Examples of problems in NP:

**SATisfiability:** INST: A Boolean formula \( \phi(x_1, \ldots, x_n) \)

EQ: \((x_1 \lor \overline{x_2} \lor \overline{x_3}) \land (\overline{x_1} \lor x_2 \lor x_3)\)  \(\overline{x_i} = \neg x_i\).

**Question:** Is there a truth assignment \( \vec{a} \in \{0, 1\}^n \) such that \( \phi(\vec{a}) = \text{true} \)? I.e., \( \vec{a} \) satisfies \( \phi \).

In the example, any assignment \( a_1 a_2 a_3 \) with \( a_3 = 1 \) works. So do \( a = 110 \) and \( 000 \). But not \( 010 \) or \( 100 \).

**3SAT** is the special case where \( \phi = C_1 \land \cdots \land C_m \)

and each clause \( C_j \) is a disjunction of literals \( x_i \) or \( \overline{x_i} \), up to three.

SAT and 3SAT belong to NP. Design a verifier \( V \) that takes both \( \phi \) and \( \vec{a} \) as inputs.

\[ V(\phi, \vec{a}) = 1 \text{ if } \phi(\vec{a}) = 1 \]

reject otherwise.

An NTM \( N \), given only \( \phi \), can guess an assignment \( \vec{a} \) that works and then run \( V(\phi, \vec{a}) \) to verify. Then \( N \), too, runs in \( O(\text{ntm}) \) time, which is linear, hence polynomial, time.

**We can evaluate a Boolean formula \( \phi \) on an input given assignment \( \vec{a} \) quickly - in \( O(\text{ntm}) \) time in the case of 3SAT.**
The complement \( \neg \Phi \) is (essentially) \( \neg (\Phi) = \phi \) is not satisfiable.

\( \phi \) is not satisfiable \( \iff \neg \phi \) is a tautology.

\[ \text{Taut} = \{ \Phi' : \Phi' \text{ is a tautology} \} \]

2. **Graph 3-coloring:**

**INSTANCE:** \( G = (V, E) \)

**QUESTIONS:** Can you assign colors \( 1, 2, 3 \) to each node so that \( G \) has no monochromatic edges?

3. **FACTORING:**

**INSTANCE:** A number \( N \) and another number \( k \in \mathbb{N} \).

**QUESTIONS:** Do \( N \) have a prime factor \( p \) s.t. \( p < k \)?

**YES case:** Verify by guessing the unique prime factorized \( p \) of \( N \) and seeing if \( p \) is part of it and \( p < k \).

**NO case:** Pits! Verify no prime in the factorization is \( \leq k \).

**FACT:** That \( p_1 \cdots p_b = N \) can be verified quickly, and that each \( p_i \) is prime.

**FACTORIZATION is in \text{NP and co-NP}.**

**Defn:** A language \( A \) **many-one reduces** to a language \( B \) in *polynomial time*, written \( A \leq^p_m B \), if there is a function \( f : \Sigma^* \rightarrow \Sigma^* \) that is computable in polynomial time s.t.

\[ f(x) = \begin{cases} \text{if } x \in A & \text{then } f(x) \in B. \end{cases} \]

**Theorem:** If \( A \leq^p_m B \) and \( B \in \text{P} \), then \( A \in \text{P} \).

**Corollary:** If \( A \leq^p_m B \) and \( A \in \text{P} \), then \( B \in \text{P} \).

**If \( A \) is \text{NP}-hard and \( A \leq^p_m B \), then \( B \) is \text{NP}-hard.**

**Time is \( O((\log k)^b) = O((\log k)^b) \).**
Defn: A language $B$ is \textit{NP-hard} (under $\leq^P_m$) if for all $A \in \text{NP}$, $A \leq^P_m B$. (under $\leq^P_m$ always understood).

It also $B \in \text{NP}$, then $B$ is \textit{NP-complete}.

\textbf{Cook-Levin Theorem:} SAT and 3SAT are NP-complete.

Proof: We've shown $\exists$SAT $\in$ NP, let any $A \in \text{NP}$ be given. Take a $\text{det}^{O(n)}$-time verifier $V$ that recognizes $\{\langle x, y \rangle : y$ is a witness for $x \in A\}$. By the principle that SAT can be reduced to hard when there is a poly-sized circuit $C_n$ of NAND gates $\text{st.}$

$C_n(x_i) = 1 \iff V(x_{i+1}) = 1$

And we can build $C_n$ in $\text{NP}$ time, $(n+p(n))\text{poly}(n)$.

Let $x = (10011\ldots0)$, say.

If $x = (10011\ldots0)$, say

$3\text{(uuvw)} \land (vuvw) \land (uvwv) \text{ is satisfied.}$

If $x = (10011\ldots0)$, say

Then $x \in A \Rightarrow \exists y \phi(x, y) = \text{true}$

$\exists y \phi(x, y) = \text{true} \Rightarrow \exists y \phi_x(y) = \text{true} \Rightarrow \phi_x \in \text{the language SAT, indeed 3SAT.}$

So. $A \leq^P_m (3\text{SAT}) \text{ via } f$, where $f(x)$ is computable in $\text{poly}$ time.
Furthermore, 3SAT \leq_m^{0 \text{ P}} 3\text{COLORING}, so 3\text{COLORING} is also NP-hard and NP-complete.

For clauses \( C_1 = (x_1 \lor x_2 \lor \overline{x_3}) \) we would use a bit to go back one to the previous state.

Given any \( \phi \) instance of 3SAT, we can build \( N_\phi \) in \( \text{POLY}(n) \) time.

If \( \phi \in \text{SAT} \) then there is some \( \alpha \) that satisfies \( \phi \), which makes \( N_\phi \) go dead on all branches, so \( \alpha \in L(N_\phi) \), so \( L(N_\phi) \in \text{ALL-NFA} \).

But if \( \phi \notin \text{SAT} \), then every \( \alpha \) makes at least one clause not satisfied, so \( \alpha \notin L(N_\phi) \).

We can build \( N_\phi \) for any \( \phi \) in Conjunctive Normal Form (CNF) by this means in linear time because the arcs directly translate the clauses one-by-one.

Because FACT is in NP \& coNP, if it were NP-complete then we would get NP = coNP. Not quite as far from P as BQP, but close. Unlike BPP \& coBPP = \text{DEC}, NP \& coNP = \text{PSPACE} is not believed to exist. Quantum computers can solve FACT in Bounded-error Quantum Polynomial time (BQP).