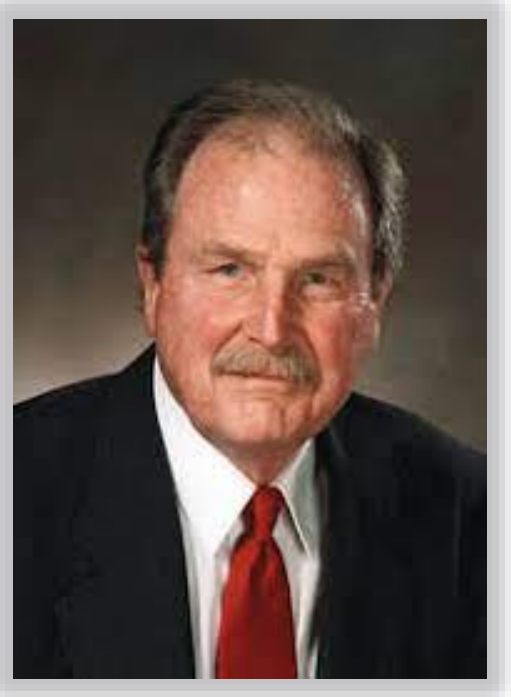
Huffman Text Compressor

Daniel Arias Cámara i David Ferré Garcia

David A. Huffman

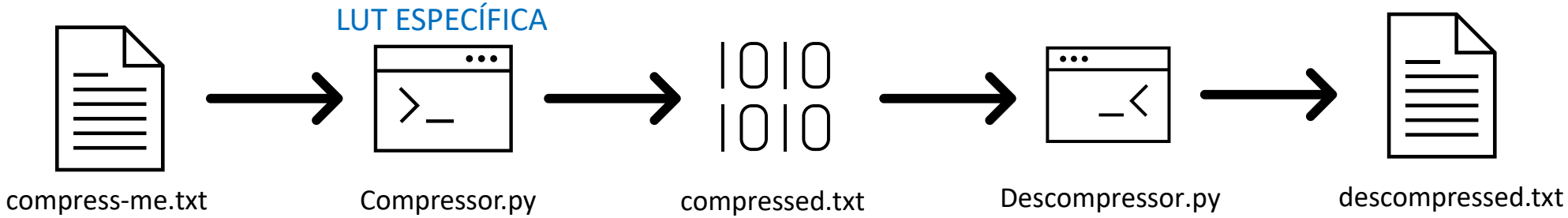
- Neix el 9 d'agost de 1925 a Ohio (EEUU)
- Mor al 7 d'octubre de 1999 als 74 anys
- Aconsegueix el títol d'enginyeria elèctrica als 18 anys a la *Universitat Estatal d'Ohio*
- Estudia un postgrau al *Massachusetts Institute of Technology* (MIT)
- Conegut pel "*codi Huffman*", un sistema de compressió i codificació variable.



