Part V: other problems
Satisfiability (SAT)

**Definition:** A **Boolean** variable takes a value either **false** (F) or **true** (T).

**Definition:** A **Boolean** formula takes any of the following forms:

- \( g \): true if \( g \) is true
- \( \neg g \): "not \( g \)" : true if \( g \) is false
- \( g \lor h \): "\( g \) or \( h \)" : true if \( g \) or \( h \) is true (or both)
\( g \land h: \) "\( g \) and \( h \)"; true if both \( g \) and \( h \) are true.

where \( g, h \) are Boolean variables or other Boolean formulas.

**Example**

\[ x \lor (\neg y \land z) \]

**Notation:**

\[ \bigwedge_{i=1,\ldots,n} x_i = x_1 \land x_2 \land \cdots \land x_n \]

\[ \bigvee_{i=1,\ldots,n} x_i = x_1 \lor x_2 \lor \cdots \lor x_n \]
Definition: An assignment for a set of variables gives a value (T or F) to each variable.

Example: \( x = T, \ y = F, \ z = T \)

Definition: A Boolean formula is satisfiable if there exists an assignment for its variables such that the value of the formula is true. Otherwise, it is unsatisfiable.
Example: \((x \lor y \lor z) \land (\neg x \lor \neg y \lor z) \land (x \lor y \lor z)\) is SAT. Proof: Let \(x = T, y = F, z = T\).

Definition: SAT problem: Given a Boolean formula, find an assignment, or prove that it is UNSAT.
Example: We organize a wedding dinner, with invitees $N = \{1, ..., n\}$ and tables $K = \{1, ..., k\}$. Each invitee hates a set of other invitees $H_i \subseteq N$, for all $i$. Find a seating arrangement such that no one is seated with someone they hate. Tables can seat any number of invitees, and invitees can be assigned multiple tables.
\[ \text{VAR: } x_{ij} = \begin{cases} \text{true} & \text{if invitee } i \text{ is at table } j, \\ \text{false} & \text{otherwise,} \end{cases} \quad \text{for } i \in N, j \in K. \]

\[ \text{MODEL: } \] 
\[ \bigwedge_{i \in N} \left( \bigvee_{j \in K} x_{ij} \right) \]
\[ \bigwedge_{i \in N} \bigwedge_{j \in K} \left( \neg x_{ij} \lor \bigvee_{l \in H_i} \neg x_{lj} \right) \]

- each invitee assigned at least one table
- either \( i \) is not at table \( j \) or none of \( H_i \) is at table \( j \).
What would an IP model look like?

\[ \text{VAR: } x_{ij} = \begin{cases} 1 & \text{if } i \text{ is at table } j \\ 0 & \text{otherwise} \end{cases} \]

\[ \text{MODEL: } \]

\[ \begin{align*}
\min & \quad 0 \\
\text{s.t. } & \quad \sum_{j \in k} x_{ij} \geq 1, \quad \forall i \in N \\
& \quad \sum_{l \in H_i} x_{lj} \leq (1-x_{ij}) \cdot |w_i|, \quad \forall i \in N, \forall j \in k \\
& \quad x_{ij} \in \{0,1\}, \quad \forall i \in N, \forall j \in k
\end{align*} \]
Boolean reformulations

1) \( \land \) and \( \lor \) are commutative and associative

2) \( x \land (y \lor z) = (x \land y) \lor (x \land z) \)

3) \( x \lor (y \land z) = (x \lor y) \land (x \lor z) \)

4) \( x \land (x \lor y) = x \)

5) \( x \lor (x \land y) = x \)

6) \( \lnot (x \lor y) = \lnot x \land \lnot y \)

7) \( \lnot (x \land y) = \lnot x \lor \lnot y \)
**Proof: 7)**

<table>
<thead>
<tr>
<th></th>
<th>x</th>
<th>y</th>
<th>$\neg(x \lor y)$</th>
<th>$\neg x \lor \neg y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

*rest: exercise*
Definition: A literal is a Boolean variable $x$ or its negation $\neg x$.

Definition: A clause is an or of literals:
$$\bigvee_{j \in C^+} x_j \lor \bigvee_{j \in C^-} \neg x_j$$

Definition: A formula is in conjunctive normal form (CNF) if it is an and of clauses:
$$\bigwedge_{i=1, \ldots, m} \left( \bigvee_{j \in C_i^+} x_j \lor \bigvee_{j \in C_i^-} \neg x_j \right)$$
Example

\((x \lor \neg y \lor z) \land (x \lor y \lor \neg z) \land (\neg x \lor y \lor z)\)

\text{is in CNF.}

Theorem Every Boolean formula can be put in CNF, whose size is polynomial in the size of the original formula (if we allow additional variables).
Remark:

- The **Disjunctive Normal Form (DNF)** is an OR of ANDs of literals.
- The size of the DNF can be exponential.
- The DNF is simply a list of the assignments that satisfy the formula.

Example:

\[(x \land \neg y \land \neg z) \lor (\neg x \land \neg y \land z)\]

Assignments:
1) \(x = T, \ y = F, \ z = F\)
2) \(x = F, \ y = F, \ z = T\)
3) \(x = F, \ y = T, \ z = T\)
How do we solve SATs

Consider a formula in CNF with variables $x_i$, $i = 1, \ldots, n$.

Naive method: enumerate all $2^n$ possible assignments. Check the formula for each.

Better method: Backtracking

Set $x_i = F$

if some clause is just $(x_i)$, UNSAT

otherwise, simplify formula, solve it.
Set $x_1 = T$
if some clause is just $(\neg x_1)$, UNSAT
otherwise, simplify formula, solve it.

Example:

$$(\neg x_1 \lor \neg x_2 \lor x_3) \land (x_1 \lor x_2)$$

$x_1 = F$

$x_1 = T$

$$(T \lor \neg x_2 \lor x_3) \land (F \lor x_2)$$

$= T \land (x_2)$

$= x_2$

$x_2 = F$

$x_2 = T$

$F \rightarrow \text{UNSAT}$

$T \rightarrow \text{SAT}$

Assignment:

$x_1 = F$

$x_2 = T$

$x_3 = T$ or $F$
Any SAT in CNF can be formulated as an IP

\[ \Lambda \left( \bigvee_{i=1}^{m} x_i \bigvee_{j \in C_i^+} \bigvee_{j \in C_i^-} \right) \]

\[ \iff \min \ 0 \]

\[ \text{s.t. } \sum_{j \in C_i^+} x_j + \sum_{j \in C_i^-} (1-x_j) \geq 1, \]

\[ \forall i = 1, \ldots, m \]

Correct in theory.

In practice, NEVER do that.
Why:
- no objective function $\Rightarrow$ no pruning
- consider a node of b8b/backtracking tree

If one constraint/ clause has just one variable, we should just fix it:

\[ x_j \geq 1 \Rightarrow x_j = 1 \]
\[ 1 - x_j \geq 1 \Rightarrow x_j = 0 \]

Otherwise: **Claim**: the LP relaxation is always feasible.
Proof: Set $x_j = \frac{1}{2}$, $\forall j$.

$\Rightarrow$ the LP relaxation tells us nothing.