Bitwise Representations & Compression & Huffman Coding

This will be handy for the Huffman assignment...
How are primitives stored?

m   a   c

“Primitive” includes “pointer” here.
How are primitives stored?

m    a    c
109   97   99

“Primitive” includes “pointer” here.
How are primitives stored?

<table>
<thead>
<tr>
<th>m</th>
<th>a</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>109</td>
<td>97</td>
<td>99</td>
</tr>
</tbody>
</table>

01101101 01100001 01100011

One binary digit ("bit")
How are primitives stored?

011011010110000101100011

“Primitive” includes “pointer” here.
How are primitives stored?

You have to know your encoding.

"Primitive” includes “pointer” here.
Text data (ASCII)

8 bits per character (one byte).

K for kilo
B for byte
≈ 508,928 characters
American Standard Code for Information Interchange, I’ve just learned.

8 bits per character (one byte).

K for kilo
B for byte
≈ 508,928 characters

“But how?”
(Lossless) data compression

8 bits per character (one byte).

<table>
<thead>
<tr>
<th>Character</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>29380</td>
</tr>
<tr>
<td>b</td>
<td>5672</td>
</tr>
<tr>
<td>c</td>
<td>9571</td>
</tr>
<tr>
<td>d</td>
<td>15972</td>
</tr>
<tr>
<td>e</td>
<td>50853</td>
</tr>
<tr>
<td>f</td>
<td>9317</td>
</tr>
<tr>
<td>g</td>
<td>7091</td>
</tr>
<tr>
<td>h</td>
<td>26624</td>
</tr>
<tr>
<td>i</td>
<td>26445</td>
</tr>
<tr>
<td>j</td>
<td>282</td>
</tr>
<tr>
<td>k</td>
<td>2351</td>
</tr>
<tr>
<td>l</td>
<td>16336</td>
</tr>
<tr>
<td>m</td>
<td>10312</td>
</tr>
<tr>
<td>n</td>
<td>25435</td>
</tr>
<tr>
<td>o</td>
<td>27513</td>
</tr>
<tr>
<td>p</td>
<td>6840</td>
</tr>
<tr>
<td>q</td>
<td>337</td>
</tr>
<tr>
<td>r</td>
<td>24075</td>
</tr>
<tr>
<td>s</td>
<td>24265</td>
</tr>
<tr>
<td>t</td>
<td>36509</td>
</tr>
<tr>
<td>u</td>
<td>9861</td>
</tr>
<tr>
<td>v</td>
<td>3680</td>
</tr>
<tr>
<td>w</td>
<td>8839</td>
</tr>
<tr>
<td>x</td>
<td>461</td>
</tr>
<tr>
<td>y</td>
<td>6534</td>
</tr>
<tr>
<td>z</td>
<td>152</td>
</tr>
</tbody>
</table>

From hawthorne.txt (from Markov)
(Lossless) data compression

8 bits per character (one byte).

a : 29380   j : 282   s : 24265
b : 5672     k : 2351   t : 36509
c : 9571     l : 16336   u : 9861
d : 15972    m : 10312   v : 3680
e : 50853    n : 25435   w : 8839
f : 9317     o : 27513   x : 461
g : 7091     p : 6840    y : 6534
h : 26624    q : 337     z : 152
i : 26445    r : 24075

What if we use variable-length encoding?

http://goo.gl/8FQbY
Encoding Design

AACCCAAABABAADAE

a : 8
c : 3
b : 1
d : 1
e : 1
Encoding Design

AACCCAAABAADAE

a : 8
c : 3
b : 1
d : 1
e : 1

http://goo.gl/oPVkR
Encoding Design

AACCCAAABAADAE

a : 8
c : 3
b : 1
d : 1
e : 1

http://goo.gl/oPVkR
http://goo.gl/vBffI
Prefix Codes

AACCCAAABAADAE

a : 8
c : 3
b : 1
d : 1
e : 1

No encoding can be a prefix of any other: if ‘a’ is 01, nothing else can start with 01.

A valid (but inefficient!) prefix code for this string:

a : 01
c : 100
b : 101
d : 110
e : 111

(Gets us to 34 bits)

010110010010001010110101011100111
Prefix Codes

AACCCAAAABAADAAD

a : 8
c : 3
b : 1
d : 1
e : 1

No encoding can be a prefix of any other: if ‘a’ is 01, nothing else can start with 01.

A valid (but inefficient!) prefix code for this string:

\[ \text{a : 01, c : 100, b : 101, d : 110, e : 111} \]

\( \text{http://goo.gl/KBk9j} \)

(Gets us to 34 bits)
Sometimes, it seems like everything is a tree. That's because it's pretty much true.
**Huffman Coding**

she sells sea shells by the sea shore

- Generate your **frequencies**.
- Make $k$ trees, one for each character.
- while (more than one tree)
  - remove the two smallest trees *cough cough priority queue*
  - merge them under a new root
- add the new tree

The per-character encoding is the path from root to leaf.

You use the tree to build the encoding, and then to decode data. For encoding, just make a map.
What if it isn’t text?
Optimality

*Optimal* for per-character encoding schemes.

In practice...trickier.
File format

File starts
Magic Number
Frequency data (or huff tree!)
Compressed data
Pseudo-EOF
File ends