Greedy Algorithms
"Greedy algorithms build up a solution piece by piece always choosing the next piece that offers the most obvious and immediate benefit."
Scheduling

Input: Collection of n jobs specified by their time intervals $[s_1, e_1], \ldots, [s_n, e_n]$

Goal: Complete as many jobs as possible without any time conflicts

Example: $[1, 3], [10, 11], [18, 20], [2, 19]$

Greedy alg: Which job to pick next?

- Pick the earliest end time? ✓
- Pick the shortest? X
- Pick the earliest start time? X

← do this 3
Claim 1: Greedy picks an optimal schedule $G = j_1 \cdots j_k$ (sorted by end time)

(Intuition) Suppose $S$ is some optimal schedule

\[ S = [s_{i_1}, e_{i_1}] \quad [s_{i_2}, e_{i_2}] \quad [s_{i_3}, e_{i_3}] \quad \cdots \quad [s_{i_k}, e_{i_k}] \]

\[ G = [d_1, d_2] \quad \cdots \quad [d_k, d_{k+1}] \quad \text{(sorted by end times)} \]

Suppose $S$ and $G$ agree on first $m$ jobs

"Convert" $S$ into $S'$ which agrees w/ $G$ on first $m + 1$

Claim 2: For $m \leq k$, $\exists$ opt sched $S$ which agrees w/ $G$ on first $m$ jobs

Proof:

By induction on $m$.

Base case: $m = 0$. ✓

Inductive step: Suppose $i_1 = j_1, \ldots, i_m = j_m$

(i) $i_{m+1} = j_{m+1}$. ✓

(ii) $i_{m+1} \neq j_{m+1}$. Let $S' = S$ w/ $i_{m+1}$ replaced by $j_{m+1}$.

Then $S'$ is optimal, has no conflicts,

agrees w/ $G$ on first $m+1$.

\[ 2 \Rightarrow 1: \text{Suppose } S \text{ is opt, agrees w/ } G \text{ on first } k. \text{ So } S = G. \]
Horn formulas

Variables: \( x_1, \ldots, x_n \in \{ \text{True, False} \} \)
literal: \( x_i \) or \( \overline{x}_i \)

Clauses
1. Implications (with unnegated variables)

\[(x_i \land x_j) \Rightarrow x_k \Rightarrow x_k \ (x_k = \text{True})\]

\[\text{AND of any number of variables}\]

2. Pure negative clauses

\[ (\overline{x}_i \lor \overline{x}_j \lor \overline{x}_k) \ (\text{OR of any number of negations})\]

Example: Dinner party among friends A, B, C, \ldots, Z. Who to invite?

\[ \Rightarrow C \ (A \land B) \Rightarrow D \ (E \lor F \lor G) \]
Horn-SAT

Input: Horn formula \( \varphi \)
Output: Satisfying assignment, if one exists

\[
\forall \varphi, \text{ set } x_0 = \text{False} \\
\text{while some implication } (x_i \land \cdots \land x_j) \Rightarrow x_k \\
\text{is not satisfied} \\
\text{set } x_k = \text{True} \\
\text{if every pure negative clause } (\overline{x_i} \lor \cdots \lor \overline{x_j}) \\
\text{is satisfied} \\
\text{Return } (x_1, \ldots, x_n) \\
\text{else} \\
\text{Return "not satisfiable"}
\]

Example

Variables
\[
\begin{align*}
  x &= T \\
  y &= T \\
  z &= T \\
  w &= T \\
\end{align*}
\]

Clauses
\[
\begin{align*}
  (w \lor y \lor z) &\not\rightarrow x & (\times) \\
  (x \lor z) &\Rightarrow w & (\checkmark) \\
  x &\exists y & (\checkmark) \\
  y &\exists x & (\checkmark) \\
  (x \lor y) &\Rightarrow w & (\checkmark) \\
  (\overline{w} \lor x \lor y) &\not\rightarrow & (\times)
\end{align*}
\]
Claim: (i) If Greedy outputs solution, then solution is satisfying
(ii) If Greedy outputs “not satisfiable”, the $\exists$ is unsatisfiable.

Pf: (i) is clear.

For (ii), we'll prove the following...

Subclaim: At all steps in Greedy, if any set of vars is assigned True, then they are also True in every satisfying assign.

Pf: By induction on while loop.

Base case: all vars set to False.

Induction step: Suppose $(\neg x_1 \land \cdots \land \neg x_j) \land x_k$ is selected.
Then LHS is true.
So LHS also true in every satisfying.
Then $x_k = T$ in sat assgn. II assignment.

For (ii), assume false. Then Greedy outputs “no”, but I sat assgn.

$\Rightarrow$ Greedy violates $(\neg x_1 \lor \cdots \lor \neg x_j)$. Then sat assgn violates clause.

Contradiction! $\square$
Data compression

We have an alphabet of 32 characters with frequencies $f_1, ..., f_{32}$. Say we have a text with $N$ characters, and we want to encode it as efficiently as possible.

Naive: every character $\to$ 5 bits  

Overall: $5N$ bits

<table>
<thead>
<tr>
<th>Freq.</th>
<th>Example</th>
<th>Encoding #1</th>
<th>Cost</th>
<th>Encoding #2</th>
<th>Cost</th>
<th>Encoding #3</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.4</td>
<td>A</td>
<td>00</td>
<td>0.4N \cdot 2</td>
<td>0</td>
<td></td>
<td>0.4N \cdot 1</td>
<td></td>
</tr>
<tr>
<td>0.2</td>
<td>B</td>
<td>01</td>
<td>0.2N \cdot 2</td>
<td>00</td>
<td>110</td>
<td>0.2N \cdot 3</td>
<td></td>
</tr>
<tr>
<td>0.3</td>
<td>C</td>
<td>01</td>
<td>0.3N \cdot 2</td>
<td>1</td>
<td>10</td>
<td>0.3N \cdot 3</td>
<td></td>
</tr>
<tr>
<td>0.1</td>
<td>D</td>
<td>11</td>
<td>$\frac{0.1N \cdot 2}{2N}$</td>
<td>01</td>
<td>110 $\frac{0.1N \cdot 3}{1.9N}$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$00|01|11|01$  
$A \quad B \quad D \quad B$

000 AAA 0110111100  
$A \quad B \quad D \quad C$

$B \quad A \quad D \quad B$
Prefix-free codes & trees

Any prefix-free code corresponds to a binary tree (and vice versa).

Example: A B C D

0 101 11 100

Prefix-free encoding:

for Horn-SAT

\[(x \implies y) \quad (\neg x \lor y)\] same

"if x is True, then y is True"

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