8 Algorithms 1 (fms27)

(a) Prove that, given any alphabet $\Sigma$, no lossless compression function $c : \Sigma^* \rightarrow \Sigma^*$ has the property that
\[
\forall s \in \Sigma^* : |c(s)| < |s|
\]
where the vertical bars around a string denote its length in symbols. [3 marks]

(b) Build the Huffman code for the following set of symbols and frequencies, drawing the corresponding tree. When combining two trees, put the tree with the smallest aggregate frequency on the 0 branch. List clearly, in order, the seven merge operations you performed. Finally, list the codeword for each symbol.

\[
\{a : 0.15; b : 0.25; c : 0; d : 0.1; e : 0.12; f : 0.18; g : 0; h : 0.2\}
\]

[8 marks]

(c) Consider a string over the following set of symbols and frequencies, encoded with one byte per symbol using the ASCII codes for the lowercase letters.

\[
\{a : 0.1; b : 0.1; c : 0.2; d : 0.2; e : 0.1; f : 0.3\}
\]

Imagine re-encoding it with the Huffman code shown below.

\[
\begin{align*}
a & : 010; \quad b : 011; \quad c : 01; \quad d : 100; \quad e : 101; \quad f : 11
\end{align*}
\]

What compression factor would you achieve? How much of this compression can be attributed specifically to Huffman coding? [3 marks]

(d) Consider strings over a set of 16 symbols. A trivial encoding, with no attempt at compression, will use codewords of 4 bits per symbol, without taking symbol frequencies into account. Prove that, when each symbol has a different frequency but the ratio between the largest and smallest frequency is strictly less than 2, Huffman coding of such strings cannot provide any compression compared to the trivial non-compressing encoding. [6 marks]