Information Theoretic Modeling – Exercise 3

Haibo Jin
Student number: 014343698

1 Problem 1

(a)

Based on Kraft inequality and Kraft-McMillan theorem, we can get the following:

SET1 should be the set of prefix(-free) codes.

SET2 should be the set of decodable codes.

SET3 should be the set of codes that satisfy the Kraft inequality.

SET4 should be the set of all possible symbol codes.

(b)

For SET1 $\subseteq$ SET2, a code such as $\{0, 01, 011, 0111\}$ satisfies, because it is decodable but not prefix(-free).

For SET2 $\subseteq$ SET3, a code such as $\{1, 10, 11\}$ satisfies, because it satisfies Kraft inequality but not decodable.

For SET3 $\subseteq$ SET4, a code such as $\{0, 1, 00\}$ satisfies, because it is just a symbol code but not satisfy Kraft inequality.

2 Problem 2

The Shannon-Fano code is simulated by a program implemented in python, please refer to *shannon_fano.py* for more details.

Based on the result of the program, the expected code-length is **1.22**, the entropy is **0.683** and the expected code-length of the Shannon code is **1.5**. Figure 1 shows the running result of the program.

```
local@tktl-2013:~/Nick/Master/courses/Information Theoretic Modeling/exercise/3$
 python shannon_fano.py
{'A': '0', 'C': '10', 'B': '1100', 'E': '1101', 'D': '1111', 'F': '1110'}
expected code-length: 1.22
entropy: 0.683639528056
expected code-length of shannon code: 1.5
```

Figure 1: Shannon-Fano code, applied to simple source of A to F

3 Problem 3

I just use Chapter I of *Alice in Wonderland* as the text for estimating the symbol occurrence probabilities. Please see the source text in *alice_preprocess* and source code in *shannon_fano2.py*.

Based on the result of the program, the expected code-length is **4.116**, the entropy is **4.058** and the expected code-length of the Shannon code is **4.563**.

I also use huffman coding to encode the same text for comparison. The source code is in *huffman2.py*, and its expected code-length is **4.105**. Figure 2 and figure 3 show the running result of Shannon-Fano code and Huffman code, respectively.

```
local@tktl-2013:~/Nick/Master/courses/Information Theoretic Modeling/exercise/3$
 python shannon_fano2.py
{' ': '000', 'a': '0111', 'c': '111010', 'b': '111101', 'e': '001', 'd': '11001'
, 'g': '111000', 'f': '111001', 'i': '1001', 'h': '1000', 'k': '1111110', 'j': '
1111111101', 'm': '111100', 'l': '11000', 'o': '0110', 'n': '1010', 'q': '111111
1110', 'p': '111110', 's': '10110', 'r': '10111', 'u': '11011', 't': '010', 'w':
 '11010', 'v': '11111110', 'y': '111011', 'x': '1111111100', 'z': '1111111111'}
expected code-length: 4.11668228679
entropy: 4.05883706245
expected code-length of shannon code: 4.56316776007
```

Figure 2: Shannon-Fano code, applied to Chapter I of *Alice in Wonderland*

```
local@tktl-2013:~/Nick/Master/courses/Information Theoretic Modeling/exercise/3$
 python huffman2.py
{' ': '00', 'a': '1001', 'c': '101100', 'b': '011101', 'e': '010', 'd': '10111',
 'g': '110001', 'f': '101101', 'i': '0110', 'h': '1000', 'k': '1100001', 'j': '1
100000001', 'm': '011110', 'l': '11001', 'o': '1010', 'n': '11111', 'q': '110000
0010', 'p': '011100', 's': '11110', 'r': '11100', 'u': '111010', 't': '1101', 'w
': '111011', 'v': '11000001', 'y': '011111', 'x': '1100000011', 'z': '1100000000
'}
expected code-length: 4.10515463918
```

Figure 3: Huffman code, applied to Chapter I of *Alice in Wonderland*

4 Problem 4

(a)

Please see source code of Huffman code in *huffman.py*. Figure 4 shows its running result.

```
local@tktl-2013:~/Nick/Master/courses/Information Theoretic Modeling/exercise/3$
 python huffman.py
{'A': '1', 'C': '00', 'B': '0111', 'E': '0101', 'D': '0100', 'F': '0110'}
expected code-length: 1.22
```

Figure 4: Huffman code, applied to simple source of A to F

(b)

Such a case can be that source code only consists A and B. A has probability of **0.0001** and B has probability of **0.9999**. Since probability of A is quite small,

so that $\lceil log_2 1/p_i \rceil$ will be large. But for Huffman code, the code-length of both A and B is one.

(c)

i.

Fewest occurrences of symbol e is *3* to keep the Huffman tree still maximally unbalanced. The number for f is *5*, for g is *8* and for h is *13*. So the sequence is $[1, 1, 1, 2, 3, 5, 8, 13, 21, 34...]$ for symbol a to j. It is Fibonacci sequence excluding the first number.

ii.

The probability of a depends on the value of m. Table 1 shows the probability of a depending on m.

m	1	2	3	4	5	6	7	8
$p(a)$	1	1/2	1/3	1/5	1/8	1/13	1/21	1/37

Table 1: Probability of a depending on m

iii.

Table 2 shows the depth of the tree depending on the value of m.

m	1	2	3	4	5	6	7	8
depth	0	1	2	3	4	5	6	7

Table 2: Depth of the tree depending on m

iv.

Since the probability of a has been calculated in **ii**, the Shannon codeword length can be computed using$\lceil log 1/p(a) \rceil$. Table 3 shows the comparison of Shannon codeword length and Huffman code.

m	1	2	3	4	5	6	7	8
Huffman code	0	1	2	3	4	5	6	7
Shannon codeword length	0	1	2	3	3	4	5	6

Table 3: Comparison of Huffman code and Shannon codeword length for symbol a.

5 Problem 5

Firstly, we should transform the real number parameter p to binary representations. We can achieve this by encoding the symbols of 0 to 9 using prefix(-free) code. Now set the parameter p as some float number $0.p_1p_2p_3...$, and the potentially infinite sequence of fair coin flips as $0.z_1z_2z_3...$ in which p_i and z_i are binaries. To simulate the generation of $Pr[x = 0] = p$, we can run a test from the first binary of p_i and z_i. If $z_i = p_i$, then continue test. Else if $z_i < p_i$, return **0** as the simulation result. Otherwise, return **1** as the result. This is because p can be seen as following the uniform distribution, thus the probability that the z_i string smaller than p_i string is p. Thus, we can get 0 at the probability of p.

The probability of only using one flip is $1/2$, and the probability of using two flips is $(1/2)^2$, and so on. So the expected number of flips is:

$$\begin{aligned} S_{n\to+\infty} &= 1 \times 1/2 + 2 \times (1/2)^2 + 3 \times (1/2)^3 + ... + n \times (1/2)^n \\ &= 2 \end{aligned}$$