THIS PAPER IS NOT TO BE REMOVED FROM THE EXAMINATION HALLS

UNIVERSITY OF LONDON 291 0325 ZB

BSc/Diploma Examination for External Students

Data Compression

Friday 21 May 2010 : 2.30 – 4.45 pm

Duration: 2 hours and 15 minutes

Answer ALL questions.

Full marks will be awarded for complete answers to ALL the THREE questions.

There are 75 marks available on this paper.

Electronic calculators may be used. The make and model should be specified on the script. The calculator must not be programmed prior to the examination. Calculators which display graphics, test or algebraic equations are not allowed.

Question 1

(a) Draw a flow chart to outline the general stages for developing a compression algorithm. [5]

(b) Discuss the absolute limit of lossless compression with the aid of an example. Explain why only a proportion of the files of size n can be compressed by one byte. What is the percentage of the files that can be compressed by one byte? [5]

(c) Consider a source (A, B, C, D) with a probability distribution (0.4, 0.3, 0.2, 0.1). Discuss whether each of the following statements is true or false. Provide supporting arguments or evidence to justify your answer. [8]

i. The binary tree representing a fixed length binary code is not optimal.
ii. The Shannon-Fano code for the given source in the question is optimal.

(d) Explain and demonstrate how the compression efficiency of the Shannon-Fano encoding can be improved by alphabet extension. Use a binary alphabet (A, B) with the probability of A being 0.3 as an example. [7]

Question 2

(a) Consider the alphabet (A, B, C, D) of a source. Discuss the possibility of finding: [5]

i. A uniquely decodable binary code in which the codeword for A is of length 2, that for B of length 1 and for both C and D of length 3.
ii. A shorter variable length prefix code than the one described in (a)i.

Provide evidence or justification for your answers.

(b) Consider a segment of bitmap image that is represented by the array of characters below: [10]

A	D	D	B	B
C	D	C	A	A
C	C	C	B	A
D	D	D	A	A

i. Compute the probability distribution of the characters.
ii. What is the minimum length of the encoded file for the segment? Show all your work and assumptions.
iii. Assume that the whole image is represented in the same distribution as that of the segment, and is stored in a text file. What is the minimum number of bits per character on average?

(c) Explain $x = Q + (S - T)/2$, a predictive rule of JPEG. Demonstrate, with the aid of a small example, how this rule can be applied in preprocessing. Assume the pixel layout as follows: [5]

T	S
Q	x?

(d) Consider a binary code that is not a prefix code. Can we conclude that the code is therefore not uniquely decodable? Explain your answer. Give one example to support your answer. [5]

Question 3

(a) Derive the *reflected Grey code* for the decimal numeral 12. [5]

(b) Draw a flowchart to outline the adaptive Huffman algorithm for encoding. [5]

(c) Draw a diagram to outline the LZ77 decoding algorithm. [10]

Following the approach of the LZ77 algorithm, decode the tokens `(0,ASCII(A))`, `(0,ASCII(A))`, `(0,ASCII(B))`, `(0,ASCII(A))`, `(0,ASCII(C))`, `(0,ASCII(C))`, `(5,2)`, `(6,2)`, `(0,ASCII(A))`, `(8,3)`, `(0,ASCII(C))`. Assume that the length of the history buffer is $H = 8$ and of the lookahead buffer is $L = 6$. The history buffer is empty initially.

(d) Explain what each of the variables L, x, $s1$, $p2$ represents in the segment of the Arithmetic decoding algorithm below. Demonstrate how the algorithm works with the aid of a small example. Assume a source (A, B) with a probability distribution (0.2, 0.8).

Hint: You may trace variable values at the end of each iteration 0–5 for an input 0.43 as suggested in the table below. Use iteration 0 to describe the initial state and add necessary assumptions for a specific source. Finally, derive the decoded text. [5]

```
1. L <- 0 and d <- 1
2. If x is within [L, L+d*p1)
     then output s1, leave L unchanged, and
          set d <- d*p1
     else if x is within [L+d*p1, L+d)
          then output s2, set L <- L+d*p1 and d <- d*p2
3. If (the_number_of_decoded_symbols
            < the_required_number_of_symbols)
     then go to step 2.
```

Iteration	L	d	d*p1	d*p2	[L, L+d*p1)	[L+d*p1, L+d)	Output
0							
1							
2							
3							
4							
5							