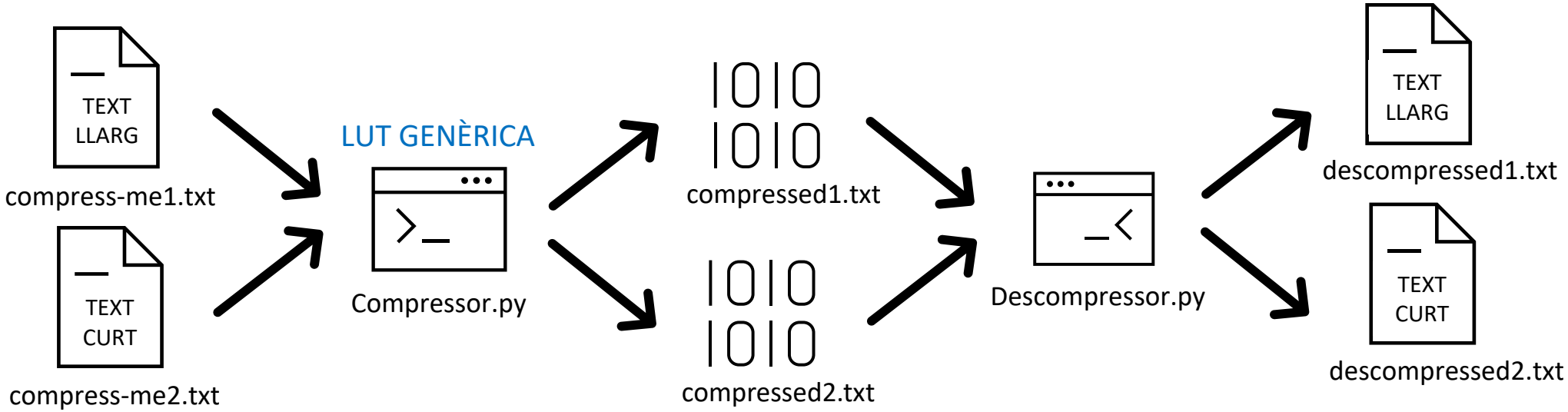
1. Introducció

https://github.com/shkolovy/huffman-compressor

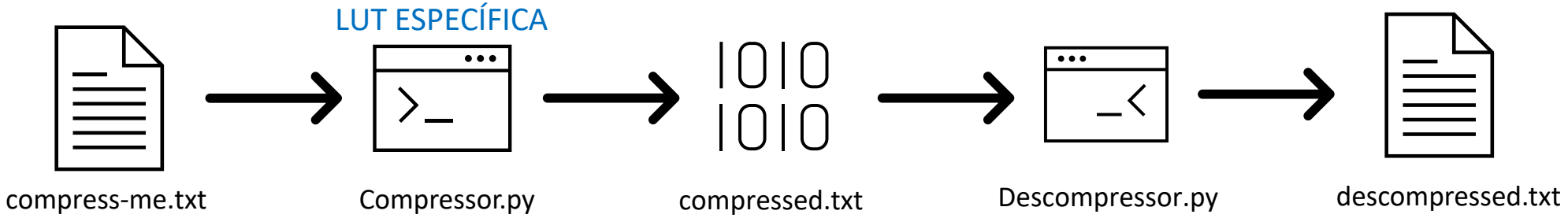
- Compressor amb una LUT específica



- Compressor amb una LUT genèrica

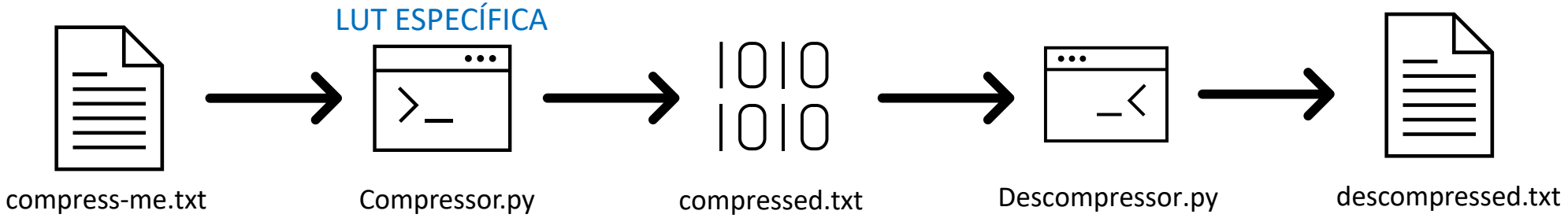


1. Compressor LUT específica




```python
if __name__ == "__main__":
    file_to_compress, compressed = "compress-me", "compressed"
    compress(file_to_compress, compressed)
```

1. Compressor LUT específica

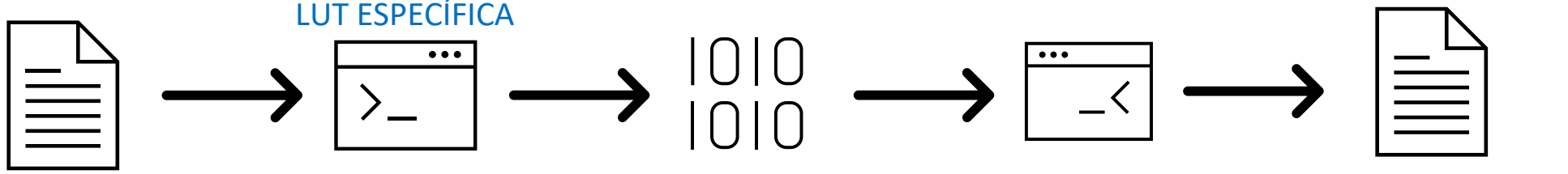



```python
if __name__ == "__main__":
    file_to_compress, compressed = "compress-me", "compressed"
    compress(file_to_compress, compressed)
```

```python
def compress(input_path, output_path):
    """Save encoded text to output file"""

    with open(input_path) as in_file, open(output_path, "wb") as out_file:
        text = in_file.read()                                          #
        encoded_text = encode(text)                                    #

        b_arr = bytearray()
        for i in range(0, len(encoded_text), 8):                       #
            b_arr.append(int(encoded_text[i:i+8], 2))

        out_file.write(b_arr)
```

1. Compressor LUT específica

LUT ESPECÍFICA

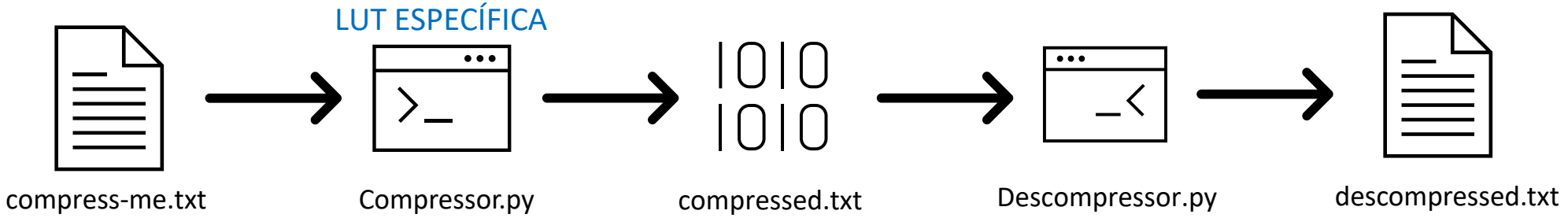



```python
if __name__ == "__main__":
    file_to_compress, compressed = "compress-me", "compressed"
    compress(file_to_compress, compressed)
```

```python
def compress(input_path, output_path):
    """Save encoded text to output file"""

    with open(input_path) as in_file, open(output_path, "wb") as out_file:
        text = in_file.read()                                          #
        encoded_text = encode(text)                                    #

        b_arr = bytearray()
        for i in range(0, len(encoded_text), 8):                       #
            b_arr.append(int(encoded_text[i:i+8], 2))

        out_file.write(b_arr)
```

```python
frequencies = Counter(text)
queue = PriorityQueue()
code_table = {}

for char, f in frequencies.items():
    queue.put(HuffmanNode(char, f))

# merge nodes
while queue.qsize() > 1:
    l, r = queue.get(), queue.get()
    queue.put(HuffmanNode(None, l.freq + r.freq, l, r))

huffman_tree = queue.get()
huffman_tree
_fill_code_table(huffman_tree, "", code_table)

encoded_text_code = ""
array_text = list(text)
i=0

for c in text:
    encoded_text_code += code_table[c]
    i= i + 1
```

