to our Head of the Department, **Dr. T.P SINGH**, for his great support in doing our **DATA COMPRESSION USING LEMPEL-ZIV-WELCH TECHNIQUE** at **SoCS**.

We are also grateful to **Dr. Manish Prateek Professor and Director SoCS** and **Dr. Kamal Bansal Dean CoES**, UPES for giving us 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 our project work. Finally we have no words to express our sincere gratitude to our **parents** who have shown us this world and for every support they have given us.

Name	**AMIT SINGH**	**ANKIT SINHA**	**RAHUL RAJ**
Roll No.	**R970206010**	**R970216014**	**R970206050**

ABSTRACT

Data Compression is one of the most fundamental problems in computer science and information technology. The most well known sequential algorithm is Lempel-Ziv-Welch (LZW) compressiontechnique. The Lempel-Ziv-Welch(LZW)datacompression algorithm is evaluated for use in the removal of the redundancy in computer les. The lempel-ziv welch algorithm is compared with respect to encoding and decoding speed, memory requirements and compression ratio also it is optimised for hardware implemntation. The project aims to design the modied LZW algorithm on source coding and this will be proposed in this paper to improve the compression efciency of the existing algorithms. Such method is to be implemented with appropriate modications that gives the best performance and satises the requirements of the applications and for the efcient Data compression. LZW is the foremost technique for general purpose data compression due to its simplicity and versatility its the basis of many PC utilities that claim to double the capacity of your harddrive if data have been losslessly compressed,then original data can be recovered exactly from the compressed data after compress/expand cycle.e LZW algorithm in future which definitely get good results like: Better compression ratio.

Keywords: LZW, Compression, Dictionary, Compression Ratio

TABLE OF CONTENTS

Contents

LIST OF FIGURES

List of Figures

1 Introduction

Lempel-ziv-Welch [1]coding is the foremost technique for general purpose data compression due to its simplicity and versatility.It address spatial redundancies in an image. It is an error free compression approach.The coding is based on a dicitionary or codeblock containing the source symbols to been coded.The coding starts with an initial dicitionary, which is enlarged with the arrival of new symbol sequences. Lempel-Ziv-Welch is a universal data compression algorithm. It is one of the adaptive technique that is been created after the size of les usually increases to a great extent when it includes lots of repetitive data like data les,documents,images and so on. LZW compression is the best technique for reducing the size of les containing more repetitive data. Its a lossless, means no data is lost when compressing, if data have been losslessly compressed, the original data can be recovered exactly from the compressed data after a compress/expand cycle. Lempel-Ziv- welch compression works by reading a sequence of symbols, grouping the symbols intos trings, and converting the strings into codes. Because the codes take up less space than the strings they replace, we get compression. The Idea relies on reoccurring patterns to save data space. LZW is the foremost technique for general purpose data compression due to its simplicity and versatility its the basis of many PC utilities that claim to double the capacity of your hard drive.

2 Background Study

These are the some references we studied and analysed for developing our project. Research paper by International Journal of Advanced Research in Computer Science and Software Engineering (03 march 2012)[3] [2] : LZW is a universal lossless data compression algorithm, which takes linear time in encoding. In this paper the problem of combining and summarizing LZW algorithm with binary search and multiple dictionaries data compression reduces the complexity of time at the maximum is analyzed. This is achieved by reducing number of comparisons and the number of shifts while creating the dictionary while encoding and decoding. The main focus of this paper in creating dictionary using LZW algorithm of any le with binary search and multiple dictionaries comprises the efciency of compression time and decompression time efciently. The experimental result shows average 99 percent decompression, comparing to other already existing algorithm, procedures and methods. The proposed work can be further enhanced and expanded for the authentication of compression and decompression techniques to obtain op timum accuracy in time. Research paper on LZW data compression algorithm by American Journal of Engineering Research (AJER) (2014) [3] : In AJER research paper the author Dheemanth H N, he explained the basics of the LZW algorithm. Dheemanth H N, has also described the encoding and the decoding process with example in his research paper. In encoding algorithm a sample string is used to demonstrate the algorithm. Lempel-Ziv-Welch (LZW) [4] Encoding Discussion and Implementation by Michael Dipperstein : In his research paper Michael Dipperstein has explained about theLZW(LZ78)and his intent is to publish an easy to follow ANSIC implementation of the Lempel-Ziv-Welch(LZW)encoding/decoding algorithm [5]. In his paper he explained about encoding of LZW that unlike LZSS, entries in the LZW dictionary are strings and every LZW code word is a references to a string in the dictionary. He also described the basic of encoding algorithm with several steps and explained about the decoding of LZW . The decoding of LZW is pretty much the opposite of how it is encoded and then he delineate the basic decoding algorithm in several steps. And in his research paper he taught about the exceptions of the decoding rule later he discussed about the implementation issues of the LZW algorithm like the layout of the dicitionary and decoding code words to strings.

