24 NP and NP-completeness

Claim: formula is satisfiable iff this graph has an independent set of size equal to the number of clauses

Proof. If the formula is satisfiable, for every clause, there must be some variable setting that shows the clause is true. Include the corresponding vertex in the independent set.
This gives a set of vertices of size equal to the number of triangles. It is independent because we pick only one vertex per triangle, and don't pick vertices at both ends of an edge between triangles.

Conversely, if there is an independent set of size equal to the number if triangles, it must have one vertex per triangle. Construct a satisfying assignment by setting the corresponding variables to the values that make those clauses true. Since an independent set can't contain vertices from different triangles corresponding to variables x_i and $\overline{x}_i$, this assignment will be consistent.
It makes every clause evaluate to true, so it satisfies the formula. □

Thus, 3-SAT $\leq_P$ Independent Set $\leq_P$ Vertex Cover $\leq_P$ Set Cover

24.1 NP and NP-completeness

We say an algorithm is efficient if it runs in time polynomial in the input size.
We can collect problems with efficient solutions into a **complexity class** called P.

We'll focus on decision problems.
Characterize such problems by a **language** $L \subseteq \Sigma^*$ (all strings of any length over alphabet Σ).
A language L is in P if there exists an algorithm running in time polynomial in n (length of input) that accepts $x \in \Sigma^n$ if $x \in L$, and rejects $x \in \Sigma^n$ if $x \notin L$.

The harder probems that we've considered (3-SAT) may not have efficient algorithms, but they all have the property that solutions can be verified efficiently.

Examples:

- for 3SAT, the satisfying assignment
- for independent set problem, the vertices of the independent set

The class NP is the set of problems whose yes instances can be efficienctly verified, i.e.,
$x \in L \implies \exists y, A(x, y)$ accepts
$x \notin L \implies \forall y, A(x, y)$ rejects
where $A(x, y)$ is an algorithm running in time poly($|x|$) (where y is some "proof", x would be the problem, y would be an example).

Clearly, $P \subseteq NP$. Is $P = NP$?

We can get evidence for the hardness of NP problems using reductions.

Definition 24.1
If $\forall Y \in NP$, $Y \leq_P X$, we say X is NP-hard.

Definition 24.2
If $X \in NP$ and X is NP-hard, then X is NP-complete.

Trivial NP-complete problem: Circuit satisfiability.
A **circuit** is a labeled DAG where **sources** (vertices with no incoming edges) are either 0/1/free. Vertices with both incoming and outgoing edges are either AND/OR/NOT gates.
There is a unique **sink** (vertex with no outgoing edges) that represents the output.

Problem: Is there an assignment of the free inputs that makes the circuit evaluate to 1?

Theorem 24.3 (Cook-Levin Theorem)
3 SAT is NP-complete.