National University of Singapore

Name : LUO ZIJIAN

Matric.No： A0224725H

MUSNET： E0572844

Subject： Information Theory

Assignment: Homework Three

Date: Aug 29th

Prof: Marco Tomamichel

Exercise 3.1

a).

$H = \sum_{x \in X} P_x \log \frac{1}{P_x} \approx 4.219$ bit

(b) By using Huffman code in my python algorithm, I got these results

$a \to 1111$	$h \to 0111$	$o \to 1011$	$u \to 01101$
$b \to 110000$	$i \to 1010$	$p \to 110001$	$v \to 0110011$
$c \to 00100$	$j \to 011001000$	$q \to 011001001$	$w \to 00111$
$d \to 11101$	$k \to 011000$	$r \to 1101$	$x \to 011001010$
$e \to 010$	$l \to 11001$	$s \to 1000$	$y \to 111001$
$f \to 00101$	$m \to 00110$	$t \to 000$	$z \to 011001011$
$g \to 111000$	$n \to 1001$		

(c) expected length $= \sum_{x \in X} P_x \cdot \text{len}(P_x) \approx 4.221$ bit
Compared to (a), is a little more than the entropy

Exercise 3.2

For (a). It is valid

For (b) It is valid

For (c) It is not valid because the number of longest symbol should be ≥ 2.
In this case, it is unreasonable.

For (d) It is not valid because this code length is wasted.
$\{0, 1\}$ is more efficient than previous one.

For (e). It is not valid because $\{1\}$ symbol is unreasonable.
In order to get reasonable, it should be $\{011\}$.

~~Exercise 3.3~~

Exercise 3.3

(a)

$a \to 8.4\%$ 00–	$h \to 6.0\%$ 010–	$o \to 7.4\%$ 11–	$u \to 2.7\%$ 110–
$b \to 1.5\%$ 0110–	$i \to 7.4\%$ 10–	$p \to 1.9\%$ 0101–	$v \to 0.9\%$ 1000–
$c \to 2.2\%$ ~~4.0%~~ 0001– ~~[illegible]~~	$j \to 0.1\%$ 1001–	$q \to 0.1\%$ 1010–	$w \to 2.5\%$ 111–
$d \to 4.2\%$ 011–	$k \to 1.3\%$ 0111–	$r \to 7.5\%$ 01–	$x \to 0.1\%$ 1011–
$e \to 11.0\%$ 0–	$l \to 4.0\%$ 100–	$s \to 6.2\%$ 001–	$y \to 2.0\%$ 0100–
$f \to 2.2\%$ 0010–	$m \to 2.4\%$ 0000–	$t \to 9.2\%$ 1–	$z \to 0.1\%$ 1100–
$g \to 2.0\%$ 0011–	$n \to 6.7\%$ 000–		

(b) Expected length $= \sum_{x \in X} P_x \cdot \text{length}(P_x) = 3.457$ bit

(c) <u>Morse code</u>. It is a little similar if we change $0 \to \cdot$ $1 \to -$ $- \to$

Exercise 3.4

(a). From the statement, we know $\left(\sum_{j=1}^{M} 2^{-l_j}\right)^n$, for each component, they are independent

$$\left(\sum_{j=1}^{M} 2^{-l_j}\right)^n = \left(\sum_{j_1=1}^{M} 2^{-l_1}\right)\left(\sum_{j_2=1}^{M} 2^{-l_2}\right)\cdots\left(\sum_{j_n=1}^{M} 2^{-l_{j_n}}\right)$$

$$= \left(\sum_{j_1=1}^{M}\sum_{j_2=1}^{M}\cdots\sum_{j_n=1}^{M} 2^{-l_j}\cdot 2^{-l_2}\cdots 2^{-l_{j_n}}\right.$$

$$= \sum_{j_1=1}^{M}\sum_{j_2=1}^{M}\cdots\sum_{j_n=1}^{M} 2^{-(l_j + l_{j_2} + \cdots l_{j_n})}$$

