Fundamental Computer Science

Malin Rau and Denis Trystram
(inspired by Giorgio Lucarelli)

February, 2020
Definition of time complexity classes

- P: problems solvable in \( O(n^k) \) time
- NP: problems verifiable in \( O(n^k) \) time

Prove that a problem belongs to NP

- give a polynomial-time verifier
- (give a Non-deterministic Turing Machine)

Reduction from problem A to problem B \((A \leq_P B)\)

1. transform an instance \( I_A \) of A to an instance \( I_B \) of B
2. show that the reduction is of polynomial size
3. prove that:
   “there is a solution for the problem A on the instance \( I_A \)
   if and only if
   there is a solution for the problem B on the instance \( I_B \)"
Definition of the class \textit{NP-complete}

\textit{SAT} is \textit{NP-complete}

Use reductions to prove \textit{NP-completeness}

Variants of \textit{SAT}
Introduction to the SAT problem
Boolean formulas

- \( x_i \): a Boolean variable, values TRUE or FALSE
- \( \overline{x}_i \): negation of \( x_i \)
- \( x_i, \overline{x}_i \): literals
- \( \lor \): logical OR
- \( \land \): logical AND
- \((x_1 \lor \overline{x}_3 \lor x_4)\): clause, a set of literals in disjunction
- \( \mathcal{F} = (x_1 \lor x_2 \lor \overline{x}_3) \land (\overline{x}_4) \land (x_1 \lor x_4)\): a Boolean formula in Conjunctive Normal Form (CNF), a set of clauses in conjunction
  - every formula can be written in CNF (focus on CNF formulas)
- **assignment**: give TRUE or FALSE value to variables
- a formula is *satisfiable* if there is an assignment evaluating to TRUE
  - i.e, \((x_1, x_2, x_3, x_4) = (\text{TRUE, TRUE, TRUE, FALSE})\) for the above formula \( \mathcal{F} \)
The satisfiability problem

- \( X = \{x_1, x_2, \ldots, x_n\} \): set of variables
- \( C = \{c_1, c_2, \ldots, c_m\} \): set of clauses
- \( \mathcal{F} = c_1 \land c_2 \land \ldots \land c_m \)

\[ \text{SAT} = \{ \langle \mathcal{F} \rangle \mid \mathcal{F} \text{ is a satisfiable Boolean formula} \} \]

- \( k\text{SAT} \): each clause has at most \( k \) literals
  (in some definitions exactly \( k \) literals)
- \textbf{example} of 2SAT: \((x_1 \lor \bar{x}_2) \land (x_2 \lor x_3) \land (x_2 \lor \bar{x}_3)\)
2SAT ∈ P
Preliminaries

- Assume that each clause has \textbf{exactly} two literals

- $x \Rightarrow y$: implication

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y$</th>
<th>$x \Rightarrow y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>FALSE</td>
<td>FALSE</td>
<td>TRUE</td>
</tr>
<tr>
<td>TRUE</td>
<td>FALSE</td>
<td>FALSE</td>
</tr>
<tr>
<td>FALSE</td>
<td>TRUE</td>
<td>TRUE</td>
</tr>
<tr>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
</tr>
</tbody>
</table>

- $x \Rightarrow y = \overline{x} \lor y$

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y$</th>
<th>$\overline{x}$</th>
<th>$\overline{x} \lor y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>FALSE</td>
<td>FALSE</td>
<td>TRUE</td>
<td>TRUE</td>
</tr>
<tr>
<td>TRUE</td>
<td>FALSE</td>
<td>FALSE</td>
<td>FALSE</td>
</tr>
<tr>
<td>FALSE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
</tr>
<tr>
<td>TRUE</td>
<td>TRUE</td>
<td>FALSE</td>
<td>TRUE</td>
</tr>
</tbody>
</table>
2SAT $\in$ P

- Construct a directed graph $G$
  - for each literal $x \in X \cup \bar{X}$, add a vertex
  - for each clause $x \lor y$, add the arcs $(\bar{x}, y)$ and $(\bar{y}, x)$
    - corresponds to implications $\bar{x} \Rightarrow y$ and $\bar{y} \Rightarrow x$

$$F = (x_1 \lor \bar{x}_2) \land (\bar{x}_1 \lor x_3) \land (x_1 \lor x_2) \land (\bar{x}_3 \lor x_4) \land (\bar{x}_1 \lor x_4)$$

We want $(\bar{x}_1 \lor x_4) = \text{TRUE}$

- arc $(x_1, x_4)$ means:
  - if $x_1 = T$ then $x_4$ should be $T$
  - if $x_4 = F$ then $x_1$ should be $F$

- arc $(\bar{x}_4, \bar{x}_1)$ means:
  - if $\bar{x}_4 = T$ then $\bar{x}_1$ should be $T$
  - if $\bar{x}_1 = F$ then $\bar{x}_4$ should be $F$
2SAT ∈ P

Lemma

If there is a path from $x$ to $y$ in $G$, then there is also a path from $\overline{y}$ to $\overline{x}$.

