Reductions let us re-use one algorithm (e.g., max-flow) to solve other computational problems (e.g., airline scheduling).

If we can't solve this efficiently, we also wouldn't be able to solve it efficiently.

If we can solve this efficiently, we can also solve these efficiently.
Some problems currently believed to be computationally hard.

Given undirected graph $G$ and a natural number $k$...

**CLIQUE:** does $G$ have a set of $k$ vertices that are all joined to one another by edges? (each pair of vertices in the set are joined by an edge)

**INDEPENDENT SET:** does $G$ have a set of $k$ vertices that have no edges between them?

**VERTEX COVER:** does $G$ have a set of $k$ vertices that covers every edge? (each edge has $\geq 1$ endpoint in the set)
These are all "computationally equivalent": an efficient algorithm for one would give us a way to solve all the others efficiently as well.

Let $\overline{E} = \{(uv) \mid (uv) \text{ is not in } E\}$

Observe: $G = (V,E)$ has a clique of size $k$ $\iff \overline{G} = (V,\overline{E})$ has independent set of size $k$

Reduction

\[
\begin{array}{c}
\text{CLIQUE} \leftarrow \rightarrow \text{INDEP SET} \\
(G, k) \quad \rightarrow \quad (\overline{G}, k) \\
(\overline{G}, k) \quad \leftarrow \quad (G, k)
\end{array}
\]

Observe: $S$ is an indep set in $G = (V,E)$ $\iff V - S$ is a vertex cover in $G$

Proof: Suppose $S$ is indep set and $e = (uv)$ is an edge. The endpoints of $e$ can't both belong to $S$ so one of them belongs to $V - S$
\[ \therefore V-S \text{ is a vertex cover.} \]

Conversely, if \( V-S \) is a vertex cover and \((u,v)\) is an edge, then at least one of \(uv\) belongs to \(V-S\) so they don't both belong to \(S\).

\[ \therefore S \text{ is an independent set.} \]

**Reduction.**

\[ \text{VERTEX COVER} \quad \longleftrightarrow \quad \text{INDEPENDENT SET} \]

\[(G,k) \quad \longrightarrow \quad (G, n-k) \]

where \( n \) = \# vertices.

\[(G, n-k) \quad \longleftrightarrow \quad (G, k) \]

What do we know about solving these?

Brute force: \( \mathcal{O}(m \cdot \binom{n}{k}) = \mathcal{O}(m \cdot n^k) \).

Fastest known: \( \mathcal{O}(n^{c \cdot k}) \) for some \( 0 < c < 1 \).

A poly-time algorithm would need a constant in the exponent, not a multiple of \( k \).
3 SAT. Given n Boolean variables \( x_1, x_2, \ldots, x_n \)
and their negations denoted \( \overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n \)
(Collectively these \( 2n \) terms are called "literals".)

Given \( m \) "clauses" each formed
by connecting \( \leq 3 \) literals using
Boolean OR operation
e.g. \( x_q \lor \overline{x}_q \lor \overline{x}_{10} \)

Does \( \exists \) a truth assignment of \( x_1, \ldots, x_n \)
that satisfies every clause?

\[
E_x : (x_1 \lor x_2 \lor x_3) \land (\overline{x}_1 \lor \overline{x}_2 \lor x_4) \land (\overline{x}_3 \lor \overline{x}_4)
\]

is satisfied by \( (x_1, x_2, x_3, x_4) = (T, F, T, F) \)

\[
(x_1 \lor x_2) \land (\overline{x}_2 \lor x_3) \land (\overline{x}_3 \lor \overline{x}_4)
\]
is satisfied by \( (x_1, x_2, x_3) = (T, F, F) \).

\[ \text{3 SAT} \quad \rightarrow \quad \text{independent set} \]
Assume you had an algorithm to solve independent set. Try using it to solve 3 SAT.
(1) This subroutine that is good at finding large vertex sets... how to manipulate it into finding a truth assignment of $n$ variables?

\[ \begin{array}{cccccc}
  x_1 & 0 & x_2 & x_3 & 0 & \ldots x_{10} \\
  x_1 & x_2 & x_3 & 0 & \ldots & x_{10} \\
  x_1 & x_2 & x_3 & x_4 & \ldots & x_{10} \\
\end{array} \]

Choosing an independent set of size $n$ in this graph is equivalent to choosing a truth assignment of $x_1, \ldots, x_n$.

\[ \begin{array}{cccccc}
  x_1 & 0 & x_2 & x_3 & 0 & \ldots x_{10} \\
  x_1 & x_2 & x_3 & 0 & \ldots & x_{10} \\
  x_1 & x_2 & x_3 & x_4 & \ldots & x_{10} \\
\end{array} \quad \text{corresponds to} \quad (x_1, x_2, x_3, x_4) \quad = (T, F, T, F). \]

(2) How to connect the "variable assignment gadgets" with connective tissue such that only the truth assignments that satisfy
every clause lead to large independent sets?

\[
x_1, \quad x_2, \quad x_3
\]

\[
\overline{x_1}, \quad \overline{x_2}, \quad \overline{x_3}
\]

\[
x_1 \lor \overline{x_2} \lor \overline{x_3}
\]

A truth assignment that satisfies the clause corresponds to a vertex set that lets me choose one of the three “null vertices”
A truth assignment that violates the clause prevents choosing any "null vertex".

So transform 3SAT formula with $n$ vars, $m$ clauses to INDEP SET problem $(G, n+m)$ where $G$ is as illustrated above.