1. Compressor LUT específica

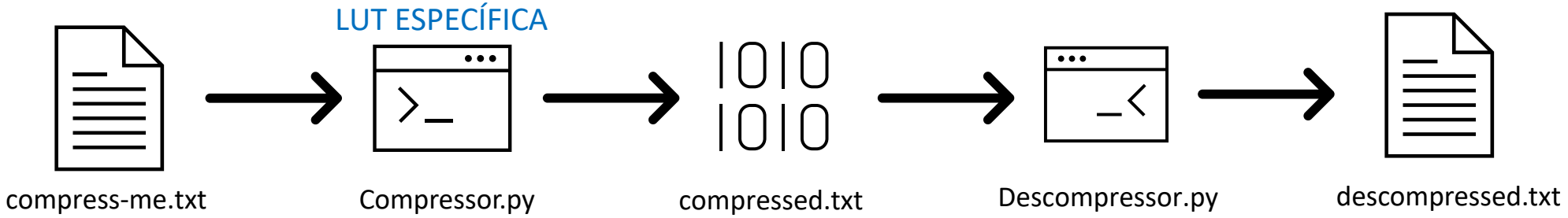



```python
if __name__ == "__main__":
    file_to_compress, compressed = "compress-me", "compressed"
    compress(file_to_compress, compressed)
```

```python
def compress(input_path, output_path):
    """Save encoded text to output file"""

    with open(input_path) as in_file, open(output_path, "wb") as out_file:
        text = in_file.read()                                          #
        encoded_text = encode(text)                                    #

        b_arr = bytearray()
        for i in range(0, len(encoded_text), 8):                       #
            b_arr.append(int(encoded_text[i:i+8], 2))

        out_file.write(b_arr)
```

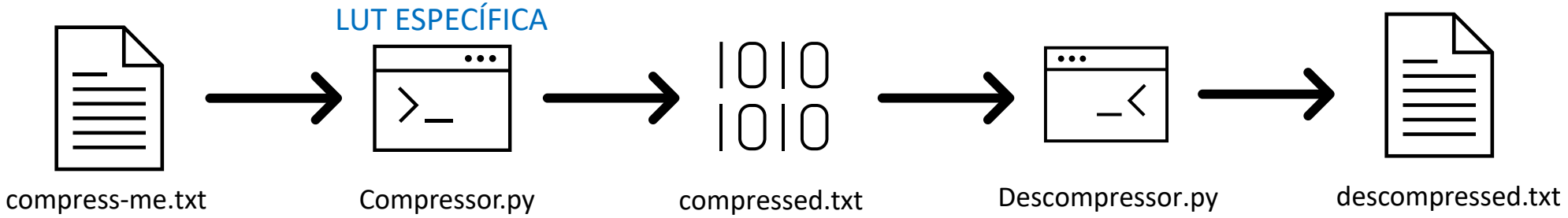
```python
encoded_tree_code = _encode_huffman_tree(huffman_tree, "")

# add extra zeros, as in python it is not possible read
# file bit by bit (min byte) so extra zeros will be
# added automatically which cause a loss of information

num = 8 - (len(encoded_text_code) + len(encoded_tree_code)) % 8
if num != 0:
    encoded_text_code = num * "0" + encoded_text_code

print("DATA:")
print(f"-> Frequencies: {frequencies}")
print(f"\n-> Charcter code: {code_table}")
print(f"\n-> Encoded huffman tree code: {encoded_tree_code}")
print(f"\n-> Encoded text code: {encoded_text_code}")

return f"{encoded_tree_code}{num:08b}{encoded_text_code}"
```

1. Compressor LUT específica




```python
if __name__ == "__main__":
    file_to_compress, decompressed, compressed = "compress-me", "decompressed", "compressed"
    _print_ratio(file_to_compress, compressed)
    decompress(compressed, decompressed)
```