Proof:

$$x \rightarrow \cdots \rightarrow a \rightarrow b \rightarrow \cdots \rightarrow y$$

- By construction:
  - we add an arc $(a, b)$ if $(\overline{a} \lor b)$ exists in $\mathcal{F}$
  - but if $(\overline{a} \lor b)$ exists in $\mathcal{F}$, then we add also the arc $(\overline{b}, \overline{a})$

- Apply the argument for all arcs in the path from $x$ to $y$

$$\overline{x} \leftarrow \cdots \leftarrow \overline{a} \leftarrow \overline{b} \leftarrow \cdots \leftarrow \overline{y}$$
Lemma

If there is a variable $x$ such that $G$ has both a path from $x$ to $\overline{x}$ and a path from $\overline{x}$ to $x$, then $F$ is not satisfiable.

\[ F = (x_1 \lor \overline{x}_2) \land (x_2 \lor \overline{x}_3) \land (x_3 \lor \overline{x}_4) \land (x_4 \lor \overline{x}_1) \land (\overline{x}_4 \lor \overline{x}_1) \land (x_2 \lor x_3) \]

If $x_1 = \text{TRUE}$, then $x_4$ should be \text{TRUE}, and then $(\overline{x}_4 \lor \overline{x}_1)$ is not satisfiable

If $x_1 = \text{FALSE}$, then $x_2$ should be \text{FALSE}, and then $\overline{x}_3$ should be \text{FALSE}, and then $(x_2 \lor x_3)$ is not satisfiable
Lemma

If there is a variable $x$ such that $G$ has both a path from $x$ to $\bar{x}$ and a path from $\bar{x}$ to $x$, then $F$ is not satisfiable.

Proof:

- assume that $F$ is satisfiable (for contradiction)
- case 1: $x = \text{TRUE}$

\[
\begin{align*}
x & \rightarrow \cdots \rightarrow a \rightarrow b \rightarrow \cdots \rightarrow \bar{x} \\
T & \quad T \quad F \quad F
\end{align*}
\]

There should be an arc $(a, b)$ with $a = T$ and $b = F$. That is, $(\bar{a} \lor b)$ is not satisfiable.

Hence, $x$ cannot be TRUE.

- case 2: $x = \text{FALSE}$

Same arguments give that $x$ cannot be FALSE on path from $\bar{x}$ to $x$.

- Then, $F$ is not satisfiable, a contradiction.
2SAT ∈ P

Algorithm

1. **while** there are non-assigned variables **do**
2. Select a literal $a$ for which there is not a path from $a$ to $\overline{a}$.
3. Set $a = \text{TRUE}$.
4. Assign TRUE to all reachable literals from $a$.
5. Eliminate all assigned variables from $G$.

$$F = \left( x_1 \lor \overline{x}_2 \right) \land \left( \overline{x}_1 \lor \overline{x}_3 \right) \land \left( x_1 \lor x_2 \right) \land \left( \overline{x}_3 \lor x_4 \right) \land \left( \overline{x}_1 \lor \overline{x}_4 \right)$$
Lemma (Correctness of the algorithm)

Consider a literal $a$ selected in Line 2 of the algorithm. There is no path from $a$ to both $b$ and $\overline{b}$.

