ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 4, Issue 3, May 2015

Study of LZ77 and LZ78 Data Compression Techniques

Suman M. Choudhary, Anjali S. Patel, Sonal J. Parmar

Abstract*— Data Compression is defined as the science and art of the representation of information in a crisply condensed form. . For decades, Data compression is considered as critical technologies for the ongoing digital multimedia revolution .There are variety of data compression algorithms which are available to compress files of different formats. This paper provides a survey of different basic lossless data compression algorithms such as LZ77 and LZ78.

***Index Terms*— Data Compression, LZ77, LZ78.**

I. INTRODUCTION

Data compression reduces the amount of space needed to store data or reducing the amount of time needed to transmit data. The size of data is reduced by removing the excessive/repeated information. The goal of data compression is to represent a source in digital form with as few bits as possible while meeting the minimum requirement of reconstruction of the original. There are two kinds of compression techniques in terms of reconstructing the original source. They are called Lossless and lossy compression.

In lossless technique, we get original data after decompression. While in lossy technique, we don't get original data after decompression. The Lempel Ziv Algorithm is an algorithm for lossless data compression. This algorithm is an offshoot of the two algorithms proposed by Jacob Ziv and Abraham Lempel in their landmark papers in 1977 and 1978 which are LZ77 and LZ78[1].

II. LZ77

Jacob Ziv and Abraham Lempel have presented their dictionary-based scheme in 1977 for lossless data compression. The LZ77 compression algorithm is the most used compression algorithm, on which program like PkZip has their foundation along with a few other algorithms. LZ77 exploits the fact that words and phrases within a text file are likely to be repeated. When there is repetition, they can be encoded as a pointer to an earlier occurrence, with the pointer followed by the number of characters to be matched. It is a very simple technique that requires no prior knowledge of the source and seems to require no assumptions about the characteristics of the source.

In the LZ77 approach, the dictionary work as a portion of the previously encoded sequence. The encoder examines the input sequence by pressing into service of sliding window which consists of two parts: Search buffer and Look-ahead buffer. A search buffer contains a portion of the recently encoded sequence and a look-ahead buffer contains the next portion of the sequence to be encoded.

The algorithm searches the sliding window for the longest match with the beginning of the look-ahead buffer and outputs a pointer to that match. It is possible that there is no match at all, so the output cannot contain just pointers. In LZ77 the sequence is encoded in the form of a triple <o, l, c>, where 'o' stands for an offset to the match, 'l' represents length of the match, and 'c' denotes the next symbol to be encoded. A null pointer is generated as the pointer in case of absence of the match (both the offset and the match length equal to 0) and the first symbol in the look-ahead buffer i.e. (0,0,"character"). The values of an offset to a match and length must be limited to some maximum constants. Moreover the compression performance of LZ77 mainly depends on these values [2].

Algorithm

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 4, Issue 3, May 2015

While (look-Ahead Buffer is not empty)
{
Get a pointer (position, length) to longest match;
if (length > 0)
{
Output (position, Longest match length, next symbol);
Shift the window by (length+1) positions along;
}
Else
{
Output (0, 0, first symbol in the look-ahead buffer);
Shift the window by 1 character along;
}
}

Example

	search buffer									look-ahead buffer										
…a	c	C	a	b	r	a	c	a	d	a	b	r	a	r	r	a	r	r	a	c…

