Computer Science I Variable Length Coding

CSCI 141 Lab

11/18/2013

1 Problem

On a digital computer, all information is ultimately represented as a sequence of binary digits, or *bits*. Zeroes and ones. The field of lossless data compression involves ways to minimize the number of bits required to represent an input data collection without losing any of the information.

Consider a long DNA chain stored in a file. DNA can be represented as a sequence of four nucleobases: guanine (G), adenine (A), thymine (T), and cytosine (C). The DNA chain is a sequence such as: ACTCGAA... Since there are are four different nucleobases, on a computer we might use two bits to represent each nucleobase as in the table below:

Symbol (Nucleobase)	Codeword (Representation)
A	00
C	01
G	10
T	11

The above is an example of what is known as a *fixed length code*. Every symbol (nucleobase) is encoded using a codeword (representation) of length 2 bits. For example, the sequence ACTCGAA would be encoded with the bits 00011101100000.

The drawback of fixed length codes is that they don't exploit the relative frequencies of the symbols that are being encoded. Imagine that in DNA the symbol A occurs 50% of the time, symbol C occurs 25% of the time, symbol G occurs 15% of the time, and symbol T occurs 10% of the time. A fixed length code would still use 2 bits for every symbol.

We could do better, however. Consider the following encoding of the DNA symbols:

Symbol (Nucleobase)	Codeword (Representation)
A	1
C	01
G	001
T	000

This code is referred to as a *variable length code*. Different symbols have different length codewords. The basic idea behind variable length coding is that if we assign short codewords to symbols that occur more frequently, and longer codewords to symbols that occur less frequently, we can arrive at an overall encoding that uses fewer expected bits than if we just used fixed length coding. We have to be careful to choose codewords that allow a unique decoding (there can never be uncertainty as to which symbol occurred - we have to be sure where each symbol's code ends), but that will not be a concern for this lab.

The example DNA chain above would be encoded as 1010000100111, using 13 bits instead of 14. This particular input works out to an average of $13/7 = 1.86$ bits per symbol, as opposed to $14/7 = 2$ bits per symbol for the fixed length code.

You likely are using variable length coding every day without even knowing it. When you use a tool such as WinZip to generate a `.zip` file, variable length coding is used to shrink your files. Every time you take a picture with your iPhone, variable length coding is used to generate your `.jpg` file.

In this lab, you will write a program that reads an input file full of symbols, and computes a variable length code to represent the symbols.

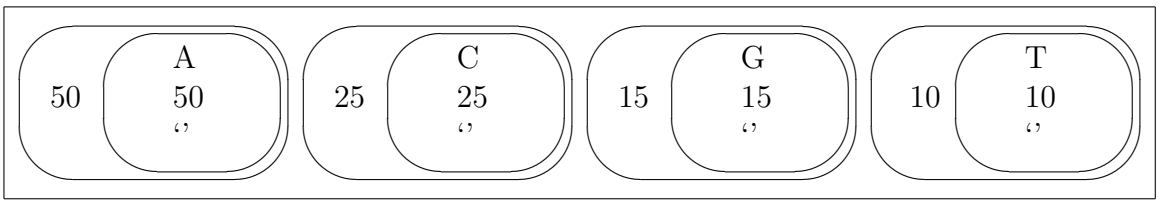
1.1 A Variable Length Coding Algorithm

The variable length coding algorithm we'll be interested in for this lab is given by the following pseudocode:

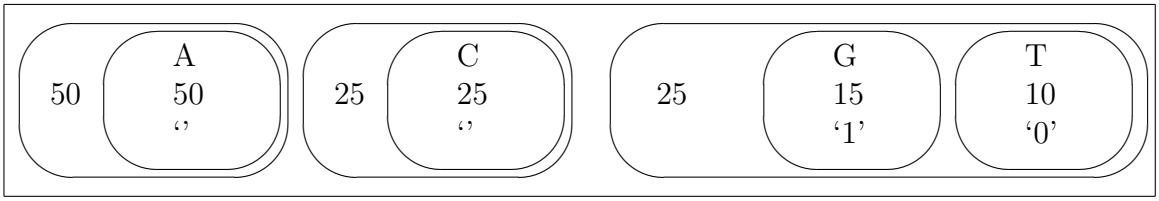
```
For each symbol, create a symbol object that contains information
  for that symbol, in particular its name, frequency and codeword
  (which is initialized to "")

Create a collection of nodes.
A node contains two pieces of information:
    1)  a cumulative frequency
    2)  a list of symbol objects
Initially, one node is created for each symbol object, and
  initialized with that symbol's frequency and a list
  containing just that symbol object.

While there remains more than one node in the collection
    Identify the node, n1, with the lowest cumulative frequency
    Identify the node, n2, with the second lowest cumulative frequency

    Remove those nodes from the collection

    For each symbol object in the symbol object list of n1
        prepend a "0" to its existing codeword
    For each symbol object in the symbol object list of n2
        prepend a "1" to its existing codeword

    Create a new node with cumulative frequency equal to the
      combined cumulative frequencies of n1 and n2,
      and with symbol object list equal to the concatenation
      of the symbol object lists from n1 and n2,
      and add this node back into the collection

At this point, the codeword for each symbol can be read
  directly from each symbol object
```

