**CMPS340 File Processing**
**Sample Problems on Coding and Huffman Compression**

**1.** The set of strings $L = \{aa, ba, abb, bbab, babb\}$ is *not* uniquely decipherable (u.d.). (Recall from class the meaning of that term.) Demonstrate this by showing two factorizations (with respect to $L$) of any string of your choosing. To receive full credit, the string you choose must be a shortest string admitting two factorizations.

Also, identify a smallest subset $L'$ of $L$ that is not u.d. ("Smallest" in this context means "having fewest members".) Justify your answer by showing that the subset $L'$ that you identified is not u.d. but that every proper subset of $L'$ is u.d. Keep in mind that any set of strings possessing either the *prefix freeness* property or the *suffix freeness* property is necessarily u.d. (A set of strings possesses the prefix (respectively, suffix) freeness property if none of its members is a proper prefix (respectively, suffix) of any other member.)

**Solution:** Two factorizations of *babbabbabb* are

$$ba \cdot bbab \cdot babb \quad \text{and} \quad babb \cdot abb \cdot abb$$

The smallest non-u.d. subset of $L$ is

$$L' = L - \{aa\} = \{ba, abb, bbab, babb\}$$

To verify this claim, we show that (a) $L'$ is not u.d., and (b) no subset of $L$ having three or fewer members is u.d.

Regarding (a), as all factors in the two factorizations of *babbabbabb* shown above are members of $L'$, it follows that $L'$ is not u.d.

Regarding (b), recall that any set of strings possessing either of the prefix-free or suffix-free properties is u.d. Hence, the subset of $L$ that we seek possesses neither of these properties. But in order for a subset of $L$ to be neither prefix-free nor suffix-free requires that all three of the strings *ba*, *abb*, and *babb* be members. (Why? Because the only violation of prefix-freeness (respectively, suffix-freeness) by $L$ arises as a result of the strings *ba* and *babb* (respectively, *abb* and *babb*) being members.) Hence, every subset of $L$ having three or fewer members is u.d., except possibly for $M = \{ba, abb, babb\}$ (i.e., the set consisting of the three strings just mentioned).

It remains only to show that $M$ is u.d. If it were not u.d., we could find an "ambiguous" string (i.e., one having two different factorizations with respect to $M$). A shortest such string must have one factorization that begins with *ba* and another that begins with *babb*. (Recall that these are the only two members of $M$ such that one is a proper prefix of the other.) But then the first factorization must continue with *bb* in order to "catch up" to the second factorization. As there is no member of $M$ that begins with *bb*, this is impossible. Hence, there exists no ambiguous string with respect to $M$, which is to say that $M$ is u.d.

**2.** Suppose that we have a file (i.e., a long bit string) that is to be compressed. Suppose, further, that we decide to view the file as being composed of 3-bit blocks. (For convenience, we assume that the length of the file is evenly divisible by three.) We interpret each block as a single character. (Under this scenario, there are eight characters in the source *alphabet*, one for each of the distinct bit strings of length three.) The frequency with which each of the characters occurs in the file is given in the following table. (For convenience, we refer to the eight characters as $a, b, \ldots, h$, as indicated in the table.)

| Character | Frequency | Character | Frequency |
|---|---|---|---|
| a (000) | 11.9% | e (100) | 6.7% |
| b (001) | 9.6% | f (101) | 7.7% |
| c (010) | 24.7% | g (110) | 9.2% |
| d (011) | 28.3% | h (111) | 1.9% |

**(a)** Construct a Huffman tree corresponding to these frequencies.

**Note:** In order to ensure that the correct answer is unique, follow this rule in using the Huffman tree to determine the codeword of each character: From each non-leaf node there are two edges connecting it to its children; one edge is to be labeled 0 and the other 1. Use 0 to label the edge that goes to the subtree having the leaf with the alphabetically smallest label (among all labels in the two subtrees). For example, if a node's two subtrees contain leaves labeled by $c$, $d$, and $f$ and by $a$ and $e$, respectively, then the edge into the latter subtree should be labeled 0, because $a$ is the smallest among all the labels in the two subtrees.

**(b)** (3 pnts) Show the binary encoding of each character (consistent with the tree constructed in (a)), i.e., show the resulting Huffman code.

**(c)** (15 pnts) Compute the ratio between the lengths of the compressed file and the original file. Show your work. Keep in mind that three bits were used to store each character in the original file, whereas the number of bits used, on average, for storing a character in the compressed file is the sum

$$\sum_{x \in \{a,b,\ldots,h\}} len(x) \cdot freq(x)$$

where $len(x)$ is the length (in bits) of the code for $x$ and $freq(x)$ is the frequency with which $x$ occurs in the original file.

**Solution:** As the answer to (a) can be obtained easily from the answer to (b), we show only the latter:

| Character | Codeword | Character | Codeword |
|---|---|---|---|
| a | 000 | e | 00100 |
| b | 100 | f | 0011 |
| c | 11 | g | 101 |
| d | 01 | h | 00101 |

(c)

$$
\begin{aligned}
& \textstyle\sum_{x \in \{a,b,\ldots,h\}} len(x) \cdot freq(x) \\
= \;& (3)(.119) + (3)(.096) + (2)(.247) + (2)(.283) + (5)(.067) + (4)(.077) + (3)(.092) + (5)(.019) \\
= \;& 2(.247 + .283) + 3(.119 + .096 + .092) + 4(.077) + 5(.067 + .019) \\
= \;& 2(.530) + 3(.307) + 4(.077) + 5(.086) \\
= \;& 1.06 + .921 + .308 + .430 \\
= \;& 2.719
\end{aligned}
$$

What this says is that, on the average, the number of bits used in encoding a character in the source (using Huffman coding) is 2.719. As the source uses three bits per character, the ratio is $\frac{2.719}{3}$, which is slightly more than 90%. This is not particularly good compression. It's even worse when you consider that it fails to account for the fact that the compressed version of the file must also contain a description of the mapping from characters to codewords.