LZ77 - Lempel Ziv 77

Group 5,
Saloni Chudgar (1641013)
Abhi Patel (1641021)
Nildeep Jadav (1641024)
Manav Chotalia (1641036)
Vishal Saha (1741004)

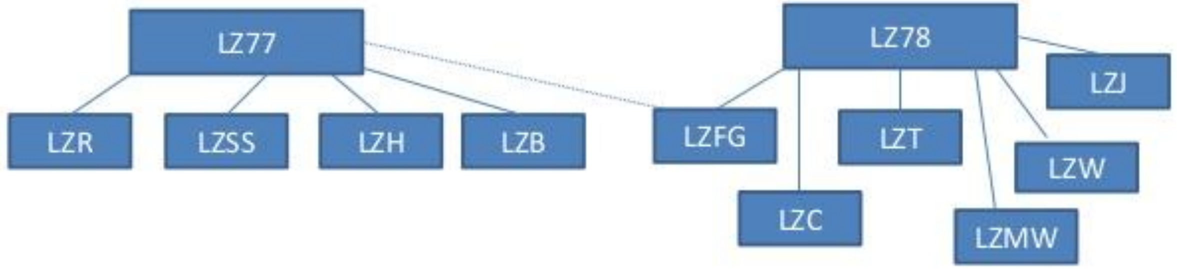
Data Compression

- Data compression deals with encoding information in as few bits as is possible or reasonable.
- Used to reduce resource requirements like
  - Storage - Hard disk, etc
  - Transmission - Time, bandwidth, etc
  - Processing - Memory(Main, Cache)
- Two Major Types
  - Lossy Compression
  - Lossless Compression
- Level of Compression can be measured by two methods
  - Compression Ratio - c/u
  - Bit Rate

LZ 77

- Lossless Data Compression
- Used for sequential data
- The encoder examines the input sequence through a sliding window
- The window consists of two parts:
  - a search buffer that contains a portion of the recently encoded sequence
  - a look-ahead buffer that contains the next portion of the sequence to be encoded
- The basic idea behind this **dictionary-based compressor** is to replace an occurrence of a particular phrase or group of bytes in a piece of data with a reference to a previous occurrence of that phrase.

