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

6.050J/2.110J **Information and Entropy** **Spring 2004**

Problem Set 5 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 $0.5^2 = 0.25$ This means that the probability that the offspring has a circular shape is 1 - 0.25 = 0.75.

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 0.5, thus the probability of inheriting a recessive gene from both parents is $0.5^2 = 0.25$, 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 0, part .

Bob has every reason to suspect that Ann has cheated on him. If he had sired their 5,000 children, statistically one would expect that only a quarter of them would be red. To produce on average half red children with Ann, the father would need to have both recessive red genes, or rr.

Solution to Problem 1, part c.

For Bob's sets of children with Carol, one would expect 75% circular and 25% square. With Dorothy, one would expect 50% circular and 50% square. With Cathy, 100% circular.

Solution to Problem 2: The Dreaded Triangle Transformation

We can draw a table of events as shown in Table 5–2.

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

$$\frac{0.001 \times 0.95}{0.001 \times 0.95 + 0.999 \times 0.004} = 19.2\% \tag{5–2}$$

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 5–2: Triangularity Test Results

Solution to Problem 3: Huffman Coding

Solution to Problem 3, part a.

To encode nine symbols we would need four bits, which would give us $2^4 = 16$ different codewords. This gives $4 \times 29 = 116$ bits needed for transmission.

Solution to Problem 3, part b.

Table 5–3 lists the calculation of the average information per symbol. Here we calculate an average of 2.749 bits per symbol, or 80 bits.

Character	Frequency	$\log_2\left(\frac{1}{p_i}\right)$	$p_i \log_2\left(\frac{1}{p_i}\right)$
a	27.59%	1.8576	0.5125
t	24.14%	2.0506	0.4950
space	13.79%	2.8582	0.3941
-	10.34%	3.2738	0.3385
s	10.34%	3.2738	0.3385
R	3.45%	4.8573	0.1676
e	3.45%	4.8573	0.1676
y	3.45%	4.8573	0.1676
h	3.45%	4.8573	0.1676
Total	100.00%		2.7490

Table 5–3: Frequency distribution of characters in “Rat-a-tat-tat as easy as that”

Solution to Problem 3, part c.

See Table 5–3.

Solution to Problem 3, part d.

A possible code is derived below and listed in Table 5–4.

Start: (a=‘NA’ $p = 0.2759$) (t=‘NA’ $p = 0.2414$) (space=‘NA’ $p = 0.1379$) (‘-’=‘NA’ $p = 0.1034$) (s=‘NA’ $p = 0.1034$) (R=‘NA’ $p = 0.0345$) (e=‘NA’ $p = 0.0345$) (y=‘NA’ $p = 0.0345$) (h=‘NA’ $p = 0.0345$)

Next: (a=‘NA’ $p = 0.2759$) (t=‘NA’ $p = 0.2414$) (space=‘NA’ $p = 0.1379$) (‘-’=‘NA’ $p = 0.1034$) (s=‘NA’ $p = 0.1034$) (R=‘NA’ $p = 0.0345$) (e=‘NA’ $p = 0.0345$) (y=‘0’, h=‘1’ $p = 0.069$)

Next: (a=‘NA’ $p = 0.2759$) (t=‘NA’ $p = 0.2414$) (space=‘NA’ $p = 0.1379$) (‘-’=‘NA’ $p = 0.1034$) (s=‘NA’ $p = 0.1034$) (R=‘0’, e=‘1’ $p = 0.069$) (y=‘0’, h=‘1’ $p = 0.069$)

Next: (a='NA' $p = 0.2759$) (t='NA' $p = 0.2414$) (space='NA' $p = 0.1379$) ('-'='NA' $p = 0.1034$) (s='NA' $p = 0.1034$) (R='00', e='01', y='10', h='11' $p = 0.138$)

Next: (a='NA' $p = 0.2759$) (t='NA' $p = 0.2414$) (space='NA' $p = 0.1379$) ('-'='0', s='1' $p = 0.2068$) (R='00', e='01', y='10', h='11' $p = 0.138$)

Next: (a='NA' $p = 0.2759$) (t='NA' $p = 0.2414$) ('-'='0', s='1' $p = 0.2068$) (space='0', R='100', e='101', y='110', h='111' $p = 0.2759$)

Next: (a='NA' $p = 0.2759$) (t='0', '-'='10', s='11' $p = 0.4482$) (space='0', R='100', e='101', y='110', h='111' $p = 0.2759$)

Next: (t='0', '-'='10', s='11' $p = 0.4482$) (a='0', space='10', R='1100', e='1101', y='1110', h='1111' $p = 0.5581$)

Final: (t='00', '-'='010', s='011', a='10', space='110', R='11100', e='11101', y='11110', h='11111' $p = 1.00$)

Character	Code
a	10
t	00
space	110
-	010
s	011
R	11100
e	11101
y	11110
h	11111

Table 5–4: Huffman code for "Rat-a-tat-tat as easy as that"

Solution to Problem 3, part e.

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

i. See Table 5–5.

Character	# of Characters	Bits per Character	Bits Needed
a	8	2	16
t	7	2	14
space	4	3	12
-	3	3	9
s	3	3	9
R	1	5	5
e	1	5	5
y	1	5	5
h	1	5	5
Total	29		80

Table 5–5: Huffman code for "Rat-a-tat-tat as easy as that"

ii. The fixed length code requires 116 bits, whereas Huffman coding requires 80 bits. So we find that the Huffman code does a better job than the fixed length code.

iii. This number compares extremely well with the information content of 116 bits for the message as a whole.

Solution to Problem 3, part f.

The original message is 29 bytes long, and with LZW we know from Problem Set 3 we can encode the message using LZW in 34 bytes, with 22 characters in the dictionary. Thus we need 34+22=56 different dictionary entries, for a total of six bits per byte. Thus we can compact the message down to $29 \times 6 = 174$ characters. Straight encoding needs 116 bits, and Huffman encoding needs 80 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 $2+2+3+3+3+5+5+5+5 = 34$ 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 $5 \times (9+1) = 50$ bits. So we have an lower-bound estimate of 84 bits.

Thus a fixed-length code requires 116 bits, LZW needs 174 bits, and Huffman coding with the transmission of the codebook requires an estimated 164 bits.