Huffman Coding
(Ch. 4.8, K&T)

Prof. Richard Zanibbi
Codes

Fixed Length Codes
Each symbol (e.g. letter) represented using same number of bits (e.g. ASCII)

Variable Length Codes
Symbols represented by different numbers of bits; normally reduces storage requirements

• Example: Morse code
(Binary) Prefix Codes

Code such that the 0/1 prefix for each symbol is unique

- Example: a: 11  b: 01  c: 001  d: 10  e: 000
- May be represented by a function, gamma:
  \[ \gamma(a) = 11 \]
- Can be decoded easily; as soon we read far ahead enough to match the code for a symbol (which is unique by def’n), we return the
Optimal Prefix Codes

We can represent the total length of our encoding in bits using the formula below

- $n$ is the number of symbols in the original message, $f_x$ the frequency of symbol $x$; $S$ the symbol alphabet

\[ \sum_{x \in S} n f_x |\gamma(x)| = n \sum_{x \in S} f_x |\gamma(x)| \]

Optimization Problem

Given an alphabet and set of frequencies for letters, we want a prefix code that minimizes the average bits per letter (red box above)
Prefix Codes as Binary Trees

Note that symbols appear at leaves: prefixes are disjoint.

Figure 4.16 Parts (a), (b), and (c) of the figure depict three different prefix codes for the alphabet $S = \{a, b, c, d, e\}$. 
Reformulation

Restatement of Optimization Problem

Find a binary tree $T$ & labeling of the leaves of this tree that minimizes average # bits/symbol:

$$\sum_{x \in S} f_x \cdot depth_T(x)$$
An insight...

There is an optimal prefix code \( (T^*) \) in which the two lowest frequency letters are siblings in \( T^* \).

**Figure 4.17** There is an optimal solution in which the two lowest-frequency letters label sibling leaves; deleting them and labeling their parent with a new letter having the combined frequency yields an instance with a smaller alphabet.
Huffman’s Algorithm

To construct a prefix code for an alphabet $S$, with given frequencies:

If $S$ has two letters then

   Encode one letter using 0 and the other letter using 1

Else

   Let $y^*$ and $z^*$ be the two lowest-frequency letters
   Form a new alphabet $S'$ by deleting $y^*$ and $z^*$ and
      replacing them with a new letter $\omega$ of frequency $f_{y^*} + f_{z^*}$
   Recursively construct a prefix code $\gamma'$ for $S'$, with tree $T'$
   Define a prefix code for $S$ as follows:
      Start with $T'$
      Take the leaf labeled $\omega$ and add two children below it
         labeled $y^*$ and $z^*$

Endif
Optimality of Huffman’s Algorithm

(4.32) $\text{ABL}(T') = \text{ABL}(T) - f_w$

(4.33) Huffman code for a given alphabet achieves the minimum average number of bits per letter of any prefix code

See course text for proofs of these properties:
- (4.32 - by definition)
- (4.33 - by induction, base case for 2 symbols; $\geq 3$ symbols by contradiction)
Analysis of Run Time

Naive implementation \( (O(k^2)) \)

\( k-1 \) recursive calls, finding low frequency symbols if \( O(k) \)

Priority Queue Implementation \( (O(k \log k)) \)

Store symbols in queue using frequency as the key: can insert and extract symbols in \( O(\log k) \) time

Each iteration: 2 deletions (min frequ. symbols), one insertion (add \( w \) for combined symbol): \( O(\log k) \)