3 Problem Statement

Technology is growing in every aspect but the basic need to save the memory and time utilization still remains.Whenever we deal with any kind of data, Users are bound to compress the data to minimize the space utilization. So this project aims to implement Lempel-ziv-welch algorithm and to modify it that helps user a special benet.

4 Objective

To design and perform the data compression and decompression and compression ratio using lempel-zivwelch technique.

5 Design Methodology

LZW is a dictionary based compression algorithm that means it encodes data by referencing a dictio- nary.

WORKING:- LZW compression works by reading a sequence of symbols, grouping the symbols into strings, and converting the strings into codes. Because the codes take up less space than the strings they replace, we get compression. LZW compression uses a code table, with 4096 as a common choice for the number of table entries. Codes 0-255 in the code table are always assigned to represent single bytes from the input file. As the encoding continues, LZW identifies repeated sequences in the data, and adds them to the code table. Decoding is achieved by taking each code from the compressed file and translating it through the code table to find what character or characters it represents.

The dictionary The dictionary [4] is an array of strings, or in other words, prefix-byte pairs. Each string in the dictionary is unique, ie. no string appears in the dictionary twice. The first 256 elements in the dictionary consist of pairs of empty prefixes and the byte values corresponding to their index in the dictionary. In other words, the first element of the dictionary is [empty] (0), the next one is [empty](1) and so on, up to [empty](255). (When optimizing the algorithm the dictionary can be initialized with less entries if the input data uses less than 256 byte values, but for the sake of simplicity well assume in this introductory section that all 256 values are used.) All the new strings added to the dictionary will be added to index positions from 256 upwards. Each new string is added to the next unused position in the dictionary.

LZW compression works by reading a sequence of symbols, grouping the symbols into strings, and converting the strings into codes Because the codes take up less space than the strings they replace, we get compression. LZW compression uses a code table or dictionary. The words are replaced by their corresponding codes and so the input le is compressed.

The algorithm works by scanning through the input string for successively longer substrings until it nds one that is not in the dictionary. When such a string is found, the index for the string less the last character is retrieved from the dictionary and sent to output, and the new string is added to the dictionary with the next available code. The last input character is then used as the next starting point to scan for substrings. The decoding algorithm works by reading value from the encoded input and output the corresponding string from the initialized dictionary. At the same time it obtains the next value from the input,and adds to the dictionary the concatenation of the string just output and the rst character of the string obtained by decoding the next input value. The decoder then proceeds to the next input value (which was already read in as the next value in the previous pass) and repeats the process until there is no more input, at which point the nal input value is decoded without any more additions to the dictionary.