Proof:

- Assume there are paths from $a$ to $b$ and from $a$ to $\overline{b}$.
- Then, there are paths from $\overline{b}$ to $\overline{a}$ and from $b$ to $\overline{a}$ (by the first lemma).
- Thus, there are paths from $a$ to $\overline{a}$ (passing through $b$ or $\overline{b}$).
- $a$ cannot be selected by the algorithm because we only select $a$ if there is not a path from $a$ to $\overline{a}$, a contradiction.
Exercise

A Horn formula has at most one positive literal per clause. Prove that \( \text{Horn-SAT} \in \mathbb{P} \), where

\[
\text{Horn-SAT} = \{ \langle \mathcal{F} \rangle \mid \mathcal{F} \text{ is a satisfiable Horn formula} \}
\]

Example:

\[
\mathcal{F} = (x_1 \lor \bar{x}_2 \lor \bar{x}_5 \lor \bar{x}_3) \land (x_2 \lor x_3 \lor \bar{x}_4) \land (\bar{x}_1 \lor \bar{x}_5) \land (x_3 \lor \bar{x}_4) \land (x_4)
\]

- negative literal \( \bar{x}_i \), \( i \in \mathbb{N} \)
- positive literal \( x_i \), \( i \in \mathbb{N} \)

Tipp:
- What has to happen to clauses that contain only one single literal?
- Consider the case that each clause contains a negative literal.
A Horn formula has at most one positive literal per clause. Prove that $\text{Horn-SAT} \in \mathbb{P}$, where

$$\text{Horn-SAT} = \{ \langle F \rangle \mid F \text{ is a satisfiable Horn formula} \}$$

Algorithm:

1. **while** there are clauses with only one literal
   1.1 pick a clause $c$ with only one literal
   1.2 set the corresponding variable to TRUE or FALSE such that the clause is satisfied
   1.3 delete all clauses that are satisfied by the assignment and remove the variable from all the other clauses

2. set all non assigned variables to FALSE

After step 1 all the clauses contain at least one negative literal. Therefore, after setting all variables to FALSE in step 2 every clause will contain at least one literal that is TRUE. Hence all the clauses are satisfied. The algorithm has a time complexity of at most $O((mn)^2)$
NP-COMPLETENESS
**NP-COMPLETENESS**

**Definition**
A language $B$ is **NP-complete** if
- $B$ is in NP, and
- every language $A$ in NP is polynomially reducible to $B$.

**Theorem**

*If $B$ is NP-complete and $B \in P$, then $P = NP$.***

**Proof:**
- direct from the definition of reducibility
Definition
A language $B$ is NP-complete if

- $B$ is in NP, and
- every language $A$ in NP is polynomially reducible to $B$.

Theorem
If $B$ is NP-complete and $B \leq_p C$ for $C \in \text{NP}$, then $C$ is NP-complete

Proof:
- initially, $C \in \text{NP}$
- we need to show: “every $A \in \text{NP}$ polynomially reduces to $C$”
  - every language in NP polynomially reduces to $B$
  - $B$ polynomially reduces to $C$
SAT ∈ NP-complete
Cook-Levin theorem

Theorem

$\text{SAT} \in \text{P} \ if \ and \ only \ if \ \text{P} = \text{NP}.$

equivalently: SAT is NP-complete.
**SAT ∈ NP-complete**

**SAT is in NP**

- given an assignment of variables, scan all clauses to check if they evaluate to TRUE

**A ≤_P SAT for every language A ∈ NP**

- \( M \): a Non-Deterministic Turing Machine that *decides* \( A \) in \( n^k \) time
- create a table of size \( n^k \times n^k \)
  - each row \( i \) corresponds to a configuration
    \( c_i = \#w_1w_2\ldots w_{\ell-1}qw_\ell\ldots w_r\# \)
  - the head is on \( w_\ell \)
  - \( c_i \vdash_M c_{i+1} \)
  - describes a branch of computation of \( M \)
- a table is **accepting** if any row is an accepting configuration
SAT ∈ NP-complete

<table>
<thead>
<tr>
<th></th>
<th>#</th>
<th>q₀</th>
<th>w₁</th>
<th>w₂</th>
<th>...</th>
<th>wₙ</th>
<th>□</th>
<th>...</th>
<th>□</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>#</td>
<td></td>
<td></td>
<td>#</td>
</tr>
<tr>
<td>#</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>#</td>
<td></td>
<td></td>
<td>#</td>
</tr>
<tr>
<td>#</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>#</td>
<td></td>
<td></td>
<td>#</td>
</tr>
</tbody>
</table>

starting configuration
second configuration

window

nᵏ-th configuration
For each $i, j, s$, where $1 \leq i, j \leq n^k$ and $s \in \Gamma \cup K$, define a variable

$$x_{i,j,s} = \begin{cases} \text{TRUE} & \text{if the cell in row } i \text{ and column } j \text{ contains the symbol } s \\ \text{FALSE} & \text{otherwise} \end{cases}$$

