ISSN No. 0976-5697

Volume 5, No. 6, July-August 2014

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Redundant Reduced LZW (RRLZW) Technique of Lossless Data Compression

Md. Kamrul Islam
Department of Computer Science and Engineering,
Jessore University of Science and Technology
Jessore, Bangladesh

Md. Yasir Arafat
Department of Computer Science and Engineering,
Jessore University of Science and Technology,
Jessore, Bangladesh

Abstract: Technology is growing in every aspect but the basic needs to save the memory and time utilization still remains. Whenever we deal with any kind of data, we are bound to compress the data to minimize the space utilization. Data compression can be lossy or lossless. Some efficient lossless data compression techniques are Huffman coding, Run-length coding, Lempel- Ziv-Welch (LZW) coding. Lempel–Ziv–Welch (LZW) is a universal lossless data compression algorithm which works best on data with repeated patterns, so the initial parts of a message will see little compression. But LZW stores redundant string pattern which increase the number of entry of the dictionary thus increases the codeword size and this reduces performance of LZW technique. In this paper, a Reduced Redundant LZW (RRLZW) has been proposed to remove this redundant string pattern from the dictionary which increases the performance of LZW technique by decreasing the required memory to save data and also increasing the dictionary capacity. Also the performances of these algorithm and proposed system are measured and compared. The comparison shows that RRLZW performs better than LZW.

Keywords: Lossless compression, LZ77, LZ78, LZW.

I. INTRODUCTION

Compression is the art of representing the information in a compact form rather than its original or uncompressed form [1]. Data can be characters in a text file, numbers, image, waveforms or sequence of numbers. The basic technique of data compression:

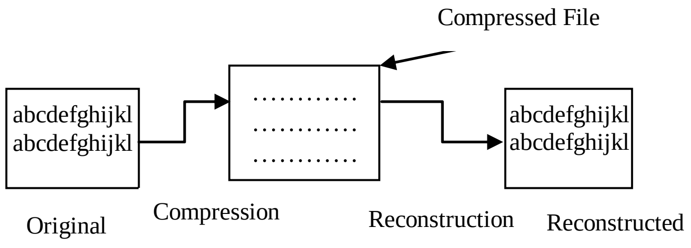


Figure 1: Basic Technique of Data Compression

Data Compression technique is divided into two broad categories lossless and lossy data compression techniques. Lossy compression techniques reconstruct the original message with loss of some information while lossless compression techniques reconstruct the original without any loss of data. These kinds of compression algorithms are called reversible compressions since the original message is reconstructed by the decompression process [2]. The statistical methods of lossless data compression capture redundancies using probabilistic methods [3] and convert characters into variable length strings of bits based on the frequency of use. They perform encoding by taking one character at a time. On the other hand the dictionary based lossless data compression encoders are also termed as substitution coders [4]. In these methods, a dictionary of certain length of input symbols is maintained and the repeated strings of characters of input data stream are replaced by the shorter code words. Both the encoder and decoder must have same dictionaries and greater the length of dictionary, more is the compression ratio. The entropy or information content provides the lower limit on the number of bits for a codeword [5]. The most popular dictionary based lossless compression methods are the Lempel Ziv (LZ1 or LZ77) [6] and LZ2 or LZ78 [7] class of dictionary based coders. There are many variants of the LZ coders namely LZSS, LZR, LZ7B, LZFG, LZT, LZW and LZJ etc. LZW maintains a dictionary at both the encoder and the decoder. The dictionary is continuously updated during encoding process [12].

In this paper, a modified LZW technique has been proposed. The proposed Redundant Reduced LZW (RRLZW) reduces the redundancy of dictionary entry and reducing the required bit for representing the entries in dictionary. Also the proposed technique increases the dictionary capacity. The next section in this paper describes the LZW efficient lossless data compression technique. In section III, the proposed RRLZW technique has been described. In section IV, the performances of these two algorithms have been examined with three samples and section V describes the conclusion.

II. DESCRIPTION OF LZW TECHNIQUE

LZW, a variation of LZ78 is named after Abraham Lempel, Jakob Ziv and Terry Welch [10], the scientists who developed this compression algorithm [11][6] proposed to remove the necessity of encoding the second element of the pair<i,c> as like LZ78. It is a lossless ‘dictionary based’ compression algorithm. Dictionary based algorithms scan a file for sequences of data that occur more than once. These sequences are then stored in a dictionary and within the compressed file, references are put where-ever repetitive data occurred. Each character has a code and index number in dictionary. Input data which we want to compress is read from file. Initially data is entered in buffer for searching in dictionary to generate its code. If there is no matching character found in dictionary. Then it will be entered as new character in dictionary and assign a code. If Character is in dictionary then its code will be generate. Output codes have less number of bits than input data. Thus

LZW compression replaces strings of characters with single codes.