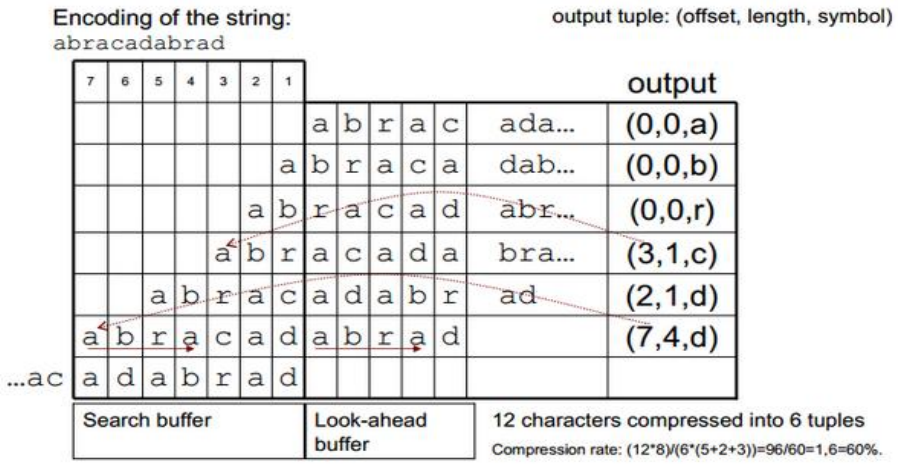

Fig: 1 Example of LZ77 Algorithm

Shortcomings of LZ77 Algorithm

- In the original LZ77 algorithm; Lempel and Ziv proposed that all string be encoded as a length and offset, even string founds no match.
- In LZ77, search buffer is thousands of bytes long, while the look-ahead buffer is tens of byte long.
- The encryption process is time consuming due to the large number of comparison done to find matched pattern.
- LZ77 doesn't have its external dictionary which cause problem while decompressing on another machine. In this algorithm whenever there is no match of any strings it encoded that string as a length and offset which will take more space and this unnecessary step also increases time taken by the algorithm.

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 4, Issue 3, May 2015

III. LZ78

The LZ78 is a dictionary-based compression algorithm that maintains an explicit dictionary. The encoded output consists of two elements: an index referring to the longest matching dictionary entry and the first non-matching symbol. The algorithm also adds the index and symbol pair to the dictionary. When the symbol is not yet found in the dictionary, the codeword has the index value 0 and it is added to the dictionary as well. With this method, the algorithm constructs the dictionary.

LZ78 algorithm has the ability to capture patterns and hold them indefinitely but it also has a serious drawback. The dictionary keeps growing forever without bound. There are various methods to limit dictionary size. the easiest one is to stop adding entries and continue like a static dictionary coder or to throw the dictionary away and start from scratch after a certain number of entries has been reached [3].

Algorithm

```
w := NIL;
While (there is input){
K  := next symbol from input;
If (wK exists in the dictionary) {
w := wK;
}
Else {
Output (index(w), K);
Add wK to the dictionary;
w := NIL;
}
}
```

Example

Encode (i.e., compress) the string **ABBCBCABABCAABCAAB** using the LZ78 algorithm.