1. Compressor LUT específica

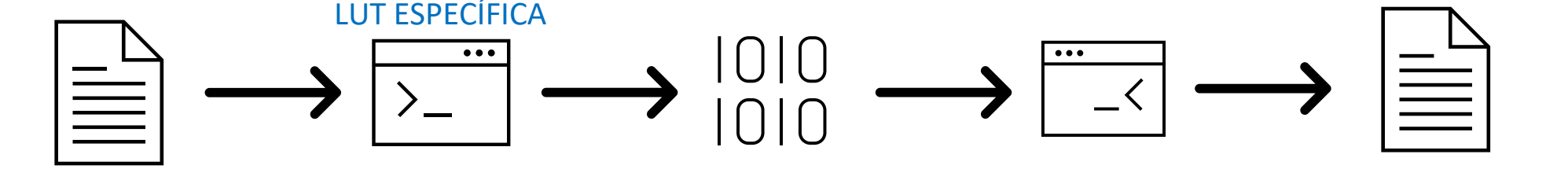



```python
if __name__ == "__main__":
    file_to_compress, decompressed, compressed = "compress-me", "decompressed", "compressed"
    _print_ratio(file_to_compress, compressed)
    decompress(compressed, decompressed)
```

```python
def decompress(input_path, output_path):
    """Save decoded text to output file"""

    with open(input_path, "rb") as in_file, open(output_path, "w") as out_file:
        encoded_text = ""

        byte = in_file.read(1)
        while len(byte) > 0:
            encoded_text += f"{bin(ord(byte))[2:]:0>8}"
            byte = in_file.read(1)

        decoded_text = decode(encoded_text)
        out_file.write(decoded_text)
```

1. Compressor LUT específica

LUT ESPECÍFICA




```python
if __name__ == "__main__":
    file_to_compress, decompressed, compressed = "compress-me", "decompressed", "compressed"
    _print_ratio(file_to_compress, compressed)
    decompress(compressed, decompressed)
```

```python
def decompress(input_path, output_path):
    """Save decoded text to output file"""

    with open(input_path, "rb") as in_file, open(output_path, "w") as out_file:
        encoded_text = ""

        byte = in_file.read(1)
        while len(byte) > 0:
            encoded_text += f"{bin(ord(byte))[2:]:0>8}"
            byte = in_file.read(1)

        decoded_text = decode(encoded_text)
        out_file.write(decoded_text)
```

```python
def decode(encoded_text):
    """Returns decoded string"""

    encoded_text_ar = list(encoded_text)
    encoded_tree = _decode_huffman_tree(encoded_text_ar)

    # remove extra zeros
    number_of_extra_0_bin = encoded_text_ar[:8]
    encoded_text_ar = encoded_text_ar[8:]
    number_of_extra_0 = int("".join(number_of_extra_0_bin), 2)
    encoded_text_ar = encoded_text_ar[number_of_extra_0:]

    # decode text
    text = ""
    current_node = encoded_tree
    for char in encoded_text_ar:
        current_node = current_node.left if char == '0' else current_node.right

        if current_node.char is not None:
            text += current_node.char
            current_node = encoded_tree

    return text
```

2. ASCII 255

- No és possible codificar tots els caràcters

☺ ♫ ♪ ∑ ☼ ♀ ♂ “ ”

- Per poder codificar els símbols anteriors és necessari utilitzar una codificació UTF-8

Number of bytes	Bits for code point	Byte 1	Byte 2	Byte 3	Byte 4
1	7	0xxxxxxx			
2	11	110xxxxx	10xxxxxx		
3	16	1110xxxx	10xxxxxx	10xxxxxx	
4	21	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

