Class definitions
(A is in the class)

- A in NP and for every B in NP, B ∈_p A
- A is accepted by nondeterministic TM within p(lxl) steps where x = input and p is polynomial
- A is accepted by deterministic TM within p(lxl) steps...

The "world view" most people believe

how to prove that A is in the class

1. Prove A ∈ NP
2. Select problem C, known to be NP-complete and show C ∈_p A

- Give a guess and check algorithm for A: (input = x)
  - Guess certificate (trivial but no such mechanism exists)
  - Verify that certificate is proof that x has a "yes" answer

- Give an algorithm for A that requires O(lxl^c) basic operations, where c is constant

Importance of NP-complete problems:
Let A be NP-complete.

1. If we prove that A ∉ P, i.e. a non-polynomial lower bound for A, then P ≠ NP (and lots of other interesting consequences follow, e.g. there exist problems in NP that are neither in P nor NP-complete).

2. If we prove that A ∈ P, i.e. find a polynomial-time algorithm for A, then P = NP -- obviously B ∈ P ⇒ B ∈ NP, now B ∈ NP means B ∈_p A and, since A ∈ P we know B ∈ P as well. The picture above degenerates to

P = NP = NP-complete

... and the world of problem complexity is a lot less interesting
The first known NP-complete problem [Cook, 1971]

\( \text{B} \in \text{NP} \) means there exists a nondeterministic Turing machine \( M \) and a polynomial \( p \), so that input \( x \in B \) (is a \text{"yes"}-instance of \( B \)) if \( M \) accepts \( x \) in \( p(|x|) \) steps.

\[
\begin{align*}
\text{X} & \quad \text{Input to TM for B} \\
\ldots & \quad \text{Transformation} \\
\text{SAT} & \quad \text{Yes/No}
\end{align*}
\]

This part is specific to \( B \) but can be done for any \( B \) in \( \text{NP} \).

So \( B \) in \( \text{NP} \) implies \( B \leq_P \text{SAT} \).

A "canonical" NP-complete problem that's easier to work with: 3-SAT [Karp, 1972]

Formula = \( C_1 \land C_2 \land \ldots \land C_m \)

where each \( C_i = (l_{i1} \lor l_{i2} \lor l_{i3}) \)

and each \( l_{ij} \) is either a variable or its negation.
A positive ("yes") instance of 3-SAT:

\[(x_1 \lor \neg x_2 \lor \neg x_4) \land (\neg x_1 \lor x_2 \lor x_3) \land (\neg x_1 \lor \neg x_2 \lor \neg x_4) \land (\neg x_2 \lor x_3 \lor \neg x_4)\]

Certificate: \[x_1 = F, x_2 = F, x_3 = T, x_4 = T\]
(one of several possibilities)

A negative ("no") instance of 3-SAT:

\[(x_1 \lor x_2 \lor x_3) \land (x_1 \lor x_2 \lor \neg x_3) \land (x_1 \lor \neg x_2 \lor \neg x_3) \land (x_1 \lor \neg x_2 \lor x_3) \land (\neg x_1 \lor \neg x_2 \lor x_3) \land (\neg x_1 \lor x_2 \lor x_3) \land (\neg x_1 \lor x_2 \lor \neg x_3)\]

Dominating Set (DS): Given an undirected graph \(G = (V, E)\) and an integer \(K\), does there exist \(S \subseteq V\) such that \(|S| \leq K\) and for all \(u, v \in V\) either \(u \in S\) or there exists \(w \in E\) with \(w \in S\)?

\[G = \]

positive instance with \(K = 3\)

negative instance with \(K < 3\)

Proof that DS is in NP:

1. Guess \(S \subseteq V\)
2. Check that \(|S| \leq K\)
   For \(u, v \in V\) do if \(u \notin S\) then \(\text{neighbor} \leftarrow \text{false}\)
   For \(v \in \text{Adj}[u]\) do if \(v \in S\) then \(\text{neighbor} \leftarrow \text{true}\)
   if not \(\text{neighbor}\) then return "no"
   return "yes"

To prove that DS is NP-complete, we will show that
3-SAT \(\leq_p DS\)

known NP-complete problem

our new problem
Idea (based on guess and check algorithms for the two problems):

<table>
<thead>
<tr>
<th>guess</th>
<th>check</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-SAT</td>
<td>truth values of variables</td>
</tr>
<tr>
<td>DS</td>
<td>$S \subseteq V$ (set of vertices)</td>
</tr>
</tbody>
</table>

Details:

Start with a 3-SAT formula $\phi = C_1 \land \ldots \land C_m$ that uses $n$ variables $x_1, \ldots, x_n$.

