Problem Set #10

1. (5 points) **Huffman codes and entropy:**

   (a) Construct a Huffman code for the probability distribution

   $$p = (.3,\ .14,\ .13,\ .12,\ .1,\ .06,\ .05,\ .05,\ .01,\ .01,\ .01,\ .01,\ .01).$$

   What is the average length of this code? What is the entropy of the distribution p?

   (b) Find a probability distribution that, when used with the code constructed above, has average length equal to its entropy.

2. (3 points) **Bad Codes:** (from Cover-Thomas) Which of these codes cannot be Huffman codes for any probability assignment and why?

   (a) $\{0, 10, 11\}$
   (b) $\{00, 01, 10, 110\}$
   (c) $\{01, 10\}$

3. (3 points) **Shannon codes and Huffman codes:** (from Cover-Thomas) Consider a random variable X that takes on four values with probabilities $(1/3,\ 1/3,\ 1/4,\ 1/12)$.

   (a) Construct a Huffman code for this random variable.
   (b) Show that there exist two different sets of optimal lengths for the codewords; namely, show that codeword length assignments $(1, 2, 3, 3)$ and $(2, 2, 2, 2)$ are both optimal.
   (c) Conclude that there are optimal codes with codeword lengths for some symbols that exceed the Shannon code length. (What are the Shannon code lengths for this distribution?)

4. (4 points) **Entropy of English:**

   (a) Find a table online of letter frequencies for the English language (there is a Wikipedia page called "letter frequency") and calculate the entropy. You can ignore space and punctuation, even though this does have a small effect on the entropy.
   (b) If the 26 letters appear with equal probability, recalculate the entropy.
   (c) The true *entropy rate* of English, which gives the fundamental lower bound for compression (allowing for compression of blocks of letters), is estimated to be about 1 bit per letter. Can you explain the difference between this and your answer to part (a)?
   (d) Find an English text from the web (no less than 1000 words—specify your source of text) and count the total number of letters, n. Then compress the .txt. file into .zip or .rar (specify your compression software), and check the number of bits, m, for this file. What is the ratio m/n? Is this compression software performing optimally? (note: These compression algorithms do not benefit from knowing that the file is English text. They are universal and simply look for patterns in the data to exploit for compression. Their performance improves as the size of the data increases.)