CSE 490GZ Midterm Solutions

1) P(a) = 1/4, P(b) = 3/4
Encode “abb” using arithmetic coding with scaling

Symbol	L	R	C	Output	
	0	1	0		
a	0	1/4	0	0	scale by 2x
	0	1/2	0	0	scale by 2x
	0	1	0		
b	1/4	1	0		
b	7/16	1	0		

The interval [7/16,1) contains 1/2 so we can use the tag = 1/2 = .1000…

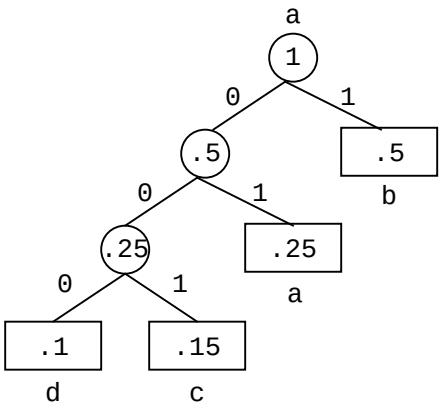
Compressed string: 001

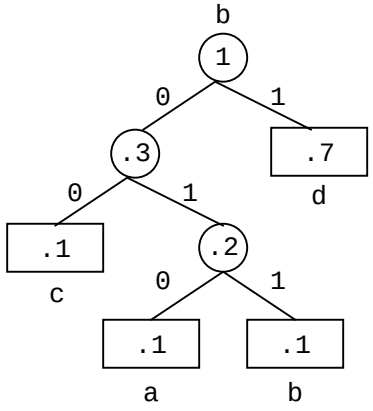
2) Decode the 1110010 using adaptive Golomb coding with doubling.

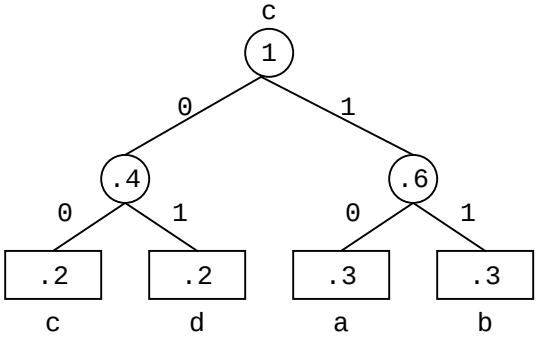
Order	Input	Output	Next Order
1	1	0	2 = 2*1
2	1	00	4 = 2*2
4	1	0000	8 = 2*4
8	0010	001	

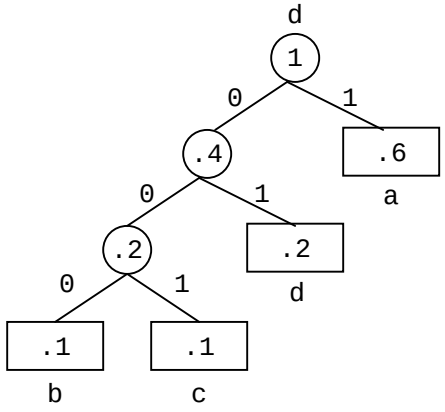
Original string: $0^9 1$

3a) Design a Huffman tree for each context:









3b) Using the Huffman trees from part a and the fixed code of "00" for "a" when there is no context, the encoding of "abcdd" is:

Input	a	b	c	d	d
Output	00	1	00	01	01

Compressed string: 00 1 00 01 01

4a) We need a fixed code for the internal nodes (these are the symbols "a" and "c"). I use the unused code "111" as the escape code and the fixed codes of length 1 for "a" and "c", that is, (a:0, c:1).

Input	aa	ab	b	cb	a
Output	001	010	000	101	1110

Compressed string: 001 010 000 101 1110

4b) P(a) = 1/2, P(b) = 1/4, P(c) = 1/4

Average bit rate = $n / (\Sigma\ p_i r_i)$
= 3 / [(1/4)*1 + (3/4)*2]
= 3 / [7/4]
= 12/7 bits per symbol

5) Decode: 4000400011 using move to front

0	1	2	3	4
a	b	c	d	e

Input: 4000400011
Output: e

0	1	2	3	4
e	a	b	c	d

Input: 4000400011
Output: ee

0	1	2	3	4
e	a	b	c	d

Input: 4000400011
Output: eee

0	1	2	3	4
e	a	b	c	d

Input: 4000400011
Output: eeee

0	1	2	3	4
e	a	b	c	d

Input: 4000400011
Output: eeeed

0	1	2	3	4
d	e	a	b	c

Input: 4000400011
Output: eeeedd

0	1	2	3	4
d	e	a	b	c

Input: 4000400011
Output: eeeeddd

0	1	2	3	4
d	e	a	b	c

Input: 4000400011
Output: eeeedddd

0	1	2	3	4
d	e	a	b	c

Input: 4000400011
Output: eeeeddddd

0	1	2	3	4
e	d	a	b	c

Input: 4000400011
Output: eeeeddddde

6) Encode: aabaabb using LZW with doubling

Dictionary, size = 4

Symbol	Code
a	00
b	01
c	10
	11

Input: aabaabb
Output: 00
Next entry: aa

Dictionary, size = 4

Symbol	Code
a	00
b	01
c	10
aa	11

Input: aabaabb
Output: 00 00
Next entry: ab

Dictionary, size = 8

Symbol	Code
a	000
b	001
c	010
aa	011
ab	100
	101
	110
	111

Input: aabaabb
Output: 00 00 001
Next entry: ba

Dictionary, size = 8

Symbol	Code
a	000
b	001
c	010
aa	011
ab	100
ba	101
	110
	111

Input: aabaabb
Output: 00 00 001 011
Next entry: aab

Dictionary, size = 8

Symbol	Code
a	000
b	001
c	010
aa	011
ab	100
ba	101
aab	110
	111

Input: aabaabb
Output: 00 00 001 011 001
Next entry: bb

Dictionary, size = 8

Symbol	Code
a	000
b	001
c	010
aa	011
ab	100
ba	101
aab	110
bb	111

Input: aabaabb

Output: 00 00 001 011 001 001

7) True or false questions
   a) True – By forming symbols from strings of length k, Huffman coding comes to within 1/k of first-order entropy.
   b) True – Arithmetic coding of string of length k have average code length within 2/k of first-order entropy.
   c) False – A single bit error can destroy most of a Golomb code.
   d) False – The first-order entropy bound applies to algorithms that only use the frequencies of the symbols. Algorithms that use context can do better than the first-order entropy bound.
   e) True – In arithmetic coding there are several alternatives for solving the problem of what code to use when seeing a symbol for the first time.
   f) False – One of our tables showed that Sequitur does indeed compress very well.
   g) False – Grade II Braille is still commonly in use today by blind people.
   h) False – If L is a repeating binary number like .010101… = 1/3 then it cannot be truncated to be a finite length code and still stay in the interval.