Mobile Programming and Multimedia - exercise on lossless compression

Amista’ Michael - 2122865

Exercise

Encode the string: abcabcabcabcffffffffffff000000000000ffffffffffffffffffffffff

using:

- the LZW algorithm and
- choose an algorithm between Shannon-Fano and Huffman.

Compare the two results in terms of compression ratio.

LZW algorithm

Here it is possible to observe the execution, step by step, of LZW encoding algorithm on the given sequence through the following table.

w	**k**	**output**	**code**	**symbol**
NULL	a			
a	b	a	256	ab
b	c	b	257	bc
c	a	c	258	ca
a	b			
ab	c	256	259	abc
c	a			
ca	b	258	260	cab
b	c			
bc	a	257	261	bca
a	b			
ab	c			
abc	f	259	262	abcf
f	f	f	263	ff
f	f			
ff	f	263	264	fff
f	f			
ff	f			

fff	f	264	265	ffff
f	f			
ff	f			
fff	f			
ffff	f	265	266	fffff
f	f			
ff	0	263	267	ff0
0	0	0	268	00
0	0			
00	0	268	269	000
0	0			
00	0			
000	0	269	270	0000
0	0			
00	0			
000	0			
0000	0	270	271	00000
0	0			
00	f	268	272	00f
f	f			
ff	f			
fff	f			
ffff	f			
fffff	f	266	273	ffffff
f	f			
ff	f			
fff	f			
ffff	f			
fffff	f			
ffffff	f	273	274	fffffff
f	f			
ff	f			
fff	f			
ffff	f			
fffff	f			
ffffff	f			
fffffff	f	274	275	ffffffff
f	f			
ff	f			
fff	f			

ffff	f			
fffff	f			
ffffff	EOF	273		

The output column represents the final compressed sequence with length equal to 21. Since 8 bits per code are not enough to represent every element of the encoded sequence, which exceeds the bound of 255 for the ASCII representation, and modern systems typically use the whole byte, instead of just 9 bits for instance, it is necessary to use 2 bytes instead of just 1. So, every encoded symbol can be represented with 2 bytes, hence 16 bits.

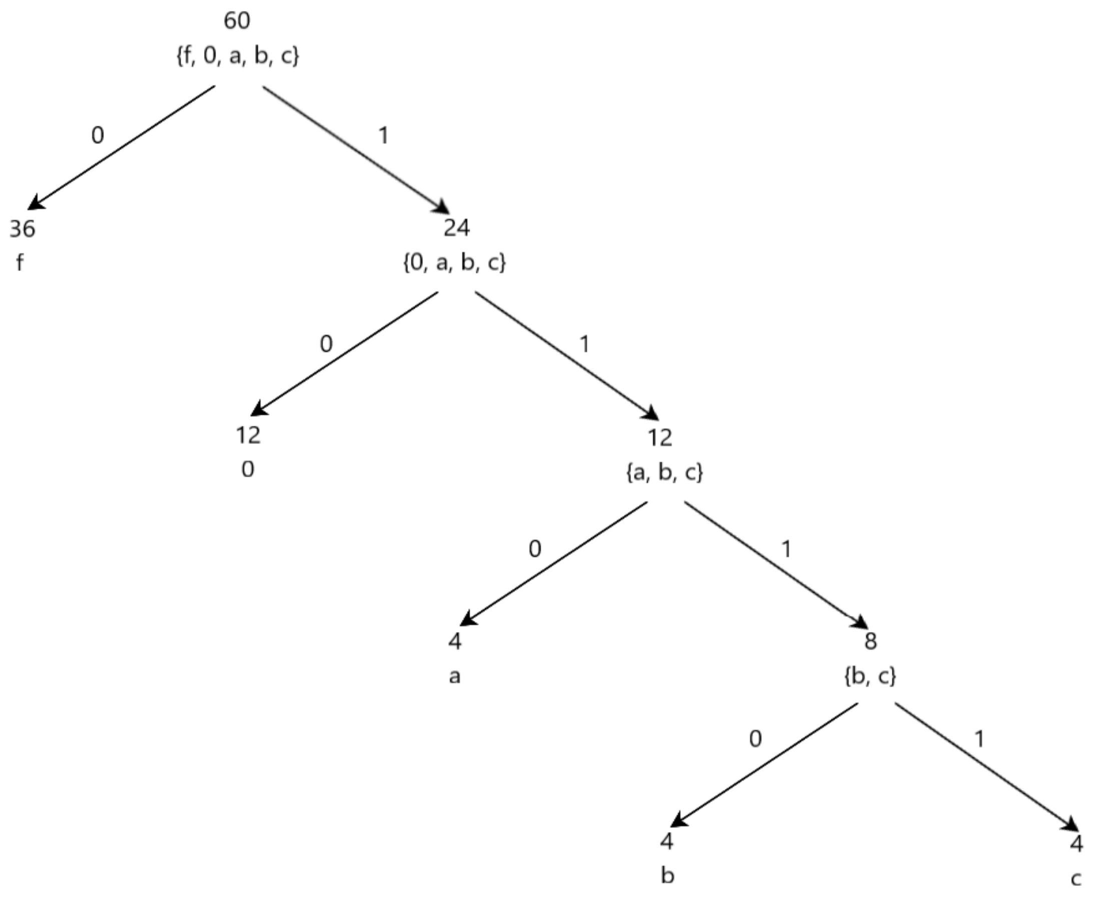
# bits used to represent the encoded sequence = 16 * 21 = 336 bits

