Name _____ ID3 _ _ _

ECS 452: Digital Communication Systems **2017/2**

HW 1 — Due: Feb 9, 4 PM

Lecturer: Prapun Suksompong, Ph.D.

Instructions

(a) This assignment has 6 pages.

(b) (1 pt) Work and write your answers **directly on these provided sheets** (not on other blank sheet(s) of paper). Hard-copies are distributed in class.

(c) (1 pt) Write your first name and the last three digits of your student ID on the upper-right corner of this page.

(d) (8 pt) Try to solve all non-optional problems.

(e) Write down all the steps that you have done to obtain your answers. You may not get full credit even when your answer is correct without showing how you get your answer.

Problem 1. To determine whether a code (or an encoder) is non-singular, uniquely decodable, or prefix-free, it is not necessary to consider the whole codebook. Many questions of this type will give only the collection (or list) of codewords that the code uses.

Consider the code $\{0, 01\}$.

(a) Is it nonsingular?

Yes, it is non singular. The two codewords are different.

(b) Is it uniquely decodable?

Yes, it is UD. Given a sequence of codewords, first isolate occurrences of 01 (i.e., find all the ones) and then parse the rest into 0's.

Remark: This code is suffix-free.

No codeword is a suffix of any other codeword.

To see this, use backward decoding. For any received sequence/string, we work backward from the end, and look for the reversed codewords. Since the codewords satisfy the suffix-free condition, the reversed codewords satisfy the prefix-free condition, and therefore we can uniquely decode the reversed sequence/string.

(c) Is it prefix-free?

No, the code is not prefix-free. The first codeword (0) is a prefix of the second codeword (01).

Reminder: prefix-free code is the same as prefix code.

This is more accurate in terms of meaning.

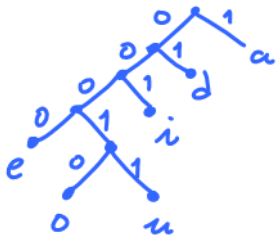
This is the name that has been traditionally used.

Remark: Observe that we can specify whether a code is nonsingular, uniquely decodable, or prefix-free even when there is no information about the shared alphabet nor the pmf of the DMS symbols

Problem 2. In this question, each output string from a DMS is encoded by the following source code:

x	Codeword $c(x)$
'a'	1
'd'	01
'e'	0000
'i'	001
'o'	00010
'u'	00011

Code tree:



(a) Is the code prefix-free?

From the code tree, all the codewords are located at leaf nodes of the tree.
So, yes, the code is prefix-free.

(b) Is the code uniquely decodable?

Yes, any prefix-free code is uniquely decodable (UD).

This is one of many reasons we consider prefix-free code.

(c) Suppose the DMS produces the string 'audio'. Find the output of the source encoder.

1000110100100010
a u d i o

(d) Suppose the output of the source encoder is
0 0 0 1 0/0 1/0 0 0 0/0 0 1/0 1/0 0 0 0/1/0 1/0 0 1/0 0 0 0
o d e i d e a d i e
Find the corresponding source string produced by the DMS. Use "/" to indicate the locations where the sting above is split into codewords.

odeideadie

Problem 3. Consider the random variable X whose support S_X contains seven values:

$$S_X = \{x_1, x_2, \ldots, x_7\}\,.$$

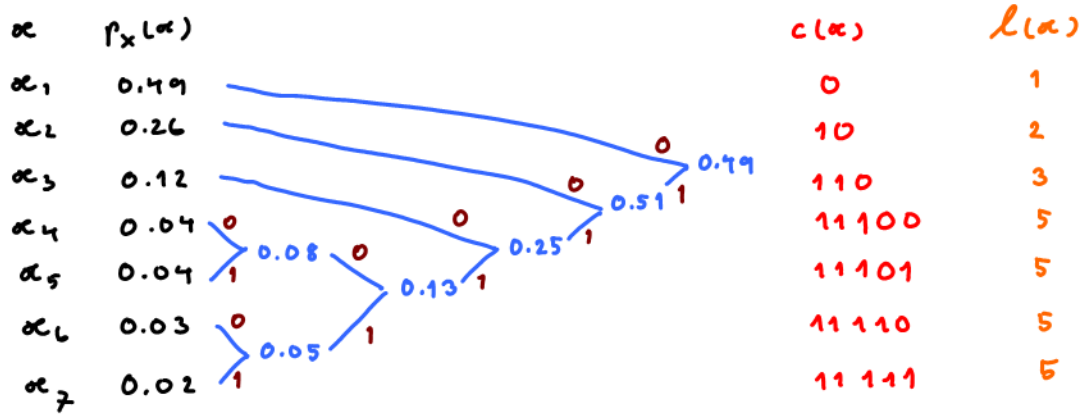
Their corresponding probabilities are given by

x	x_1	x_2	x_3	x_4	x_5	x_6	x_7
$p_X(x)$	0.49	0.26	0.12	0.04	0.04	0.03	0.02

(a) Find the entropy $H(X)$.