The algorithm of LZW is divided into two parts encoding and decoding. The encoder would only send the index to the dictionary. In order to do this, the dictionary has to be primed with all the letters of the source alphabet. The input to the encoder is accumulated in a pattern p as long as p is contained in the dictionary. If the addition of another letter a results in a pattern p*a(*denotes concatenation)that is not in the dictionary, then the index of p is transmitted to the receiver ,the pattern p*a is added to the dictionary, and another pattern with the letter a is started.

The LZW decompression creates the same string table during decompression. It reads the input codeword sequentially add them to the decompressed string and add combining the previous string with next first string to the dictionary. Thus the in decompression the same encoding dictionary is constructed.

III. PROPOSED REDUNDANT REDUCED LZW (RRLZW)

The proposed Reduced Redundant LZW (RRLZW) for data compression is the modification of a famous data compression technique LZW technique. It reduces the redundant entry of the dictionary with replacing with the next entry of the dictionary. From LZW techniques, if aba is a new entry in the next entry in the dictionary then ab contains in the dictionary. So ab is a redundant entry in dictionary. The proposed method replaces the redundant entry ab with aba and thus reduces the number of entry of dictionary consequently reduce the codeword size. The basic principle of RRLZW method is

> If α and αβ are two input sequence, then if αβ is an entry of the dictionary, then α must be an entry in the dictionary. So we have no need to store α. Now, we can replace α with αβ. Here length of α must be greater than or equal to 2.

In this method a special command is also used that is 0. So the dictionary entry is started from 1. This special command is used when αβ and αμ entered in the dictionary or matched.

The encoding flowchart of RRLZW is given as figure 2 and the decoding flowchart is given as figure 3.

To illustrate the encoding technique, consider the following string to be encoded.
PPPQPPQQQ
Here the alphabet = {P,Q}, so the initial dictionary is as like Table 1

Table 1: Initial dictionary of PPPQPPQQQ

Index	Entry
P	1
Q	2

The encoder first gets the letter P. This pattern is already in the dictionary, so it concatenates the next symbol P to obtain PP. So next entry of the dictionary is PP and encoder output is the code of index P is 1. The dictionary looks like as Table 2

Table 2: Constructing the 3rd entry

Index	Entry
P	1
Q	2
PP	3

The encoder output: 1

Now, the current letter of the input is P which exists in the dictionary, so concatenate with the next letter to form PP which is also exists in the dictionary. Now concatenate the next letter with PP and get the pattern PPQ. But PP and PPQ both need not be store in the dictionary because PP is a sub string of PPQ. So according to the proposed RRLZW, PP is replaced by PPQ and this is the 3rd entry of the dictionary. And the dictionary looks like Table 3 and the encoder output will be 3.

Table 3: Replacing PP with PPQ in the 3rd entry

Index	Entry
P	1
Q	2
PPQ	3

The encoder output: **1, 3**

Now, the current letter is Q which exists in the dictionary, so concatenate the next with the next letter to form the pattern QP, which does not exist in the dictionary. So it adds to the dictionary in the 4th entry. Then the dictionary looks like Table 4 and the decoder output is 2.

Table 4: Constructing the 4th entry

Index	Entry
P	1
Q	2
PPQ	3
QP	4

The encoder output: **1, 3, 2**

Now the current letter is P that exists in the dictionary, then concatenate with the next letter P to form the pattern PP which is the substring of entry 3 which start with PP, so proceeds next to form the pattern PPQ which exists in the dictionary, so proceeds to the next letter and concatenate with the next letter and form the pattern PPQQ, that pattern does not exists in the dictionary. So replace PPQ with PPQQ in the 3rd entry. Then the dictionary looks like Table 5 and the encoder output is 3.