Limpel Ziv Algorithm Family



APPLICATIONS:
- ZIP
- GZIP
- STACKER

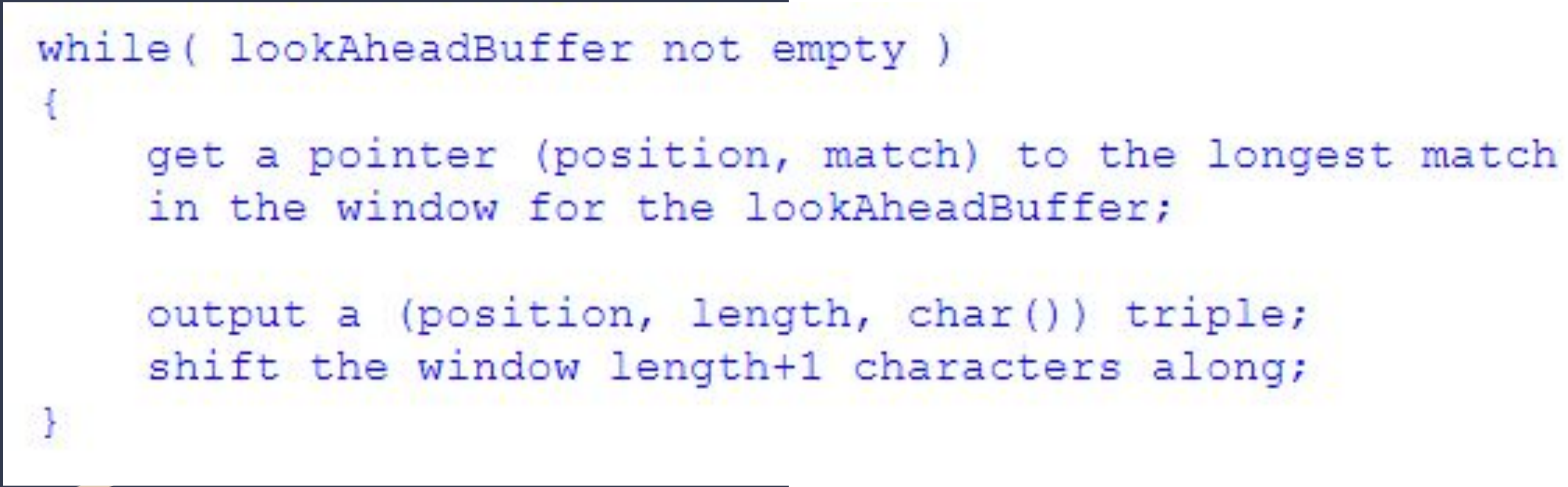
APPLICATIONS:
GIF
V.42
COMPRESS

Fig 1: Limpel Ziv Algorithm Family

Working Principle – Compression

- The compression algorithm searches the window for the longest match with the beginning of the lookahead buffer and then outputs a pointer to that match.
- Because even a 1-byte match might not be found, the output cannot only contain pointers.
- The compression algorithm solves this problem by outputting after the pointer the first byte in the lookahead buffer after the match.
- If no match is found, the algorithm outputs a null-pointer and the byte at the coding position.

