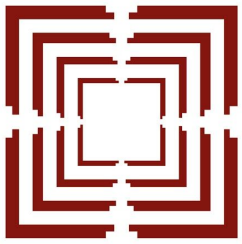


Ahmedabad University

LZ77 - Lempel Ziv 77

By,

Group - 13

Saloni Chudgar - 1641013

Abhi Patel - 1641021

Nildeep Jadav - 1641024

Manav Chotalia - 1641036

Vishal Saha - 1741004

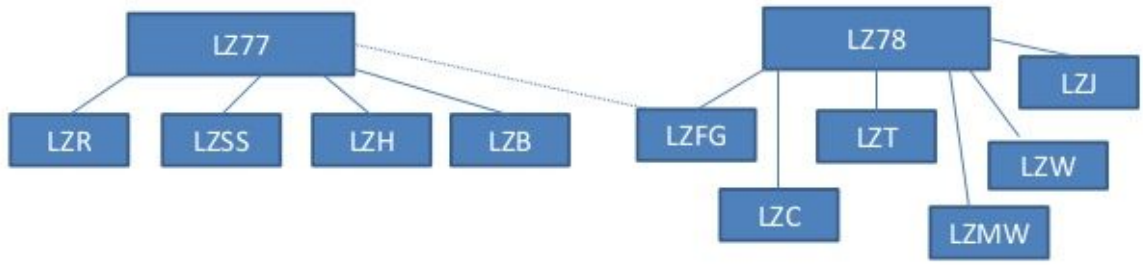
Introduction:

LZ77 is a data compression algorithm published by Abraham Lempel and Jacob Ziv in 1977[1]. It is also known as LZ1. Along with LZ78, it forms the basis for many variations including LZW, LZSS, LZMA, and others. These algorithms formed the basis of several ubiquitous compression schemes, including GIF and the DEFLATE algorithm used in PNG and ZIP. Theoretically, it is a dictionary coder[A dictionary coder, also sometimes known as a substitution coder, is a class of lossless data compression algorithms which operate by searching for matches between the text to be compressed and a set of strings contained in a data structure (called the 'dictionary') maintained by the encoder. When the encoder finds such a match, it substitutes a reference to the string's position in the data structure.]. LZ77 maintains a sliding window during compression. The window consists of two parts:

1. a search buffer that contains a portion of the recently encoded sequence
2. a look-ahead buffer that contains the next portion of the sequence to be encoded

LZ77 algorithms achieve compression by replacing repeated occurrences of data with references to a single copy of that data existing earlier in the uncompressed data stream. A match is encoded by a pair of numbers called a lengthy-distance pair, which is equivalent to the statement "each of the next length characters is equal to the characters exactly distance characters behind it in the uncompressed stream". (The distance is sometimes called the offset instead.) To spot matches, the encoder must keep track of some amount of the most recent data, such as the last 2 kB, 4 kB, or 32 kB. The structure in which this data is held is called a sliding window, which is why LZ77 is sometimes called sliding-window compression. The encoder needs to keep this data to look for matches, and the decoder needs to keep this data to interpret the matches the encoder refers to. The larger the sliding window is, the longer back the encoder may search for creating references.

Limpel Ziv Algorithm Family



APPLICATIONS:
- ZIP
- GZIP
- STACKER

APPLICATIONS:
- GIF
- V.42
- COMPRESS

Fig 1: Limpel Ziv Algorithm Family

Algorithm :

Compression :