STANDARD ASCII (0-127)							EXTENDED ASCII (128-255)							
Control Codes	Keyboard and <Alt+number>						<Alt+number>							
0 Null	32	sp	64	@	96	`	128	Ç	160	á	192	+	224	α
1 Start of heading (SOH)	33	!	65	A	97	a	129	ü	161	í	193	-	225	ß
2 Start of text (STX)	34	"	66	B	98	b	130	é	162	ó	194	-	226	Γ
3 End of text (ETX)	35	#	67	C	99	c	131	â	163	ú	195	+	227	π
4 End of transmit (EOT)	36	$	68	D	100	d	132	ä	164	ñ	196	-	228	Σ
5 Enquiry (ENQ)	37	%	69	E	101	e	133	à	165	Ñ	197	+	229	σ
6 Acknowledge (ACK)	38	&	70	F	102	f	134	å	166	ª	198	¦	230	µ
7 Audible bell (BEL)	39	'	71	G	103	g	135	ç	167	º	199	¦	231	τ
8 Backspace (BS)	40	(	72	H	104	h	136	ê	168	¿	200	+	232	Φ
9 Horizontal tab (HT)	41	)	73	I	105	I	137	ë	169	¬	201	+	233	Θ
10 Line feed (LF)	42	*	74	J	106	j	138	è	170	¬	202	-	234	Ω
11 Vertical tab (VT)	43	+	75	K	107	k	139	ï	171	½	203	-	235	δ
12 Form feed (FF)	44	,	76	L	108	l	140	î	172	¼	204	¦	236	∞
13 Carriage return (CR)	45	-	77	M	109	m	141	ì	173	¡	205	-	237	φ
14 Shift out (SO)	46	.	78	N	110	n	142	Ä	174	«	206	+	238	ε
15 Shift in (SI)	47	/	79	O	111	o	143	Å	175	»	207	-	239	∩
16 Data link escape (DLW)	48	0	80	P	112	p	144	É	176	¦	208	-	240	≡
17 Device control 1 (DC1)	49	1	81	Q	113	q	145	æ	177	¦	209	-	241	±
18 Device control 2 (DC2)	50	2	82	R	114	r	146	Æ	178	¦	210	-	242	≥
19 Device control 3 (DC3)	51	3	83	S	115	s	147	ô	179	¦	211	+	243	≤
20 Device control 4 (DC4)	52	4	84	T	116	t	148	ö	180	¦	212	+	244	(
21 Neg. acknowledge (NAK)	53	5	85	U	117	u	149	ò	181	¦	213	+	245	)
22 Synchronous idle (SYM)	54	6	86	V	118	v	150	û	182	¦	214	+	246	÷
23 End trans. Block (ETB)	55	7	87	W	119	w	151	ù	183	+	215	+	247	≈
24 Cancel (CAN)	56	8	88	X	120	x	152	ÿ	184	+	216	+	248	°
25 End of medium (EM)	57	9	89	Y	121	y	153	Ö	185	¦	217	+	249	·
26 Substitution (SUB)	58	:	90	Z	122	z	154	Ü	186	¦	218	+	250	·
27 Escape (ESC)	59	;	91	[	123	{	155	¢	187	+	219	¦	251	√
28 File separator (PS)	60	<	92	\	124	|	156	£	188	+	220	_	252	n
29 Group separator (GS)	61	=	93	]	125	}	157	¥	189	+	221	¦	253	²
30 Record separator (RS)	62	>	94	^	126	~	158	₱	190	+	222	¦	254	¦
31 Unit separator (US)	63	?	95	_	127	¦ DEL	159	ƒ	191	+	223	¯	255	

3. Compressió Codi C

- Compressor LUT específica

  Abans: 841 bytes

  Després: 610 bytes

  Compressió 27.5%

" " → 010 "a" → 1100

"i" → 1001 "e" → 1010

Sprecific LUT Code C					
Character	Frequency Table	Look up Table	Character	Frequency Table	Bit Table
' '	92	"010"	/	5	"0000011"
a	58	"1100"	<	4	"11110000"
i	54	"1001"	>	4	"11010100"
e	54	"1010"	b	4	"11011100"
n	53	"1000"	N	4	"11011011"
\n	52	"0111"	{	4	"11110001"
t	48	"0011"	}	4	"11010101"
r	41	"0001"	0	4	"11110010"
l	34	"11101"	@	3	"01100100"
o	29	"10111"	=	3	"01100101"
d	26	"10110"	A	2	"110111011"
m	22	"00100"	T	2	"111000110"
(	19	"111110"	5	2	"1111001111"
)	19	"111111"	g	2	"111000010"
\t	18	"111101"	L	2	"111000100"
s	16	"111001"	V	2	"111000001"
*	12	"011000"	:	2	"110111010"
;	12	"011010"	+	2	"111000111"
c	11	"001010"	%	2	"111000000"
u	11	"001011"	1	2	"110110100"
.	11	"000011"	?	2	"110110000"
p	10	"000000"	!	2	"1101100100"
f	8	"1101011"	M	1	"1101101010"
v	8	"1101111"	X	1	"1101100101"
E	7	"1101001"	I	1	"1101101011"
"'"	7	"1101000"	S	1	"1111001100"
h	6	"0110011"	z	1	"1110000111"
,	6	"0110111"	U	1	"1110000110"
"	6	"0110110"	-	1	"1101100100"
#	5	"0000010"	Q	1	"1110001010"
_	5	"0000101"	y	1	"1110001011"
			\\	1	"1111001101"
Before:	841 bytes		w	1	"1101100110"
After:	610 bytes		q	1	"1101100111"
Compression:	27.5%		&	1	"1111001110"

