ECE 499/599 Data Compression/Information Theory
Spring 06

Dr. Thinh Nguyen

Homework 3
Due 05/04/06 at the beginning of the class

Problem 1: For an alphabet A = {a1, a2, a3} with probabilities P(a1) = 0.6, P(a2) = 0.3, and P(a3) = 0.1. (3pts)

(a) Design a 3-bit Tunstall code for this alphabet.
(b) Find the redundancy (code rate – entropy rate).

(a) Tunstall coding for the alphabet A = {a1, a2, a3} with P(a1) = 0.6, P(a2) = 0.3, and P(a3) = 0.1.
Choose m = 3,

Sequences	Probability	Codeword
a2	0.3	000
a3	0.1	001
a1a2	0.18	010
a1a3	0.06	011
a1a1a1	0.216	100
a1a1a2	0.108	101
a1a1a3	0.036	110

(b)

Code Rate = $\frac{3}{0.3*1+0.1*1+0.18*2+0.06*2+0.216*3+0.108*3+0.036*3} = 1.5306$

Entropy Rate = $-(0.6*\log 0.6+0.3*\log 0.3+0.1*\log 0.1) = 1.2955$

Redundancy = 1.5306-1.2955 = 0.2351

Problem 2: Suppose you saw this one game in which, a guy repeatedly tosses a fair coin (at least he claims that it is a fair coin, and hence with probability P(head) = 1/2) until either (a) the outcome is head or (b) the number of consecutive tail outcomes reaches 4. (8pts)

(a) Code these outcomes using Golomb code with m = 4. What is average code rate? (note that the uncoded outcomes have the forms: 1, 01, 001, 0001, 0000, with 1 representing head and 0 representing tail.)

(Assume bit "1" represents head and "0" for tail. Then:

Sequences	**Probability**	**Codeword**

0000	1/16	1
0001	1/16	011
001	1/8	010
01	1/4	001
1	1/2	000

Average code rate = $\dfrac{1/16*1+(1/16+1/8+1/4+1/2)*3}{1/16*4+1/16*4+1/8*3+1/4*2+1/2*1}=\dfrac{23}{15}=1.533$

(b) Being a very observant person, you notice that the guy is actually cheating his audiences by using a biased coin, in which the probability of head P(head) is not 1/2. Using m = 4, can you derive the equation for the average Golomb code rate as a function of P(head)?

With P(head) = p and P(tail) = 1-p, thus:

Sequences	**Probability**	**Codeword**
0000	$(1-p)^4$	1
0001	$p\,(1-p)^3$	011
001	$p(1-p)^2$	010
01	$p(1-p)$	001
1	p	000

$$\text{Average code rate}=\frac{(1-p)^4+3p(1-p)^3+3p(1-p)^2+3p(1-p)+3p}{4(1-p)^4+4p(1-p)^3+3p(1-p)^2+2p(1-p)+p}$$
$$=\frac{(1-p)^4+3\left(1-(1-p)^4\right)}{4(1-p)^4+4(1-p)^3p+3(1-p)^2p+2(1-p)p+p}$$
$$=\frac{3-2(1-p)^4}{4(1-p)^4+4(1-p)^3p+3(1-p)^2p+2(1-p)p+p}$$

(c) If you are going to code these outcomes using runlength code, what is the average code rate as a function of probability P(head), assuming you always code the run-lengths of tails using 2-bit fixed-lengh code?

With P(head) = p and P(tail) = 1-p
Code the run-length of 0's using 2-bit run-length coding:

Sequences	Probability	Codeword
0000	(1-p)4	1101

0001	p(1-p)3	1100
001	p(1-p)2	10
01	p(1-p)	01
1	p	00

$$\text{Average code rate} = \frac{4(1-p)^4 + 4p(1-p)^3 + 2p(1-p)^2 + 2p(1-p) + 2p}{4(1-p)^4 + 4p(1-p)^3 + 3p(1-p)^2 + 2p(1-p) + p}$$

Problem 3: Do problem 7 in chapter 4. This problem refers to integer arithmetic coding with scaling. Show steps by steps during encoding and decoding. (8pts)

(a)

Let:

n_1: the number of times that symbol a occurs

n_2: the number of times that symbol b occurs

n_3: the number of times that symbol c occurs

And, $C_i = n_1 + n_2 + ... + n_i$

Then, $n_1 = 37; n_2 = 38; n_3 = 25$

$C_0 = 0; C_1 = 37; C_2 = 75; C_3 = 100$

Thus, the total interval range has to be greater than 400.