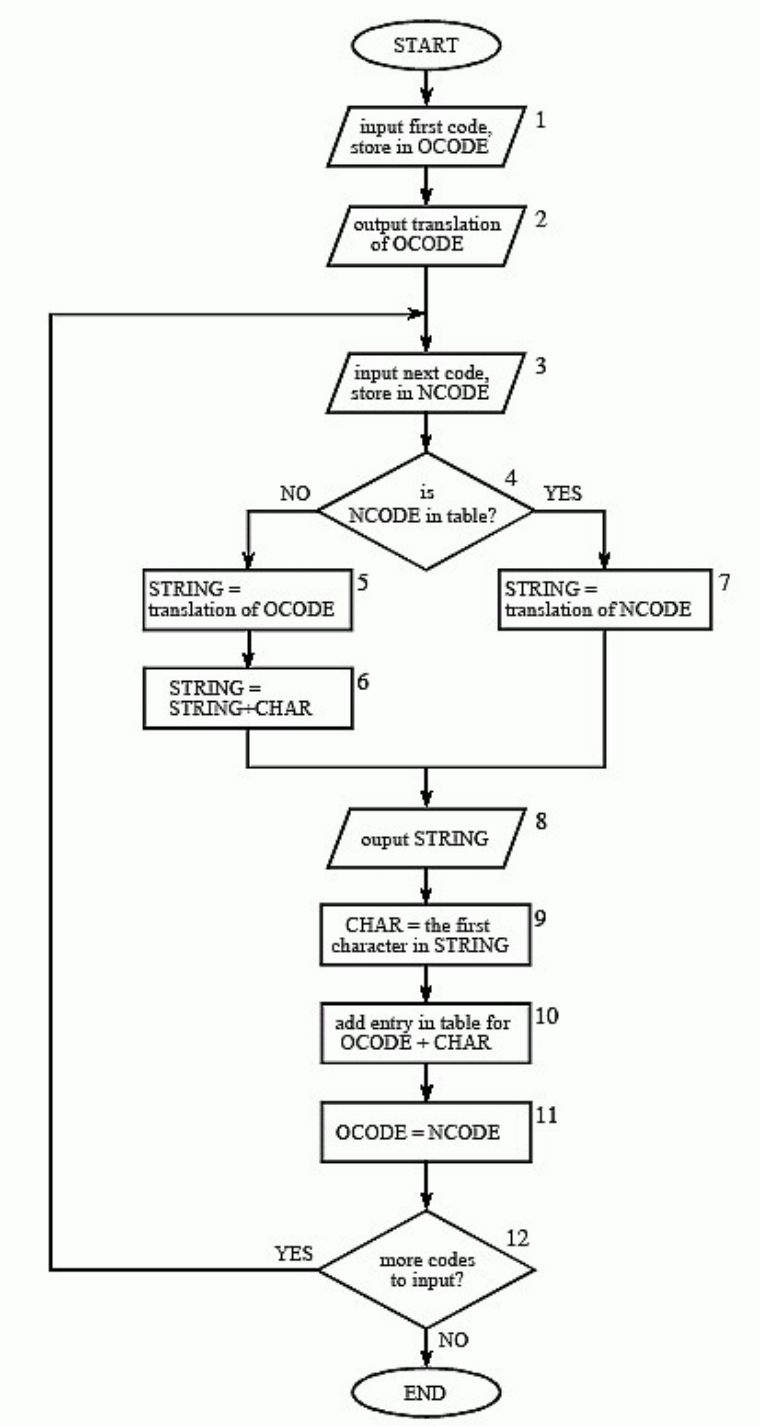

Figure 1: *Decoding algorithm*

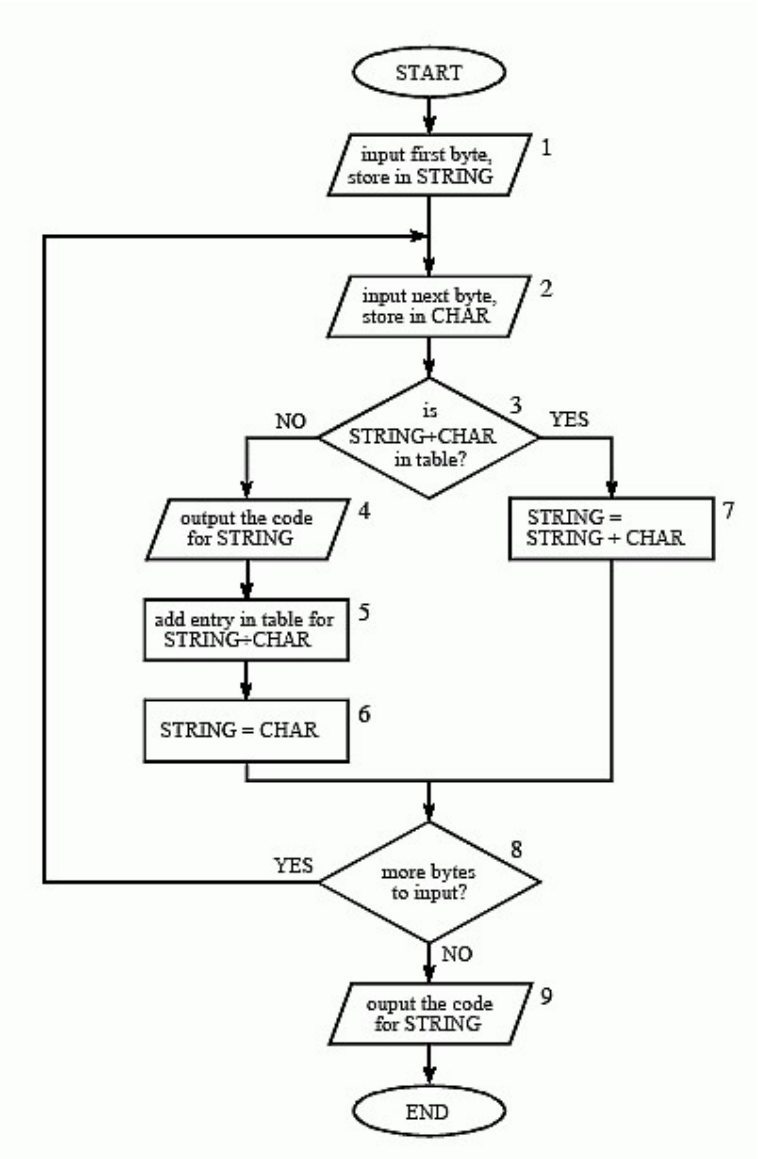

Figure 2: *Encoding algorithm*

6 Implementation

A dictionary keeps a correspondence between the longest encountered words and a list of code values. The words are replaced by their corresponding codes and so the input file is compressed. Therefore, the efficiency of the algorithm increases as the number of long, repetitive words in the input data increases Any project needs a conceptual model(Software development life cycle) because it describe the stages and tasks involve in each step of a project to write and deploy software.In this project we are using Incremental Model which broke down requirements into multiple standalone modules. Incremental development is done in steps from analysis design, implementation, testing or verification, maintenances. Each iteration passes through the require- ments, design, coding and testing phases. And each subsequent release of the system adds function to the previous release until all designed functionality has been implemented. Incremental Model consist of many stages. In this project we are performing 3 stages to do analysis, design, implemen- tation, testing, deployment and maintainance. Each stage perform certain task and then execute it without concerning other two.

- ENCODING

A dictionary is initialized to contain the single-character strings corresponding to all the possible input characters (and nothing else except the clear and stop codes if they're being used). The algorithm works by scanning through the input string for successively longer substrings until it finds one that is not in the dictionary. When such a string is found, the index for the string less the last character (i.e., the longest substring that is in the dictionary) is retrieved from the dictionary and sent to output, and the new string (including the last character) is added to the dictionary with the next available code. The last input character is then used as the next starting point to scan for substrings. In this way, successively longer strings are registered in the dictionary and made available for subsequent encoding as single output values. The algorithm works best on data with repeated patterns, so the initial parts of a message will see little compression. As the message grows, however, the compression ratio tends asymptotically to the maximum.