The uncompressed original sequence can be represented using the standard ASCII representation, so allocating 8 bits for each character. The length of the original given sequence is 60, so the uncompressed sequence can be represented using 60 * 8 = 480 bits.

Compression ratio = uncompressed size / compressed size = 480 / 336 = **1,428**

Shannon-Fano algorithm

Here it is possible to observe the tree, resulting from the execution of Shannon-Fano's algorithm on the given sequence. Each node is characterized by the total number of occurrences for the characters in the set {...}, initially ordered in a descendent way. The set is recursively divided in two portions in a way to obtain balanced divisions of the set with more or less the same number of occurrences.



The following table presents the algorithm results, with encodings of every character.

Character	Code
f	0
0	10
a	110
b	1110
c	1111

# bits used to represent the encoded sequence = (1*36) + (2*12) + (3*4) + (4*4) + (4*4)
= 104 bits

Since the above table is crucial to decode any output sequence of Shannon-Fano's algorithm, the table must be contained in the compressed result of the algorithm and due to this it should be considered in the total amount of bits used to represent the Shannon-Fano's final output sequence. In particular, to represent this table, it is possible to allocate 8 bits for each different character that appears in the given sequence and 8 bits to represent each character's encoding. Since the table contains five different characters (f,0,a,b,c) the calculation is the following.

# bits used to represent the table = (8 * 5) * 2 = 80 bits

# total bits used to represent Shannon-Fano's output sequence (including the table) = 184 bits

The uncompressed original sequence can be represented using the standard ASCII representation, so allocating 8 bits for each character. The length of the original given sequence is 60, so the uncompressed sequence can be represented using 60 * 8 = 480 bits.

Compression ratio = uncompressed size / compressed size = 480 / 184 = **2,608**

Conclusions

The compression ratio calculated for both LZW and Shannon-Fano algorithms is respectively 1,428 and 2,608 meaning that Shannon-Fano's algorithm performs better than LZW on the given sequence. This is not so surprising since the number of different characters (or symbols) that appear in the given sequence are "just" five which is not a huge number and it is known that the weight of the table, necessary to decode Shannon-Fano's outputs, increases as the number of different symbols increases in any input sequence. This means that Shannon-Fano performs better than LZW on this example, but this is not generally true for all the possible input sequences.