Table 5: Replacing PPQ with PPQQ

Index	Entry
P	1
Q	2
PPQQ	3
QP	4

Encoder output: **1, 3, 2, 3**

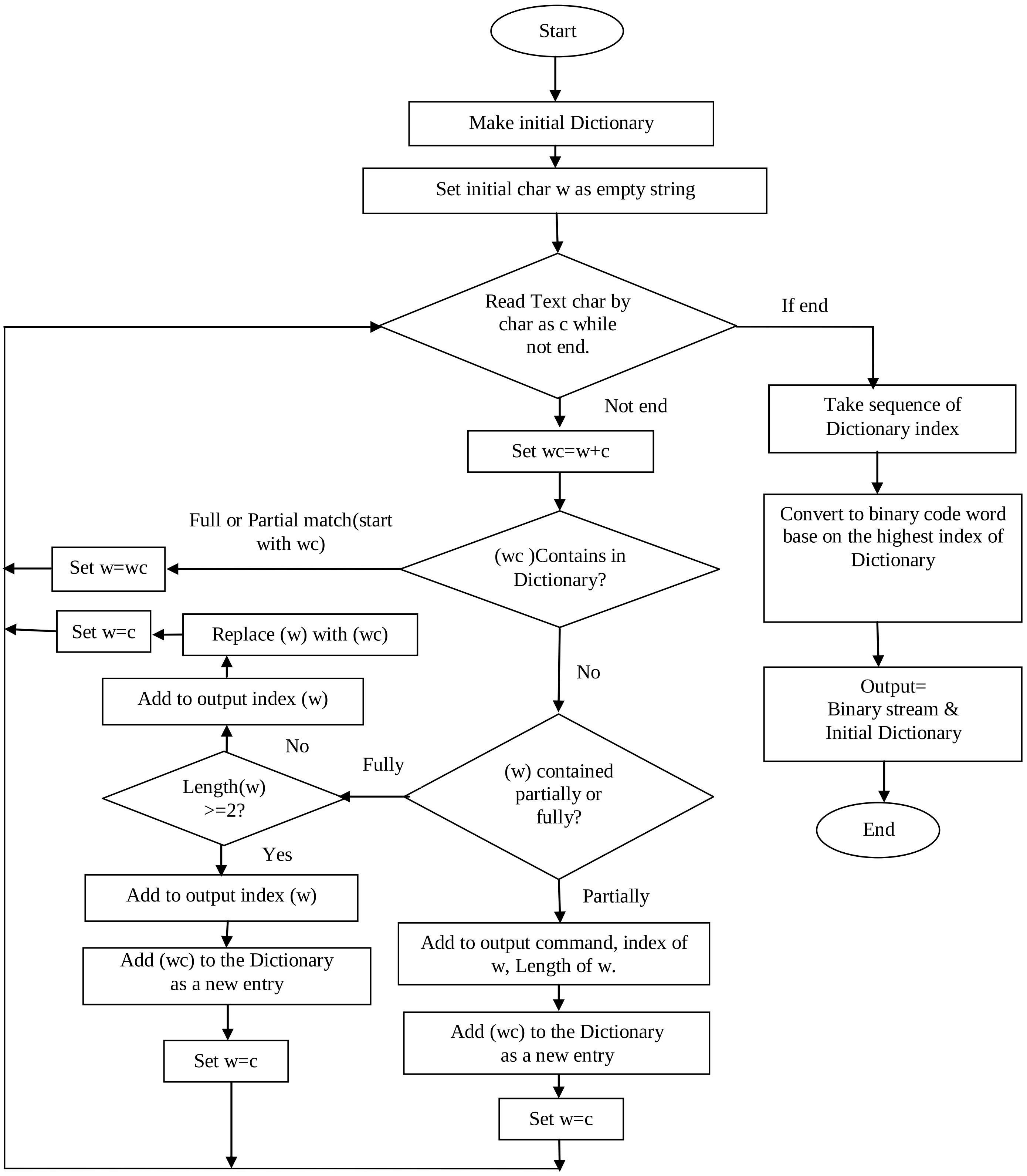
The current letter is Q that exists in the dictionary, so form QQ that does not in the dictionary, so add it to the next entry

to the dictionary and encoder output is 2. Then the dictionary is looks like as Table 6.