- DECODING

The decoding algorithm works by reading a value from the encoded input and outputting the corresponding string from the initialized dictionary. At the same time it obtains the next value from the input, and adds to the dictionary the concatenation of the string just output and the first character of the string obtained by decoding the next input value. The decoder then proceeds to the next input value (which was already read in as the "next value" in the previous pass) and repeats the process until there is no more input, at which point the final input value is decoded without any more additions to the dictionary. In this way the decoder builds up a dictionary which is identical to that used by the encoder, and uses it to decode subsequent input values. Thus the full dictionary does not need be sent with the encoded data; just the initial dictionary containing the single-character strings is sufficient (and is typically defined beforehand within the encoder and decoder rather than being explicitly sent with the encoded data.)

6.1 Pseudocode

- ENCODING ALGORITHM:-
  - Initialize table with single character strings
  - P= first input character
  - While not end of input stream
  - C= next input character
  - IF P+C 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

  - DECODING 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

6.2 Output Screen

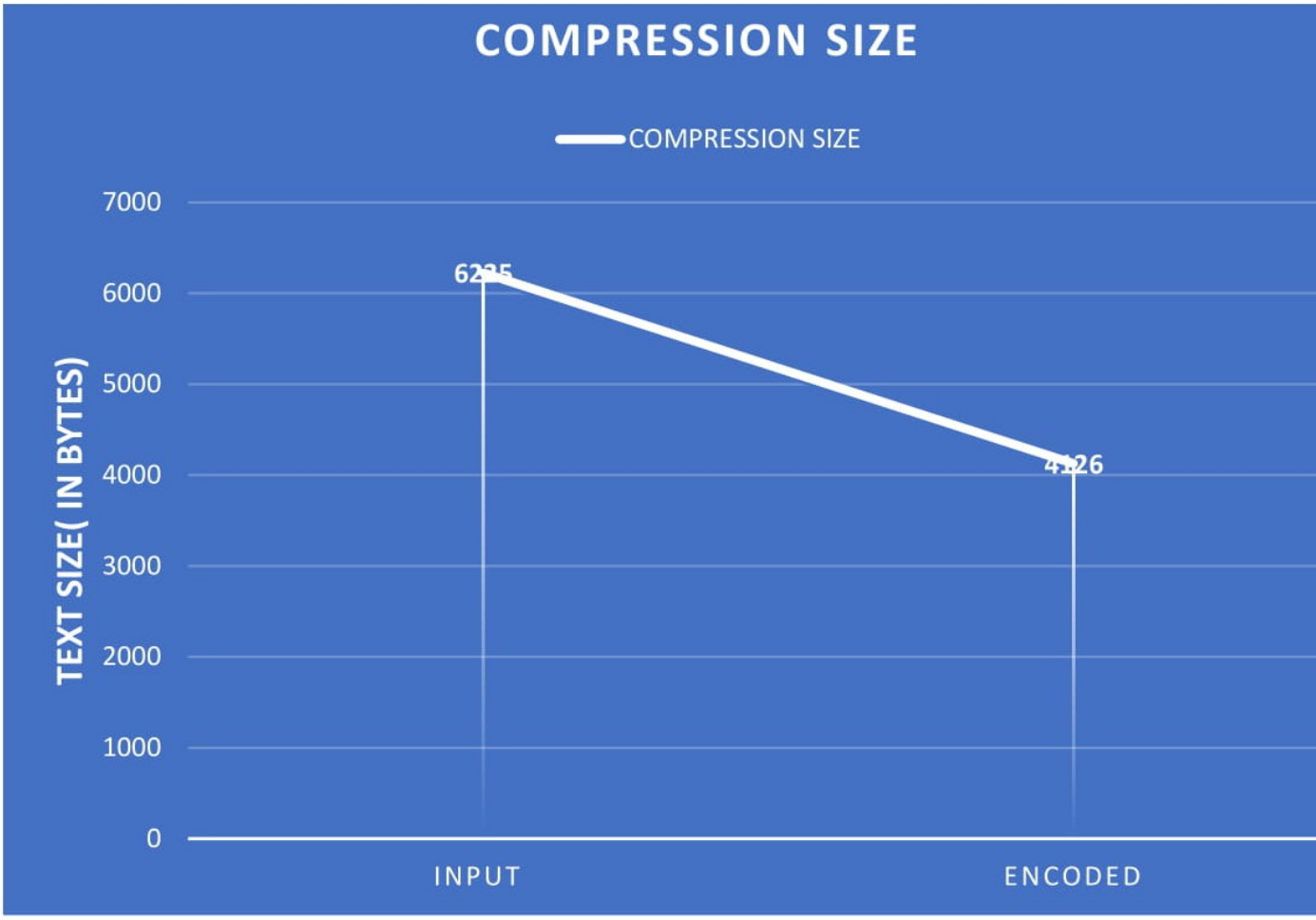

Figure 3: *COMPRESSION RATO GRAPH*

This shows the compression ratio of lempel-ziv-welch(LZW). The compression ratio expresses the difference between the file size of an uncompressed text file, and the size of the same text file when compressed. The compression ratio is equal to the size of the orginal file divided by the size of the compressed file. The ratio gives an indication of how much compression is achieved for a particular file. Most algorithms have a typical range of compression ratios that they can achieve over a variety of files. Because of this, it is actually more useful to look at an average compression ratio for an particular method.

(COMPRESSION RATIO= SIZE OF ORIGINAL FILE/ SIZE OF COMPRESSED FILE)

USING LZW,50-60 PERCENT OF COMPRESSION RATIO CAN BE ACHIEVED.

