Programming Assignment

Due: September 25, 2003.

Option 1 [100 points]:

1. (a) For a binary source with probabilities p(0)=0.9, p(1)=0.1, design a Huffman code for the source obtained by blocking *m* bits together, *m=1,2,…,8*. Plot the average lengths versus *m*. Comment on your result. Repeat for p(0)=0.99, p(1)=0.01.
2. Generate a binary sequence of length *L* with p(0)=0.8, and use the arithmetic coding algorithm to encode it. Plot the difference of the rate in bits/symbol and the entropy as a function of *L*. Comment on the effect of *L* on the rate.

Option 2 [100 points, 20 points extra credit] (more challenging):

1. Consider a specific type of source that generates English characters with a high probability of groups. For example: aaaaddddddddcccqqqqhwwww. The number of repetitions may be up to a few thousand. Using Order-0 Huffman or Arithmetic coding directly may not have optimal compression ratio. A typical methodology in data compression is to preprocess/transform the text into another form to explore the redundancy of the source so that Huffman or Arithmetic coding may compress the text better. For example, using Move-to-Front algorithm [1] to convert the above text into(assuming the alphabet has 26 letters):

   10004000000400(17)0008(23)0000

   The resulting string has a more skewed probability distribution so it could have a better compression. But it is not always the case. What is the problem when probability of some symbol such as '0' is greater than 0.5? Find at least two methods to solve this problem.
2. Read the technical report "Improving Huffman Coding" and understand it. Illustrate the problem with examples in the section "Use of Multiple Tables" and explain clearly with some worked out examples how the problem has been solved using multiple Huffman tables for adaptation during encoding and decoding. Download the paper and the code and modify the code as appropriate to illustrate your examples. http://www.cs.ucf.edu/~nzhang/CAP5015/AS2

Deliverables:

1. Report, including method description, algorithm, performance analysis.
2. Source code (c/c++), should be compiled with and Microsoft/Unix c++ compiler.
3. Test data used (file or hyperlink).
4. Pack everything into a zip file and email to nzhang@cs.ucf.edu .

[1] http://www.arturocampos.com/ac_mtf.html