Table 6: Constructing the 5th entry

Index	Entry
P	1
Q	2
PPQQ	3
QP	4
QQ	5

Encoder output: 1,3,2,3,2
Now the present letter is Q this exists in the dictionary and it is the last character in the input.
So it is encoded as 2.
So finally the encoder output is: 1,3,2,3,2,2



Start
Make initial Dictionary
Set initial char w as empty string
Read Text char by char as c while not end.
If end
Not end
Take sequence of Dictionary index
Set wc=w+c
Convert to binary code word base on the highest index of Dictionary
Full or Partial match(start with wc)
Set w=wc
(wc)Contains in Dictionary?
Set w=c
Replace (w) with (wc)
No
Output= Binary stream & Initial Dictionary
Add to output index (w)
No
Fully
Length(w) >=2?
(w) contained partially or fully?
End
Yes
Add to output index (w)
Partially
Add (wc) to the Dictionary as a new entry
Add to output command, index of w, Length of w.
Add (wc) to the Dictionary as a new entry
Set w=c
Set w=c

Figure 2: RRLZW encoding flowchart

For the above example, the decoding technique works as follows

Decoder input is: 1,3,2,3,2,2

Decoder uses the same initial dictionary as like:

Table 7: Decoder initial dictionary

Index	Entry
P	1
Q	2

Now the decoder first input is 1 which corresponds to P which is exists in the dictionary, so decoder output is: P and the next input is 3 that is under construction but its first character is known that is P, so we concatenate them to form the pattern PP that is not exists in the dictionary, so add it to the dictionary to the next entry. Then the dictionary looks like as Table 8.

Table 8: Constructing 3rd entry while decoding

Index	Entry
P	1
Q	2
PP	3

Decoder output: **P**

Current input is 3 which is exists in the dictionary that corresponds to the pattern PP, so add it to the decoder output and concatenate with the next input corresponding value Q, formed PPQ and Replace the existing previous entry PP. then the dictionary is looks like Table 9.

Table 9: Replacing PP with PPQ in the 3rd entry while decoding

Index	Entry
P	1
Q	2
PPQ	3

Decoder output: **PPP**

Now the input is 2, corresponds to Q, that exists in the dictionary next input is 3 corresponds PPQ, now concatenating P to Q form the pattern QP that does not exists in the dictionary, so it is the next entry to the dictionary and decoded as Q, then the dictionary is looks like Table 10

Table 10: Constructing the 4th entry

Index	Entry
P	1
Q	2
PPQ	3
QP	4

Decoder output: **PPPQ**

Now the present entry is 3 corresponds to PPQ exists in the dictionary, and next input is 2 corresponds to Q, next entry would be PPQQ, that does not exists in the dictionary, so replace PPQ with PPQQ in the 3rd entry and decoded PPQ. Then the dictionary looks like as Table 11.

Table 11: Replacing PPQ with PPQQ

Index	Entry
P	1
Q	2
PPQQ	3
QP	4

Decoder output: **PPPQPPQ**

Now the present entry is 2 corresponds to Q, and next entry is 2 corresponds to Q, concatenating we get the pattern QQ, that is the next entry to the dictionary. Then the dictionary would be as following Table 12

Table 12: Constructing the 5th entry while decoding

Index	Entry
P	1
Q	2
PPQQ	3
QP	4
QQ	5

Decoder output: **PPPQPPQQ**

Now the input is 2 and this is the last entry of input and it corresponds to Q. so the decoder final output is as like which is the same as encoder input string.

