Huffman Coding

How do we represent numbers?

Binary code: \( \{9, \ldots, 33\} \rightarrow \{0000, \ldots, 1001\} \)

4 bits

Letter: \( \{a \ldots z\} \rightarrow 5 \) bits

26 letters 2 things

\( \{a \ldots z, A \ldots Z\} \rightarrow 6 \) bits

International Morse Code

<table>
<thead>
<tr>
<th>Letter</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>12.02</td>
</tr>
<tr>
<td>T</td>
<td>9.10</td>
</tr>
<tr>
<td>A</td>
<td>8.12</td>
</tr>
<tr>
<td>O</td>
<td>7.66</td>
</tr>
<tr>
<td>I</td>
<td>7.31</td>
</tr>
<tr>
<td>N</td>
<td>6.95</td>
</tr>
<tr>
<td>S</td>
<td>6.26</td>
</tr>
<tr>
<td>R</td>
<td>6.02</td>
</tr>
<tr>
<td>H</td>
<td>5.92</td>
</tr>
<tr>
<td>D</td>
<td>4.32</td>
</tr>
<tr>
<td>L</td>
<td>3.98</td>
</tr>
<tr>
<td>U</td>
<td>2.88</td>
</tr>
<tr>
<td>C</td>
<td>2.71</td>
</tr>
<tr>
<td>M</td>
<td>2.61</td>
</tr>
<tr>
<td>F</td>
<td>2.30</td>
</tr>
<tr>
<td>Y</td>
<td>2.11</td>
</tr>
<tr>
<td>W</td>
<td>2.09</td>
</tr>
<tr>
<td>G</td>
<td>2.03</td>
</tr>
<tr>
<td>P</td>
<td>1.82</td>
</tr>
<tr>
<td>B</td>
<td>1.43</td>
</tr>
<tr>
<td>V</td>
<td>1.11</td>
</tr>
<tr>
<td>K</td>
<td>0.68</td>
</tr>
<tr>
<td>X</td>
<td>0.17</td>
</tr>
<tr>
<td>Q</td>
<td>0.11</td>
</tr>
<tr>
<td>J</td>
<td>0.10</td>
</tr>
<tr>
<td>Z</td>
<td>0.07</td>
</tr>
</tbody>
</table>

1 2 1 3 2 1 3 2 1

abracadabra

3 \times 13 = 39 \text{ bits}

alphabet = \{a, b, c, d, r, \} \rightarrow 3 \text{ bits}
- Variable length encoding
- More frequent letters are represented with shorter codes

- Morse code

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>5</td>
<td>001</td>
</tr>
<tr>
<td>001090</td>
<td>r</td>
<td>01</td>
</tr>
</tbody>
</table>

Morgue \[\text{---} = \text{J}\]

Morse has pauses:
- Between letters: pause as long as 3 dots
SOS

excepted with no pause,

Variable length without separators like pauses in ambiguous.
Pauses do not contribute to compression.

How can we resolve ambiguity?

Prefix code: an encoding of a letter is not the beginning of the encoding of another letter.

g 11
b 01
001 000 001 11011
c 001
10
c 000

c e c d a
unique decoding
Binary Tree for Prefix Code

letters are leaves.

left for 0
right for 1

Why is it always prefix?
Because letters are only at the leaves.

(same length = perfectly balanced tree)

What is the optimal code (tree) given the frequency of each letter?
* optimal w.r.t. qshet criterion.

\[ S = \{ a, b, c \ldots \} \text{ alphabet} \]

\[ x \in S \text{, } y(x) \text{ encoding } y(x) = 001 \]

\[ \text{depth}_T (x) = |y(x)| \text{ length} \]

\[ |1 1 0| = 3 \]

\[ \text{depth}_T = 3 \]

**optimality : minimum average bit length**

\[ ABL(T) = \sum_{x \in S} f_x |y(x)| \]

\[ = \sum_{x \in S} f_x \text{ depth}_T(x) \]

\[ f_a = 0.5 \]

\[ f_b = 0.2 \]

\[ f_y = 0.3 \]

\[ ABL(T) = 0.5 \cdot 1 = 0.5 \text{ bit/r} \]

\[ + 0.2 \cdot 2 \]

\[ + 0.3 \cdot 2 \]
Solve \( \min_{T} ABL(T) \)

3 letters mean binary tree with 3 leaves

We need an algorithm for an arbitrary \# of letters!

**Observation (Lemmata)**

If a code (tree) is optimal then each parent has two children, [binary tree in "full"]

\[ ABL(T) > ABL(T') \]
Idea: construct a tree from bottom:

\[
ABL(\tau) = \sum x \cdot \text{depth}
\]

\[
= 3f_y + 3f_z + 2f_x + 1 \cdot f_s
\]

\[
= 2f_w + 2f_x + 1 \cdot f_s
\]

\[
f_w = f_y + f_z + (f_y + f_z)
\]

Start with the **two deepest** (= least frequent) letters and replace them with one virtual letter whose frequency is the sum of the original two.

\[
ABL(\text{new tree}) = ABL(\text{tree}) - f_y - f_z
\]
Continue with the new tree \( w, x, s \) and do the same: Choose the two deepest and replace them!

Huffman \((S, t_{265})\)

\[ Q = \text{build-min-heap} (S, t_{265}) \]

for \( i = 1 \) to \( |S| \)

\[ x = \text{extract Min} (Q) \]

\[ y = \text{extract Min} (Q) \]

\[ z = \text{new node} (T) \]

2. left = \( x \)

2. right = \( y \)

\[ t_z = t_x + t_y \]

\[ \text{insert} (Q, (z, t_z)) \]

return \( \text{extract Min} (Q) \)

\( \text{root} \)
\( S = \{ a, b, c, d, r \} \)

\[
\begin{array}{cccc}
0.5 & 0.2 & 0.05 & 0.05 & 0.2 \\
\end{array}
\]

\( Q : \ c, d, b, r, a \)

\[
\begin{array}{ccc}
x & y & \\
\end{array}
\]

\( f_2 = 0.1 \)

\( Q : \ cd, b, r, a \)

\[
\begin{array}{cccc}
0.1 & 0.2 & 0.2 & 0.5 \\
\end{array}
\]

\[
\begin{array}{ccc}
x & y & \\
\end{array}
\]

\( f_2 = 0.3 \)

\( Q : \ r, cd, a \)

\[
\begin{array}{cccc}
0.2 & 0.3 & 0.5 \\
\end{array}
\]

\[
\begin{array}{ccc}
x & y & \\
\end{array}
\]

\( f_2 = 0.5 \)

\( Q : \ a, cdbr \)

\[
\begin{array}{cccc}
0.5 & 0.5 \\
\end{array}
\]
\[ \text{ABL} = 1 \cdot 0.5 + 2 \cdot 0.2 + 3 \cdot 0.2 \]
\[ + 4 \cdot 0.05 \]
\[ + 4 \cdot 0.05 \]
\[ = 1.9 \text{ bits} \]

next time proof
of why is it optimal!

1. prefix \( \rightarrow \) binary tree
2. optimal \( \rightarrow \) which full binary tree has
   mis \( \text{mi}_3 \), \( \text{ABL} \)!