$H(X) = -\mathbb{E}[\log_2 p_X(X)] = -\sum_{\alpha} p_X(\alpha) \log_2 p_X(\alpha) = 2.0128$ bits per symbol.

(b) Find a binary Huffman code for X.



(c) Find the expected codelength for the encoding in part (b).

$\mathbb{E}[\ell(X)] = 1\times0.49 + 2\times0.26 + 3\times0.12 + 5\times(\underbrace{2\times0.04}_{0.08} + 0.03 + 0.02)$ (0.13)

$= 2.02$ bits per symbol

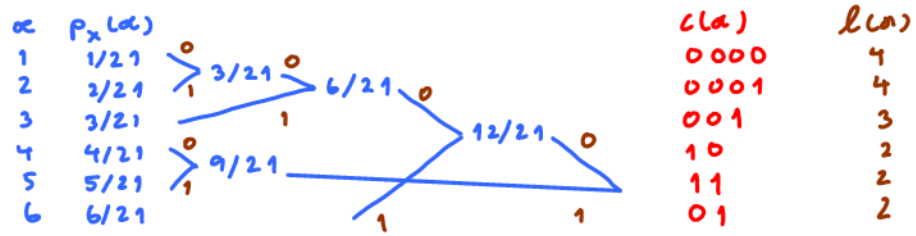
Problem 4. Find the entropy and the binary Huffman code for the random variable X with pmf

$$p_X(x) = \begin{cases} \frac{x}{21}, & x = 1, 2, \ldots, 6, \\ 0, & \text{otherwise.} \end{cases}$$

Also calculate $\mathbb{E}[\ell(X)]$ when Huffman code is used.

MATLAB

$H(X) = -\sum_{\alpha} p_X(\alpha) \log_2 p_X(\alpha) = 2.3983$ bits per symbol



$\mathbb{E}[\ell(X)] = \frac{17}{7} \approx 2.43$ bits per symbol

Problem 5. Consider a random variable X whose support is $S_X = \{x_1, x_2, \ldots, x_n\}$. Let $p_k = p_X(x_k)$. Then, the entropy of X is

$$H(X) = -\sum_{k=1}^{n} p_X(x_k) \log_2 p_X(x_k) = -\sum_{k=1}^{n} p_k (\log_2 p_k).$$

As discussed in class, observe that the entropy of X does not depend on the specific values $x_1, x_2, \ldots, x_n$ in its support. The entropy is a function of the probability values $p_1, p_2, \ldots, p_n$ in the pmf. Therefore, to calculate entropy, it is enough to specify these probabilities. From this observation, we may write the entropy as a function $H(\underline{\mathbf{p}})$ of a probability vector $\underline{\mathbf{p}}$ constructed from (positive probabilities in) the pmf.

In general, given a probability vector $\underline{\mathbf{p}} = [p_1, p_2, \ldots, p_n]$ whose elements are nonnegative and sum to one, we calculate the corresponding entropy value by

$$H(\underline{\mathbf{p}}) = -\sum_{k=1}^{n} p_k (\log_2 p_k).$$

In each row of the following table, compare the entropy value in the first column with the entropy value in the third column by writing ">", "=", or "<" in the second column. Watch out for approximation and round-off error.

Each entropy value can be calculated directly from the given row vectors,
Here, we use a MATLAB script (entropy2.m) to do this task. See 2.44 in the lecture note.

0.8813 ≈	$H(\underline{\mathbf{p}})$ when $\underline{\mathbf{p}} = [0.3, 0.7]$.	a)	>	$H(\underline{\mathbf{q}})$ when $\underline{\mathbf{q}} = [0.8, 0.2]$.	≈ 0.7219
1.5710 ≈	$H(\underline{\mathbf{p}})$ when $\underline{\mathbf{p}} = [0.3, 0.3, 0.4]$.	b)	=	$H(\underline{\mathbf{q}})$ when $\underline{\mathbf{q}} = [0.4, 0.3, 0.3]$.	≈ 1.5710
1.9710 ≈	$H(X)$ when $p(x) = \begin{cases} 0.3, & x \in \{1,2\}, \\ 0.2, & x \in \{3,4\}, \\ 0, & \text{otherwise.} \end{cases}$	c)	>	$H(\underline{\mathbf{q}})$ when $\underline{\mathbf{q}} = [0.4, 0.3, 0.3]$.	≈ 1.5710

A smarter way to solve this problem (without a calculator)

a) Note that $H(\underline{\mathbf{p}}) = H(0.3)$ and $H(\underline{\mathbf{q}}) = H(0.8)$
where the $H(\ )$ here denotes the binary entropy function. (See 2.47 in lecture note.)
This function gives larger value when its argument is closer to 0.5.
Here, $|0.3 - 0.5| = 0.2$, $|0.8 - 0.5| = 0.3$ } ⇒ Therefore, 0.3 is closer to 0.5. Hence, $H(0.3) > H(0.8)$

b) Note that the numbers inside both $\underline{\mathbf{p}}$ and $\underline{\mathbf{q}}$ are exactly the same (with different ordering). Therefore, by the commutative property of addition, the corresponding entropy values should be the same.

c) $H(X) = -0.3\log_2 0.3 - 0.3\log_2 0.3 - 0.2\log_2 0.2 - 0.2\log_2 0.2$
$H(\underline{\mathbf{q}}) = -0.3\log_2 0.3 - 0.3\log_2 0.3 - 0.4\log_2 0.4$
↳ only need to compare this part.

