3-SAT

Theorem. 3-SAT is NP-complete.

Proof. First, 3-SAT ∈ NP, since we can non-deterministically “guess” assignment and verify that it satisfies the given 3-CNF-formula.

Second, let

\[ \phi = (l_{1,1} \lor l_{1,2} \lor \ldots \lor l_{1,i_1}) \land (l_{2,1} \lor l_{2,2} \lor \ldots \lor l_{2,i_2}) \land \ldots \land (l_{k,1} \lor l_{k,2} \lor \ldots \lor l_{k,i_k}), \]

where \( l_{i,j} \) are literals (can contain “not” sign). We convert each clause into “and” of clauses with 3 literals, as follows:

\[ \varphi_j = (l_{j,1} \lor l_{j,2} \lor \ldots \lor l_{j,i_j}) \]

is converted into

\[ \hat{\varphi}_j = (l_{j,1} \lor l_{j,2} \lor z_1) \land (\bar{z}_1 \lor l_{j,3} \lor z_2) \land (\bar{z}_2 \lor l_{j,4} \lor z_3) \land \ldots \land (\bar{z}_{i_j-3} \lor l_{i_j-1} \lor l_{i_j}), \]

where \( z_1, z_2, \ldots, z_{i_j-3} \) are new Boolean variables. Then \( f(\phi) = \hat{\varphi}_1 \land \hat{\varphi}_2 \land \ldots \land \hat{\varphi}_k. \)

Correctness. Prove that \( \phi \in SAT \) if and only if \( f(\phi) \in 3\text{-SAT}. \) In other words, \( \phi \) is satisfiable if and only if \( f(\phi) \) is satisfiable.

Direction 1) Let \( \phi \) be satisfiable, and consider assignment to literals that satisfy \( \phi. \) Consider clause \( \varphi_j = (l_{j,1} \lor l_{j,2} \lor \ldots \lor l_{j,i_j}). \) There exists \( l_{j,r} \) in that
assignment such that \( l_{j,r} = \text{TRUE} \). Then we choose assignments to \( z_1, \ldots, z_{i_j-3} \) such that all clauses with three variables are satisfied:

\[
\begin{array}{cccc}
T & F & T & F \\
(l_{j,1} \lor l_{j,2} \lor z_1) \land (\bar{z}_1 \lor l_{j,3} \lor z_2) \land (\bar{z}_2 \lor l_{j,4} \lor z_3) \land \\
F & T & F & T \\
\ldots \land (\bar{z}_{r-2} \lor l_{j,r} \lor z_{r-1}) \land \ldots \land (\bar{z}_{i_j-3} \lor l_{j,i_j-1} \lor l_{i_j}).
\end{array}
\]

Here T and F stand for \text{TRUE} and \text{FALSE}, respectively.

Direction 2) Let \( f(\phi) \) be satisfiable, and consider assignment to \( l_{j,r} \) and \( z_r \) that satisfy \( f(\phi) \). Then, the same assignment to \( l_{j,r} \) satisfies \( \phi \) because there are \( i_j - 2 \) clauses with three variables, but only \( i_j - 3 \) variables \( z_r \) can be \text{TRUE}. So at least one of the clauses is satisfied with \( l_{j,r} = \text{TRUE} \).

Polynomiality. \( f(\phi) \) can be constructed from \( \phi \) in time polynomial in length of \( \phi \).

\[
\begin{proof}
\end{proof}
\]

**Practise session**

1. **Definition:** given an undirected graph \( G \), independent set \( S \) is a subset of vertices of \( G \), such that for any two vertices \( u, v \in S \), there is no edge between \( u \) and \( v \) in \( G \).

Define the language \text{INDEPENDENT-SET}:

\[
\text{INDEPENDENT-SET} = \left\{ \langle G, k \rangle \mid G \text{ is an undirected graph, \hspace{1cm} which has an independent set with } k \text{ vertices} \right\}.
\]

In this exercise, we will show that \text{INDEPENDENT-SET} is \( \mathcal{NP} \)-complete.

(a) Show that \text{INDEPENDENT-SET} \in \mathcal{NP}.

(b) Show that \text{INDEPENDENT-SET} is \( \mathcal{NP} \)-hard.

\[
\begin{proof}
\end{proof}
\]

(a) We use a non-deterministic algorithm, which chooses non-deterministically a set \( S \) of \( k \) vertices. Then it checks that there is no edge between any two vertices in that set \( S \). If all checks are satisfied – accepts, otherwise – rejects.

The checks are performed in polynomial time, in particular to check that there are no edges between any two vertices in \( S \) requires at most
\(O(n^2)\) steps, where \(n\) is the number of vertices in \(G\) (here, \(O(n^2)\) is an upper bound on the number of pairs of vertices). Consequently, this non-deterministic algorithm is polynomial.

