EE3980 Algorithms

Homework 9. Encoding ASCII Texts

Due: May 17, 2020

The American Standard Code for Information Interchange (ASCII) is a standard encoding method for English letters and symbols. It is a fixed length encoding scheme, which means each letter takes a fixed number of bits to represent the letter. Since we know that some of the letters are not used as often as others, this fixed length encoding scheme may not be the most efficient in information storage or exchange.

To increase the storage efficiency, the variable length encoding scheme can be adopted. In this scheme, a less frequently used letter can be encoded with more bits, while more frequently used letters with fewer number of bits. In the end, the total number of bits for information storage should be minimized.

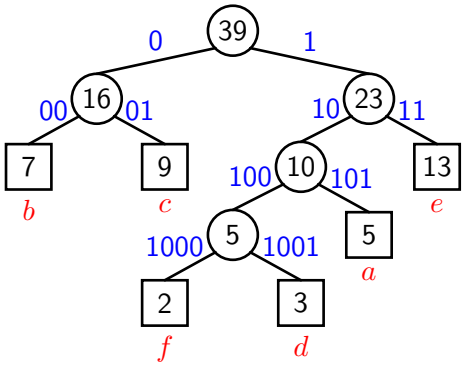
Let $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$ be the set of letters used, and $\mathcal{F} = \{f_1, f_2, \ldots, f_n\}$ be the corresponding frequency of the letters. In ASCII encoding, each letter takes 8 bits. Thus the total number of bits to store the information is

$$S_{ASCII} = \sum_{i=1}^{n} 8 \times f_i = 8 \sum_{i=1}^{n} f_i. \tag{9.1}$$

In a variable length encoding scheme, we assume the number of bits for each letters are $\mathcal{L} = \{\ell_1, \ell_2, \ldots, \ell_n\}$, then the total number of bits is

$$S_{VL} = \sum_{i=1}^{n} \ell_i \times f_i. \tag{9.2}$$

The Huffman encoding scheme has been shown to achieve minimum storage requirement for any given set of letters, $\mathcal{A}$, and the corresponding frequencies, $\mathcal{F}$. As an example, suppose 6 letters, $\mathcal{A} = \{a, b, c, d, e, f\}$ are used to store some information and the corresponding frequencies are also known, $\mathcal{F} = \{5, 7, 9, 3, 13, 2\}$. Then the Binary Merge Tree algorithm (5.3.4) constructs the following binary tree such that $\sum_{i=1}^{n} d_i f_i$ is minimum, where d_i is the depth of the leave node and f_i is the frequency.



From the tree, we get the following Huffman encoding table.

letter	code
a	101
b	00
c	01
d	1001
e	11
f	1000

The goal of this homework is to construct the Huffman code given a text file and to show the ratio between the Huffman encoded storage and the standard ASCII storage. Five text files are given as test cases. The output of your program taking `talk1.txt` as input is shown at the end of this file.

Notes.

1. One executable and error-free `C` source file should be turned in. This source file should be named as `hw09.c`.
2. A `pdf` file is also needed. This report file should be named as `hw09a.pdf`.
3. Submit your `hw09.c` and `hw09a.pdf` on EE workstations using the following command:

   `$ ~ee3980/bin/submit hw09 hw09.c hw09a.pdf`

   where `hw09` indicates homework 9.
4. Your report should be clearly written such that I can understand it. The writing, including English grammar, is part of the grading criteria.

Example program output:

```
$ ./a.out < talk1.txt
Huffman coding:
  ' ': 000
  s: 00100
  y: 001010
  I: 0010110
  v: 0010111
  t: 0011
  w: 010000
  c: 010001
  h: 01001
  C: 01010000000
  4: 01010000001000
  U: 01010000001001
```

8: 01010000001010
$: 01010000001011
9: 0101000000110
K: 0101000000111
?: 010100000100
G: 0101000001010
;: 0101000001011
j: 01010000011
0: 0101000010
x: 0101000011
5: 010100010000
F: 010100010001
2: 010100010010
X: 010100010011
W: 0101000101
H: 01010001100
E: 01010001101
": 0101000111
,: 0101001
m: 010101
d: 01011
'\n': 011000
f: 011001
l: 01101
o: 0111
a: 1000
g: 100100
p: 100101
S: 100110000
-: 1001100010
3: 10011000110
L: 10011000111
B: 1001100100
M: 1001100101
T: 100110011
': 10011010
P: 10011011000
O: 10011011001
Y: 10011011010
R: 10011011011
7: 10011011100
q: 10011011101
D: 1001101111
.: 100111

```
  n: 1010
  u: 10110
  1: 1011100000
  :: 1011100001
  N: 1011100010
  z: 10111000110
  J: 101110001110
  6: 101110001111
  A: 10111001
  k: 1011101
  b: 101111
  e: 110
  i: 1110
  r: 1111
Number of Chars read: 11949
  Huffman Coding needs 53830 bits, 6729 bytes
  Ratio = 56.3143 %
```