Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science
Department of Mechanical Engineering

6.050J/2.110J **Information and Entropy** **Spring 2003**

Problem Set 4 Solutions

Solution to Problem 1: Meet the Box People

Solution to Problem 1, part a.

The probability that one of the box people's offspring has different phenotypes is as follows:

i. An offspring has a circular phenotype unless s/he has genes cc. The probability of having cc is equal to the probability that each parent transmits gene c, squared (because they are independent events). Thus, this probability is $0.7^2 = 0.49$ This means that the probability that the offspring has a circular shape is 1 - 0.49 = 0.51.

ii. An offspring has a square phenotype when it inherits the recessive gene from both parents. Inheriting a recessive gene from one parent occurs with a probability of 0.7, thus the probability of inheriting a recessive gene from both parents is $0.7^2 = 0.49$, as before.

iii. Using the same reasoning, we get that the probability of a blue phenotype is 0.25.

iv. Similarly, the probability of a red phenotype is 0.75.

Solution to Problem 1, part b.

Given that an offspring is circular, the probability of other phenotypes are as follows:

i. Because the genes are independent, the probability of having a blue phenotype given a circular shape is equal to the probability of having a blue phenotype (gene rr), which is 0.25.

ii. Similarly, the probability of having a red phenotype given a circular shape is 0.75.

Solution to Problem 1, part c.

Given that an offspring is red, the probability of other phenotypes are as follows:

i. Again, the genes are independent, thus the probability of a box shape given a red color is 0.49.

ii. The probability of a circular shape given a red color is 1 - 0.49 = 0.51.

Solution to Problem 1, part d.

We can draw a table of events as shown in Table 4–2.

We see that the probability that a person has the disease given that the test is positive, is:

$$\frac{0.001 \times 0.95}{0.001 \times 0.95 + 0.999 \times 0.004} = 19.2\% \tag{4–2}$$

Have Disease?	Percent	Test Results	Percent	Total
Yes	0.001	Positive	0.95	0.00095
		Negative	0.05	0.00005
No	0.999	Positive	0.004	0.003996
		Negative	0.996	0.95504

Table 4–2: Triangularity Test Results

Solution to Problem 2: Huffman Coding

Solution to Problem 2, part a.

To encode fourteen symbols we would need four bits, for $2^4 = 16$ different codewords. This gives $4 \times 44 = 176$ bits.

Solution to Problem 2, part b.

Table 4–3 lists the calculation of the average information per symbol. Here we calculate an average of 3.33 bits per symbol, or 147 bits.

Character	Frequency	$\log_2\left(\frac{1}{p_i}\right)$	$p_i \log_2\left(\frac{1}{p_i}\right)$
p	20.46%	2.29	0.46
e	18.18%	2.45	0.44
space	15.91%	2.65	0.42
c	6.82%	3.87	0.26
i	6.82%	3.87	0.26
k	6.82%	3.87	0.26
r	6.82%	3.87	0.26
d	4.55%	4.45	0.20
a	2.27%	5.46	0.12
f	2.27%	5.46	0.12
l	2.27%	5.46	0.12
o	2.27%	5.46	0.12
s	2.27%	5.46	0.12
t	2.27%	5.46	0.12
Total	100.00		3.33

Table 4–3: Frequency distribution of characters in “peter piper picked a peck of pickled peppers”

Solution to Problem 2, part c.

See Table 4–3.

Solution to Problem 2, part d.

A possible code is derived below and listed in Table 4–4.

Start: (p=‘NA’ $p = 0.2046$) (e=‘NA’ $p = 0.1818$) (space=‘NA’ $p = 0.1591$) (c=‘NA’ $p = .0682$) (i=‘NA’ $p = .0682$) (k=‘NA’ $p = .0682$) (r=‘NA’ $p = .0682$) (d=‘NA’ $p = .0455$) (a=‘NA’ $p = .0227$) (f=‘NA’ $p = .0227$) (l=‘NA’ $p = .0227$) (o=‘NA’ $p = .0227$) (s=‘NA’ $p = .0227$) (t=‘NA’ $p = .0227$)

Next: (p='NA' $p = 0.2046$) (e='NA' $p = 0.1818$) (space='NA' $p = 0.1591$) (c='NA' $p = .0682$) (i='NA' $p = .0682$) (k='NA' $p = .0682$) (r='NA' $p = .0682$) (d='NA' $p = .0455$) (a='NA' $p = .0227$) (f='NA' $p = .0227$) (l='NA' $p = .0227$) (o='NA' $p = .0227$) (s='0', t='1' $p = .0454$)

Next: (p='NA' $p = 0.2046$) (e='NA' $p = 0.1818$) (space='NA' $p = 0.1591$) (c='NA' $p = .0682$) (i='NA' $p = .0682$) (k='NA' $p = .0682$) (r='NA' $p = .0682$) (d='NA' $p = .0455$) (a='NA' $p = .0227$) (f='NA' $p = .0227$) (l='0', o='1' $p = .0454$) (s='0', t='1' $p = .0454$)

Next: (p='NA' $p = 0.2046$) (e='NA' $p = 0.1818$) (space='NA' $p = 0.1591$) (c='NA' $p = .0682$) (i='NA' $p = .0682$) (k='NA' $p = .0682$) (r='NA' $p = .0682$) (d='NA' $p = .0455$) (a='0', f='1' $p = .0454$) (l='0', o='1' $p = .0454$) (s='0', t='1' $p = .0454$)

