Week 6: Compression (II) (Solutions)

1 Huffman Algorithm

1.3 Exercise 1

The binary trees for the two different codes appear in Figure 1. The dictionary obtained by the Shannon-Fano code is $a \to 00, b \to 01, c \to 10, d \to 110, e \to 111$.

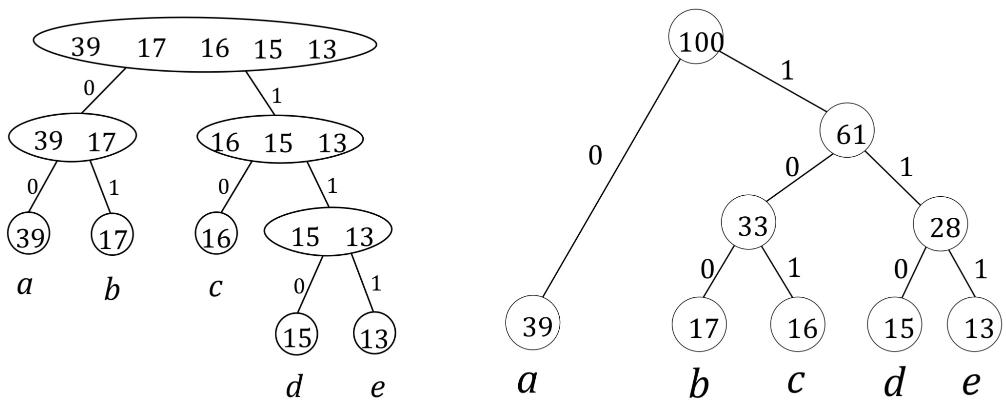


Figure 1: Shannon-Fano tree Huffman tree

The average number of bits used by the Shannon-Fano code is, therefore,
$\frac{(39+17+16)\times2+(15+13)\times3}{100} = \frac{228}{100} = 2.28$ bits/letter

The dictionary obtained by Hufman's code is $a \to 0, b \to 100, c \to 101, d \to 110, e \to 111$.

The average number of bits used by the Huffman code is, therefore,
$\frac{(39\times1)+(17+16+15+13)\times3}{100} = \frac{222}{100} = 2.22$ bits/lettre.

The theoretical limit given by Shannon's theory is the entropy:
Entropy $= -0.39\log_2(0.39) - 0.17\log_2(0.17) - 0.16\log_2(0.16) - 0.15\log_2(0.15) - 0.13\log_2(0.13) \approx 2.18$ bits/letter.

1.4 Exercise 2

The binary tree for the two different codes appears in Figure 2 above. The dictionary obtained by the Shannon-Fano code is $a \to 0, b \to 100, c \to 101, d \to 110, e \to 1110, f \to 1111$.

The average number of bits used by the Shannon-Fano code is, therefore,
$\frac{45\times1+(16+13+12)\times3+(9+5)\times4}{100} = \frac{224}{100} = 2.24$ bits/letter.

The dictionary obtained by Hufman's code is:
$a \to 1, b \to 000, c \to 010, d \to 011, e \to 0010, f \to 0011$.

The average number of bits used by the Huffman code is, therefore,
$\frac{45\times1+(16+13+12)\times3+(9+5)\times4}{100} = \frac{224}{100} = 2.24$ bits/letter.