(b) Recall the language CLIQUE defined as follows:

\[
\text{CLIQUE} = \{ \langle G, k \rangle \mid G \text{ is an undirected graph, which has a clique of size } k \}.
\]

It was shown in the class that CLIQUE is \(\mathcal{NP}\)-complete.

Define a function \(f\) that maps the pair \(\langle G, k \rangle\) to the pair \(\langle \overline{G}, k \rangle\), where \(\overline{G}\) is a complement of the graph \(G\). Specifically, there is an edge between any two vertices \(u\) and \(v\) in \(G\) if and only if there is no edge between \(u\) and \(v\) in \(\overline{G}\). The function \(f\) can be computed in time polynomial with respect to the number of vertices and edges in the graph \(G\), because we need to look at all vertex pairs in the graph \(G\) (there are polynomial number of them) and reverse the presence/absence of an edge in each pair (can be done in polynomial time).

If \(\langle G, k \rangle \in \text{CLIQUE}\), then the graph \(G\) contains a click \(S\) with \(k\) vertices. There are edges between any two vertices of \(S\) in \(G\). Then, there are no edges between any two vertices of \(S\) in \(\overline{G}\), and thus \(S\) forms an independent set with \(k\) vertices in \(\overline{G}\). Consequently, \(\langle \overline{G}, k \rangle \in \text{INDEPENDENT-SET}\). Similarly, we get that if \(\langle \overline{G}, k \rangle \in \text{INDEPENDENT-SET}\), then \(\langle G, k \rangle \in \text{CLIQUE}\) (all logical transitions are invertible).

Polynomiality: we see that \(f\) is a polynomial-time reduction from the language CLIQUE to the language INDEPENDENT-SET.

Since CLIQUE is \(\mathcal{NP}\)-complete, and CLIQUE \(\leq_P\) INDEPENDENT-SET, we conclude that INDEPENDENT-SET is \(\mathcal{NP}\)-complete.

2. **Definition:** given an undirected graph \(G\), vertex cover \(C\) is a subset of vertices of \(G\) such that any edge \(e\) in \(G\) has at least one of its endpoints in \(C\).

Define the language VERTEX-COVER:

\[
\text{VERTEX-COVER} = \{ \langle G, k \rangle \mid G \text{ is an undirected graph, which has a vertex cover on } k \text{ vertices} \}.
\]

In this exercise, we will show that VERTEX-COVER is \(\mathcal{NP}\)-complete.

(a) Show that VERTEX-COVER \(\in \mathcal{NP}\).
(b) Show that VERTEX-COVER is $\mathcal{NP}$-hard.

Solution.

(a) Similar to the previous exercise, we use a non-deterministic algorithm, which chooses non-deterministically a set $C$ of $k$ vertices. Then it checks that every edge in $G$ has an endpoint in $C$. If all checks are satisfied – accepts, otherwise – rejects.

The checks are performed in polynomial time. This requires at most $O(m)$ steps, where $m$ is the number of edges in $G$. Consequently, this non-deterministic algorithm is polynomial.

(b) In the previous exercise we showed that INDEPENDENT-SET is $\mathcal{NP}$-complete. In this exercise, we use reduction from INDEPENDENT-SET to VERTEX-COVER.

Define a function $f$ that maps the pair $\langle G, k \rangle$ to the pair $\langle G, n - k \rangle$, where $n$ is the number of vertices in $G$. The function $f$ can be computed in time polynomial with respect to the number of vertices and edges in the graph $G$, because we need only to replace $k$ by $n - k$.

Take a set $S$ with $k$ vertices in $G$. Denote by $C$ a set of all vertices of $G$, which are not in $S$. Its size is $n - k$. Next, we show that $S$ is an independent set if and only if $C$ is a vertex cover.

1. Let $S$ be an independent set. Consider an arbitrary edge $e$ connecting two vertices $u$ and $v$ in $G$. It is impossible that both $u, v \in S$ (since $S$ is an independent set). Thus, at least one of the vertices $u, v$ is in $C$, and the edge $e$ is “covered” by $C$. Thus, $C$ is a vertex cover.

2. Let $C$ be a vertex cover. Then, every edge in $G$ is “covered” by $C$. Take an edge $e$ in $G$. At least one of its endpoints is in $C$, and therefore at most one of its endpoints is in $S$ (since $S$ and $C$ complement each other). Thus, every two vertices in $S$ are not connected by an edge, and therefore $S$ is an independent set.

We conclude that $\langle G, k \rangle \in \text{INDEPENDENT-SET}$ if and only if $\langle G, n - k \rangle \in \text{VERTEX-COVER}$.

Polynomiality: we see that $f$ is a polynomial-time reduction from the language $\text{INDEPENDENT-SET}$ to the language $\text{VERTEX-COVER}$.

Since $\text{INDEPENDENT-SET}$ is $\mathcal{NP}$-complete, and $\text{INDEPENDENT-SET} \leq_p \text{VERTEX-COVER}$, we conclude that $\text{VERTEX-COVER}$ is $\mathcal{NP}$-complete.