Goals
- Describe pre-fix free codes & why they are desirable
- Describe Huffman's algorithm
- Prove correctness of Huffman's algorithm

Reminder: Spring Symposium Bonus

QuickSort Clarification

I wrote:

\[ \Pr(z_i, z_j \text{ compared}) = \Pr(z_i, z_j \text{ chosen as pivot} \mid \text{an element of } \{z_i, z_{i+1}, \ldots, z_j\} \text{ chosen as pivot} ) \]

but \[ P(A) = P(A \mid B)P(B) \] (Bayes Rule)

I should have written:

\[ \Pr(z_i, z_j \text{ compared}) = \Pr(z_i, z_j \text{ chosen as pivot} \mid \text{an element of } \{z_i, z_{i+1}, \ldots, z_j\} \text{ chosen as pivot} ) \times \Pr(\text{an element of } \{z_i, z_{i+1}, \ldots, z_j\} \text{ chosen as pivot} ) \]

Equals 1

Because we get to subarrays of size 1, one is eventually always chosen as a pivot.
Huffman Codes

Binary code: \( f: \Sigma \rightarrow \{0,1,\}^* \) (f maps characters to bit strings)

\[ \Sigma = \{a, b, \ldots, z, \ldots, 0, \ldots, 9 \} \]

32 letters \( \rightarrow \) each letter gets unique string from \( \{0,1,3,5\} \) \( \rightarrow \) ASCII

e.g. \( \Sigma = \{a, b, c\} \)

Suppose you have a message where a occurs 50%, b occurs 30%, c occurs 20%.

Q. What is the best and most efficient binary encoding of a, b, c to send message?

A) \( a \rightarrow 00 \)
\( b \rightarrow 01 \)
\( c \rightarrow 10 \)

B) \( a \rightarrow 0 \)
\( b \rightarrow 1 \)
\( c \rightarrow 01 \)

C) \( a = 0 \)
\( b = 11 \)
\( c = 01 \)

D) \( a = 0 \)
\( b = 10 \)
\( c = 11 \)

Average length: 2

\[ 0110 = abba \] or \( cba \)

\[ \text{Length of encoding of } a = \text{prob of } a \times \text{length of encoding of } a \]

\[ \text{Average length} = L(f) = \sum_{i \in \Sigma} l(f(i)) p_i \]

\[ 0110 \]
\[ \text{Start of b, end of c} \]
\[ \text{Need to go to } y \text{ to figure out} \]

\[ 0111 \]
\[ a \ c \ a \]

average length \( = 5 \cdot 1 + 5 \cdot 2 = 1.5 \)
Prefix-free code: \( \forall i, j \in \Sigma, f(i) \) is not prefix of \( f(j) \)

\[
\begin{array}{c|c}
  i & f(i) \\
  \hline 
  a & 0 \\
  b & 11 \\
  c & 01 \\
\end{array}
\]

\( f(a) \) is prefix for \( f(c) \)

Don't know if 0 is a or start of c

Problem: Given probability \( p_i \) for each \( i \in \Sigma \)
What is prefix-free code w/ smallest average length?

Subproblem: how to ensure prefix-free?

Fact: Binary codes \( \leftrightarrow \) binary trees

- \( \forall b \in \Sigma \), there is a vertex with label \( b \)
- Encoding of \( b \) is path from root to vertex \( b \)
Q: Create trees for each of the following codes. What property must the tree have to correspond to prefix-free code?

B) \( a \rightarrow 0 \)  
\( b \rightarrow 1 \)  
\( c \rightarrow 01 \)

C) \( a \rightarrow 0 \)  
\( b \rightarrow 11 \)  
\( c \rightarrow 01 \)

D) \( a \rightarrow 0 \)  
\( b \rightarrow 10 \)  
\( c \rightarrow 11 \)

Prefix-free codes correspond to trees where no letter node is ancestor of any other. 
\Rightarrow all letters are at leaves

As desired. Decoding is simple: follow path until hit a leaf.