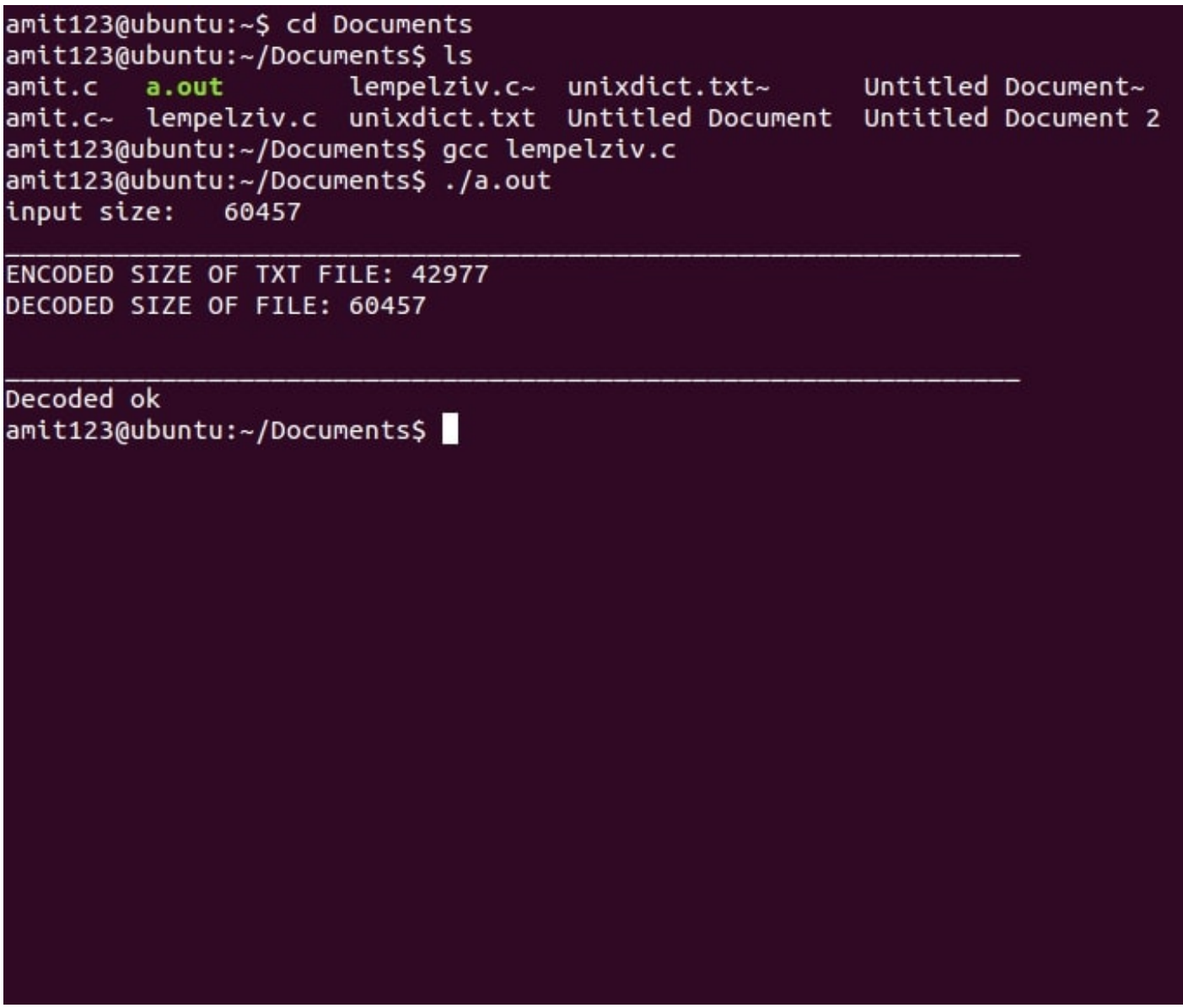

Figure 4: *OUTPUT SCREEN*

6.3 Result Analysis

This project addresses the necessity of the data size in storage devices. Problems related to data storage have been analyzed and found a better solution for it. Basically the LZW technique finally gives the compressed file that is half the size of the actual file and algorithm is designed such that it compress any text file better than any other compressing algorithm. The computational and space complexity of LZW data compression algorithm is purely depends on the effective implementation of data structure. LZW data structure implementation gives the better performance on the following operation- Insertion of new pattern to the dictionary, searching of a given pattern and returning the matching code word if it is present in the dictionary as a phrase. The characters were imported using the file handling named as unixdict.txt. Then we implemented the set of rules for encoding by using array and a dictionay created using structure. LZW has various advantages when being used to compress large text data, in English language which has high redundancy. It's a lossless compression algo, Hence no information lost, the disadvantage or we can say there is a case when dicitionary becomes too large. One approach is to throw the dicitionary away when it reaches a certain size. Useful only for a large amount of text data where redundancy is high.

7 Conclusion and Future Scope