PPPQQPPQQQ

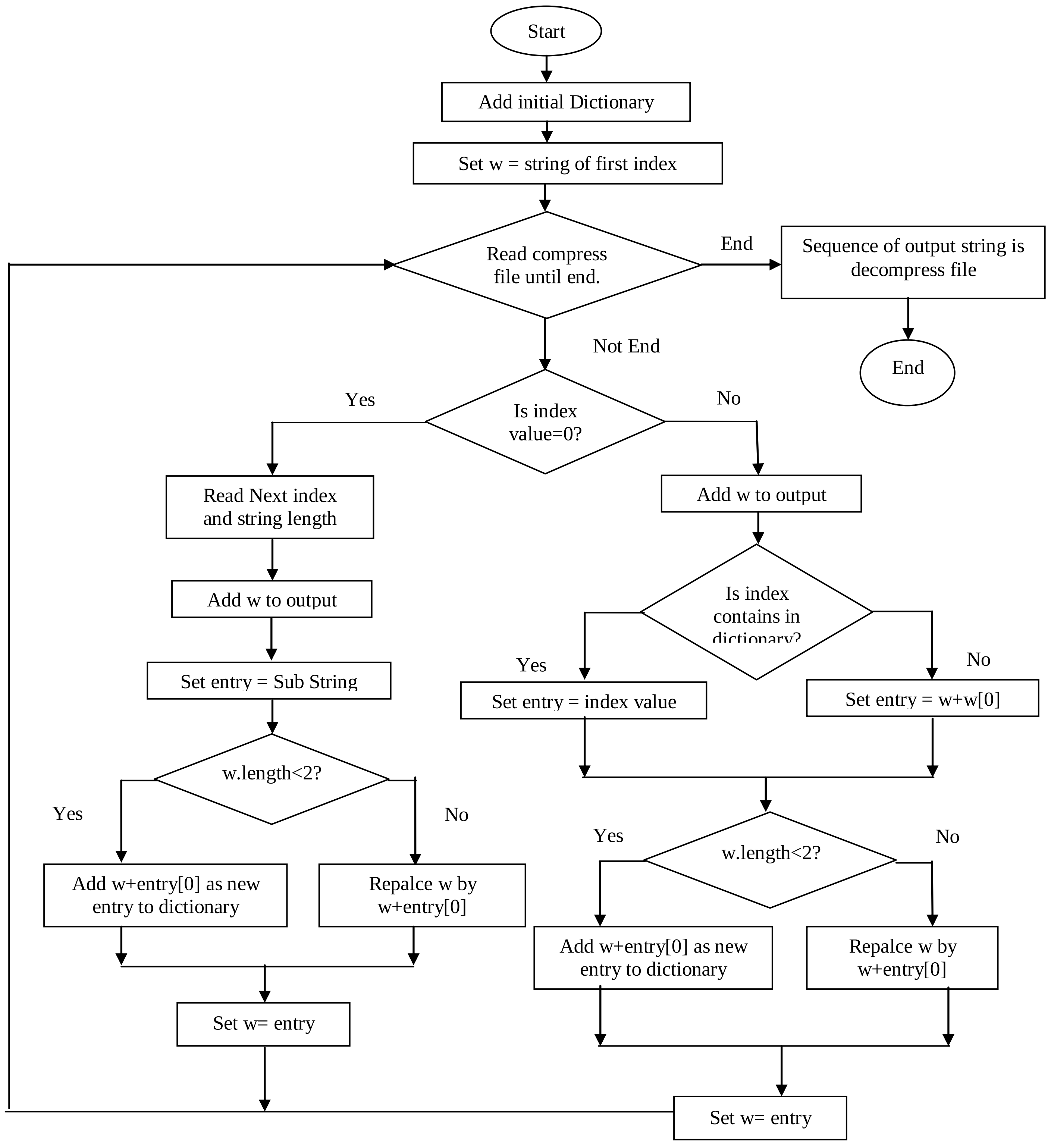


Figure 3: RRLZW decoding flowchart

IV. PERFORMANCE ANALYSIS

The performance of a data compression technique is measured here in terms of the encoded data rate using the following equation [11].

$$\text{Performance} = \frac{\textit{input file size} - \textit{compressed file size}}{\textit{input file size}}$$

$$= \frac{\textit{reduction}}{\textit{input file size}} * 100\%$$

From the above equation it is clear that if the reduction of required space is so high, the performance is also high.

To compare the performance of , the universal technique LZW and the proposed method RRLZW the following samples have been taken.

Sample 1:

wabba@wabba@wabba@wabba@woo@woo@woo

Sample2:

abcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghij abcdefghij

Sample 3:

asdfghjklasdfghjklasdfghjklasdfghjklasdfghjklasdfghjkl asdfghjklasdfghjklasdfghjklasdfghjklasdfghjklasdfghjklasdf ghjklasdfghjklasdfghjklasdfghjklasdfghjklasdfghjklasdfghjk lasdfghjkl

Sample 4:

Qwertyuioppoiuytreqqwertyuiopasdfghjklqwertyuiopas dfghjklzxcvbnmasdfghjklqwertyuiopzxcvbnmqwertyuiopaas dfghjklaaaaaawwwwwweeeeeeqwertyuiopasdfghjklzxcvbn maaaaaaaaaasssssasdfghjklasdfghjklqwertyuiopzxcvbnmqwe rtasdfghjklzxcv

Then the performance of three compression techniques Huffman coding, LZW, RRLZW is measured for the above four samples. In Table 13 the performance data has been given.

Table 13: Performance data for the above samples

Compression Technique	Performance			
	Sample 1	Sample 2	Sample 3	Sample 4
LZW	48.21	55.00	65.56	16.29
RRLZW	52.86	67.71	75.90	27.15

From the above Table 10, it is seen that in all samples the performance RRLZW is higher than LZW.

