NP-completeness

1. The complexity class P
   a. Instance encodings
   b. Formal languages

2. The complexity class NP
   a. NP-completeness and reducibility
   b. proofs of NP-completeness

3. Some NP-complete Problems
   a. SAT
   b. 3-SAT
   c. CLIQUE
   d. PARTITION
   e. $k$-COLORING
   f. 3D-MATCHING
1a. The complexity class P

An abstract problem is a relation on the sets of problem instances and problem solutions.

**Example 1** The **shortest path** problem:
Instance: A simple graph $G$ and two vertices $u, v$.
Output: A shortest path $u \sim v$ in $G$ (it such exists).

**Decision problems:**
A solution is of the form “Y” or “N”.

**Example 2**
Instance: A simple graph $G$, two vertices $u, v$, and $k > 0$.
Question: $\exists u \sim v$ in $G$ of length $\leq k$?

**Optimization problems:**
Some function should be minimized or maximized.

Any “discrete optimization problem” can be formulated as a decision problem.

**Remark 1**
Optimization problem is “easily solvable” $\Rightarrow$ corresponding decision problem is also “easily solvable”.

Optimization problem is “hardly solvable” $\Rightarrow$ corresponding decision problem is also “hardly solvable”.
An encoding is a mapping of the set of abstract object into the set of binary strings.

Any algorithm that “solves” an abstract decision problem works with an encoding of this problem. We call a problem with encoded instance concrete problem.

**Definition 1** We say that an algorithm solves a problem in time $O(T(n))$ if for any instance encoding of length $n$ the algorithm computes a solution in time $O(T(n))$.

**Definition 2** A concrete problem with instance encoding of size $n$ is solvable in polynomial time, if there exists an algorithm for solving the problem in time $O(n^k)$ for some constant $k$ (independent on $n$).

**Definition 3** The complexity class $P$ consists of the concrete decision problems solvable in polynomial time.

Let $f$ be a mapping $f : \{0, 1\}^* \mapsto \{0, 1\}^*$. $f$ is said to be computable in polynomial time, if there exists an algorithm that for any $x \in \{0, 1\}^*$ constructs a sequence $f(x)$ in polynomial time.

Let $I$ be the set of all problem instances. We call two encodings $e_1$ and $e_2$ polynomially equivalent, if there exist computable in polynomial time functions $f_{12}$ and $f_{21}$ such that for any $i \in I$ one has: $f_{12}(e_1(i)) = e_2(i)$ and $f_{21}(e_2(i)) = e_1(i)$.

**Lemma 1** Let $Q$ be an abstract decision problem and $e_1, e_2$ be polynomially equivalent encodings of the set $I = \{i\}$. Then $Q(e_1(i)) \in P \iff Q(e_2(i)) \in P.$
1b. A formal language framework

Let $\Sigma = \{0, 1\}$. A language $L$ is a subset of $\Sigma^*$. Any decision problem $Q$ can be represented as the following language:

$$L = \{x \in \Sigma^* \mid Q(x) = 1\}.$$

**Definition 4** An algorithm $A$ accepts $x \in \Sigma^*$, if its output $A(x) = 1$. The algorithm $A$ rejects a string $x \in \Sigma^*$ if $A(x) = 0$.

The set $L = \{x \in \Sigma^* \mid A(x) = 1\}$ is the language accepted by algorithm $A$.

A language $L$ is decided by an algorithm $A$ if for any $x \in \Sigma^*$ either $A$ accepts $x$ or $A$ rejects $x$.

**Definition 5** A language $L$ is accepted by algorithm $A$ in polynomial time, if any $x \in L$ with $|x| = n$ is accepted by $A$ in time $O(n^k)$.

The language $L$ is decided in polynomial time by an algorithm $A$, if any $x \in \Sigma^*$ with $|x| = n$ is decided by $A$ in time $O(n^k)$.

Further definitions for the class $P$:

$$P = \{L \subseteq \Sigma^* \mid \exists A \text{ which decides } L \text{ in polynomial time}\}.$$  

**Theorem 1** (Theorem 34.2, p.977)

$$P = \{L \subseteq \Sigma^* \mid L \text{ is accepted by a polyn.-time algorithm}\}.$$
2a. The complexity class NP

Let $x$ and $y$ be binary strings.