Transform arbitrary 3-SAT instance $\phi$ to a DS instance $f(\phi)$ as follows:

In $f(\phi)$, $G = (V,E)$ and $K = n$, where

- $V = \{ u_i, \overline{u_i}, w_i \mid 1 \leq i \leq n \} \cup \{ v_j \mid 1 \leq j \leq m \}$
- $E = \{ u_i \overline{u_j}, u_i w_j, \overline{u_i} w_j \mid 1 \leq i \leq n \}$
  $\cup \{ u_i v_j \mid x_i$ is in $C_j \}$ $\cup \{ \overline{u_i} v_j \mid \overline{x_i}$ is in $C_j \}$

For example, if $\phi = (x_1 \lor \overline{x_2} \lor x_3) \land (x_1 \lor x_2 \lor \overline{x_3})$

then $f(\phi)$ is:

```
\begin{tikzpicture}
  \node [circle, draw] (u1) at (0,0) {$u_1$};
  \node [circle, draw] (u2) at (1,1) {$u_2$};
  \node [circle, draw] (u3) at (2,0) {$u_3$};
  \node [circle, draw] (w1) at (0,2) {$w_1$};
  \node [circle, draw] (w2) at (1,3) {$w_2$};
  \node [circle, draw] (w3) at (2,2) {$w_3$};
  \node [circle, draw] (v1) at (1,0) {$v_1$};
  \node [circle, draw] (v2) at (2,1) {$v_2$};
  \node [circle, draw] (v3) at (3,0) {$v_3$};

  \draw (u1) -- (u2);
  \draw (u2) -- (u3);
  \draw (u3) -- (u1);
  \draw (u1) -- (w1);
  \draw (u2) -- (w2);
  \draw (u3) -- (w3);
  \draw (u1) -- (v1);
  \draw (u2) -- (v2);
  \draw (u3) -- (v3);
  \draw (v1) -- (v2);
  \draw (v2) -- (v3);
  \draw (v3) -- (v1);

  \node at (3,2) {$K = 3$};
\end{tikzpicture}
```

Note: the $w_i$'s ensure that $S$ includes one of $u_i, \overline{u_i}, w_i$ for each $i$ and no $v_j$ vertices.
Obviously the transformation $f$ can be computed in polynomial time. So it remains to show that $\phi$ is a positive instance iff $f(\phi)$ is a positive instance. Usually this is done by showing that a certificate for $\phi$ translates into one for $f(\phi)$ and vice versa.

A certificate $y$ for $\phi$ is an assignment of truth values that makes each clause true.

If $x_i = T$ in $y$ then put $u_i$ in $S$

If $x_i = F$ in $y$ then put $\overline{u}_i$ in $S$

The resulting $S$ is a certificate for $f(\phi)$ because $|S| \leq K$

and $u_i$ is either in $S$ or its neighbor $\overline{u}_i$ is

$\overline{u}_i$ is $u_i$ is

$\overline{u}_i$ has a neighbor (either $u_i$ or $\overline{u}_i$) in $S$

$y$ has a neighbor in $S$: let $l$ be a literal that makes $C_j$ true under $y$; if $l = x_k$ then $u_k, v_j \in E$ and $u_k \in S$; if $l = \overline{x}_k$ then $\overline{u}_k, v_j \in E$ and $\overline{u}_k \in S$.

Conversely, let $S$ be a certificate for $f(\phi)$. Since each $u_i$ has only two neighbors, $u_i$ and $\overline{u}_i$, at least one of $u_i, \overline{u}_i, w_i$ must be in $S$ for $1 \leq i \leq n = K$. That leaves no room for other vertices to be in $S$. To derive a certificate for $\phi$

let $x_i = \begin{cases} T & \text{if } u_i \in S \text{ or } w_i \in S \\ F & \text{if } u_i \in S \end{cases}$

Since no $v_j$ can be in $S$, each $v_j$ must have a neighbor in $S$.

If that neighbor is $u_i$ for some $i$ then $C_j$ is true because $x_i \in C_j$; if the neighbor is $\overline{u}_i$ for some $i$ then $C_j$ is true because $\overline{x}_i \in C_j$. $\square$
Proof that CLIQUE is NP-complete (see pp. 947-949 in CLR) uses a different guess and check algorithm for 3-SAT.

\[ \text{guess} \quad \text{check} \]

<table>
<thead>
<tr>
<th>3-SAT</th>
<th>CLIQUE</th>
<th>SEV</th>
</tr>
</thead>
<tbody>
<tr>
<td>( l_i )</td>
<td>( l_{i1} \lor l_{i2} \lor l_{i3} )</td>
<td>( SEV )</td>
</tr>
</tbody>
</table>

Note: The CLIQUE problem is as follows, "Given an undirected graph \( G = (V,E) \) and an integer \( k \), does there exist \( SEV \) such that for every pair \( \{u, v\} \subseteq S \), \( uv \in E \)?"

Transformation \( f(\phi) \) where \( \phi = C_1 \land \ldots \land C_m \):

let \( C_i = (l_{i1} \land l_{i2} \land l_{i3}) \)

Then \( f(\phi) = (V,E), k \) where

- \( V = \{v_{ij} \mid 1 \leq i \leq m, 1 \leq j \leq 3\} \)
- \( E = \{v_{ij}v_{jk} \mid l_{ij} \neq l_{jk}\} \)
- \( k = m \)

If \( \phi = (x_1 \lor x_2 \land x_3) \land (x_1 \lor x_2 \land x_3) \)

\[ f(\phi) = \]

Let \( \overline{G} = (V, \overline{E}) \) where \( \overline{E} = \{uv \mid u \neq v \text{ and } uv \notin E\} \)

If \( S \) is a clique in \( G \), then \( S \) is an \textbf{independent set} in \( \overline{G} \) (a set of vertices that have no edges among them).

If \( S \) is an independent set in \( G \), then \( V - S \) is a \textbf{vertex cover} in \( G \) (set of vertices such that every edge has at least one vertex in the set).

\( \square \) means \( x_i = T \), the others don't matter

\( \square \) means \( x_3 = T \), \( x_2 = T \) and \( x_i \) doesn't matter