Next: (p='NA' $p = 0.2046$) (e='NA' $p = 0.1818$) (space='NA' $p = 0.1591$) (c='NA' $p = .0682$) (i='NA' $p = .0682$) (k='NA' $p = .0682$) (r='NA' $p = .0682$) (d='NA' $p = .0455$) (a='0', f='1' $p = .0454$) (l='00', o='01', s='10', t='11' $p = .0908$)

Next: (p='NA' $p = 0.2046$) (e='NA' $p = 0.1818$) (space='NA' $p = 0.1591$) (c='NA' $p = .0682$) (i='NA' $p = .0682$) (k='NA' $p = .0682$) (r='NA' $p = .0682$) (d='0', a='10', f='11' $p = .0909$) (l='00', o='01', s='10', t='11' $p = .0908$)

Next: (p='NA' $p = 0.2046$) (e='NA' $p = 0.1818$) (space='NA' $p = 0.1591$) (c='NA' $p = .0682$) (i='NA' $p = .0682$) (k='0',r='1' $p = .1364$) (d='0', a='10', f='11' $p = .0909$) (l='00', o='01', s='10', t='11' $p = .0908$)

Next: (p='NA' $p = 0.2046$) (e='NA' $p = 0.1818$) (space='NA' $p = 0.1591$) (c='0', i='1' $p = .1364$) (k='0', r='1' $p = .1364$) (d='0', a='10', f='11' $p = .0909$) (l='00', o='01', s='10', t='11' $p = .0908$)

Next: (p='NA' $p = 0.2046$) (e='NA' $p = 0.1818$) (space='NA' $p = 0.1591$) (c='0', i='1' $p = .1364$) (k='0', r='1' $p = .1364$) (d='00', a='010', f='011', l='100', o='101', s='110', t='111' $p = .1817$)

Next: (p='NA' $p = 0.2046$) (e='NA' $p = 0.1818$) (space='NA' $p = 0.1591$) (c='00', i='01', k='10', r='11' $p = .2728$) (d='00', a='010', f='011', l='100', o='101', s='110', t='111' $p = .1817$)

Next: (p='NA' $p = 0.2046$) (e='NA' $p = 0.1818$) (c='00', i='01', k='10', r='11' $p = .2728$) (space='0', d='100', a='1010', f='1011', l='1100', o='1101', s='1110', t='1111' $p = .3408$)

Next: (p='0', e='1' $p = 0.3864$) (c='00', i='01', k='10', r='11' $p = .2728$) (space='0', d='100', a='1010', f='1011', l='1100', o='1101', s='1110', t='1111' $p = .3408$)

Next: (p='0', e='1' $p = 0.3864$) (c='000', i='001', k='010', r='011', space='10', d='1100', a='11010', f='11011', l='11100', o='11101', s='11110', t='11111' $p = .6136$)

Final: (p='00', e='01', c='1000', i='1001', k='1010', r='1011', space='110', d='11100', a='111010', f='111011', l='111100', o='111101', s='111110', t='111111' $p = 1.0000$)

Solution to Problem 2, part e.

When the sequence is encoded using the codebook derived in part d…

i. See Table 4–5.

ii. The fixed length code requires 176 bits, whereas Huffman coding requires 149 bits. So we find that the Huffman code does a better job than the fixed length code.

Character	Code
p	00
e	01
space	110
c	1000
i	1001
k	1010
r	1011
d	11100
a	111010
f	111011
l	111100
o	111101
s	111110
t	111111

Table 4–4: Huffman code for “peter piper picked a peck of pickled peppers”

Character	# of Characters	Bits per Character	Bits Needed
p	9	2	18
e	8	2	16
space	7	3	21
c	3	4	12
i	3	4	12
k	3	4	12
r	3	4	12
d	2	5	10
a	1	6	6
f	1	6	6
l	1	6	6
o	1	6	6
s	1	6	6
t	1	6	6
Total	44		149

Table 4–5: Huffman code for “peter piper picked a peck of pickled peppers”

iii. This number compares extremely well with the information content of 147 bits for the message as a whole.

Solution to Problem 2, part f.

The original message is 44 bytes long, and with LZW we know from Problem Set 2 we can encode the message using LZW in 32 bytes, with 31 characters in the dictionary. Thus we need 32+14=46 different dictionary entries, for a total of six bits per byte. Thus we can compact the message down to $32 \times 6 = 192$ characters. Straight encoding needs 176 bits, and Huffman encoding needs 149 bits. Thus Huffman encoding does the best job of compacting the material.

A lower bound on sending the Huffman codebook is the number of bits in the code, total. This is equal to $2+2+3+4+4+4+4+5+6+6+6+6+6+6 = 64$ bits. If we imagine that we need to send some control bits along, perhaps it is something like five bits between each code (a reasonable estimate), this is an additional $5 \times (14+1) = 75$ bits. So we have an lower-bound estimate of 139 bits.

Thus a fixed-length code requires 176 bits, LZW needs 192 bits, and Huffman coding with the transmission of the codebook requires an estimated 296 bits.

6.050J / 2.110J Information and Entropy
Spring 2008