So the required word length is: m = 9

(b) Using m = 9, encode the input sequence *abacabb*

Step	**Symbol**	**L, R**	**R-L+1**	**C**	**Output Bit**
Initial		$L_0 = 0 = 000000000$ $R_0 = 511 = 111111111$	512	0	
1	a	$L_1 = 0 + \left\lfloor \frac{512 \times 0}{100} \right\rfloor = 0 = 000000000$ $R_1 = 0 + \left\lfloor \frac{512 \times 37}{100} \right\rfloor - 1 = 188 = 010111100$ Fall in lower half , output bit 0 and shift left: $L_1 = 000000000$ $R_1 = 101111001 = 377$	189 378	0	 0
2	b	$L_2 = L_1 + \left\lfloor \frac{378 \times 37}{100} \right\rfloor = 139 = 010001011$	144	0	

		$R_2 = L_1 + \left\lfloor \frac{378 \times 75}{100} \right\rfloor - 1 = 282 = 100011010$ Fall in middle, update C and shift left: L_2 = 000010110 = 22 R_2 = 100110101 = 309	288	1	
3	a	L_3 = 000010110 =22 R_3 = 001111111 = 127 Fall in lower half , output bit 0 and (C = 1) bit 1, then shift left: L_3 = 000101100 = 44 R_3 = 011111111 = 255 Fall in lower half , output bit 0 and shift left: L_3 = 001011000 = 88 R_3 = 111111111 = 511	106 212 424	1 0	01 0
4	c	L_4 = 110010110 = 406 R_4 = 111111111 = 511 Fall in upper half , output bit 1 and shift left: L_4 = 100101100 = 300 R_4 = 111111111 = 511 Fall in upper half , output bit 1 and shift left: L_4 = 001011000 = 88 R_4 = 111111111 = 511	106 212 424		1 1
5	a	L_5 = 001011000 = 88 R_5 = 011110011 = 243 Fall in lower half , output bit 0 and shift left: L_5 = 010110000 = 176 R_5 = 111100111 = 487	156 312		0
6	b	L_6 = 100100011 = 291 R_6 = 110011001 = 409 Fall in upper half, output bit 1 and shift left: L_6 = 001000110 = 70 R_6 = 100110011 = 307	119 238		1
7	b	L_7 = 010011110 = 158 R_7 = 011110111 = 247 Fall in lower half, output bit 0 and shift left: L_7 = 100111100 = 316 R_7 = 111101111 = 495 Fall in upper half, output bit 1 and shift left: L_7 = 001111000 = 120 R_7 = 111011111 = 479	90 180 360		0 1

To this point, we generate the output binary sequence **0 0 1 0 1 1 0 1 0 1.** We want to terminate the encoding at this point, so we send the current tag (that is L_7 = 001111000). Thus the final transmitted sequence is **0 0 1 0 1 1 0 1 0 1 0 0 1 1 1 1 0 0 0**

(c) Decoding sequence **0 0 1 0 1 1 0 1 0 1 0 0 1 1 1 1 0 0 0**

- Using the same word length with the encoder, we form tag T:

  T = 001011010 = 90

  We initialize the upper and lower limit as:

  $L_0 = 0 = 000000000$

  $R_0 = 511 = 111111111$

  Then we compute:

  $$\left\lfloor \frac{(T - L + 1) \times Total - 1}{R - L + 1} \right\rfloor = 17$$

  Since 0<17<37, we decode symbol **a**

- Update the upper and lower limit as:

  R = 010111100 = 188

  L = 000000000 = 0

  Fall in the lower half, shift left:

  L = 000000000 = 0

  R = 101111001 = 377

  T = 010110101 = 181

  Compute:

  $$\left\lfloor \frac{(T - L + 1) \times Total - 1}{R - L + 1} \right\rfloor = 48$$

  Since 37< 48 < 75, we decode symbol **b**

- Update the upper and lower limit as:

  R = 100011010 = 282

  L = 010001011 = 139

  Fall in the middle, shift left and complement the MSB bit:

  L = 000010110 = 22

  R = 100110101 = 309

  T = 001101010 = 106

  Compute:

  $$\left\lfloor \frac{(T - L + 1) \times Total - 1}{R - L + 1} \right\rfloor = 29$$

  Since 0 < 29 < 37, we decode symbol **a**

- Update the upper and lower limit as:

  R = 001111111 = 127

  L = 000010110 = 22

  Fall in the lower half, shift left:

L = 000101100 = 44

R = 011111111 = 255

T = 011010100 = 212

Fall in the lower half, shift left:

L = 001011000 = 88

R = 111111111 = 511

T = 110101001 = 425

Compute:

$$\left\lfloor \frac{(T-L+1)\times Total-1}{R-L+1} \right\rfloor = 79$$

Since 75 < 79 < 100, we decode symbol **c**

- Update the upper and lower limit as:

R = 111111111 = 511

L = 110010110 = 406

Fall in the upper half, shift left:

L = 100101100 = 300

R = 111111111 = 511

T = 101010011 = 339

Fall in the upper half, shift left:

L = 001011000 = 88

R = 111111111 = 511

T = 010100111 = 167

Compute:

$$\left\lfloor \frac{(T-L+1)\times Total-1}{R-L+1} \right\rfloor = 18$$

Since 0 < 18 < 37, we decode symbol **a**

- Update the upper and lower limit as:

R = 011110011 = 243

L = 001011000 = 88

Fall in the lower half, shift left:

L = 010110000 = 176

R = 111100111 = 487

T = 101001111 = 335

Compute:

$$\left\lfloor \frac{(T-L+1)\times Total-1}{R-L+1} \right\rfloor = 51$$

Since 37 < 51 < 75, we decode symbol **b**

- Update the upper and lower limit as:

R = 110011001 = 409

L = 100100011 = 291

Fall in the upper half, shift left:

L = 001000110 = 70

R = 100110011 = 307

T = 010011110 = 158

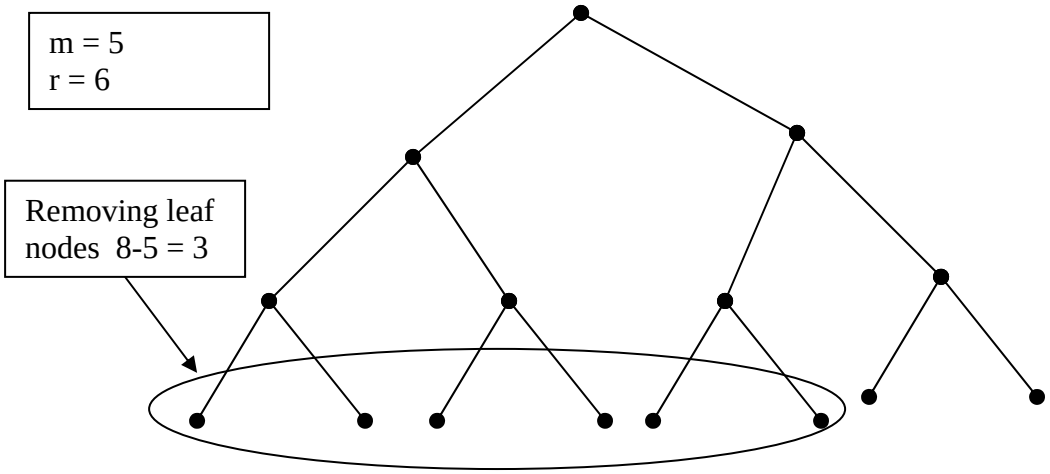
Compute:

$$\left\lfloor \frac{(T - L + 1) \times Total - 1}{R - L + 1} \right\rfloor = 37$$

Since 37 <= 37 < 75, we decode symbol **b**

Thus, the decode sequence is **a b a c a b b**

Problem 4: (bonus) Show that the remainder bits in Golomb code can be viewed as a prefix code (1pt).



The proof is based on the fact that if we can construct a tree that represents the code for the remainder then we prove that the code is a prefix code.

Note that a leaf of a full binary tree of L level is basically the traditional binary encoding using L bits. Suppose floor[log(m)] = k and therefore ceil(log(m)] = k + 1.

Clearly, 2^k <= m <= 2^(k+1). Now 2^(k+1) – m <= 2^k. So, we can remove all the children belonging to 2^(k+1) – m nodes out of 2^k at the k level of the tree from left to right. So the tree now is unbalanced with the left side is shorter than right side. You can see that the leaf nodes of the shorter branches are precisely the k-bit binary coding for the first 2^(k+1) – m values in the the remainder. The number of leaf nodes remains in the (k+1) level is now 2^(k+1) – 2(2^(k+1) – m) = 2m-2^(k+1)

The number of values remains in the remainder is r – 2^(k+1) + m which is smaller than 2m-2^(k+1) since r < m. Hence we can assign the rest of the values in the remainder with the codes represented by the leaf nodes in the k+1 level. This is precisely the binary representation of r +2^{k+1} – m.