Note: Always assign 0 for left (or lower) branch in any binary tree.

Information Theory

Suppose that X is a random variable that takes on values from an M-letter alphabet. Show that $0 \le H(X) \le \log_2 M$

An experiment has 2 binary random variable outputs X & Y, with the shown joint probabilities. Calculate the amount of information gained, or expected, from knowing each of the following:

P(X,Y)			
		Y	
		0	1
X	0	0.2	0.1
	1	0.1	0.6

a) Y = 1. {2}

b) The value of X. {2}

c) The value of Y, Given that X= 0. {2}

d) The values of both X, Y. {1}

e) The average common information between X and Y. {1}

	a	b	C	d	E	Units
Info =						

Markov finite state Model

A source emits iid symbols form the alphabet {a, b, & c}, with the following probabilities. Calculate the entropy of that source.

Symbol	a	B	C
Prob.	0.25	0.25	0.5
Entropy =		Bytes	

The probability of the current symbol, X_n, emitted form the previous source, is found to be related to the previous symbol, X_{n-1}, as shown in table. Recalculate the entropy of the source.

$P(X_n / X_{n-1})$				
		X_n		
		A	b	C
X_{n-1}	a	$\frac{1}{2}$	$\frac{1}{8}$	$\frac{3}{8}$
	b	$\frac{1}{8}$	$\frac{1}{2}$	$\frac{3}{8}$
	c	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{3}{4}$

Entropy = Bytes

The bases of a genomic sequence have the probabilities shown. Calculate the self-information of “C”, and the entropy of the sequence. {8}

Base	A	C	G	T
Prob.	0.125	0.125	0.25	0.5
Self-Information of C				
The Entropy of the Seq.				

The probability of the any base, B_n, in the previous sequence, is found to be related to its previous base, B_{n-1}, by the probabilities given in the following table. Recalculate the entropy of the sequence. {12}

		$P(B_n / B_{n-1})$			
		B_n			
		A	C	G	T
B_{n-1}	A	0.4	.0.2	0.1	0.3
	C	0.2	0.4	0.2	0.2
	G	0.1	0.1	0.3	0.5
	T	0.05	0.05	0.275	0.625

Decodability

Determine whether each of the following codes is uniquely decodable (UD), prefix, instantaneous, or near instantaneous:

Code	UD	Prefix	Instantaneous	Near-Instant.
{0,01,11,111}				
{0,01,110,111}				
{0,10,110,111}				
{1,10,110,111}				

An alphabet of 7 symbols, had code lengths of 2, 2, 3, 3, 3, 3, & 3 bits. Is this code uniquely decodable? Why? {4}
Answer:
Reasons:

What the largest number is of symbols that can be coded using a ternary (i.e. 3 symbols) code, while the code length of each symbol does not exceed 3? Why? {3}
Answer:
Reasons:

14 symbols are coded using a ternary (i.e. 3) code. The code lengths are 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3,

3, 3, & 3. Is this code uniquely decodable? Why? {2}
Answer:
Reasons:

An alphabet of 7 symbols, had code lengths of 2, 2, 3, 3, 3, 3, & 3 bits. Is this code uniquely decodable? Why? {2}
Answer:
Reasons:

Using Kraft's Inequality, check the decidability of the following code table. Next, check the code table. Is the code decodable? Why? Calculate the entropy, the average code length, and the redundancy		Prob	Code
	a	0.125	0
	b	0.125	10
	c	0.25	110
	D	0.5	101

Kraft's inequality result:
Decodability based on Kraft's inequality:
Is the code decodable?
Why?
Average Code Length =
Redundancy =

Assume a 5-symbol alphabet {A, B, C, D, & E}. Build a prefix code for the symbols.

Symbol	**A**	**B**	**C**	**D**	**E**
Code					

Code the following sequence: CBABAA

Symbol	**C**	**B**	**A**	**B**	**A**	**A**
Code						

Huffman Coding

17/5/2004

2. Design each of the following codes, and then calculate its average coding length, and redundancy, per symbol.

a) A minimum variance Huffman code for the alphabet {A, B, C, D}, with probabilities, 0.1, 0.2, 0.3, & 0.4, respectively.

b) A minimum variance extended Huffman code for the binary alphabet {0, 1}, with probabilities 0.1 and 0.9, respectively. Generate a code word for every block of 3 symbols.

c) A 3-bit Tunstall code for the above binary alphabet.

Min-Var Huffman		Huffman		Extended Huffman		Tunstall	
Symbol	Code	Symbol	Code	Symbol	Code	Symbol	Code
$L_{Average}$		$L_{Average}$		$L_{Average}$		$L_{Average}$	
Red		Red		Red		Red	

An alphabet is composed of the 6 symbols {A, C, N, T, Y, Δ}. Design a minimum length code for the alphabet. Assign short codes for small index symbols. Use this code as a starting code during next problems.			
Code Tree	**index**	**Sym**	**Code**
	1	**A**	
	2	**C**	
	3	**N**	
	4	**T**	
	5	**Y**	
	6	**Δ**	

Using adaptive Huffman coding, a) code the following: "ACACTAN". Show the development of the coding tree, and the code generated for each symbol								
Adaptive Huffman	**Symbol**	**A**	**C**	**A**	**C**	**T**	**A**	**N**
	Code							
b) Decode the first 5 symbols of the same alphabet, form the stream "011101100101100110110000111101100010100110001101 0101"								
	Symbol							
	Code							

Use the Golomb code, with m=2, to code the above sequence.								
Golomb	**Symbol**	**A**	**C**	**A**	**C**	**T**	**A**	**N**
	Code							
Code the above sequence using Move-To-Front.								
	Symbol	**A**	**C**	**A**	**C**	**T**	**A**	**N**
	Code							

Code the number 7 using

a) Golomb code, with m=3.

b) Rice code, fundamental sequence.

c) Rice code, split sample option, with m=3, and k=5.