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;
}
```

(Source : https://cs.stanford.edu/people/eroberts/courses/soco/projects/data-compression/lossless/lz77/algorithm.htm)

Algorithm- Compression

- Set the coding position to the beginning of the input stream.
- Find the longest match in the window for the lookahead buffer.
- If a match is found, output the pointer P. Move the coding position (and the window) L bytes forward.
- If a match is not found, output a null pointer and the first byte in the lookahead buffer. Move the coding position (and the window) one byte forward.
- If the lookahead buffer is not empty, return to step 2.

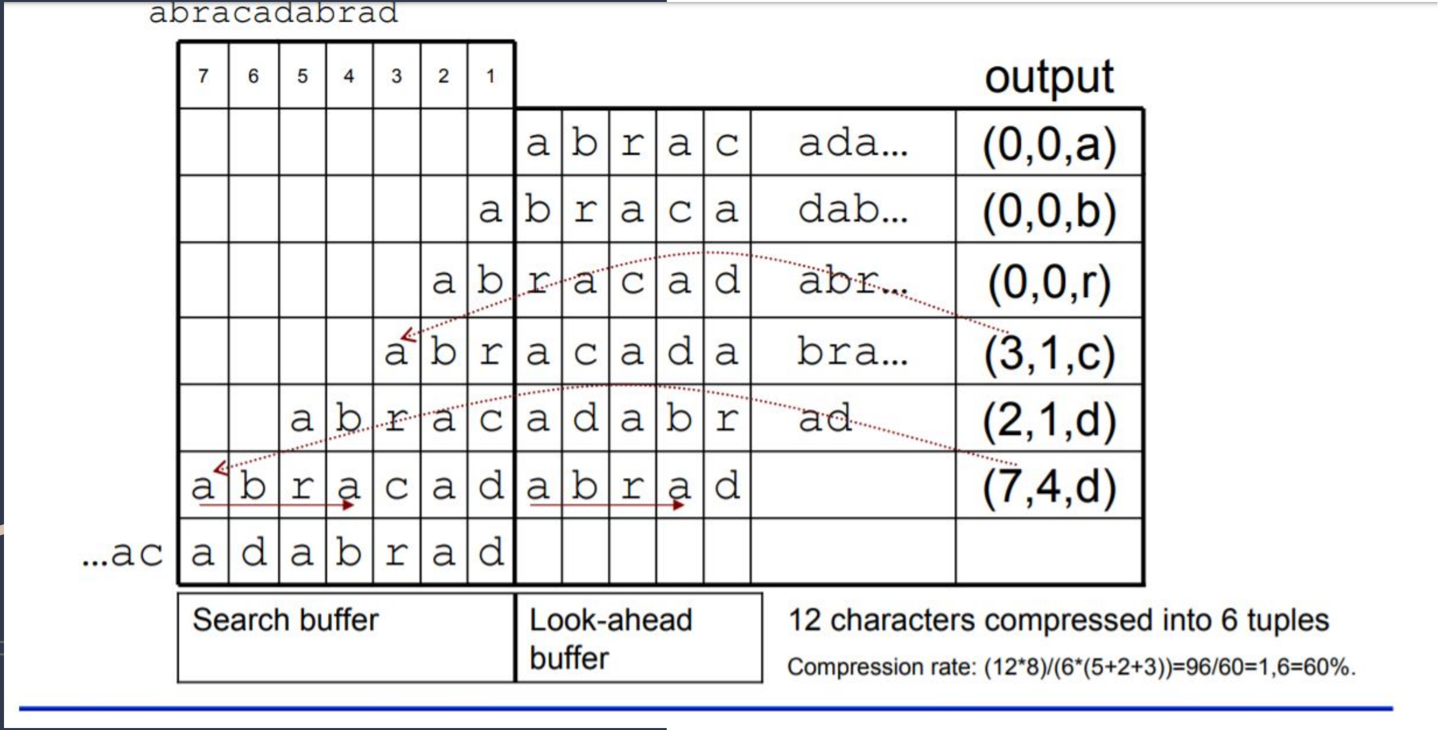
Algorithm- Compression : Terms Used

- Offset (o) - The distance of the pointer from the look-ahead buffer is called the offset.
- Length of Match (l) - The number of consecutive symbols in the search buffer that match consecutive symbols in the look-ahead buffer, starting with the first symbol, is called the length of the match.