Define clauses to guarantee the calculation of $M$

- there is exactly one symbol in each cell

$$\phi_{\text{cell}} = \bigwedge_{1 \leq i,j \leq n^k} \left[ \left( \bigvee_{s \in \Gamma \cup K} x_{i,j,s} \right) \wedge \left( \bigwedge_{s,t \in \Gamma \cup K, s \neq t} (\bar{x}_{i,j,s} \lor \bar{x}_{i,j,t}) \right) \right]$$
\textbf{SAT} \in \textbf{NP-complete}

\begin{itemize}
  \item the first row corresponds to the starting configuration
    \[
    \phi_{\text{start}} = x_{1,1},\# \land x_{1,2},q_0 \land \\
    x_{1,3},w_1 \land x_{1,4},w_2 \land \ldots \land x_{1,n+2},w_n \land \\
    x_{1,n+2,\sqcup} \land \ldots \land x_{1,n^k-1,\sqcup} \land x_{1,n^k},\#
    \]
  
  \item there is an accepting state
    \[
    \phi_{\text{accept}} = \bigvee_{1 \leq i,j \leq n^k} x_{i,j,yes}
    \]
\end{itemize}
**SAT ∈ NP-complete**

- every window is legal
  - example: legal configurations for
    \[ \Delta(q_1, a) = \{(q_1, b, \rightarrow)\} \text{ and } \Delta(q_1, b) = \{(q_2, c, \leftarrow), (q_2, a, \rightarrow)\} \]

\[
\begin{array}{c|c|c}
(a) & (b) & (c) \\
\hline
a & q_1 & b \\
q_2 & a & c \\
\hline
\end{array}
\]

\[
\begin{array}{c|c|c}
(d) & (e) & (f) \\
\hline
\# & b & a \\
\# & b & a \\
\hline
\end{array}
\]

- then,

\[
\phi_{\text{legal}}^{i,j} = \bigwedge_{a_1, \ldots, a_6} \left( x_{i,j-1, a_1} \land x_{i,j, a_2} \land x_{i,j+1, a_3} \land x_{i+1,j-1, a_4} \land x_{i+1,j, a_5} \land x_{i+1,j+1, a_6} \right)
\]

\[
\phi_{\text{move}} = \bigwedge_{1 \leq i, j \leq n^k} \phi_{\text{legal}}^{i,j}
\]
SAT $\in$ NP-complete

Construct $F = \phi_{\text{cell}} \land \phi_{\text{start}} \land \phi_{\text{accept}} \land \phi_{\text{move}}$

- $F$ has $n^{O(k)}$ variables and clauses

**Theorem:** $F$ is satisfiable if and only if $A$ is decided by $M$
$3\text{SAT} \in \text{NP-complete}$
3SAT ∈ NP-complete

3SAT Problem

- as SAT but each clause has at most 3 literals

How can we prove a problem A is NP-complete?

- show that the problem is NP
- find a suitable problem B that is NP-complete
- show that $B \leq_p A$
  - find a polynomial transformation that transforms each instance $I_B$ of B to an instance $I_A$ of A
  - prove that there is a solution for the problem B on the instance $I_B$ if and only if there is a solution for the problem A on the instance $I_A$. 
3SAT ∈ NP-complete

3SAT is in NP

- given an assignment of variables, scan all clauses to check if they evaluate to TRUE

SAT ≤_P 3SAT

Transformation: given any formula $F$ of SAT in CNF with $m$ clauses and $n$ variables, we construct a formula $F'$ of 3SAT:

- replace each clause $(a_1 \lor a_2 \lor \ldots \lor a_\ell)$ in $F$ with $\ell - 2$ clauses

$$
(a_1 \lor a_2 \lor z_1) \land (\bar{z}_1 \lor a_3 \lor z_2) \land (\bar{z}_2 \lor a_4 \lor z_3) \land \ldots \land (\bar{z}_{\ell-3} \lor a_{\ell-1} \lor a_\ell)
$$

1. Polynomiality: $F'$ has $O(nm)$ variables and clauses

2. $F$ is satisfiable iff $F'$ is satisfiable
3SAT ∈ NP-complete

SAT ≤ₚ 3SAT
Transformation: given any formula \( F \) of SAT in CNF with \( m \) clauses and \( n \) variables, we construct a formula \( F' \) of 3SAT:

\[
\text{replace each clause } (a_1 \lor a_2 \lor \ldots \lor a_\ell) \text{ in } F \text{ with } \ell - 2 \text{ clauses }
\]

\[
(a_1 \lor a_2 \lor z_1) \land (\bar{z}_1 \lor a_3 \lor z_2) \land (\bar{z}_2 \lor a_4 \lor z_3) \land \ldots \land (\bar{z}_{\ell-3} \lor a_{\ell-1} \lor a_\ell)
\]

