UNIVERSITY OF LONDON

GOLDSMITHS COLLEGE

B. Sc. Examination 2002

COMPUTING AND INFORMATION SYSTEMS

IS520010A(CIS325) Data Compression

Duration: 2 hours and 15 minutes

Date and time:

Answer FOUR questions.
Full marks will be awarded for complete answers to FOUR questions.
There are 100 marks available on this paper
Electronic calculators may be used. The make and model should be specified on the script and the calculator must not be programmed prior to the examination.

THIS EXAMINATION PAPER MUST NOT BE REMOVED FROM THE EXAMINATION ROOM

Question 1 (a) Explain the meaning of the following terms: [8]

(i) entropy

(ii) lossless compression

(iii) prefix codes

(iv) basic quantisation

(b) Explain the meaning of the following terms relating to text compression algorithms and provide one example for each case. [8]

(i) fixed-variable model

(ii) adaptive coding

(c) With the aid of an example, describe why the unique decodability is not a problem for any fixed length codeword set. [4]

(d) Explain, with the aid of an example, why the following statement is considered to be true: [5]

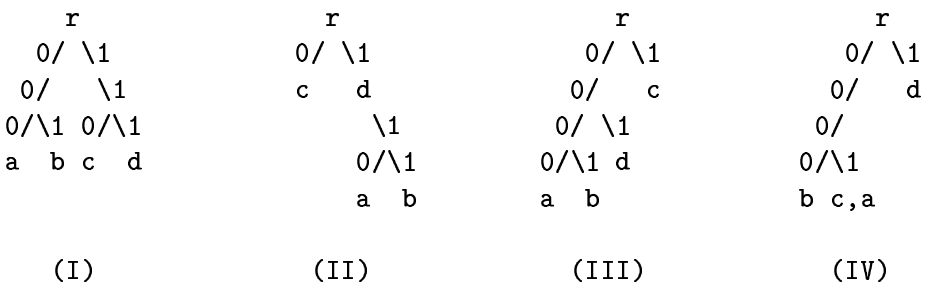
`The Huffman code in general is not optimal unless all probabilities are negative powers of 2.`

Question 2 (a) Explain why the following statements are considered to be true in describing the absolute limits on lossless compression. [9]

- No algorithm can compress all files, even by one byte.
- No algorithm can compress even 1% of all files, even by one byte.

(b) Why is it possible to derive two different Huffman codes for the same probability distribution of an alphabet? Explain briefly what is meant by *Canonical* and *Minimum-variance* Huffman coding. [4]

(c) A binary tree (0-1 tree) can be used to represent a code containing a few codewords of variable length. Consider each of the four binary trees below and decide which binary tree(s) represents a prefix code. Give your reasons. [12]



Question 3 (a) Given an alphabet of four symbols {A,B,C,D}, is it possible to assign a codeword of length 1 to A, 2 to B and 3 to C and D, so that the code is uniquely decodable? Give your reasons. [4]

(b) Describe briefly, with the aid of an example, how Shannon-Fano encoding differs from *static* Huffman coding. [6]

Hints: You may give an example by deriving a code for {A,B,C,D} with probability distribution 0.5, 0.3, 0.1, 0.1 using the two algorithms, and show the difference.

(c) What does the HDC algorithm stand for? What would be the output if the HDC algorithm is applied to the sequence below? Explain the meaning of each control symbol that you use. [7]

`TTU␣␣␣␣␣␣␣␣␣␣K␣␣RR33333333333333␣␣PPPEE`

(d) One way to improve the efficiency of Huffman coding is to maintain two sorted lists during the encoding process. Using this approach, derive a canonical minimum variance Huffman code for the alphabet {A,B,C,D,E,F} with the probabilities (in %) 34,25,13,12,9,7 respectively. [8]

Question 4 (a) If a code is *not* a prefix code, can we conclude that the code is not uniquely decodable? Give reasons, with the aid of an example, to support your argument. [4]

(b) Determine whether the following codes for the alphabet {A, B, C, D} are *uniquely decodable*. Give your reasons for each case. [12]

(i) {1, 10, 101, 0101}

(ii) {000, 001, 010, 111}

(iii) {0, 001, 10, 011}

(iv) {000, 010, 011, 1}

(c) Consider a source with a binary alphabet {A,B}. Under what condition on the source does the static Huffman coding algorithm achieve the best performance? Under what condition on the source does the algorithm do the worst? Give your reasons. [9]

Question 5 (a) Huffman coding can perform badly when it is applied to fax images. For example, the canonical minimum-variance Huffman code is about 37% worse than its optimal when the probabilities of the white pixels $Pr(White) = 0.2$ and $Pr(Black) = 0.8$. Demonstrate step by step how this situation can be improved by grouping two symbols at a time. [12]

Hint: $\log_{10} 2 \approx 0.3$; $\log_{10} 0.8 \approx -0.1$; $\log_{10} 0.2 \approx -0.7$.

(b) What are the two main disadvantages of Huffman coding? How does Arithmetic coding solve these problems? [13]

Question 6 (a) A sequence of four symbols from alphabet {A,B} was encoded using Arithmetic coding. Suppose the probability distribution is $Pr(A) = 1/5$ and $Pr(B) = 4/5$. If the encoded output is 0.24, derive the decoded output step by step. [10]

The decoding algorithm is given below.

```
1. (Initialise) L <- 0 and d <- 1
2. (Probabilities) For the next symbol,
     Let Pr[s1] be p1, the probability for s1
     and Pr[s2] be p2, the probability for s2
3. (Update) If x is a member of [L,L+d*p1)
     then output s1, leave L unchanged, and
          set d<-d*p1
     else if x is a member of [L+d*p1, L+d)
          then output s2, set L<- L+d*p1 and d<-d*p2
4. (Done?) If the_number_of_decoded_symbols
              < the_required_number_of_symbols
     then go to step 2.
```

(b) Demonstrate step by step how the Basic LZW *encoding* and *decoding* algorithms maintain the same version of a dictionary without ever transmitting it in a separate file, using a small string `AABABC` as an example. [15]

The LZW encoding and decoding algorithms are given below.

(i) Encoding

```
word='';
while not end of file
  x=read next character;
  if word+x is in the dictionary
    word=word+x;
  else
    send the dictionary number for word (to output)
    add word+x to the dictionary
    word=x
end of while loop
output the number for word
```

(ii) Decoding

```
read a character x from compressed file;
write x to uncompressed version;
word=x's uncompressed version;
while not end of the compressed file do
  begin
    read x
    look up dictionary element corresponding to x;
    output element
    add word+first char of element to the dictionary
    word=element
  endwhile
```