Proving NP-Completeness

We show problems are NP-complete by reducing from known NP-complete problems.
Proving $\mathsf{NP}$-Completeness by Reduction

To prove a problem is $\mathsf{NP}$-complete, use the earlier observation:

If $S$ is $\mathsf{NP}$-complete, $T \in \mathsf{NP}$ and $S \leq_p T$, then $T$ is $\mathsf{NP}$-complete.
Recall 3SAT:

Input: $\phi$ a boolean formula in 3CNF
Question: is there a satisfying assignment?

The language 3SAT is a restriction of SAT, and so 3SAT $\in \mathcal{NP}$. 

Proof that 3SAT is $\mathcal{NP}$-complete
Reducing 3SAT to SAT

We reduce SAT to 3SAT. The task is to describe a polynomial-time algorithm for:

**input:** a boolean formula $\phi$ in CNF  
**output:** a boolean formula $\psi_\phi$ in 3CNF such that $\phi$ is satisfiable exactly when $\psi_\phi$ is.
Substituting Clauses

We replace each clause $C$ of $\phi$ by family $D_C$ of clauses that preserves satisfiability.

For example, say $C = a \lor b \lor c \lor d \lor e$. One can simulate this by

$$D_C = (a \lor b \lor x) \land (\bar{x} \lor c \lor y) \land (\bar{y} \lor d \lor e)$$

where $x$ and $y$ are new variables. Need to verify:

1) If $C$ is FALSE, then $D_C$ is FALSE; and
2) If $C$ is TRUE, then one can make $D_C$ TRUE.

Clauses of other sizes are handled similarly.
So this yields $\psi_\phi$ in 3CNF. If $\phi$ is satisfiable, then there is assignment where each clause $C$ is TRUE; this can be extended to make each $D_C$ TRUE. Further, if assignment evaluates $\phi$ to FALSE, then some clause say $C'$ must be FALSE and thus the corresponding family $D_{C'}$ in $\psi_\phi$ is FALSE.

The last thing to check is that the conversion process can be encoded as a polynomial-time algorithm. Thus, we have shown that SAT reduces to 3SAT, and so 3SAT is $\mathcal{NP}$-complete.
Proof that DOMINATION is NP-complete

Recall that a dominating set $D$ is such that every other node is adjacent to a node in $D$; and that the DOMINATION problem is:

Input: graph $G$ and integer $k$
Question: is there dominating set of at most $k$ nodes?

We reduce 3SAT to DOMINATION. That is, we describe a procedure that takes boolean formula $\phi$, and produces graph $G_\phi$ and integer $k_\phi$ such that $\phi$ is satisfiable exactly when there is a dominating set of $G_\phi$ of $k_\phi$ nodes.
Suppose $\phi$ in 3CNF has $c$ clauses and $m$ variables. For each clause, create a node. For each variable $v$, create a triangle with one node labeled $v$ and one labeled $\overline{v}$. Then for each clause, join the clause-node to the three nodes corresponding to its literals. The result is graph $G_\phi$.

For example, the graph for $(x \lor y \lor z) \land (\overline{x} \lor y \lor \overline{z})$: 

![Graph Diagram](image-url)
Set $k_\phi = m$ (the number of vars). Claim: the mapping $\phi$ to $\langle G_\phi, k_\phi \rangle$ is the desired reduction.

If $\phi$ has satisfying assignment, then let $D$ be the $m$ nodes corresponding to TRUE literals in the assignment. Then each triangle is dominated, as is each clause-node. So $D$ is dominating set.

Conversely, suppose $G_\phi$ has dominating set $D$ of size $m$. Then $D$ must be one node from each triangle, and every clause must be connected to one literal in $D$. So setting all the literals corresponding to nodes in $D$ to TRUE is satisfying.
That is, we have shown that 3SAT reduces to DOMINATION, and so DOMINATION is \( \mathcal{NP} \)-complete.
Gadgets

The above reduction illustrates a common pattern. To reduce from 3SAT, create a “gadget” for each variable and a “gadget” for each clause, and connect them up somehow.
Proof that \textsc{subset-sum} is $\mathcal{NP}$-complete

Recall that input to Subset sum problem is set $A = \{a_1, a_2, \ldots, a_m\}$ of integers and target $t$. The question is whether there is $A' \subseteq A$ such that elements in $A'$ sum to $t$.

We prove this problem is $\mathcal{NP}$-complete. This is again a reduction from 3SAT. The previous example suggests the approach: define numbers $x_i$ and $\bar{x}_i$ and a target $t$ such that one can take only one of $x_i$ and $\bar{x}_i$, and then some constraint is to be satisfied.
Suppose that one has \textit{vectors} instead of numbers. Two vectors (of same length) can be added component-wise. The question now is whether there is subset whose sum equals a specified vector.