In this age the main problem is to store the data without loosing it. The limited amount of store becames a big problem to the users. Hence,the compression technique has a great scope in todays environment. The use of LZW algorithm has a great value in compression technique.We have seen that with help of LZW how we can reduce the size of any text file and maximize the capacity of the hard drives.The LZW has found various Applications from text compression to multimedia.

The Lempel Ziv Algorithm although quite old but still is the standard algorithm for various proposes ranging from text compression to image and video compression. Although it's a Dictionary Based algorithm, but as we know the dictionary structure gets flatten out after compression process, In Future I would like to implement an LZW variant using Hash Sets and find out the performance with respect to LZW and it's a lossless in nature and incorporated as the standard algorithm for compression. LZW requires no prior information about the input data stream and also it can compress the input stream in one single pass. Another advantage of LZW its simplicity, allowing fast execution.

References

[1] R. M. Chezian, "International Journal of Advanced Research in Computer Science and Software Engineering Enhanced LZW (Lempel-Ziv-Welch) Algorithm by Binary Search with Multiple Dictionary to Reduce Time Complexity for Dictionary Creation in Encoding and Decoding," vol. 2, no. 3, 2012.

[2] M. Heggeseth, "Compression Algorithms: Huffman and LZW," pp. 19–32, 2003.

[3] M. A. Smadi and Q. A. B. U. Al-haija, "a Modified Lempel Ziv Welch Source Coding," vol. 61, no. 1, pp. 200–205, 2014.

[4] M. Nelson, "LZW Data Compression," *October*, no. 02, pp. 1–14, 1989.

[5] S. Mishra and A. Singh, "Image compression and enhancement by using the LZW and BHEPL," vol. 7, no. 5, pp. 235–238, 2017.