Proving \( F \) is satisfiable iff \( F' \) is satisfiable

1. \( F' \) is satisfiable if \( F \) is satisfiable
   - assume that \( F \) is satisfiable
   - then some \( a_i \) is TRUE for all clauses
   - use the same assignment for the common variables of \( F \) and \( F' \)
   - set \( z_j = \text{TRUE} \) for \( 1 \leq j \leq i - 2 \)
   - set \( z_j = \text{FALSE} \) for \( i - 1 \leq j \leq \ell - 3 \)
   - all clauses of \( F' \) are satisfied

Example

\[
(a_1 \lor a_2 \lor z_1) \land (\bar{z}_1 \lor a_3 \lor z_2) \land (\bar{z}_2 \lor a_4 \lor z_3) \land (\bar{z}_3 \lor a_5 \lor a_6)
\]

\[
(F \lor F \lor z_1) \land (\bar{z}_1 \lor T \lor z_2) \land (\bar{z}_2 \lor F \lor z_3) \land (\bar{z}_3 \lor F \lor F')
\]

\[
(F \lor F \lor T) \land (F \lor T \lor F') \land (T \lor F \lor F') \land (T \lor F \lor F')
\]
**3SAT \in \text{NP-complete}**

\textbf{SAT} \leq_p \text{3SAT}

Transformation: given any formula $\mathcal{F}$ of SAT in CNF with $m$ clauses and $n$ variables, we construct a formula $\mathcal{F}'$ of 3SAT:

- replace each clause $(a_1 \lor a_2 \lor \ldots \lor a_\ell)$ in $\mathcal{F}$ with $\ell - 2$ clauses

\[
(a_1 \lor a_2 \lor z_1) \land (\overline{z_1} \lor a_3 \lor z_2) \land (\overline{z_2} \lor a_4 \lor z_3) \land \ldots \land (\overline{z_{\ell-3}} \lor a_{\ell-1} \lor a_\ell)
\]

Proving $\mathcal{F}$ is satisfiable iff $\mathcal{F}'$ is satisfiable

1. $\mathcal{F}'$ is satisfiable if $\mathcal{F}$ is satisfiable \checkmark
2. $\mathcal{F}$ is satisfiable if $\mathcal{F}'$ is satisfiable
   - assume that $\mathcal{F}'$ is satisfiable
   - at least one of the literals $a_i$ should be TRUE for each clause
   - if not, then $z_1$ should be TRUE which implies that $z_2$ should be TRUE, etc
   - hence, the clause $(\overline{z_{\ell-3}} \lor a_{\ell-1} \lor a_\ell)$ is not satisfiable, contradiction
   - then there is an assignment that satisfies $\mathcal{F}$
3SAT ∈ NP-COMPLETE

- 3SAT is in NP ✓
- give a transformation form SAT to 3SAT ✓
- it is polynomial ✓
- \( F \in SAT \) is satisfiable iff \( F' \in 3SAT \) is satisfiable ✓

\[ \Rightarrow 3SAT \in NP\text{-COMPLETE} \]
MAX-2SAT ∈ NP-complete
MAX-2SAT ∈ NP-complete

**MAX-2SAT** = \{⟨\mathcal{F}, k⟩ \mid \mathcal{F} \text{ is a formula with } k \text{ TRUE clauses}\}

**MAX-2SAT** is in NP

- given an assignment of variables, scan all clauses to check if there are at least \(k\) of them evaluated to TRUE

**3SAT** \(\leq_P** MAX-2SAT

1. given any formula \(\mathcal{F}\) of 3SAT, we construct a formula \(\mathcal{F}'\) of MAX-2SAT
   - replace each clause \((x \lor y \lor z)\) with
     \[(x) \land (y) \land (z) \land (\bar{x} \lor \bar{y}) \land (\bar{y} \lor \bar{z}) \land (\bar{z} \lor \bar{x}) \land (w) \land (\bar{w} \lor x) \land (\bar{w} \lor y) \land (\bar{w} \lor z)\]
   - \(k = 7m\) (\(m\) is the number of clauses)

