Lecture #10
2 More Greedy Algorithms

• Huffman Encoding
  - used to compress data
    - gzip, jpeg, mp3, ...
  - Intro to Entropy

• Horn Clauses
  - special case of satisfiability
    - of a Boolean expression
      (x or \overline{y}) \land (\overline{z} or \overline{w}) \land ...
  - used in logic programming (Prolog,...)
Data Compression Problem - 1

Given string of chars: ABACCDDBB
from alphabet {A, B, C, D} how many bits do we need to encode it?

Obvious way

4 chars \Rightarrow 2 \text{ bits/char} \Rightarrow 2n \text{ bits for } n \text{ chars}

Can we do better?

Need more information
Data Compression Problem - 2

- Suppose we also know the frequency with which each character appears:
  char: A B C D
  freq: .4 .3 .2 .1

- Can we use shorter codes (fewer bits) for more common characters?
  Try A=0, B=1, C=00, D=01

  What is 000? AAA? AC? CA?

  Goal: avoid common "prefix", 0 for A and C
Prefix-free Codes

<table>
<thead>
<tr>
<th>char</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>freq</td>
<td>.4</td>
<td>.3</td>
<td>.2</td>
<td>.1</td>
</tr>
<tr>
<td>code</td>
<td>0</td>
<td>10</td>
<td>110</td>
<td>111</td>
</tr>
</tbody>
</table>

Total #bits for n chars
= n \cdot 0.4 \cdot 1 + n \cdot 0.3 \cdot 2 + n \cdot 0.2 \cdot 3 + n \cdot 0.1 \cdot 3
= 1.9n versus 2n for 2-bit encodings

Def: Code is prefix free if no char is prefix of another

Fact: 1-1 correspondence between prefix free codes and full binary trees (i.e. nodes have 0 or 2 children) with characters at leaves
Prefix free Codes - 2

Goal: Find full binary tree with minimal cost, given

\[ n = \# \text{chars}, \quad f_i = \text{frequency of char } i \quad (\sum_{i=1}^{N} f_i = 1) \]

\[ \text{cost} = \sum_{i=1}^{N} f_i \cdot (\text{length of encoding of char } i) \]

\[ = \sum_{i=1}^{N} f_i \cdot (\text{depth of leaf } i \text{ in tree}) \]

sending \( m \) chars requires \( m \cdot \text{cost bits} \)

Greedy Intuition: Who goes at bottom of tree?

i.e. longest encodings? Least frequent char.

smallest \( f_i \) at bottom
Cost of an Optimal Tree

\[ \text{Cost} = \frac{1}{2} \sum_{i=1}^{n} f_i \cdot (\text{depth of leaf } i \text{ in tree}) \]

Claim: Suppose \( f_1 \leq f_2 \leq \ldots \leq f_n \)

\( f_1 \) and \( f_2 \) are siblings at bottom of tree.

Proof: Suppose not

then swap \( f_1 \) & \( f_2 \)

\[ f_2 \rightarrow f_1 \]

Claim: Lower cost! Swap \( f_1 \) & \( f_a \) changes cost by

\[ (d_i \cdot f_i + d_a \cdot f_a) - (d_i \cdot f_a + d_a \cdot f_i) \]

\[ = (d_a - d_i) \cdot (f_a - f_i) \geq 0 \]

Def: for any node except root, \( \text{cost}(v) \leq f_i \) of all leaves in subtree rooted at \( v \)

Claim: Cost of tree = \( \sum_{v \text{ except root}} \text{cost}(v) \)
Greedy Algorithm for an Optimal Tree - 1

1. pick 2 least frequent chars $f_1, f_2, \ldots$
2. replace them by 1 char with frequency $f_1 + f_2$
3. find optimal tree with $n-1$ chars
4. replace leaf for $f_1 + f_2$ by $\hat{f_1 f_2}$

proof: Induction on $n = \#\text{chars}$

Base: $n = 1$ or 2 chars, use 1 bit

$T' = \text{tree found on } f_1 + f_2, f_3, f_4, \ldots, f_n$ (optimal by induction)

$\hat{T} = T'$ with $\hat{f_1 f_2}$ added to $f_1 + f_2$

if $u \in \hat{T}$ and $\hat{T}'$ then $\text{cost}(\hat{v})$ same

$\Rightarrow \text{cost}(T) = \text{cost}(T') + f_1 + f_2$

if $T$ not optimal $\Rightarrow \exists$ better tree for $f_1, f_2, f_3, \ldots$

$\Rightarrow T'$ not optimal (replace $f_1, f_2$ by $f_1 + f_2$, siblings at bottom)

contradiction
Greedy Algorithm for an Optimal Tree - 2

1. pick 2 least frequent chars \( f_1 \leq f_2 \leq \ldots \)
2. replace them by 1 char with frequency \( f_1 + f_2 \)
3. find optimal tree with \( n-1 \) chars
4. replace leaf for \( f_1 + f_2 \) by \( \wedge \)

