Suppose we have $n$ boolean variables $x_1, \ldots, x_n$.

A literal is either a variable or its complete negation, $x_i$ or $\neg x_i$.

If $x_i$ is assigned to $T$, $\neg x_i$ evaluates to $F$ under this assignment.

If $x_i$ is assigned $F$, $\neg x_i$ evaluates to $T$ under this assignment.

A clause is a \textit{disjunction} of literals. For example, $x_1 \lor \neg x_2 \lor \neg x_4$.

\textit{Disjunction} $\equiv \text{OR}$

Under an assignment to the variables, a clause evaluates to whatever the OR of the literals evaluates to.
For example,
\[ x_1 = F \quad x_2 = F \quad x_3 = ? \quad x_4 = T \]
causes the clause to evaluate to true.

Whereas,
\[ x_1 = F \quad x_2 = T \quad x_3 = ? \quad x_4 = T \]
causes the clause to evaluate to false.

So an assignment

So a clause evaluates to true under an assignment if and only if at least one literal in the clause evaluates to true under the assignment.

A CNF formula \( \phi \) is a collection of clauses, treated as a conjunction (AND). \( \phi \) evaluates to true under an assignment if each clause evaluates to true under the assignment.
So, $\phi$ evaluates to true under an assignment to the variables if and only if each at least one literal in each clause of $\phi$ evaluates to true under the assignment.

$$\phi = (x_1 \lor x_2) \land (\neg x_1 \lor \neg x_2) \land (x_1 \lor \neg x_2)$$

$\phi$ evaluates to true under $x_1 = T, \ x_2 = T$.

$\phi$ evaluates to False under $x_1 = F, \ x_2 = T$.

The CNF-SAT problem is to determine, given a CNF formula $\phi$, whether there is an assignment that satisfies $\phi$, that is, causes it to evaluate to true.
In the above example, \( \phi \) had a satisfying assignment.

Here is a \( \phi \) that does not have a satisfying assignment:

\[
\phi: (x, v x_2) \land (x, v \neg x_2) \land (\neg x, v x_2) \\
\land (\neg x, v \neg x_2)
\]

Here is another:

\[
\phi: (x, v x_2) (x, v \neg x_2) (\neg x, v x_2) \\
(\neg x, v \neg x_2 v x_3) (\neg x, v \neg x_2 v \neg x_3)
\]

CNF-SAT arises in a number of constraint satisfaction applications. We are interested in 3CNF-SAT, the special case in which each formula in \( \phi \) has exactly 3 literals.
Such a formula $\phi$ is called a 3CNF-
formule.

We do not know if 3CNF-SAT has a polynomial time algorithm.

What we will show is that

$3CNF\text{-SAT} \leq_p \text{Independent-Set}$.

This means

(a) A polynomial time algorithm for
independent set implies a poly-time
algorithm for $3CNF\text{-SAT}$.

(b) If there is no poly-time algo
for independent set $3CNF\text{-SAT}$,
there is no poly-time algo for
independent set.
3CNF-SAT $\leq_p$ Independent Set

We describe an algorithm that takes as input any instance

$\phi = C_1 \land C_2 \ldots \land C_m$ of 3CNF-SAT and in variables $x_1, \ldots, x_n$ and constructs an instance of independent set such that the answer to the independent set instance is yes if and only if there is an assignment that satisfies $\phi$. The algorithm will have polynomial running time.

The algorithm will simply feed the independent set instance to the independent set black box and return whatever the black box returns - yes/no.
The graph is constructed as follows. Corresponding to $C_i$, we add vertices $v_i^1, v_i^2, v_i^3$. $v_i^1$ corresponds to first literal in $C_i$, $v_i^2$ to second literal in $C_i$, and so on.

So there are $3m$ vertices.

Now for the edges: For each $i$, we add $(v_i^1, v_i^2)$, $(v_i^2, v_i^3)$, $(v_i^3, v_i^1)$. Any $i \neq j$ let $v_i^s$ and $v_j^t$ be two vertices with $i \neq j$. We add an edge between $v_i^s$ and $v_j^t$ if corresponding literals are negations of each other — $x_k$ and $\neg x_k$. 
That completes the description of the graph. Finally, the independent set instance asks if the graph has an independent set of size at least \( m \)?

\[
\phi : (x_1 \lor x_2 \lor \neg x_3) \land (x_3 \lor x_4 \lor \neg x_1) \land (x_2 \lor \neg x_4 \lor x_5)
\]
All we need to do is to argue that \( \phi \) has a satisfying assignment if and only if the corresponding graph has an independent set of size \( m \) or greater. Notice that an independent set can include at most one vertex from each "triangle", so it has size at most \( m \).

Suppose \( \phi \) has a satisfying assignment. This assignment causes one literal in each clause to evaluate to true. The corresponding vertices in the graph form an independent set of size \( m \). (If \( \neg \phi \)-assignment, the set at literals we picked cannot include \( \neg \xi \) and \( \neg \neg \xi \), because both literals must evaluate to true under assignment.)
Conversely, suppose the graph has an independent set of size $m$. This must include exactly one vertex from each triangle. So we get one literal from each clause. Since we started from an independent set, this set of literals does not both have $x_i$ and $\neg x_i$ for any $i$. So there is any assignment that causes each literal in the set to evaluate to true, and thus each clause to evaluate to true.
Set Cover:

Given a set $U$ of $n$ elements, a collection $S_1, \ldots, S_m$ of subsets of $U$, and a number $K$, does there exist a collection of at most $K$ of these sets whose union is equal to $U$?
In the above example, answer is yes with $K=3$, but no with $K=2$.

Vertex Cover $\leq_p$ Set-Cover.
This is simply because vertex cover is a special case of set cover, we'll show why via an example.
In the corresponding set cover instance,

\[ U = \{ a, b, c, d, e, f, g, h, i, j \} \]

the set of edges. There is a set corresponding to each vertex - the set of edges incident to it,

\[ S_1 = \{ a, e \} \]
\[ S_2 = \{ e, b, g, i \} \]
\[ S_3 = \{ a, b, c, d \} \]
\[ S_4 = \{ g, h \} \]
\[ S_5 = \{ i, j \} \]
\[ S_6 = \{ d, f \} \]
\[ S_7 = \{ c, f, h, j \} \]