Tech	Golomb	Rice, Fund.	Rice, Split
Code			

Code the number 6. First, use Golomb code with m=3. Next use Recursive indexing, with a representation alphabet of size 3. Use variable length code for entropy coding, with smaller numbers having short length. Draw the VLC tree. Label different parts of each code.		**VLC Tree**
	Golomb Code:	**Recursive indexing**
Code		
Labels		

Using adaptive Huffman coding, code the string "COCOAC". Show the development of the coding tree. Assume that the used alphabet has only the 3 characters A, C, and O. Use VLC to for new character coding.
code:
Coding Trees

Arithmetic

Given the frequency counts shown:			
Symbol	**A**	**C**	**G**
Count	**9**	**2**	**17**

What is the word length required for a digital arithmetic encoder to unambiguously encode the above symbols?

Word Length =   bits

Calculate the cumulative count and the total count for the above symbols.

Symbol	**A**	**C**	**G**	**Total Count**
CumCount				

Using a word length of 8 bits, encode the symbols "CACG". Use the table given below to report your results. Give *l* and *u* in binary form. Use E to identify the scale operation. Assume the given initial *l* & *u* values, and initial scale3 = 4. {14}

Step	**Symbol**	***L***	***u***	**E(Scale)**	**scale3**	**Code**
n-1		**00111111**	**11000001**	**--------**	**4**	**-------------------------**
n	**C**					
n+1						
n+2						
n+3						
n+4						
n+5						
n+6						
n+7						
n+8						
n+9						
n+10						
n+11						
n+12						

Using a word length of 8 bits, decode the first 5 symbols of the stream "01110001000001111100100000011110010101010010100 10101". Use the table given below to report your results. Give *l* and *u* in binary form. Use E to identify the scale operation. Assume the given initial *l* & *u* values, and initial scale3 = 4. {14}

Step	Symbol	*L*	*u*	Tag	E(Scale)	scale3
n-1		00111111	11000001		----------	4
n	C					

Dictionary

Given the sequence {10101101001000100110101010010111111000000}, use(LZ78/W/Y) to encode/decode the first 7 phrases. Show the generated phrases, the corresponding code, and the encoder tree after each coding/decoding cycle. Use suppression, excision and prefix coding. {21}																
Phrase 1																
Code 1																
Phrase 2																
Code 2																
Phrase 3																
Code 3																
Phrase 4																
Code 4																
Phrase 5																
Code 5																
Phrase 6																
Code 6																
Phrase 7																
Code 7																

Coding Trees		
Initial Tree	**Tree 1**	**Tree 2**
Tree 3	**Tree 4**	**Tree 5**

Tree 6	Tree 7

Using LZ77 (Tail biting, Future), encode, in binary, the first four phrases following the underlined window. Indicate the meaning of each part of the code, e.g. new character, Length, …etc. <u>011001001111010001001111110110011101110</u> 1000100111111001100111011101001110001001111000111000001111
First Phrase:
Second Phrase:
Third Phrase:
Fourth Phrase:

Code the first 6 phrases of the sequence {1011011011100100101111111001…}. Use LZ77. Show the generated phrases, the corresponding code, and the search window after each coding cycle. {10}

Phrase 1																
Code 1																
Window1																
Phrase 2																
Code 2																
Window2																
Phrase 3																
Code 3																
Window3																
Phrase 4																
Code 4																
Window4																
Phrase 5																
Code 5																
Window5																
Phrase 6																
Code 6																
Window6																

Predictive

Use ppm technique, and the following probabilities:
a) Show the different context tables after coding the sequence “AGAAGT”. Assume max context of 2.

Base	A	C	G	T
Prob.	0.125	0.125	0.25	0.5

a) The sequence “CAGT” is encoded. Show the generated counts and the cumulative counts. Modify the above context tables during coding. Use the exclusion principle.

Base	C		A		G		T	
	Count	Cum Count	Count	Cum Count	Count	Cum Count	Count	Cum Count

BWT:

Using Burrows-Wheeler transform, and the alphabet { A, C, T, N, & Δ }:

a) Code the sequence {ACACAT}. Let row index start at 0. **{4}**

0															
1															
2															
3															
4															
5															

Code :

Find the Move to Front code for the sequence {NNΔΔΔN}. **{2}**

Code:

b) Find the inverse Burrows Wheeler transform of {3, CNACAAA}. Row numbers are given at the left most column. **{4}**

0															
1															
2															
3															
4															
5															
6															

Decoded Sequence:

Associative Coder of Buyanovsky (ACB)

Assume that the underlined part of the sequence "ACGACGACATGCGA" has already been encoded. Encode next 2 phrases using the Associative Coder of Buyanovsky (ACB) {16}