To realize this fact in the table, the following performance histogram is drawn

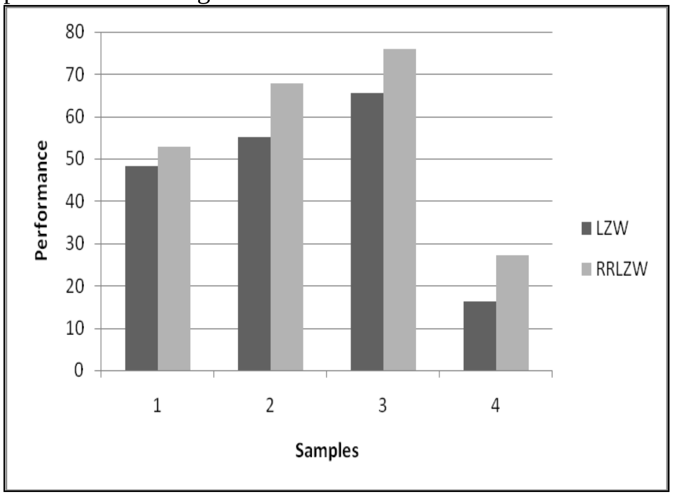


Figure 4: Performance histograms of LZW and RRLZW for the above samples

From this histogram graph it can be concluded that the proposed method (RRLZW) performs better than the standard LZW in terms of compression ratio.

V. CONCLUSION

Lossless data compression is a class of data compression algorithms that allows the exact original data to be reconstructed from the compressed data. LZW is one of the most popular and efficient technique for lossless data compression. But the technique suffers from redundancy in its dictionary. In this paper, a modified LZW called Redundant Reduced (RRLZW) has been proposed which reduces the redundant entry from the dictionary and thus reduce the code word size. This method also increases the capacity of its dictionary. To measure the performance of proposed RRLZW technique, four samples have been taken in section IV. For these samples, the performance of LZW and RRLZW have been measured which are recorded in the Table 13. From Figure 4, it is seen that the performance of the proposed technique RRLZW is greater than LZW. So it can be concluded here, the proposed RRLZW works better than formal LZW by reducing the redundant entry in its dictionary. If the complexity is increased than LZW but it would not be burden because our modern hardware speed is increased rapidly.

VI. REFERENCES

[1]. Bentley J.L., Sleator D.D., Tarjan R.E., and Wei V.K. “A Locally Adaptive Data Compression Scheme.”, Communications of ACM 29, pp. 320-330, April 1986.

[2]. Jung B., Burleson W. P., “A VLSI Systolic Array Architectures for Lempel_Ziv Based Data Compression”, Proceedings of IEEE Symposium on Circuits and Systems, 1994.

[3]. Milward M., Nunez-Yanez J. L. and Mulvaney D. , “Lossless Paralle Compression Systems”, Electronic Systems and Control Division Research Department of Electronic and Electrical Engineering, Loughborough University, LE11 3TU, UK, 2003.

[4]. Welsh T., “A Technique for high-Performance Data Compression”, IEEE Computer, vol. 17, pp 8-19, June 1984.

[5]. Huang D. T., “Fast and Efficient Algorithms for Text and Video Compression”, A PhD. Dissertation, Brown University, Rhode Island, 1997

[6]. Ziv J and Lempel A., “A universal algorithm for sequential data compression", IEEE Transactions on Information Theory, vol.-23, no. 3, pp. 337–343, Mar.1977.

[7]. Ziv J. and Lempel A., “Compression of Individual Sequences via Variable-Rate Coding", IEEE Transactions on Information Theory, vol.-23, no. 3, pp. 337–343, Mar.1978.

[8]. Yin Z. and Leung V. C.M., “A Proxy Architecture to Enhance the Performance of WAP 2.0 by Data Compression” , EURASIP Journal on Wireless Communications, 2005.

[9]. SAYOOD K., “Introduction to Data Compression”, Academic Press, SanDiego, CA, 2000.

[10]. Shannon C. E. , “A Mathematical Theory of Communication”, The Bell System Technical Journal, Vol. 27, pp. 379–423, 623–656, July-October, 1948.

[11]. Daitx F.F., Rosa, V.S., Costa, E., Flores, P., Bampi, S., “VHDL Generation of Optimized FIR Filetrs”, 2nd International Conference on Signals, Circuits and Systems, pp(s) 1 – 5,7-9 Nov, 2008.