**Definition 6** A verification algorithm is an algorithm with two parameters. We say $A$ verifies a string $x$ if $\exists y$ such that $A(x, y) = 1$.

An algorithm $A$ verifies a language $L$ if:

$$L = \{ x \in \Sigma^* \ | \ \exists y \in \Sigma^* \text{ with } A(x, y) = 1 \}.$$

**Definition 7** The complexity class NP is the set of all languages $L$ for which there exists a polynomial-time verification algorithm $A$ and a constant $c$ such that:

$$L = \{ x \in \Sigma^* \ | \ \exists y \text{ with } |y| = O(|x|^c), \ A(x, y) = 1 \}.$$

Obviously, $P \subseteq NP$. The principal question is whether $P \neq NP$.

**Definition 8** A language $L_1$ is called polynomial-time reducible to a language $L_2$ if there exists a polynomial-time computable function $f : \Sigma^* \mapsto \Sigma^*$ such that for all $x \in \Sigma^*$ one has:

$$x \in L_1 \iff f(x) \in L_2.$$

(denotation $L_1 \leq_P L_2$).

We write $L_1 \equiv L_2$ if $L_1 \leq_P L_2$ and $L_2 \leq_P L_1$. 
Lemma 2 Let $L_1, L_2 \subseteq \Sigma^*$ be languages and $L_1 \leq_P L_2$. $L_2 \in P$ implies $L_1 \in P$.

Definition 9 Let $L \subseteq \Sigma^*$ be a language.

1. $L$ is called **NP-hard** if $L' \leq_P L$ for any language $L' \in NP$.

2. The language $L$ is called **NP-complete** if $L$ is NP-hard and $L \in NP$ (denotation $L \in NPC$).

Theorem 2

1. If some NP-complete problem is solvable in polynomial time then $P=NP$.

2. If some problem of NP is not solvable in polynomial time then no other NP-complete problem is solvable in polynomial time.

Proof.

1. Let $L \in NPC$ and $L \in P$.
   $\Rightarrow L' \leq_P L$ for any problem $L' \in NP$ (Definition 9).
   $\Rightarrow L' \in P$ (Lemma 2).

2. Assume $\exists L \in NP$ with $L \notin P$.
   Let $L' \in NPC$. $\Rightarrow L \leq_P L'$ (Definition 9).
   Now if $L' \in P$ then $L \in P$ (Lemma 2), a contradiction. $\Box$
2b. Proofs of NP-completeness

**Lemma 3** Let $L$ be a language such that $L' \leq_P L$ for some $L' \in \text{NPC}$. Then $L$ is NP-hard. If additionally, $L \in \text{NP}$, then $L \in \text{NPC}$.

*Proof.* $L' \in \text{NPC} \Rightarrow L'' \leq_P L'$ for any $L'' \in \text{NP}$. Furthermore, since $L' \leq_P L \Rightarrow L'' \leq_P L \Rightarrow L$ is NP-hard. $\Rightarrow L \in \text{NPC}$ if $L \in \text{NP}$.

To prove NP-completeness:

1. Show: $L \in \text{NP}$.
2. Choose an appropriate language $L'$ (problem) for which it is know that it is NP-complete.
3. Design an algorithm that computes a function $f$ mapping every instance $x$ of $L'$ to an instance $f(x)$ for $L$.
4. Prove that $x \in L' \iff f(x) \in L$ for any $x \in \{0, 1\}^*$.
5. Show that the function $f$ is polynomial-time computable.

**Theorem 3** (Cook).

*One has:*

\[ \text{SAT} \in \text{NPC}. \]
3. Some NP-complete problems

Satisfiability (SAT):
Instance: Boolean formula $F$.
Question: Is $F$ satisfiable?

3-SAT:
Similar to SAT, but each clause in the formula has 3 literals.

Clique:
Instance: Graph $G$ and $k \in \mathbb{N}$.
Question: Does $G$ contain a $k$-clique?

Ham-Cycle (HC):
Instance: Graph $G = (V, E)$.
Question: Does $G$ contain a simple cycle of length $|V|$?

Vertex-Cover (VC):
Instance: Graph $G$ and $k \in \mathbb{N}$.
Question: Is there a set $C \subseteq V$ of size $k$ such that any edge of $G$ is incident to some vertex of $C$?
Theorem 4 \(3\text{-SAT} \in \text{NPC}\).