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 4, Issue 3, May 2015

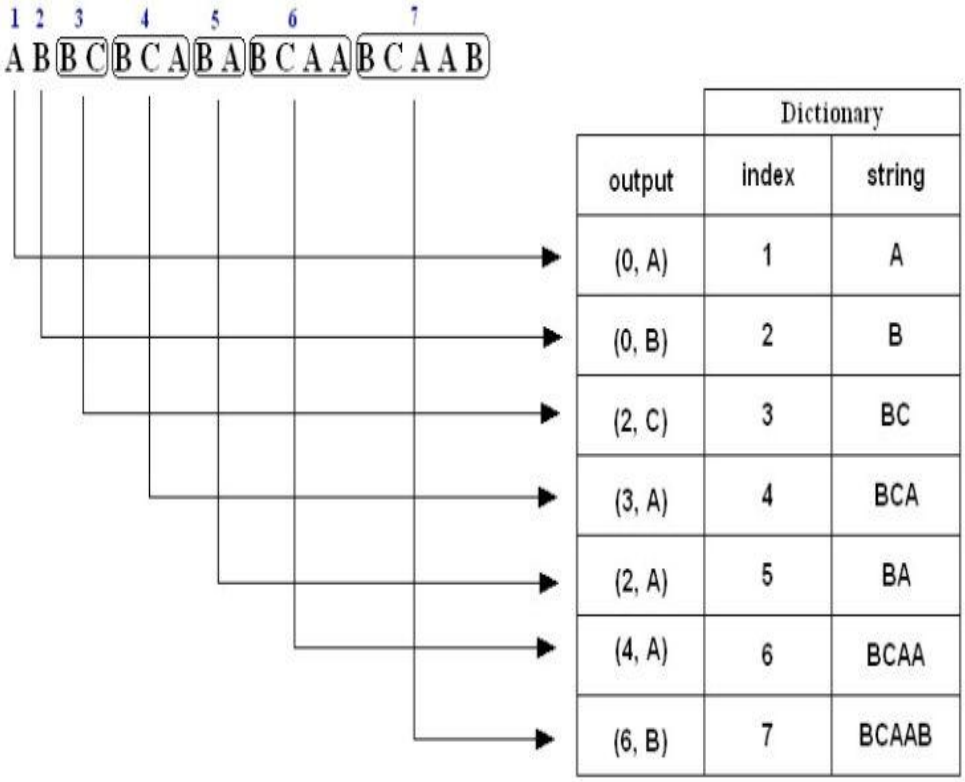

Fig: 2 Example of LZ78 Algorithm

The compressed message is: (0,A)(0,B)(2,C)(3,A)(2,A)(4,A)(6,B).

IV. COMPARISON OF LZ77 AND LZ78

Table 1: Comparison between LZ77 and LZ78 Algorithms

LZ77	LZ78
1) The LZ77 algorithm works on past data 2) The LZ77 is slower 3) The output format of lz77 is triplet <o, l, c> Where o=offset, l=length of the match, c=next symbol to be encoded. 4) Application: This algorithm is open source and used in what is widely known as ZIP, and by the formats PNG, TIFF, PDF and many others. LZ77 is used in gzip, Squeeze, LHA, PKZIP, and ZOO.	1) The LZ78 algorithm attempts to work on future data. 2) LZ78 is faster than LZ77 3) And the output format of lz78 is pair <i,c>.Where i=index and c=next character. 4) Application: Lz78 has various applications in the field of information theory such as random number generation, hypothesis testing, parsing of the string etc.LZ78 is used in compress, GIF, CCITT (modems), ARC, PAK.

The LZ77 algorithm works on past data whereas LZ78 algorithm attempts to work on future data. It does this by forward scanning the input buffer and matching it against a dictionary it maintains. It will scan into the buffer until it cannot find a match in the dictionary. At this point it will output the location of the word in the dictionary, if one is available, the match length and the character that caused a match failure. The resulting word is then added to the dictionary.

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 4, Issue 3, May 2015

LZ78, like LZ77, has slow compression but very fast decompression. LZ78 is faster than LZ77 but doesn't always achieve as high a compression ratio as LZ77. The biggest advantage LZ78 has over the LZ77 algorithm is the reduced number of string comparisons in each encoding step [4].

V. CONCLUSION

Lempel Ziv scheme which is a dictionary based technique is divided into two families: one derived from LZ77 and the other derived from LZ78. The study of two main dictionary based lossless compression algorithms i.e. LZ77 and LZ78 for text data is carried out. After studying and comparing LZ77 and LZ78 algorithms, we found that LZ78 is better and faster than LZ77 algorithm.

ACKNOWLEDGMENT

We are thankful to our parents and friends for motivating us to write research paper. We are very much thankful to our professor Kruti J. Dangarwala for guiding us to write the research paper.

REFERENCES

[1] Data compression, I. Pu , CO0325 2004 by Goldsmiths, University of London.

[2] TEXT COMPRESSION ALGORITHMS - A COMPARATIVE STUDY, S. Senthil and L. Robert, ICTACT JOURNAL ON COMMUNICATION TECHNOLOGY, DECEMBER 2011, VOLUME: 02, ISSUE: 04.

[3] The Lempel Ziv Algorithm Christina Zeeh Seminar"Famous Algorithms" January 16, 2003.

[4] The Data Compression book, second edition by Mark Nelson and Jean-Loup Gailly.

AUTHOR BIOGRAPHY

Suman M. Choudhary, B. E. from Shri S'ad Vidya Mandal Institute Of Technology, Bharuch, Gujarat , India.

.

Anjali S. Patel, B. E. from Shri S'ad Vidya Mandal Institute Of Technology, Bharuch, Gujarat , India.

Sonal J. Parmar, B. E. from Shri S'ad Vidya Mandal Institute Of Technology, Bharuch, Gujarat , India.