3. Compressió Codi C

- Compressor LUT específica

  Abans: 841 bytes

  Després: 610 bytes

  Compressió 27.5%

“ ” → 010 “a” → 1100

“i” → 1001 “e” → 1010

- Compressor LUT genèrica

  Abans: 862 bytes

  Després: 650 bytes

  Compressió 24.5%

“ ” → 1100 “a” → 1001

“i” → 0101 “e” → 0111

Generic LUT Code C					
Character	Frequency Table	Look up Table	Character	Frequency Table	Bit Table
' '	92	"1100"	/	5	"0110111"
a	58	"1001"	<	4	"101001000"
i	54	"0101"	>	4	"1101100000"
e	54	"0111"	b	4	"1101110"
n	53	"11111"	N	4	"101001010"
\n	52	"11100"	{	4	"01101100"
t	48	"1000"	}	4	"01101011"
r	41	"0000"	0	4	"11011001"
l	34	"10110"	@	3	"11011000011"
o	29	"0001"	=	3	"1110111"
d	26	"01100"	A	2	"11010001"
m	22	"101000"	T	2	"111010110"
(	19	"101111"	5	2	"0100011110"
)	19	"101110"	g	2	"1110110"
\t	18	"0011"	L	2	"11101010"
s	16	"10101"	V	2	"0110110111"
*	12	"0110100"	:	2	"11110111010"
;	12	"111100"	+	2	"111010111"
c	11	"00101"	%	2	"010001110"
u	11	"00100"	1	2	"110110001"
.	11	"010010"	?	2	"011010101110"
p	10	"110101"	!	2	"01101010110"
f	8	"1110100"	M	1	"011010100"
v	8	"0100001"	X	1	"011011010000"
E	7	"01001100"	I	1	"01001110"
"'"	7	"1111011110"	S	1	"0100000"
h	6	"11011111"	z	1	"11110111111"
,	6	"1101101"	U	1	"01101101001"
"	6	"1010011"	-	1	"011011010101"
#	5	"0100011011"	Q	1	"11011011010"
_	5	"101001011"	y	1	"0100011010"
			\\	1	"11011000010"
Before:	862 bytes		w	1	"11110110"
After:	650 bytes		q	1	"1101111011"
Compression:	24.5%		&	1	"1111011100"

4. Compressió Ourfather

- Compressor LUT específica

  Abans: 290 bytes

  Després: 199 bytes

  Compressió 31.4%

“ ” → 110 “e” → 000

“a” → 1010 “t” → 1001

Character	Frequency Table	Bit Table
' '	45	"110"
e	27	"000"
a	20	"1010"
t	18	"1001"
s	18	"0111"
o	16	"0101"
i	16	"0100"
n	15	"0011"
r	13	"11111"
h	12	"11110"
\n	10	"11100"
d	10	"10110"
u	9	"10001"
l	8	"01101"
v	7	"00100"
,	6	"111010"
m	6	"111011"
w	5	"101110"
y	5	"101111"
b	4	"100000"
g	4	"001011"
f	3	"001010"
p	3	"1000011"
;	2	"1000010"
.	2	"0110011"
O	1	"01100000"
F	1	"01100100"
k	1	"01100101"
c	1	"01100001"
G	1	"01100010"
A	1	"01100011"
Before:	290 bytes	
After:	199 bytes	
Compression:	31.4%	

4. Compressió Ourfather

- Compressor LUT específica

  Abans: 290 bytes

  Després: 199 bytes

  Compressió 31.4%

" " → 110 "e" → 000

"a" → 1010 "t" → 1001

- Compressor LUT genèrica

  Abans: 300 bytes

  Després: 244 bytes

  Compressió 18.7%

" " → 111 "e" → 001

"a" → 1010 "t" → 1100

Character	Frequency Table	Bit Table
' '	45	"110"
e	27	"000"
a	20	"1010"
t	18	"1001"
s	18	"0111"
o	16	"0101"
i	16	"0100"
n	15	"0011"
r	13	"11111"
h	12	"11110"
\n	10	"11100"
d	10	"10110"
u	9	"10001"
l	8	"01101"
v	7	"00100"
,	6	"111010"
m	6	"111011"
w	5	"101110"
y	5	"101111"
b	4	"100000"
g	4	"001011"
f	3	"001010"
p	3	"1000011"
;	2	"1000010"
.	2	"0110011"
O	1	"01100000"
F	1	"01100100"
k	1	"01100101"
c	1	"01100001"
G	1	"01100010"
A	1	"01100011"
Before:	290 bytes	
After:	199 bytes	
Compression:	31.4%	

Sprecific LUT Ourfather		
Character	Frequency	Bit Table
' '	45	"111"
e	27	"001"
a	20	"1010"
t	18	"1100"
s	18	"0001"
o	16	"1000"
i	16	"0110"
n	15	"0101"
r	13	"11011"
h	12	"11010"
\n	10	"100100"
d	10	"01110"
u	9	"00000"
l	8	"10011"
v	7	"1011110"
,	6	"011110"
m	6	"00001"
w	5	"100101"
y	5	"010010"
b	4	"010000"
g	4	"010011"
f	3	"101101"
p	3	"011111"
;	2	"0100011101000"
.	2	"1011111"
O	1	"10110000100"
F	1	"101100111010"
k	1	"10110010"
c	1	"101110"
G	1	"01000111110"
A	1	"1011001111"
Before:	300 bytes	
After:	244 bytes	
Compression:	18.7%	

5. Compressió Parenostre

- Compressor LUT específica

  Abans: 416 bytes

  Després: 269 bytes

  Compressió 35.3%

“ ” → 110 “e” → 011

“a” → 1011 “s” → 1010

Sprecific LUT Parenostre		
Character	Frequency Table	Bit Table
' '	58	"110"
e	49	"011"
a	28	"1011"
s	28	"1010"
n	26	"1000"
o	23	"0101"
l	23	"0100"
r	20	"0010"
t	20	"0001"
u	16	"11110"
i	15	"11100"
\n	11	"00110"
d	11	"00111"
c	10	"00001"
m	9	"111111"
,	7	"100111"
.	7	"100110"
v	6	"100100"
p	6	"100101"
g	4	"1110100"
Ã	4	"1110110"
q	3	"0000011"
S	2	"11101011"
f	2	"0000000"
" ' "	2	"11101111"
x	2	"11101010"
\xad	2	"11101110"
" - "	2	"11111000"
I	2	"11111011"
P	1	"111110100"
V	1	"111110011"
F	1	"111110101"
E	1	"111110010"
y	1	"00000010"
³	1	"00000101"
A	1	"00000011"
©	1	"00000100"