- the encoder encodes it with a term <o, l, c>
- Here (c) is the codeword corresponding to the symbol in the look-ahead buffer that follows the match.

Algorithm-Compression - Result

- The result of compression, a series of bytes and optional metadata that indicates whether that byte is preceded by some sequence of bytes that is already in the output.

Example – Compression



abracadabrad

	7	6	5	4	3	2	1							output
								a	b	r	a	c	ada...	(0,0,a)
							a	b	r	a	c	a	dab...	(0,0,b)
						a	b	r	a	c	a	d	abr...	(0,0,r)
					a	b	r	a	c	a	d	a	bra...	(3,1,c)
			a	b	r	a	c	a	d	a	b	r	ad	(2,1,d)
	a	b	r	a	c	a	d	a	b	r	a	d		(7,4,d)
...ac	a	d	a	b	r	a	d							

Search buffer

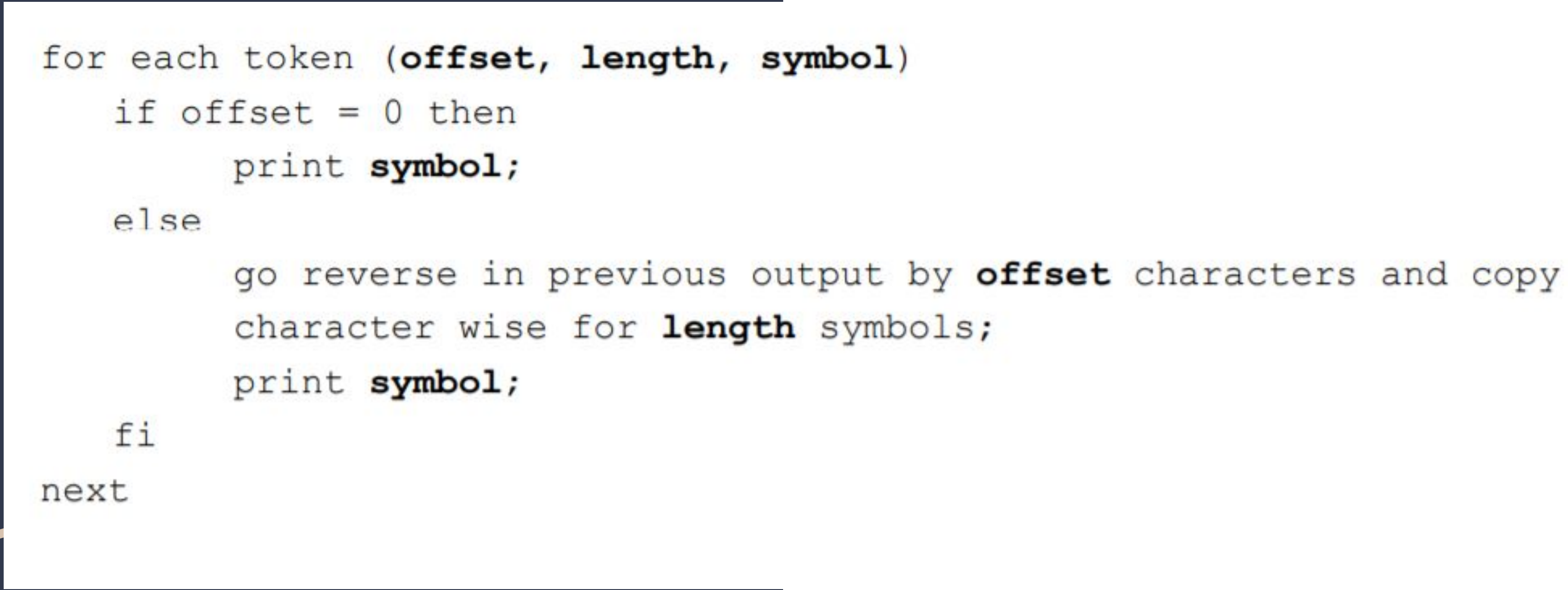
Look-ahead buffer

12 characters compressed into 6 tuples

Compression rate: (12*8)/(6*(5+2+3))=96/60=1,6=60%.

Sourc

Algorithm – Decompression