Proof. We show \(\text{SAT} \leq_p 3\text{-SAT}\).

Given a boolean formula \(f\) (instance for \(\text{SAT}\)), we construct an instance \(f'\) for \(3\text{-SAT}\).

Step 1. For any “internal subformula” we create a new variable \(y_i\).

Example 3

\[
f = ((x_1 \rightarrow x_2) \lor \neg((\neg x_1 \leftrightarrow x_3) \lor x_4)) \land \neg x_2
\]

\[
f' = y_1 \land (y_1 \leftrightarrow (y_2 \land \neg x_2)) \\
\land (y_2 \leftrightarrow (y_3 \lor y_4)) \\
\land (y_3 \leftrightarrow (x_1 \rightarrow x_2)) \\
\land (y_4 \leftrightarrow \neg y_5) \\
\land (y_5 \leftrightarrow (y_6 \lor x_4)) \\
\land (y_6 \leftrightarrow (\neg x_1 \leftrightarrow x_3))
\]
Step 2. Write every clause $C_i$ in $f'$ in CNF and obtain a formula $f''$.

<table>
<thead>
<tr>
<th>$y_1$</th>
<th>$y_2$</th>
<th>$x_2$</th>
<th>$(y_1 \leftrightarrow (y_2 \land \neg x_2))$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

$C_i = (\neg y_1 \lor \neg y_2 \lor \neg x_2) \land (\neg y_1 \lor y_2 \lor \neg x_2) \land (\neg y_1 \lor y_2 \lor x_2) \land (y_1 \lor \neg y_2 \lor x_2)$.

Step 3. Expand every clause $C_i$ in $f''$ to make it depending on exactly 3 variables and obtain a formula $f'''$.

- $C_i = l_1 \lor l_2 \lor l_3 \Rightarrow C'_i := C_i \in f'''$.
- $C_i = l_1 \lor l_2 \Rightarrow C''_i := (l_1 \lor l_2 \lor p) \land (l_1 \lor l_2 \lor \neg p)$.
- $C_i = l \Rightarrow C''_i := (l \lor p \lor q) \land (l \lor \neg p \lor q) \land (l \lor p \lor \neg q) \land (l \lor \neg p \lor \neg q)$.

The formula $f$ is satisfiable $\iff$ $f'''$ is satisfiable.

The formula $f'''$ is constructible in polynomial time.

Therefore, 3-SAT $\in$ NP.  \qed
**Clique:**

**Instance:** A graph $G$ and $k \in \mathbb{N}$.

**Question:** Does $G$ contain a clique of size $k$?

**Theorem 5**  \(\text{Clique} \in \text{NPC}\).

**Proof.** Obviously, \(\text{Clique} \in \text{NP}\).

We show: \(3\text{-SAT} \leq_p \text{Clique}\). Let \(f = C_1 \land C_2 \land \cdots \land C_k\) be an instance for \(3\text{-SAT}\) with \(C_i = l_i^1 \lor l_i^2 \lor l_i^3\).

Construct a graph \(G = (V, E)\) with \(V = \{v_i^1, v_i^2, v_i^3 \mid i = 1, \ldots, k\}\) and \((v_i^r, v_j^s) \in E\) iff \(i \neq j\) and \(l_i^r \neq \neg l_j^s\).

**Example:**

\[
f = (x_1 \lor \neg x_2 \lor \neg x_3) \land (\neg x_1 \lor x_2 \lor x_3) \land (x_1 \lor x_2 \lor x_3).
\]

The formula \(f\) is satisfiable \(\iff\) \(G\) contains a clique with \(k\) vertices.

The graph \(G\) is constructible in polynomial time. \(\Box\)
**Vertex Cover (VC):**

**Instance:** A graph $G = (V, E)$ and $k \in \mathbb{N}$.

**Question:** Is there a subset $C \subset V$ with $|C| = k$ s.t. each edge of $G$ is incident to some vertex of $C$?

**Theorem 6** \( VC \in \text{NPC} \).

**Proof.** Obviously, $VC \in \text{NP}$.

We show: $\text{CLIQUE} \leq_P VC$.