Before:	416 bytes
After:	269 bytes
Compression:	35.3 %

5. Compressió Parenostre

- Compressor LUT específica

  Abans: 416 bytes

  Després: 269 bytes

  Compressió 35.3%

“ ” → 110 “e” → 011

“a” → 1011 “s” → 1010

- Compressor LUT genèrica

  Abans: 416 bytes

  Després: 312 bytes

  Compressió 25%

“ ” → 010 “e” → 1010

“a” → 1100 “s” → 1111001100

Sprecific LUT Parenostre		
Character	Frequency Table	Bit Table
' '	58	"110"
e	49	"011"
a	28	"1011"
s	28	"1010"
n	26	"1000"
o	23	"0101"
l	23	"0100"
r	20	"0010"
t	20	"0001"
u	16	"11110"
i	15	"11100"
\n	11	"00110"
d	11	"00111"
c	10	"00001"
m	9	"111111"
,	7	"100111"
.	7	"100110"
v	6	"100100"
p	6	"100101"
g	4	"1110100"
Ã	4	"1110110"
q	3	"0000011"
S	2	"11101011"
f	2	"0000000"
"'"	2	"11101111"
x	2	"11101010"
\xad	2	"11101110"
"-"	2	"11111000"
I	2	"11111011"
P	1	"111110100"
V	1	"111110011"
F	1	"111110101"
E	1	"111110010"
y	1	"00000010"
³	1	"00000101"
A	1	"00000011"
©	1	"00000100"

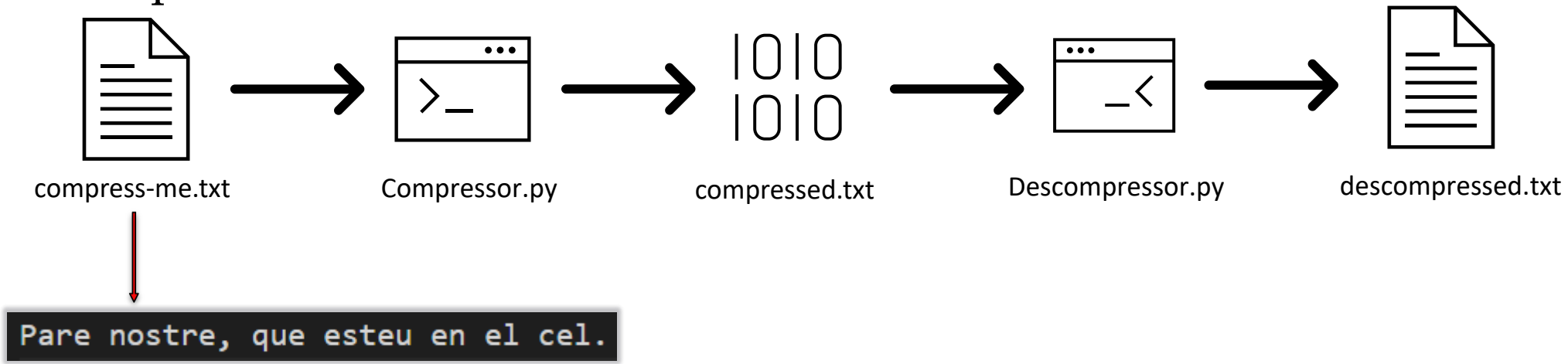
Before:	416 bytes
After:	269 bytes
Compression:	35.3 %

Generic LUT Parenostre		
Character	Frequency Table	Bit Table
' '	58	"010"
e	49	"1010"
a	28	"1100"
s	28	"1111001100"
n	26	"0111"
o	23	"10111"
l	23	"1101101011"
r	20	"0001"
t	20	"0011"
u	16	"001011"
i	15	"1001"
\n	11	"1000"
d	11	"10110"
c	10	"001010"
m	9	"00100"
,	7	"0110111"
.	7	"000011"
v	6	"1101111"
p	6	"000000"
g	4	"111000010"
Ã	4	"101111"
q	3	"1101100111"
S	2	"1111001100"
f	2	"1101011"
"'"	2	"1101000"
x	2	"0000100"
\xad	2	"011010011"
"-"	2	"1101100100"
I	2	"11101"
P	1	"1011100010"
V	1	"111000001"
F	1	"111000111"
E	1	"1101001"
y	1	"1110001011"
³	1	"111000111"
A	1	"110111011"
©	1	"01101110"