(b). We rewrite $l = l_{j_1} + l_{j_2} + \cdots l_{j_n}$

$$\Rightarrow \left(\sum_{j=1}^{M} 2^{-l_j}\right)^n = \sum_{j_1=1}^{M}\sum_{j_2=1}^{M}\cdots\sum_{j_n=1}^{M} 2^{-l}$$

$$= \sum_{j=n}^{M\cdot l_n} I^{-l}$$

$$= \sum_{l=n}^{n\, l_{max}} A_l \cdot 2^{-l}$$

(c) From (b), and using $\left(\sum_{j=1}^{M} 2^{-l_j}\right)^n = \sum_{n}^{n l_{max}} A_l \cdot 2^{-l}$

We know $\left(\sum_{j=1}^{M} 2^{-l_j}\right)^n \le 1 \Rightarrow A_l \cdot 2^{-l} \le 1$

$\Downarrow$

$A_l \le 2^l$

$\Downarrow$

$A_i \le 2^i$

Hence $\left(\sum_{j=1}^{M} 2^{-l_j}\right)^n \le n \cdot l_{max}$

Exercise 3.5.

(a). For any d, x be a random variable on $\{0, 1, 2, \cdots d-1\}$

First $\left\lceil \log_2 \frac{1}{P_x(x)} \right\rceil$ expression is uniquely decodable

because $\left\lceil \log_2 \frac{1}{P_x(x)} \right\rceil \geq \log_2 \frac{1}{P_x(x)}$

(b). $0 \to 0.1 \; (\frac{1}{2}) \Rightarrow 1$

$1 \to 0.\cancel{1001}\,0010101 \; (\frac{1}{6}) \Rightarrow 001$

$2 \to \cancel{0.100100}\,0010101 \; (\frac{1}{6}) \Rightarrow 001$

$3 \to \cancel{0.100100}\,0.0010101 \; (\frac{1}{6}) \Rightarrow 001$

$H = \frac{1}{2} \times 1 + \frac{1}{6} \times 3 \times 3 = 2$

(c) If we use Huffman code, we express

$0 \to 0$

$1 \to 10$

$2 \to 110$

$3 \to 111$

$H = 1 \times 0.5 + 2 \times \frac{1}{6} + 2 \times 3 \times \frac{1}{6} = 1.8333\cdots$

Therefore, the expected length of Huffman code is shorter

```
# -*- coding: utf-8 -*-
"""
Created on Sun Aug 29 19:29:37 2021

@author: 15193
"""


# A Huffman Tree Node
class node:
    def __init__(self, freq, symbol, left=None, right=None):
        # frequency of symbol
        self.freq = freq

        # symbol name (character)
        self.symbol = symbol

        # node left of current node
        self.left = left

        # node right of current node
        self.right = right

        # tree direction (0/1)
        self.huff = ''

# utility function to print huffman
# codes for all symbols in the newly
# created Huffman tree


def printNodes(node, val=''):
    # huffman code for current node
    newVal = val + str(node.huff)

    # if node is not an edge node
    # then traverse inside it
    if(node.left):
        printNodes(node.left, newVal)
    if(node.right):
        printNodes(node.right, newVal)

        # if node is edge node then
        # display its huffman code
    if(not node.left and not node.right):
```

```
            print(f"{node.symbol} -> {newVal}")


# characters for huffman tree
chars = ['a', 'b', 'c', 'd', 'e', 'f','g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w',
'x', 'y', 'z']

# frequency of characters
freq = [84,15,22,42,110,22,20,60,74,1,13,40,24,67,74,19,1,75,62,92,27,9,25,1,20,1]

# list containing unused nodes
nodes = []

# converting ccharacters and frequencies
# into huffman tree nodes
for x in range(len(chars)):
    nodes.append(node(freq[x], chars[x]))

while len(nodes) > 1:
    # sort all the nodes in ascending order
    # based on theri frequency
    nodes = sorted(nodes, key=lambda x: x.freq)

    # pick 2 smallest nodes
    left = nodes[0]
    right = nodes[1]

    # assign directional value to these nodes
    left.huff = 0
    right.huff = 1

    # combine the 2 smallest nodes to create
    # new node as their parent
    newNode = node(left.freq+right.freq, left.symbol+right.symbol, left, right)

    # remove the 2 nodes and add their
    # parent as new node among others
    nodes.remove(left)
    nodes.remove(right)
    nodes.append(newNode)

# Huffman Tree is ready!
printNodes(nodes[0])
```