$0.4\log_2 0.4 = (0.2+0.2)\log_2 0.4 = 0.2\log_2 0.4 + 0.2\log_2 0.4$
$\log_2(\ )$ is an increasing func. → $> 0.2\log_2 0.2 + 0.2\log_2 0.2$
$-0.4\log_2 0.4 < -0.2\log_2 0.2 - 0.2\log_2 0.2$

Problem 6. These codes cannot be Huffman codes. Why?

(a) $\{00, 01, 10, 110\}$

(b) $\{01, 10\}$

(c) $\{0, 01\}$

(a) The code {00,01,10,110} can be shortened to {00,01,10,11}. The prefix-free property is preserved; so the shortened code is still UD. Regardless of the probabilty associated with each source symbol (as long as the last symbol has non-zero probability), the shortened code will have smaller expected length. Because Huffman code is optimal, the original code can't be Huffman.

(b) The code {01,10) can be shortened to {0,1}. The prefix-free property is preserved; so the shortened code is still UD. Regardless of the probabilty associated with each source symbol, the shortened code will have smaller expected length. Because Huffman code is optimal, the original code can't be Huffman.

(c) The code {0,01} is not prefix-free. Any Huffman code must be prefix-free. Therefore, it cannot be a Huffman code.

Alternatively, one can also use the same reasoning in part (a) and (b): The code {0,01) can be shortened to {0,1}. The new code is prefix-free and hence still UD. Regardless of the probabilty associated with each source symbol, the shortened code will have smaller expected length. Because Huffman code is optimal, the original code can't be Huffman.

Extra Questions

Here are some optional questions for those who want more practice.

Problem 7. (Optional) The following claim is sometimes found in the literature:

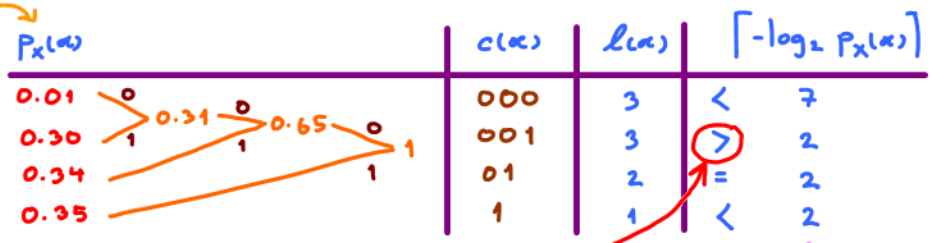
> "It can be shown that the length $\ell(x)$ of the Huffman code of a symbol x with probability $p_X(x)$ is always less than or equal to $\lceil -\log_2 p_X(x) \rceil$".

Even though it is correct in many cases, this claim is not true in general.

Find an example where the length $\ell(x)$ of the Huffman code of a symbol x is greater than $\lceil -\log_2 p_X(x) \rceil$.

Hint: Consider a pmf that has the following four probability values $\{0.01, 0.30, 0.34, 0.35\}$.

Following the hint, we create a Huffman code from the suggested probability values



This column is added for comparison

Use ceil(-log2(·)) in MATLAB

The claim says $\ell(x) \le \lceil -\log_2 p_X(x) \rceil$ for any x.

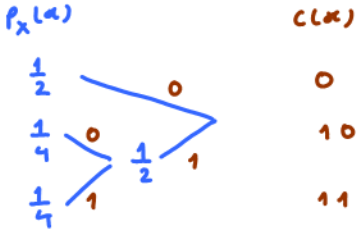
Here, we found a case where $\ell(x) > \lceil -\log_2 p_X(x) \rceil$.

Therefore, the claim is not true in general.

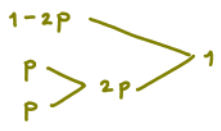
Problem 8. (Optional) Construct a random variable X (by specifying its pmf) whose corresponding Huffman code is $\{0, 10, 11\}$.

$\{0, 10, 11\}$ is a Huffman code for

$$p_X(x) = \begin{cases} 1/2, & x = 1 \\ 1/4, & x = 2, 3 \\ 0, & \text{otherwise.} \end{cases}$$



Note that the answer for this question is not unique. You may check that



$$p_X(x) = \begin{cases} 1-2p, & x = 1 \\ p, & x = 2, 3 \\ 0, & \text{otherwise} \end{cases}$$

works when

$1 - 2p \ge p$

$\Updownarrow$

$p \le \frac{1}{3}$

This condition is here to guarantee that x = 2 and x = 3 are grouped frist.

Note also that when $p = \frac{1}{3}$, we have uniform pmf.