NP-Completeness

Thm 7.27 [Cook-Levin]: SAT is in P iff P = NP.
Def 7.29: Language A is **polynomial-time reducible** to language B, written $A \leq_p B$, if a polynomial-time computable function $f: \Sigma^* \rightarrow \Sigma^*$ exists such that for every $w$,

$$w \in A \text{ iff } f(w) \in B$$

The function $f$ is called **polynomial-time reduction** of A to B.

Thm 7.31: If $A \leq_p B$ and $B \in P$, then $A \in P$. 

**def** fun for A .

**(poly-time)**

preprocessing

call fun for B (...) 

return ... 

if B poly-time, then this is a poly-time algo for A
Thm 7.32: 3SAT is polynomial-time reducible to CLIQUE, where

\[ 3SAT = \{ <\phi> \mid \phi \text{ is a satisfiable } 3\text{-cnf formula} \} . \]
Def 7.34: A language $B$ is **NP-complete** if it satisfies both conditions:

- $B$ is in NP, and
- every $A$ in NP is polynomial-time reducible to $B$. 

**Assuming $P \neq NP$**
NP-Completeness

Def 7.34: A language $B$ is **NP-complete** if it satisfies both conditions:
- $B$ is in NP, and
- every $A$ in NP is polynomial-time reducible to $B$.

Thm 7.35: If $B$ is NP-complete and $B \in P$, then $P = NP$.

If we find a poly-time algo for $B$, then every $A \in NP$ has a poly-time algo (by using $B$)

$\Rightarrow P = NP$
Def 7.34: A language $B$ is **NP-complete** if it satisfies both conditions:

- $B$ is in NP, and
- every $A$ in NP is polynomial-time reducible to $B$.

Thm 7.36: If $B$ is NP-complete and $B \leq_P C$ for some $C \in \text{NP}$, then $C$ is NP-complete.
Def 7.34: A language $B$ is **NP-complete** if it satisfies both conditions:
- $B$ is in NP, and
- every $A$ in NP is polynomial-time reducible to $B$.

Thm 7.37 [Cook-Levin]: SAT is NP-complete.

Note: a long list of known NP-complete problems.