CSCI 3104 Summer 2017 - Assignment # 4

Due: June 30th 2:30pm

Be sure to justify all of your work. Submit all of your code to the moodle submission link.

1. Recall that Huffman's algorithm takes as input a vector f of length n, containing the frequencies of the input symbol set C and returns a Huffman encoding tree. We can make the process of encoding real strings faster by instead returning a *codebook*, which is a dictionary ADT T that contains n key-value pairs (x_i, y_i), where $x_i \in C$ and y_i is the Huffman binary encoding of x_i. Let $T(x_i) = y_i$. Hence, the encoding of $x = x_1x_2x_3 \ldots x_l$ is simply $y = T(x_1)T(x_2)T(x_3) \ldots T(x_l)$.
   In this problem, you will implement and apply three functions: (i) `strToFreq`, (ii) `huffmanEncode`, and (iii) `encodeStr`.

   a) Implement the three functions, i.e. `strToFreq`, `huffmanEncode`, `encodeStr`, **from scratch**. For Huffman's algorithm, use a min-heap as a priority queue. Within this priority queue, ties should be broken uniformly at random.

   b) What is the running time of the call
   `y = encodeStr(x, huffmanEncode(strToFreq(x)))`?
   Justify your answer using asymptotic analysis.

   c) Pick your favorite quote, poem, or piece of text that has length $l > 500$ characters (not too long either, for simplicity). Given that *ASCII* characters are usually represented using 8 bits, how many bits are required to represent the entire text?

   d) Claude Shannon famously proved that the lower bound on the number of encoded bits per input symbol for a prefix-free binary encoding of a string x is given by the entropy function, defined as

   $$H = -\sum_{i=1}^{|C|} \left(\frac{f_i}{l}\right) \log_2 \left(\frac{f_i}{l}\right)$$

   where f_i is the frequency in x of the ith symbol in C, and l is the length of x. Because we take $\log_2$, H has units of *bits*. Using this equation, compute (i) the predicted number of bits per symbol needed to optimally encode the text you chose in part (c), and (ii) the predicted number of bits needed to encode the entire text you chose.

   e) Now use your Huffman encoder from part (a) to encode your text and report the number of bits in the encoded string. Compare this number to the lower bound obtained in part (d) and comment on the comparison.

2. Here, you will modify and then use your Huffman encoder from part (1) to perform a numerical experiment to show that your encoder uses $\mathcal{O}(n \log n)$ time on an input of size n. The deliverables here are (i) a nice figure showing this result and (ii) a brief description

of how you modified your code and ran your experiment.
First, implement a new function `makeHuffmanInput`, which takes an argument n, the size of the symbol set C, and returns a vector of counts f of length n, in which each f_i is a positive integer chosen uniformly at random from the interval $[1, 100]$. Second, modify your `huffmanEncode` function by adding measurement code that counts the number of atomic operations it performs before returning.