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 $w$,

$$w \in A \iff f(w) \in B$$

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 $A \in P$. 

If we have a poly-time algo for $B$, then the box is a poly-time algo for $A$. 

---
Thm: HAMPATH is polynomial-time reducible to LONGESTPATH, where

LONGESTPATH = \{ <G,s,t,k> \mid G \text{ is digraph and there exists a path from } s \text{ to } t \text{ of length } \geq k \}.

Thm: HAMPATH \leq_p LONGESTPATH
**Thm:** HAMCYCLE is polynomial-time reducible to TSP, where

HAMCYCLE = \{ <G> | G is a graph that contains a cycle through all vertices \}

TSP = \{ <G_w,k> | G_w is a complete weighted graph that contains a cycle through all vertices of length \( \leq k \) \}

\[ G_w \rightarrow _{f} G \rightarrow _{h} <G_w,k> \\
\text{edges in } G \rightarrow \text{weights of } \text{other edges } \# \text{vertices} + 1 \]

\[ k = \# \text{vertices} \]
Thm 7.32: 3SAT is polynomial-time reducible to CLIQUE, where

\[ 3SAT = \{ \langle \phi \rangle \mid \phi \text{ is a satisfiable 3-cnf formula} \} \].

Example:

\[ \begin{align*} & (x_1 \lor x_2 \lor \neg x_3) \land (x_1 \lor \neg x_2 \lor x_3) \land (x_1 \lor \neg x_2 \lor \neg x_3) \land (x_1 \lor x_4 \lor x_5) \land (\_ \lor \_ \lor \_ ) \land (\_ \lor \_ \lor \_ ) \land (\_ \lor \_ \lor \_ ) \\ & \text{3 variables per clause} \end{align*} \]

Note:
- did the edges only for the first clause, need to do also the other clauses
- we connect every vertex to every other vertex in other clauses, except for the vertex's negation
- \( k = \# \text{ clauses} \)
- observe: \( k \)-clique exists iff \( \phi \) is satisfiable
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.
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 ∈ P, then P = NP.

if have poly-time algo for B, then for every A the reduction ≤_P B gives a poly-time algo for A

⇒ 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.