1.2 Example

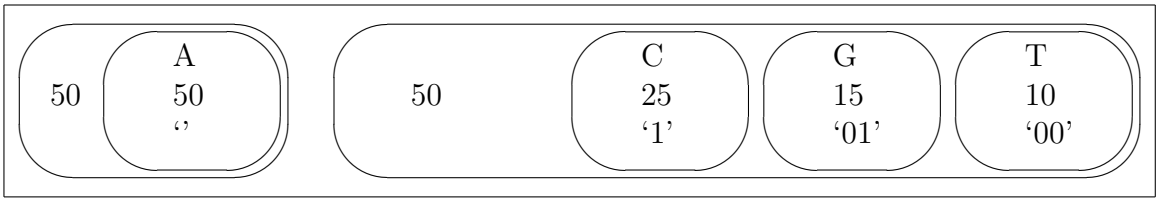
The DNA example above is illustrated graphically below. Initially, four symbol objects are created. The symbol objects contain the symbol name (often a single character), the symbol frequency (often an integer, but could also be a , and the symbol codeword (initially an empty string). One node is created for each symbol object. Each node contains a cumulative frequency (initially just the frequency of the sole symbol object contained in the node), and a list of symbol objects (initially just a single symbol object).



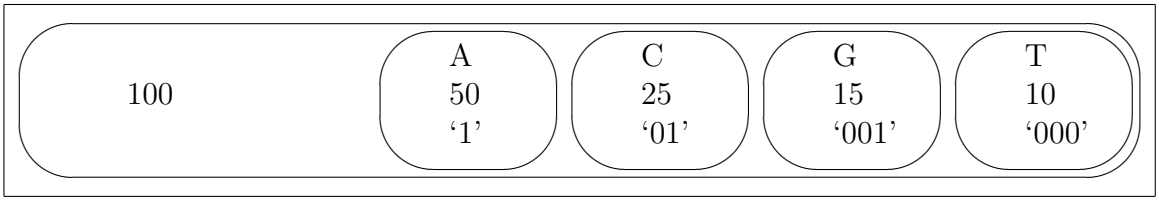
After one iteration, the two least frequently occurring nodes have been combined and an aggregate node replaces it. Each symbol object in the least frequently occurring node has a ‘0’ prepended to its codeword. Each symbol object in the second least frequently occurring node has a ‘1’ prepended to its codeword.



Again the two least (cumulative) frequently occurring nodes are combined. Note that when selecting nodes, ties in cumulative frequency can be broken arbitrarily. In this case, the node containing symbol objects for both G and T is chosen as the least frequently occurring node, so each codeword has a ‘0’ prepended to it. The node containing the symbol object for C is chosen as the second least frequently occurring node, so a ‘1’ is prepended to the codeword for C.



Finally, the last two remaining nodes are combined, and the codewords for each symbol object are complete.



1.3 Pre-Lab (5%)

5% of your grade for this lab is based on reading the first 3 pages of the lab write-up, completing the example variable length coding exercise below, and bringing your solution to your lab session.

Please use the provided sheet (linked with this week's lecture materials, and also handed out in lecture) to show your work. Make sure your name is on your solution. You must work individually and bring your individual solution to the lab session.

Variable Length Coding Exercise: We are interested in designing a variable length code for English text. We know that the letter 'E' occurs much more frequently than the letter 'Z', and want to design an encoding that exploits the relative frequencies of the letters.

To keep things simple, for this exercise we will consider a reduced alphabet containing only the letters: A,E,I,O,U,Y.

For the frequencies listed in the table below, generate codewords for the symbols using the algorithm described in this lab. Show all of the individual steps involved in building up the codewords bit by bit. Your final step will produce the complete codeword for each symbol.

Symbol (Letter)	Frequency
A	82
E	127
I	70
O	75
U	27
Y	20