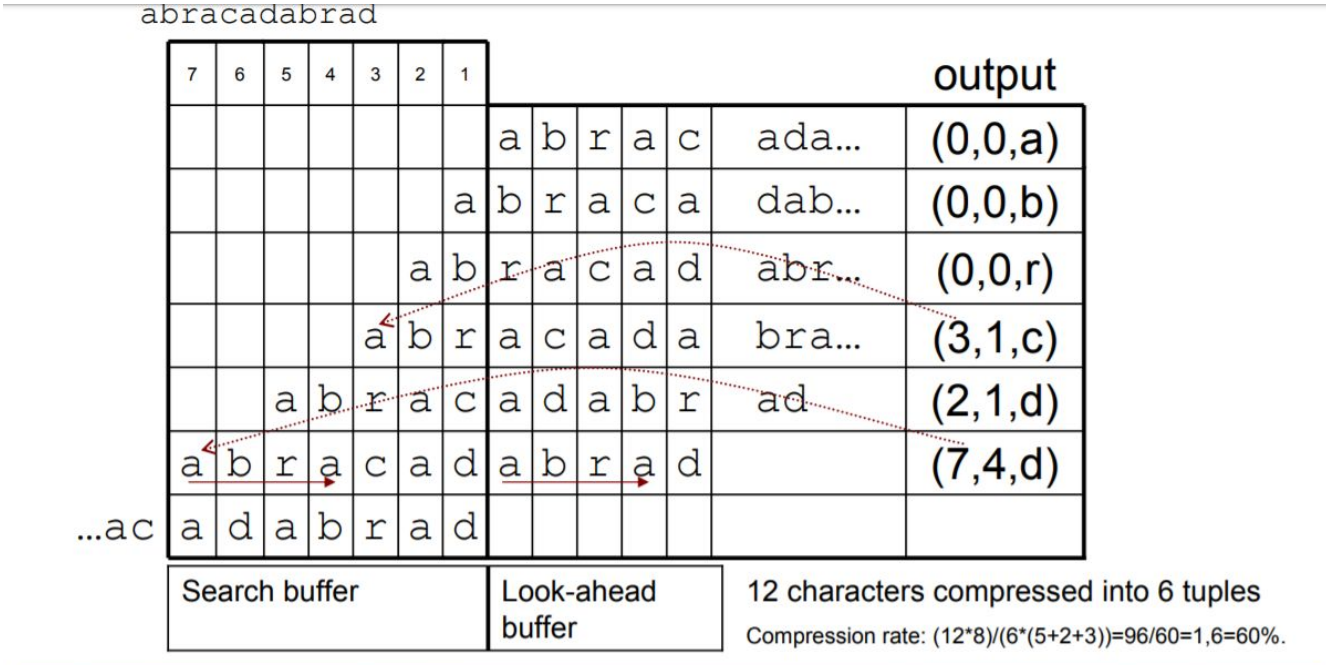
```
while( lookAheadBuffer not empty )
{
    get a pointer (position, match) to the longest match
    in the window for the lookAheadBuffer;

    output a (position, length, char()) triple;
    shift the window length+1 characters along;
}
```

De - Compression :

```
for each token (offset, length, symbol)
   if offset = 0 then
        print symbol;
   else
        go reverse in previous output by offset characters and copy
        character wise for length symbols;
        print symbol;
   fi
next
```

Example :



Results :

- Compression of "**alice29.txt**"

```
vishal@vishal-Vostro-15-3568:~$ ./a.out e alice29.txt compalice.txt

Compressing alice29.txt to compalice.txt

Size of the original file is 148575 Bytes

Size of the compressed file is 69059 Bytes

EXECUTION TIME = 0.008849
vishal@vishal-Vostro-15-3568:~$ ./a.out d compalice.txt decompalice.txt

Decompressing compalice.txt to decompalice.txt

Size of the compressed file is 69059 Bytes

Size of the decompressed file is 148575 Bytes

EXECUTION TIME = 0.003244
vishal@vishal-Vostro-15-3568:~$
```

➔ The compression ratio of the "alice29.txt" above is **2.1514**.