Create Priority Queue \( H \) of \( \{1, 2, \ldots, n\} \)
ordered by \( f_1 \leq f_2 \leq \ldots \leq f_n \)

for \( k = n+1 \) to \( 2n-1 \)

\[
\begin{cases}
  i = \text{deletemin}(H), \ j = \text{deletemin}(H) \\
  \text{create node } k \text{ with children } i \text{ and } j \\
  \hat{f}_k = f_i + f_j \ ; \ \text{insert}(k)
\end{cases}
\]

\[ \text{cost of } A \ y = O(n \cdot \text{cost(deletemin)} + n \cdot \text{cost(insert)}) = O(n \log n) \]
Huffman Encoding Example

<table>
<thead>
<tr>
<th>char</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>freq</td>
<td>.4</td>
<td>.3</td>
<td>.2</td>
<td>.1</td>
</tr>
</tbody>
</table>

0 10 110 111

1 + 2 = .3
.3 + .3 = .6
Horn Formulas

- Can we satisfy a Boolean expression?
  - Notation  \( w = \) Boolean variable (\( \text{TorF} \))
    \[ \land = \text{and}, \quad \lor = \text{or}, \quad \overline{w} = \text{not} \ w \]
  - Are there values (\( \text{TorF} \)) of Bool.\ vars. that make this true?
    \[ (w \land y \land z) \land (x \land y \Rightarrow w) \land (\overline{u} \lor \overline{v} \lor \overline{v} \lor z) \ldots \]

- General case: \( \text{NP} \) (hard (chap 8))

- Horn Formulas - special case - clauses must be
  1) implication  \( (u \land v \land w) \Rightarrow x \)
     with all positive vars (no \( \overline{u}, \overline{v}, \text{etc} \))
     includes  \( \Rightarrow x \) (\( x \) must be \( T \))
     recall  \( a \Rightarrow b = \overline{a} \lor b \)
  2) negative clauses  \( (x \lor \overline{v} \lor \overline{z}) \)
Horn Formula: Example

\( x = \) murder took place in kitchen
\( y = \) butler innocent
\( z = \) colonel asleep at 8pm
\( w = \) murder took place at 8pm
\( u = \) colonel innocent
\( v = \) professor innocent

Implication \((z \land w) \Rightarrow u\)

Negative clause \((\overline{0} v y v \overline{v})\)

See Prolog
Greedy Algorithm for Horn Formulas

3 kinds of formulas: 
1) \((z \land \omega) \Rightarrow u\)
2) \(\Rightarrow x\) \((x = T)\)
3) \((\bar{u} \lor \bar{v} \lor \bar{w})\)

Set all variables to false \((3\) ok \(2\) not ok \(1\) ok\)

While an implication not satisfied, set rhs = T

If all negative clauses satisfied
Satisfiable: return assignment
else
\(\text{Not satisfiable}\)

Example:
\((w \lor y \lor z) \Rightarrow x, (x \land z) \Rightarrow w, x \Rightarrow y, \Rightarrow x, (x \land y) \Rightarrow w, (\bar{w} \lor \bar{x} \lor \bar{v})\)
\(\text{OK} \quad \text{OK} \quad T \quad T \quad T \quad \text{oops} \quad 12\)
Back to Huffman: Intro to Entropy

- What is Entropy?
  - If a random event occurs, and I tell you the outcome, how much Information is that?
  - \( I(p) = \text{information from being told a random event with probability } p \text{ occurred} \)
- Base case: flip fair coin, be told H or T: \( I(\frac{1}{2}) = 1 \text{ bit} \)
  - If \( p > \frac{1}{2} \), \( I(p) < 1 \), if \( p < \frac{1}{2} \), \( I(p) > 1 \)
  - Two independent events, probs \( p_1 \) and \( p_2 \) ⇒ \( I(p_1; p_2) = I(p_1) + I(p_2) \)
  ⇒ \( I(p) = \log_2 \left( \frac{1}{p} \right) \)
Intro to Entropy -2

- Suppose n possible outcomes, probabilities $p_1, p_2, \ldots, p_n$

Expected information $= E(I) = \sum_{i=1}^{n} p_i I(p_i) = \sum_{i=1}^{n} p_i \log_2 \left(\frac{1}{p_i}\right) = \text{Entropy}$, measures how "random" distribution is

Ranges from 0 (one $p_i = 1$, rest 0) to $\log_2 n \left(\text{all } p_i = \frac{1}{n}\right)$

- In Huffman, suppose all $f_i = p_i = \frac{1}{2^k_i}$ for some $k_i$
  - can show depth of $f_i$ in Huffman tree $= k_i$

- To encode $m$ chars with frequencies $p_i$, takes $m \cdot \sum_{i=1}^{n} p_i$ \cdot \text{depth of } f_i \text{ in tree} = m \cdot E(I) = m \cdot \text{Entropy}$

- Thm (Shannon) $m \cdot \text{Entropy}$ is a lower bound on any encoding scheme