Ex: 010110 \rightarrow a b c a

\text{Using this tree}
Optimal Encoding Problem

Input: \( P_i \) for each \( i \in \Sigma \)

Output: [Binary tree \( T \) with all letters at leaves] & Smallest average length: \( L(T) = \sum_{i \in \Sigma} P_i \cdot \text{depth of node } i \)

Bottom up Approach: Merge trees

Always get prefix free tree!
Q: What is the length of $f(i)$ for $i \in \Sigma$ if use merging strategy to create tree

A) # of mergers involving $i$

B) $\log_2 i$

C) $2 ^{\text{# of mergers involving } i}$

D) # of siblings of node $i$

Q:

(a) $p_a$

(b) $p_b$

(c) $p_c$

\[ p = ? \]

Prob of entering subtree is prob of $b$ or $c$ occurring

$P(b \lor c) = P(b) + P(c)$

(Prob of union of disjoint events)
Huffman's Strategy

While (> 1 tree to be merged)
  • Find 2 trees with smallest probability
  • Merge into new tree with new probability = sum of old probabilities

Q: Create tree using this algorithm for
  30%  25%  20%  15%  10%
  a  b  c  d  e

  36%  25%  20%  25%
  a  b  c  d e

  30%
  a

  45%
  a

  45%
  a

What is average length?
Write + encode a message

\[ L = 1.3 + 0.15 \times 3 + 0.75 \times 2 = 2.25 \]
**Thm:** Huffman’s Algorithm Produces a Tree \( T \) with minimum average length

\[
L(T) = \sum_{i \in \Sigma} P_i [\text{depth node } i] \]

**Pf:** Induction on \( n = |\Sigma| \). Assume \( n \geq 2 \)

**Base Case:** If \( n = 2 \), \( \emptyset \) is optimal

**Inductive Step:** Let \( a, b \) be the letters with the lowest frequency.

\[1. \] There is a tree with optimal \( L \) s.t. \( a, b \) are siblings. [Use exchange!] For contradiction, suppose \( T^* \) is optimal tree.

Let \( x, y \) be siblings at deepest level of tree

Let \( T \) be tree where \( x \leftrightarrow a \)
\( y \leftrightarrow b \)
Q. What is \( L(T^*) - L(T) \)?

\[
L(T^*) = \sum_{i \in \Sigma - \{a, b, x, y\}} p_i d_i + p_a d_a + p_b d_b + \left(p_x', p_y\right) d_{x/y} \\
L(T) = \sum_{i \in \Sigma - \{a, b, x, y\}} p_i d_i + p_a d_a + p_b d_b + p_x d_x + p_y d_y \\
L(T^*) - L(T) = p_x (d_{x/y} - d_a) + p_y (d_{x/y} - d_b) + p_a (d_a - d_{x/y}) \\
+ p_b (d_b - p_{x/y}) \\
= (p_x - p_a) (d_{x/y} - d_a) + (p_y - p_b) (d_{x/y} - d_b)
\]

So new tree \( T \) must also have optimal average length, but \( T \in X_{ab} \). So there is always an optimal tree with \( a, b \) siblings.
From [1], without loss of generality, there is an optimal tree $T$ where $a/b$ are siblings.

Let $T'$ be the tree that is same as $T$, but with $a/b$ replaced by

one letter $a/b$ with probability $p_{ab}$. 

Huffman Codes Page 10
Q: What is $L(T) - L(T')$

$A: \quad L(T') = \sum_{i \in \Sigma - \{a,b\}} p_i d_i + (p_a + p_b) d_{a/b}^\prime$

$\quad L(T) = \sum_{i \in \Sigma - \{a,b\}} p_i d_i + (p_a + p_b)(d_{a/b} + 1)$

$\quad L(T) - L(T') = p_a + p_b$

---

Set of all trees where $a,b$ are siblings

Tree with minimum $L$ in $X_{ab}$, can be found by finding tree with minimum $L$ in $X_{ab}$

By inductive assumption, Huffman does this! and replacing $a/b$ with $a/b$.