- Compression of "**server.c**"

```
vishal@vishal-Vostro-15-3568:~$ ./a.out e server.c compserver.txt

Compressing server.c to compserver.txt

Size of the original file is 8740 Bytes

Size of the compressed file is 2730 Bytes

EXECUTION TIME = 0.000650
vishal@vishal-Vostro-15-3568:~$ ./a.out d compserver.txt serverdecomp.c

Decompressing compserver.txt to serverdecomp.c

Size of the compressed file is 2730 Bytes

Size of the decompressed file is 8740 Bytes

EXECUTION TIME = 0.000226
vishal@vishal-Vostro-15-3568:~$
```

➔ The compression ratio of the 'server.c" code file above is **3.014**.

- Compression of "**Los Angeles.mp3**"

```
vishal@vishal-Vostro-15-3568:~$ ./a.out e Los\ Angeles.mp3 compmp3.txt

Compressing Los Angeles.mp3 to compmp3.txt

Size of the original file is 9340897 Bytes

Size of the compressed file is 10201355 Bytes

EXECUTION TIME = 0.586274
vishal@vishal-Vostro-15-3568:~$ ./a.out d compmp3.txt decomp.mp3

Decompressing compmp3.txt to decomp.mp3

Size of the compressed file is 10201355 Bytes

Size of the decompressed file is 9340897 Bytes

EXECUTION TIME = 0.106526
vishal@vishal-Vostro-15-3568:~$
```

➔ The compression ratio of the "Los Angeles.mp3" above is **0.9156**.

Conclusion :

LZ77 with Different text File.

File Name	Size	Compression Ratio
alice29.txt	145kB(original)	2.15
	67.4kB (compressed)	
asyoulik.txt	122kB(original)	2
	61kB (compressed)	
lcet10.txt	409kB(original)	2.24
	182kB (compressed)	

General Comparision with Run-length - Shannon - Huffman coding:

Input File (.txt)	Input File Size (KB)	Characteristics	Run length Coding	Shannon Fano Coding	Huffman Coding	Repeated Huffman Coding				
						Pass 1	Pass 2	Pass 3	Pass 4	Pass 5
alice29	148	Output File Size	283 KB	3.96 KB	91.5 KB	91.5 KB	102 B	12 B	3 B	2 B
		Compression Ratio	1.91	0.03	0.62	0.62	0.0011	0.12	0.25	0.67
		Avg. Code Length	-	6.22	9.78	9.78	6.32	4.32	2.4	1
		Standard Deviation	-	0.17	14.65	14.65	0.58	0.22	0.24	0
asyoulik	122	Output File Size	233 KB	96.8 KB	81.8 KB	81.8 KB	361 B	3 B	2 B	-
		Compression Ratio	1.91	0.79	0.67	0.67	0.0043	0.0083	0.67	-
		Avg. Code Length	-	6.12	9.13	9.13	7.23	2.67	1	-
		Standard Deviation	-	0.10	12.82	12.82	1.12	0.22	0	-
lcet10	416	Output File Size	387 KB	341 KB	651 KB	651 KB	297 B	20 B	4 B	2 B
		Compression Ratio	0.93	0.82	1.56	1.56	0.00045	0.067	0.20	0.5
		Avg. Code Length	-	6.46	28.10	28.10	8.16	4.77	2.86	1.67
		Standard Deviation	-	0.25	401.79	401.79	4.10	0.18	0.12	0.22
plrabn12	470	Output File Size	916 KB	366 KB	785 KB	785 KB	1.93 KB	4 B	2 B	-
		Compression Ratio	1.95	0.78	1.67	1.67	0.0025	0.0020	0.5	-
		Avg. Code Length	-	6.38	33.14	33.14	9.31	3	1	-
		Standard Deviation	-	0.24	449.86	449.86	5.62	0	0	-

References :

[1] - Ziv, Jacob, and Abraham Lempel. "A universal algorithm for sequential data compression." IEEE Transactions on information theory 23.3 (1977): 337-343.

[2] - Sailunaz, Kashfia & Kotwal, Mohammed & Huda, Mohammad. (2014). Data Compression Considering Text Files. International Journal of Computer Applications. 90. 10.5120/15765-4456.