Assignment 1 COMP 423 Prof. M. Langer

- Posted Mon. January 21, 2008. Due **IN CLASS** on Mon. Feb. 4, 2008.
- Assignment will be marked out of 25.
- Late penalty is 2 points (out of 25) per day.
- Include a printout of source code for any programs you write. BE NEAT! Points will be taken off for excessive sloppiness.

Introduction

In this assignment, you will investigate the effectiveness of specific compression techniques for a sequence of independent random floating point numbers. Floating point numbers are represented using the IEEE (754) floating point standard. Most of you will have learned about IEEE floating point numbers either in COMP 273 or in COMP 350. If you have not learned about them, then now is the time.[1] In particular, we consider *single precision* (32 bit).

We strongly suggest that you use C to solve this assignment. C has the advantage that it allows bit string manipulations. Examples of C code to get you started are included with the assignment.

Part 1

You need to generate a sequence of floating point numbers between 0 and 1,

$$\{f_0, f_1, \ldots, f_{n-1}\}$$

where $n = 100,000$. We will consider various schemes for compressing this sequence.

1. **(3 marks)** For each value f_i, compute the exponent e_i. Estimate the entropy of these exponents based on the relative frequencies of the different exponents. Although the exponent is eight bits and the floats are randomly generated, you should find that the entropy is less than eight bits. Explain why. Your explanation should relate to how the IEEE defines the exponent component and to the fact that the numbers are uniformly distributed over the range $[0, 1]$.

2. **(5 marks)** Let the j^{th} bit of f_i be denoted as f_i^j. Let b_j denote the bitstring $f_o^j ... f_{n-1}^j$. For example, the bitstring b_{31} is a bitstring of length n comprised of all the sign bits of the original floating point numbers.

   For each of the 32 bitstrings b_j, compute the relative frequency (the percentage) of the bits that are 1. Explain this result for each of:

   - the sign bit,
   - each of the 8 exponent bits,
   - each of the 23 mantissa bits.

3. **(2 marks)** One way to compress the set of n floating point numbers would be to compress each of the 32 bitstrings b_j individually. Using the relative frequencies computed in question 2, estimate the entropy of each of the 32 bits.

[1] See en.wikipedia.org/wiki/IEEE_floating_point or see Prof. Langer's lecture notes for COMP 273 (lecture 2).

Part 2

Consider these bitstrings b_j to be Bernoulli trials. The following experiments should be done for the bitstrings b_0, b_{24}, b_{25}, where b_0 is the least significant bit of the mantissa, and b_{24}, b_{25} are two of the bits of the exponent. (The exponent bits are $b_{23}, \ldots, b_{30}$.)

In this part of the assignment, you apply run-length coding to these bitstrings. Here you will consider runs of 1's, followed by a 0. For example, 11110 is a run of length 5. Note this is the opposite of what was seen in class, where we looked at runs of 0's followed by a 1.

1. **(3 marks)** Count the relative frequencies of runs of length i. Compare these frequencies to the values predicted by the formula for $p(i)$ we saw in class, based on p_1 which is the relative frequency of 1's in the bitstring (as calculated in Part 1).

2. **(2 marks)** Using these relative frequencies of run lengths as an estimate of the probability of runlengths, estimate the entropy of run length. (Assign probability of zero to any run length that does not occur e.g. a run length that is longer than the longest run in that sequence.)

3. **(2 marks)** Verify that the average run length follows the formula derived in class (namely $\frac{1}{1-p_1}$, since we are considering runs of 1's followed by a 0).

4. **(4 marks)** If we were to code the run lengths with a Golomb code, what parameter b would yield the shortest average code length? Verify any theoretical prediction you make with an experiment. Note that b does not need to be a power of 2. For each b that you consider, what is the *compression ratio*, namely the ratio of the number of bits in the encoded sequence to the number of bits in the original string ?

5. **(4 marks)** If we were to encode the run lengths with an Elias code, what would be the average code length? (Consider both Elias1 and Elias2.) What would be the compression ratio?