```
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
```

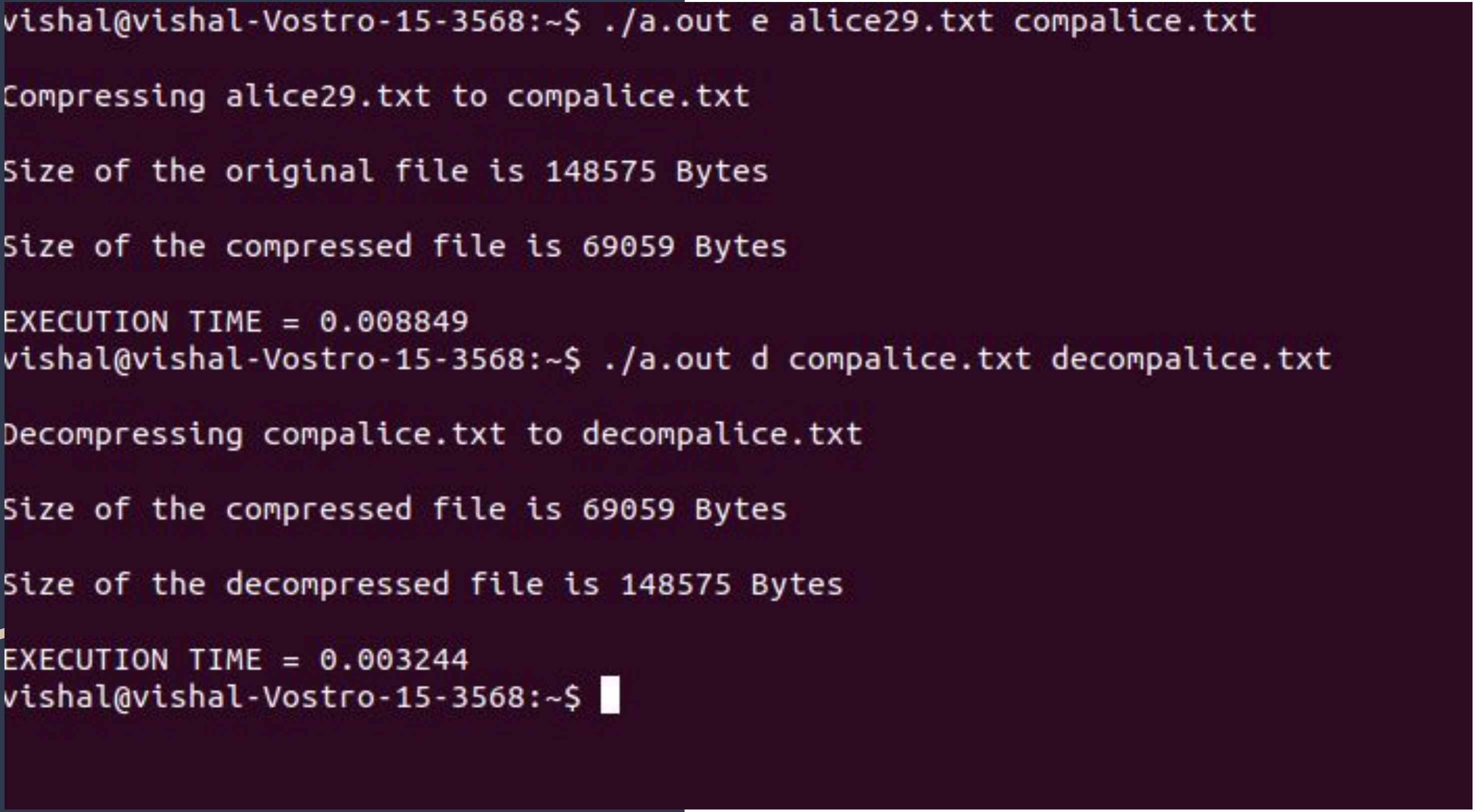
Algorithm- Decompression

- The decompression algorithm processes the compressed stream from start to end.
- For each null pointer, it appends the associated byte directly to the end of the output stream.
- For each non-null pointer, it reads back to the specified offset from the current end of the output stream and appends the specified number of bytes to the end of the output stream.

Importance

- 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.

Implementation

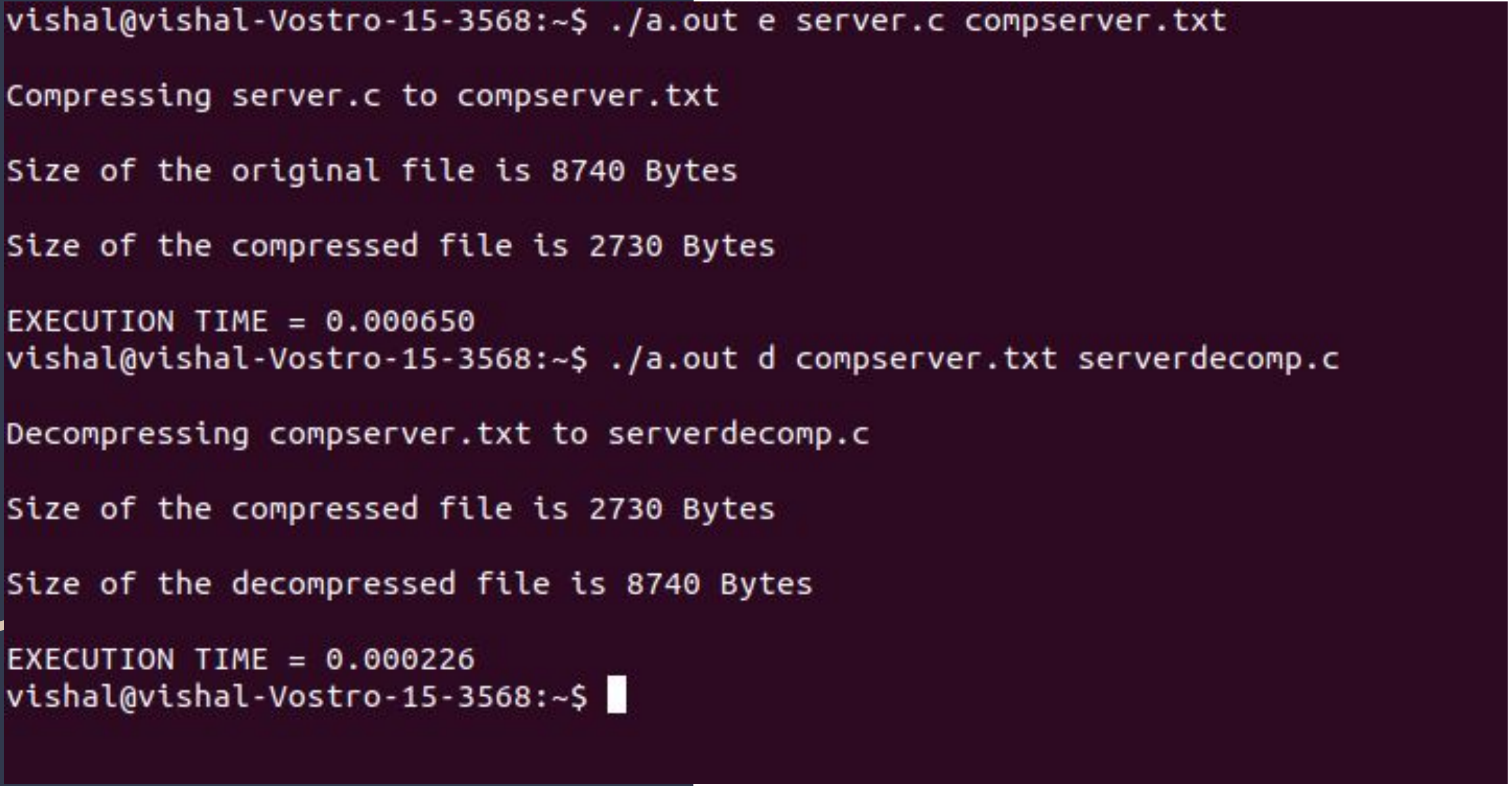