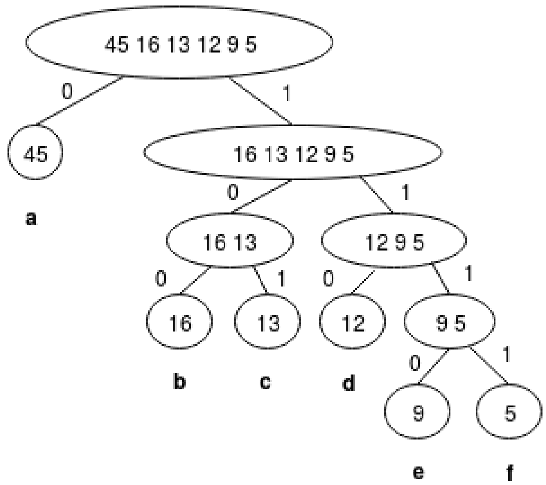
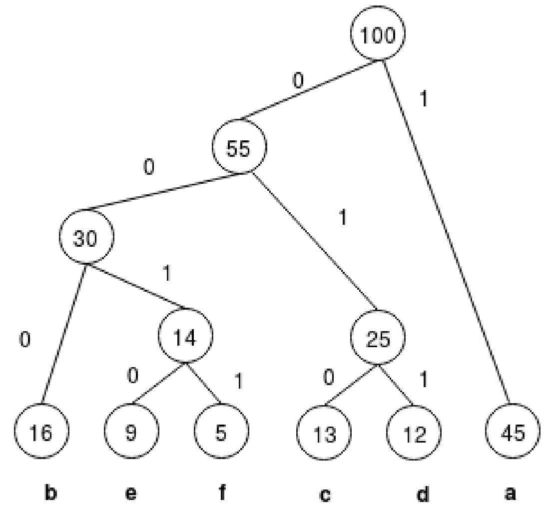


Figure 2: Shannon-Fano tree



Huffman tree

The theoretical limit given by Shannon's theory is the entropy:
Entropy $= -0.45\log_2(0.45) - 0.16\log_2(0.16) - 0.13\log_2(0.13) - 0.12\log_2(0.12) - 0.09\log_2(0.09) - 0.05\log_2(0.05) \approx$ 2.22 bits/letter.

2 Conceptual Question

Huffman encoding is a lossless compression technique. Therefore, option a) is false. In addition, the Huffman encoding is optimal, therefore b) is false. Option c) is true as this is the basis of decoding of a message from a given code.

3 Encoding and Decoding

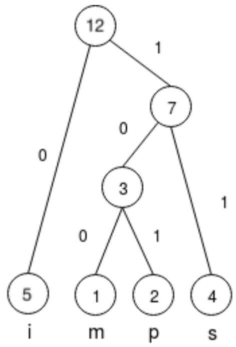
1. Following is the frequency table of characters in 'mississippii' in non-decreasing order of frequency:

character	frequency
m	1
p	2
s	4
i	5

One generated Huffman tree is the one displayed in the next page (note, however, that there might be other possible solutions).

We can then obtain the encoding:

character	frequency	code	code length
m	1	100	3
p	2	101	3
s	4	11	2
i	5	0	1



Total number of bits = freq(m) * codelength(m) + freq(p) * codelength(p) + freq(s) * codelength(s) + freq(i) * codelength(i) = 1*3 + 2*3 + 4*2 + 5*1 = 22.

Average bits per character = total number of bits required / total number of characters = 22/12 = 1.833

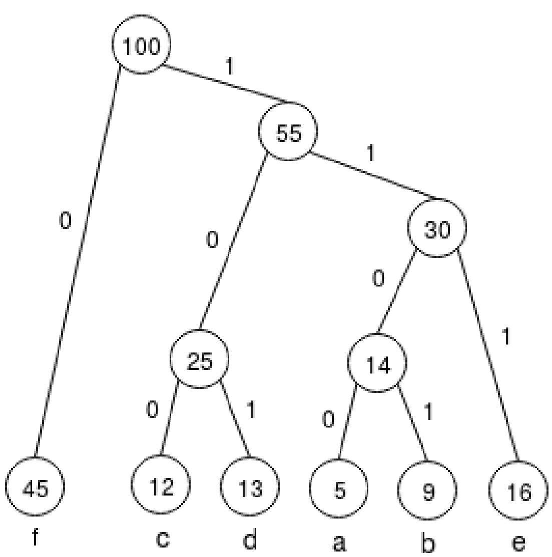
2. Using prefix matching, the given string can be decomposed as follows (the answer might be different if you arrived at a different Huffman tree in 1):

11	0	11	100	11	101	0
s	i	s	m	s	p	i

4 Number of bits

Compute the number of bits without using Huffman:
Total number of characters = sum of frequencies = 100
Size of 1 character = 1 Byte = 8 bits
Total number of bits = 8*100 = 800

Using Huffman Encoding,



Total number of bits needed can be calculated as: 5*4 + 9*4 + 12*3 + 13*3 + 16*3 + 45* 1 = 224
Bits saved = 800-224 = 576.