Today: more NP-complete problems.

**clique**: Given a graph \( G = (V, E) \) and \( k \in \mathbb{N} \), does \( G \) have a clique of size \( \geq k \)?

Observe \( C \subseteq V \) is a clique in \( G \) iff \( C \) is an independent set in \( G^c \), the complement of \( G \).

- \( G^c \) - vertices \( V \)
- edge \((u,v)\) iff \((u,v) \notin E(G)\).

**Thm**: Clique is NP-complete.

**Proof**:

1. Clique \( \in \text{NP} \)
   - **Certificate**: the vertices of the clique
   - **Verification**: - check \( \geq k \) vertices
   - check every 2 joined by edge
   - obviously correct and poly-time

2. Ind Set \( \leq_p \) Clique
   - **Alg. for Ind. Set**: assuming poly-time alg. for clique
     - give input \( G^c, k \) to clique alg.
     - return YES/NO answer.
   - This takes poly-time

It is correct because \( \times \)
Today: more NP-complete problems.

**Vertex Cover**

Input: Graph $G = (V, E)$, number $k \in \mathbb{N}$

Question: Does $G$ have a vertex cover of size $\leq k$?

- A set $S \subseteq V$ s.t. every edge $(u, v) \in E$ has $u$ or $v$ (or both) in $S$

  e.g.

  ![Vertex Cover Example](image)

  observe: $V - S$ is an independent set

**Theorem** Vertex Cover is NP-Complete

**Pf.**

1. **Vertex Cover $\in$ NP**
   - **Certificate:** the set $S$
   - **Verification:** check that every edge has endpoint in $S$ and check $|S| \leq k$. This is correct & poly-time.

2. **Ind. Set $\leq_p$ Vertex Cover**
   - Assume we have an alg. for Vertex Cover.
   - Give an alg. for Ind. Set.
   - We use relationship between Vertex Cover & Ind. Set in $G$.

   **Claim** $S$ is a vertex cover if $V - S$ is an Ind. set.

**Pf. Exercise**

Here's our alg. for Ind. Set

input $G, k$

- give $G, n - k$ to Vertex Cover alg.
- output YES/NO answer correct by poly-time (assuming Vertex Cover alg. is poly-time)
Note: for all these reductions $A \leq_p B$
for NP-completeness, our algorithm for $A$ uses
the algorithm for $B$ only once and returns its
YES/NO answer. "one-shot" - good way to remember
what many-one means.
This is called a many-one reduction
More general reduction is a Turing reduction.
For NP-completeness proofs, use one-shot reductions.

Road Map of NP-completeness Reductions

\[
\text{INDSET} \leq_p \text{VERTEX COVER} \leq_p \text{SET COVER}
\]
\[
\text{Circuit-SAT} \leq_p 3\text{-SAT} \leq_p \text{HAM. CYCLE} \leq_p \text{TSP}
\]
\[
\leq_p \text{SUBSET SUM}
\]

History
proof that 3-SAT is NP-complete
due to Prof. Stephen Cook, U. of Toronto, '71

and independently to Levin.

Other "first" NP-completeness proofs above due
to Richard Karp.

Directed Hamiltonian Cycle

Input: a directed graph $G = (V, E)$

Question: Does $G$ have a directed Hamiltonian cycle, i.e., a directed cycle that visits every vertex exactly once.

The Directed Hamilton cycle is NP-complete.

Proof 1: $\text{NP}$ certificate - order of visiting vertices.
Verification - check for directed edge between each pair of vertices, and that all vertices visited once, correct and poly. time.

Proof 2: $3$-SAT $\leq_p$ Directed Hamilton cycle.
Assume we have a poly. time alg. for Directed Hamilton cycle.
Design a poly. time alg. for $3$-SAT.
Input: clauses $C_1$ $C_m$, each clause has 3 literals.
Variables $x_1$ $x_n$.
Construct gadget to choose $T/F$ for each variable.

Example clause $(x_1 \lor \neg x_2 \lor x_3)$
Going left \rightarrow right on path for $x_i$ corresponds to $x_i = \text{True}$ and right \rightarrow left corresponds to $x_i = \text{False}$.

A 3-clam cycle must use one or the other path for each $x_i$.

Clause gadget for clause $C = (x_1 \lor \neg x_2 \lor x_3)$

Idea: visit $C$ by detouring off $x_1$ True path or $x_2$ False path or $x_3$ True path.
Note: make sure to leave a spare vertex between 2 clause detours

\[ \text{not OK} \quad \text{OK} \]

\[ \text{not OK} \]

To prove this construction is correct we must prove there's no other way to visit C.

Claim \( G \) has a directed Ham. cycle iff all clauses satisfyable

\[ \text{Pf} \leq \text{traverse the variable paths in True/False direction. For each clause C, at least one literal is set True — take a detour from that path to node C.} \]

\[ \Rightarrow \text{Suppose G has a Ham. cycle} \]

Claim visiting C must happen as a detour off a path.

\[ \text{Suppose we use (a, c) but not (c, b) (e.g. s_2, b, c, a, c then to different chain).} \]

\[ \text{can't use (a, s_1) so must enter s_1 from left. Must use (s_1, a).} \]

\[ \text{But then we can only enter b from s_2, and can only exit b to s_2, contradiction!} \]

Thus the Ham. cycle must traverse a T or F path for each variable, and must visit each clause vertex off such a path.

So this corresponds to satisfying truth value assignment.

Claim This construction takes poly. time.