```
vishal@vishal-Vostro-15-3568:~$ ./a.out e alice29.txt compalice.txt

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:~$ █
```

Implementation

- The above image is for the compression of text file.
- The compression ratio of the above is 2.1514. I.e 54% of the file is compressed and 46% remains

Implementation

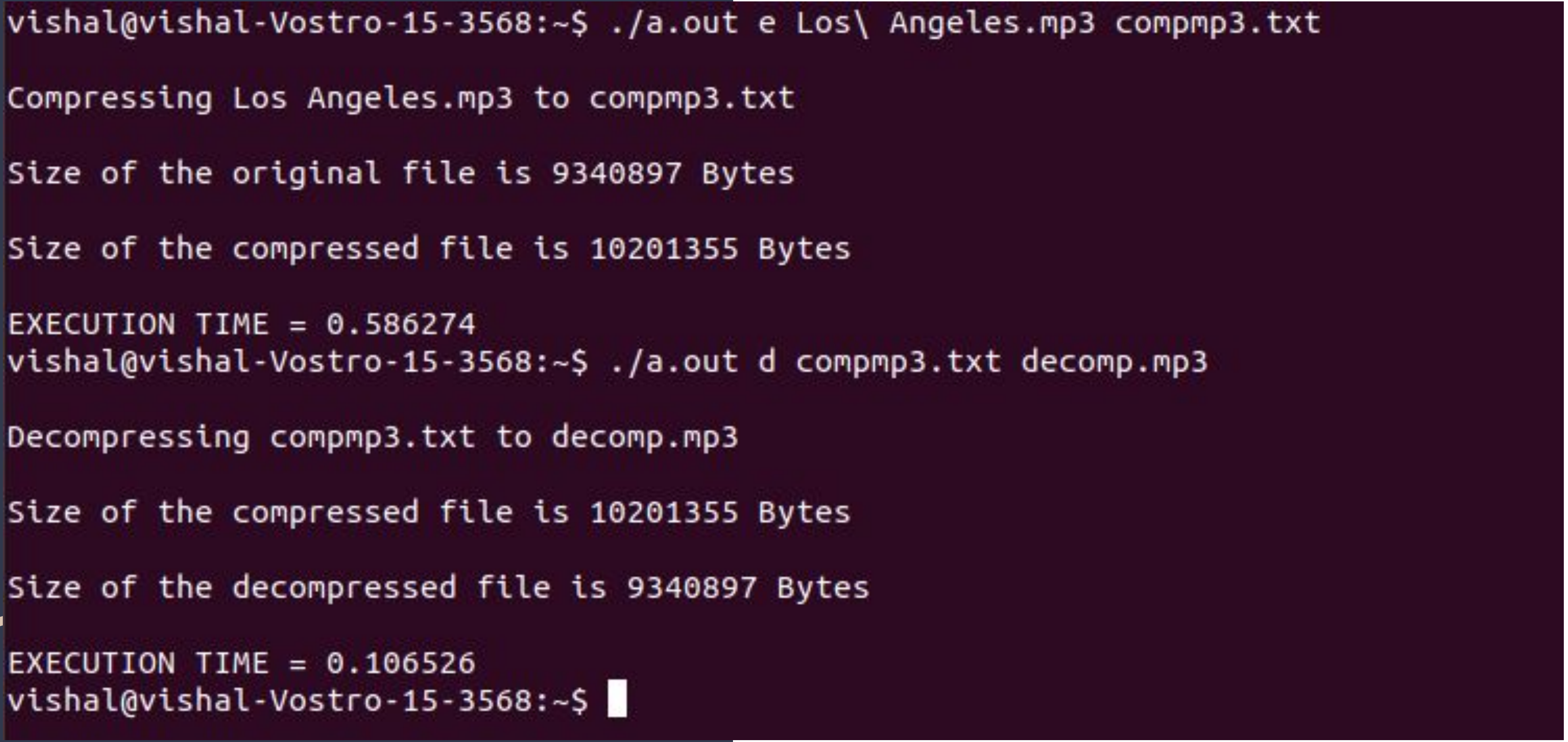


```
vishal@vishal-Vostro-15-3568:~$ ./a.out e server.c compserver.txt

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:~$
```

Implementation

- The above image is for the compression of C file
- The compression ratio of the above is 3.014. I.e 68.7% of the file is compressed and 31.3% remains.

Implementation



```
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:~$
```

Implementation

- The above image is for the compression of .mp3 file
- The compression ratio of the “Los Angeles.mp3” above is 0.9156.

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	-

Compression Ratio For LZ77

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)	
plrabn12.txt	460kB(original)	1.88
	244kB (compressed)	

Thank You!