For a graph $G = (V, E)$ we define its complement $\overline{G} = (V, \overline{E})$.

Then $G$ has a clique of size $k$ iff $\overline{G}$ has a VC of size $|V| - k$.

Indeed:
If $G$ has a $k$-clique $V' \subset V$ then $V \setminus V'$ is a VC.

On the other hand, if $\overline{G}$ has a VC $V'$ of size $|V'| = |V| - k$, then
\[ \forall u, v \in V \text{ if } (u, v) \in \overline{E} \text{ then } u \in V' \text{ or } v \in V'. \]

The contraposition of this implication is:
\[ \forall u, v \in V \text{ if } u \notin V' \text{ and } v \notin V' \text{ then } (u, v) \in E. \]

In other words, $V \setminus V'$ is a clique. \( \square \)
**Partition:**

**Instance:** A set $S = \{s\}$ of integers and $t \in \mathbb{IN}$.

**Question:** Is there a subset $S' \subseteq S$ with $\sum_{s \in S'} s = t$?

**Theorem 7**  

**Partition** $\in$ **NPC**.

**Proof.** Obviously, **Partition** $\in$ **NP**. We show: $\text{VC} \leq_P \text{Partition}$.

Let $G = (V, E)$ be an instance for **VC** with

$V = \{v_0, \ldots, v_{n-1}\}$ and $E = \{e_0, \ldots, e_{m-1}\}$.

We represent $G$ by its $n \times m$ incidence matrix $B = \{b_{ij}\}$, where

$$b_{ij} = \begin{cases} 1, & \text{if } e_i \text{ is incident to } v_j \\ 0, & \text{otherwise} \end{cases}$$

<table>
<thead>
<tr>
<th>$V$</th>
<th>$e_0$</th>
<th>$e_1$</th>
<th>$v_0$</th>
<th>$v_1$</th>
<th>$v_2$</th>
<th>$v_3$</th>
<th>$v_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_0$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$v_1$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$v_2$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$v_3$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$v_4$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

$B = \begin{bmatrix} e_4 & e_3 & e_2 & e_1 & e_0 \end{bmatrix}$

$\begin{bmatrix} e_0 & e_1 & e_2 & e_3 & e_4 \end{bmatrix}$
Example 4

For $i = 0, \ldots, n-1$ and $j = 0, \ldots, m-1$ put:

$$x_i = 4^m + \sum_{j=0}^{m-1} b_{ij}4^j, \quad y_j = 4^j, \quad t = k \cdot 4^m + \sum_{j=0}^{m-1} 2 \cdot 4^j$$

and extend the matrix $B$:

<table>
<thead>
<tr>
<th></th>
<th>$e_0$</th>
<th>$e_1$</th>
<th>$e_2$</th>
<th>$e_3$</th>
<th>$e_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_0$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$v_1$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$v_2$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$v_3$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$v_4$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

It holds: $G$ has vertex cover of size $k \iff \exists S' \subseteq S$ with $\sum_{s \in S'} s = t$. \qed
**3-Coloring:**

**Instance:** A graph $G = (V, E)$.

**Question:** Is $G$ 3-colorable?

**Theorem 8** 3-Coloring $\in$ NPC.

**Proof.** Obviously, 3-Coloring $\in$ NP.

We show: 3-SAT $\leq_P$ 3-Coloring.

Let $f = C_1 \land C_2 \land \cdots \land C_m$ (here $f = f(x_1, \ldots, x_n)$) be an instance for 3-SAT. We construct for every clause $C_i = l_1 \lor l_2 \lor l_3$ a graph $G_i = (V_i, E_i), 1 \leq i \leq m$:

Assume the vertices $l_1, l_2, l_3$ are colored with color 0 or 1. Then $v_6$ can be colored with color 1 or 2 $\iff$ $\exists l_i, 1 \leq i \leq 3$ colored with 1.

We construct an instance $G = (V, E)$ for 3-Coloring:

$$V = \{a, b\} \cup_{i=1}^{m} V_i$$

$$E = \{(a, b)\} \cup \{(a, x_i), (a, \overline{x_i}), (x_i, \overline{x_i}) \mid 1 \leq i \leq n\} \cup_{i=1}^{m} E_i.$$

It holds: $f$ is satisfiable $\iff$ $G$ is 3-colorable. \qed