2. \(\mathcal{F}'\) has \(O(n + m)\) variables and \(O(m)\) clauses
MAX-2SAT $\in$ NP-complete

3SAT $\leq_p$ MAX-2SAT

1. recall: replace each clause $(x \lor y \lor z)$ with
   
   $$(x) \land (y) \land (z) \land (\bar{x} \lor \bar{y}) \land (\bar{y} \lor \bar{z}) \land (\bar{z} \lor \bar{x}) \land (w) \land (\bar{w} \lor x) \land (\bar{w} \lor y) \land (\bar{w} \lor z)$$

3. $F$ is satisfiable iff $F'$ has at least $k$ satisfied clauses
   - assume that $F$ is satisfiable
   - if $x = T$, $y = F$ and $z = F$, then set $w = F$: 7 satisfied clauses
   - if $x = T$, $y = T$ and $z = F$, then set $w = F$: 7 satisfied clauses
   - if $x = T$, $y = T$ and $z = T$, then set $w = T$: 7 satisfied clauses
   - in all cases, there are 7 satisfied clauses in $F'$ for each clause of $F$

   - contrapositive: assume that $F$ is not satisfiable
   - there is one clause for which $x = y = z = F$
   - then, in $F'$ we correspondingly have:
     - 4 satisfied clauses if $w = T$
     - 6 satisfied clauses if $w = F$
   - hence, in $F'$ there are less than $k$ clauses that are satisfied
CLIQUE ∈ NP-complete
CLIQUE $\in$ NP-

CLIQUE is in NP

- given a set of vertices, check if there is an edge between any pair of them

3SAT $\leq_p$ CLIQUE

1. given any formula $\mathcal{F}$ of SAT, we construct an instance $I = \langle G, k \rangle$ of CLIQUE

   - add a vertex for each literal
   - add an edge between any two literals except:
     (a) literals in the same clause
     (b) a literal and its negation
   - $k = m$ (number of clauses)
   - example: $\mathcal{F} = (x_1 \lor x_2 \lor \overline{x}_3) \land (x_1 \lor x_3 \lor x_4) \land (\overline{x}_2 \lor x_3 \lor \overline{x}_4)$
CLIQUE $\in$ NP-complete

3SAT $\leq_P$ CLIQUE

2. $|V| = 3m$, $|E| = O(m^2)$

3. $\mathcal{F}$ is satisfiable iff there is a clique of size $k$ in $G$
   - assume that $\mathcal{F}$ is satisfiable
   - at least one literal is TRUE in any clause
   - there is an edge between such literals (why?)
   - hence, the corresponding vertices form a $k$-clique

   - assume there is a $k$-clique in $G$
     - this clique contains at most one vertex from each clause
     - $k = m$, hence the clique contains exactly one vertex from each clause
     - each pair of these vertices is compatible (no a literal and its negation)
     - set the corresponding literals to TRUE
     - $\mathcal{F}$ is satisfiable
Summarize: NP-COMPLETENESS proofs
1. Prove that the problem is in NP (give a verifier)

2. Give a polynomial time reduction from a known NP-complete problem
   ▶ important: choose the correct problem
NP-complete problems

SAT

3SAT

MAX-2SAT

CLIQUE

IndSet

VCover
Exercises

- Show that **Independent Set** is **NP-complete** by a reduction from 3-SAT or **Clique**, where

  \[
  \text{Independent Set} = \{ \langle G, k \rangle \mid G = (V, E) \text{ is a graph with a set } A \subseteq V \text{ such that } |A| = k \text{ and for each } x, y \in A \text{ with } x \neq y, \text{ it holds that } \{x, y\} \notin E \}\.
  \]

- Show that **Vertex Cover** is **NP-complete** by a reduction from 3-SAT, **Clique** or **Independent Set**, where

  \[
  \text{Vertex Cover} = \{ \langle G, k \rangle \mid G = (V, E) \text{ is a graph with a set } A \subseteq V \text{ such that } |A| = k \text{ and every } e \in E \text{ is incident to a vertex in } A \}\.
  \]

- Show that **3-Coloring** is **NP-complete** by a reduction from 3-SAT where

  \[
  \text{3-Coloring} = \{ \langle G, k \rangle \mid G = (V, E) \text{ is a graph and there exists a function } f : V \to \{1, 2, 3\} \text{ such that for every edge } \{u, v\} \in E \text{ we have } f(u) \neq f(v) \}\.
  \]