Before:	416 bytes
After:	312 bytes
Compression:	25%

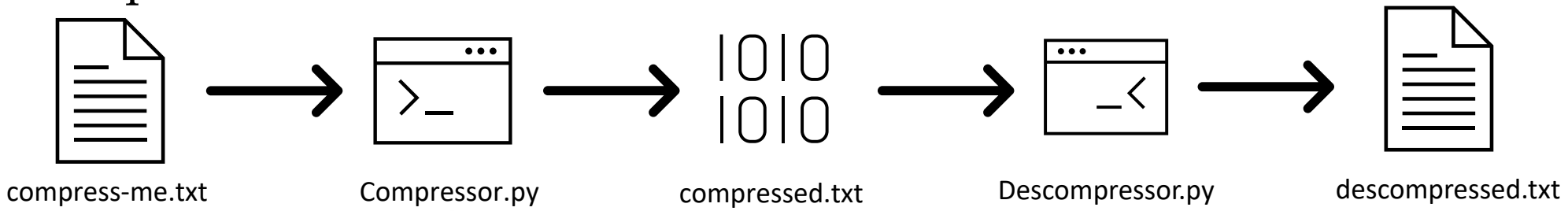
6. Conclusions

- Mitja de compressió amb LUT específica = 31.4% ✔
- Mitja de compressió amb LUT genèrica = 22.73% ✘
- Exemple amb un text curt:



6. Conclusions

- Mitja de compressió amb LUT específica = 31.4% ✔
- Mitja de compressió amb LUT genèrica = 22.73% ✘
- Exemple amb un text curt:

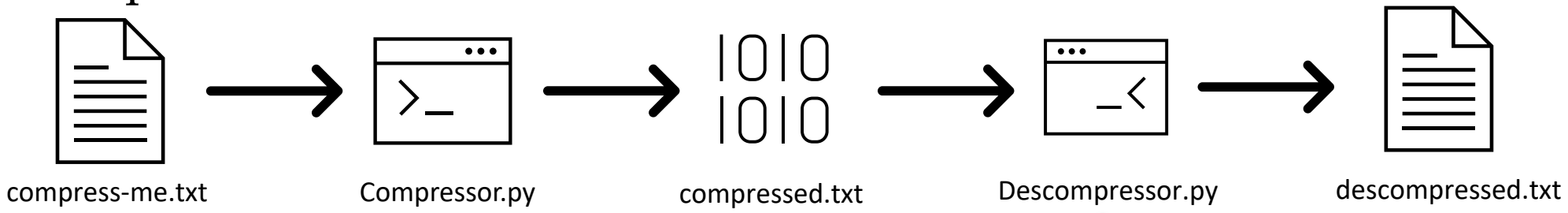



```
DATA:
-> Frequencies: Counter({'e': 8, ' ': 6, 'r': 2, 'n': 2, 's': 2, 't': 2, 'u':
2, 'l': 2, 'P': 1, 'a': 1, 'o': 1, ',': 1, 'q': 1, 'c': 1, '.': 1, '\n': 1})

-> Charcter code: {'u': '0000', 'q': '00010', '\n': '00011', 'a': '00100',
',': '00101', 't': '0011', 'e': '01', 'r': '1000', 'P': '10010', 'o': '10011',
's': '1010', 'l': '1011', 'n': '1100', 'c': '11010', '.': '11011', ' ': '111'}
```

```
-> Encoded huffman tree code:
0000101110101010111000110000101000101100001100101100101110100101100101000101110
0100101010000101101111010111001110110110000101101110010110001110010111010010000
000

-> Encoded text code:
0000000100100010010000111111001001110100011100001001011110001000000111101101000
0110100001110111001110110111111101001101111011001100011
```

6. Conclusions

- Mitja de compressió amb LUT específica = 31.4% ✔
- Mitja de compressió amb LUT genèrica = 22.73% ✘
- Exemple amb un text curt:

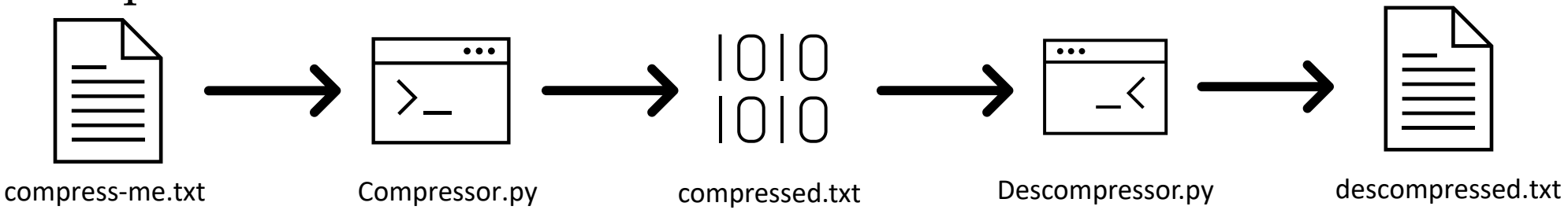



```
return f"{encoded_tree_code}{num:08b}{encoded_text_code}"
```

```
-> Before: 34bytes
-> After: 37bytes
-> Compression -8.8%
```

6. Conclusions

- Mitja de compressió amb LUT específica = 31.4% ✔
- Mitja de compressió amb LUT genèrica = 22.73% ✘
- Exemple amb un text curt:




```
-> Before: 34bytes
-> After: 37bytes
-> Compression -8.8%
```

```
return f"{encoded_tree_code}{num:08b}{encoded_text_code}"
```

Solució en LUTs genèriques:

```
return f"{encoded_ ee_code}{num:08b}{encoded_text_code}"
```

Gràcies per la vostra atenció