Suppose input $\phi$ has $c$ clauses and $m$ variables. The vectors will have length $c + m$. For each vector, the first $m$ positions will specify which variable by a 1 in the appropriate position. The second part records the clauses each literal is in.
Example Vectors

For example, if $\phi$ is

$$(x_2 \lor x_3 \lor \bar{x}_4) \land (x_1 \lor \bar{x}_3 \lor x_4) \land (x_1 \lor \bar{x}_2 \lor x_4)$$

then vectors corresponding to the variables are

$x_1 = (1, 0, 0, 0; 0, 1, 1)$ and $\bar{x}_1 = (1, 0, 0, 0; 0, 0, 0)$

$x_2 = (0, 1, 0, 0; 1, 0, 0)$ and $\bar{x}_2 = (0, 1, 0, 0; 0, 0, 1)$

$x_3 = (0, 0, 1, 0; 1, 0, 0)$ and $\bar{x}_3 = (0, 0, 1, 0; 0, 1, 0)$

$x_4 = (0, 0, 0, 1; 0, 1, 1)$ and $\bar{x}_4 = (0, 0, 0, 1; 1, 0, 0)$
A target of all 1’s would force selection of exactly one of each variable and its negation. However, some clauses might have multiple true literals. So define $t$ as all 1’s for variables and all 3’s for clauses: $t = (1, 1, 1, 1; 3, 3, 3)$. 

Constructing the Target
Then add *slack variables*. These are vectors that one can use to round sum up to \( t \). Specifically, add *two* copies of each clause:

\[
c_1 = (0, 0, 0, 0; 1, 0, 0) \text{ and } c'_1 = (0, 0, 0, 0; 1, 0, 0), \text{ etc.}
\]

Note that to reach 3 in a component, at least one 1 must be supplied by a literal.
The Reduction for Vectors

That is, we have built a set of vectors and a target vector such that there is a subset of vectors that sums to the target vector exactly when the boolean formula has a satisfying assignment.

(Well, actually we do have to argue this both ways.)
Finally, we go from vectors to numbers. Just think of the vector as the number in decimal:

\[ t = 1111333 \text{ and } x_1 = 1000011, \bar{x}_1 = 1000000, \]
\[ x_2 = 0100100, \bar{x}_2 = 0100001, \text{ etc.} \]

A worry is that one might be able to reach the target in unintended way, but that does not happen. So we have shown a reduction from 3SAT to \textsc{Subset Sum}, and so \textsc{Subset Sum} is \textsc{NP}-complete.

\textit{Believe it or not, these reductions become routine, eventually.}
Show that VERTEX_COVER is \( \mathcal{NP} \)-complete.

(Recall that the removal of a vertex cover destroys every edge, and that the input to VERTEX_COVER is graph \( G \) and integer \( k \).)

(Hint: Reduce from 3SAT using two connected nodes for each variable and three connected nodes for each clause.)
Solution to Practice

We first show that VERTEX_COVER is in $\mathcal{NP}$. The nondeterministic program guesses $k$ nodes and then checks they form a vertex cover.

We then reduce 3SAT to VERTEX_COVER. We describe a procedure to take a boolean formula $\phi$, and produce graph $G_\phi$ and integer $k_\phi$, such that $\phi$ is satisfiable exactly when there is vertex cover of $G_\phi$ of $k_\phi$ nodes.
Assume $\phi$ has $c$ clauses and $m$ variables. For each variable $v$, create two adjacent nodes labeled $v$ and $\bar{v}$. For each clause, create three adjacent nodes and join each to a literal in the clause.

For example, the graph $G_{\phi}$ for $(x \lor y \lor z) \land (\bar{x} \lor y \lor \bar{z})$:
Let $k_\phi = m + 2c$. Claim: the mapping $\phi$ to $\langle G_\phi, k_\phi \rangle$ is the desired reduction. The main part is to show that the mapping preserves the answer.

Suppose $G_\phi$ has vertex cover $D$ of size $k_\phi$. It contains at least one node from each node-pair and two nodes from each clause-triangle. Since $D$ has size $m + 2c$, this is exactly what $D$ is. Thus when we remove $D$, for each clause one node remains, and so the other end of that edge is in $D$. That is, the literals in $D$ are a satisfying assignment.
Conversely, suppose $\phi$ has a satisfying assignment. Then let $D$ be the $m$ nodes corresponding to the TRUE literals in the assignment. Then each clause-triangle is dominated. So one can add two nodes from each clause-triangle and all edges incident with that clause are taken care of. It follows that $G_\phi$ has a vertex cover of size $m + 2c$.

That is, we have shown that 3SAT reduces to VERTEX_COVER, and so VERTEX_COVER is $NP$-complete.
Summary

A new problem can be proven \( \mathcal{NP} \)-complete by reduction from a problem already known to be \( \mathcal{NP} \)-complete.