To prove Clique is NP-complete, need to show:

a) Clique ∈ NP:
   - can verify solutions in poly-time

b) ∀A ∈ NP, A → Clique.
   - take known NP-complete problem B & show:
     B → Clique.

Thus, if can solve Clique in poly-time then can solve every A ∈ NP in poly-time.
In other words, if P ≠ NP then Clique cannot be solved in poly-time.
Shows that clique is intractable.

We used Independent Set to prove Clique is NP-complete.

Later: SAT → 3SAT to show 3SAT is NP-complete & 3SAT → Independent Set to show Independent set is NP-complete.
**Vertex Cover (VC):**

For $G=(V,E)$, $S \subseteq V$ is a vertex cover if it covers every edge:

- for every edge $(x, y) \in E$, either $x \in S$ &/or $y \in S$.

(Also one or both endpoints are in the VC)

**Example:**

$S = \{a, b, d, e, f\}$ is a vertex cover.

$S' = \{b, d, f\}$ is a vertex cover of min size.

Want to find smallest vertex cover.
Vertex Cover Problem:

**Input:** \( G = (V,E) \) & goal/budget \( b \)

**Output:** Vertex cover \( S \) of size \( \leq b \) if one exists

**No** otherwise.

**Theorem:** Vertex Cover is \( NP \)-complete.

**Proof:**

a) \( VC \in NP \):

Given \( G, b, S \) in \( O(n+m) \) time we can check for every edge \( (x,y) \in E \) that \( x \in S \) &/or \( y \in S \).

b) **Independent Set \( \rightarrow \) Vertex Cover.**

Given input \( (G,g) \) for IS,

run Vertex Cover on \( (G, n-g) \)

For solution \( S \) to VC, output \( S \) as solution to IS

& if NO solution to VC, then NO solution to IS.
Claim: $S$ is a vertex cover of size $|S| \leq b$ where $b = n - q$ if and only if $S$ is an independent set of size $|S| \geq g$. 

Proof:

$\Rightarrow$ Consider VCS in $G$. Hence, for every $(x, y) \in E$, $x \in S$ or $y \in S$ and thus $x \notin S$ or $y \notin S$. Therefore, $S$ is an independent set.

And if $|S| \leq n - g$ then $|S| \geq g$.

$\Leftarrow$ Consider independent set $S$ in $G$. Hence, for every $(x, y) \in E$, $x \in S$ or $y \in S$. Thus, $x \in S$ or $y \in S$ and $S$ is a VC. And if $|S| \geq g$ then $|S| \leq n - g$. $\blacksquare$
Let's start back at the beginning (almost). Assume SAT is NP-complete.
Let's now prove 3SAT is NP-complete.

**SAT:**

*input:* Boolean formula \( f \) in CNF with:
- \( n \) variables \( x_1, \ldots, x_n \)
- \( m \) clauses \( C_1, \ldots, C_m \)

*output:* satisfying assignment for \( f \) if one exists
NO if none exist.

**3SAT:**

*input:* \( f \) in CNF with \( x_1, \ldots, x_n, C_1, \ldots, C_m \)
where for every clause \( |C_i| \leq 3 \)
\(( \leq 3 \) literals per clause\)

*output:* satisfying assignment if one exists
NO if none exist.
Theorem: 3SAT is NP-complete

Proof:

a) $3\text{SAT} \in \text{NP}$:

Given $f$ & assignment $\sigma$, for each clause $C_i$, we can check in $O(1)$ time that at least one literal in $C_i$ is satisfied by $\sigma$.

Hence in $O(m)$ total time we can check that $\sigma$ satisfies every clause.

b) $\text{SAT} \rightarrow 3\text{SAT}$.

Consider input $f$ to SAT. We will create input $f'$ for 3SAT.

Denote variables in $f$ as $x_1, \ldots, x_n$

& clauses as $C_1, \ldots, C_m$.

For clause $C_i$ if $|C_i| \leq 3$ then we can leave it as is for 3SAT input $f'$.

What if $|C_i| > 3$?
Take for example $C_i = (\overline{x_3} \lor x_2 \lor x_5 \lor \overline{x_1})$.

Let's create a new variable call it $y_i$.

Consider the pair of clauses:

$$S_i = (\overline{x_3} \lor x_2 \lor y_i) \land (x_5 \lor \overline{x_1} \lor y_i)$$

Claim: $C_i$ is satisfiable $\iff S_i$ is satisfiable.

$(\Rightarrow)$ Take an assignment to $x_3, x_2, x_5, x$, satisfying $C_i$.

Use that same assignment in $S_i$. At least one of the clauses in $S_i$ is satisfied & we can use $y_i$ to satisfy the other.

$(\Leftarrow)$ Take an assignment to $x_3, x_2, x_5, x, y_i$ satisfying $S_i$.

Note $y_i = T$ or $y_i = F$ satisfying the 1st or 2nd clause, but the other clause must be satisfied by $x_3, x_2, x_5$ or $x_1$. That same literal will satisfy $C_i$. 


What if \(|C| = 5\)?

Example: \(C_i = (x_3 \vee x_2 \vee \overline{x}_5 \vee x_1 \vee \overline{x}_6)\).

We'll add 2 new variables \(y_{i,1}\) & \(y_{i,2}\)

We'll replace \(C_i\) by these 3 clauses:

\[S_i = (x_3 \vee x_2 \vee y_{i,1}) \land (y_{i,1} \vee x_5 \vee y_{i,2}) \land (y_{i,2} \vee x_1 \vee \overline{x}_6)\]

Note, \(y_{i,1}\) & \(y_{i,2}\) satisfy \(\leq 2\) of the clauses

(can set to satisfy any 2 we want)

& the remaining clause must be satisfied

by \(x_3, x_2, x_5, \overline{x}_1, \) or \(\overline{x}_6\).

In general, for clause \(C_i = (a_1 \vee a_2 \vee \ldots \vee a_k)\)

where \(k > 3\) & \(a_1, \ldots, a_k\) are the literals

\((s_0 a_1 = \overline{x}_3, a_2 = \overline{x}_2, a_3 = x_5, a_4 = \overline{x}_1, a_5 = \overline{x}_6)\)

in above example.
add \( k-3 \) new variables: \( Y_{i1}, Y_{i2}, \ldots, Y_{i,k-3} \)
and replace \( C_i \) by \( k-2 \) clauses:
\[
S_i = (a, va_{a_2} v Y_{i1}) \land (\overline{Y_{i1}}, va_{a_3} v Y_{i2}) \land (Y_{i2}, va_{a_4} v Y_{i3}) \land \ldots \land (\overline{Y_{i,k-4}}, va_{a_{k-2}} v Y_{i,k-3}) \land (Y_{i,k-3}, va_{a_{k-1}} v a_k)
\]
We do this for every clause \( C_i \) where \( |C_i| > 3 \),
& if \( |C_i| \leq 3 \) then leave as is.
This gives a new formula \( f' \).
Note \( f' \) has \( \leq n + m(n-3) = O(nm) \) variables
\& \( \leq O(nm) \) clauses.

Run 3SAT on \( f' \).
Ignore the settings for the new variables \( Y_i \)'s.
Take the setting for the original variables \( X_1, \ldots, X_n \)
& this satisfies \( f' \).
If NO for \( f' \) then return NO for \( f \).
Claim: $f$ is satisfiable $\iff f$ is satisfiable

$\Rightarrow$ Take setting of $x_1, \ldots, x_n$ that satisfies $C_1, \ldots, C_m$.
Consider $C_i = (a_1 \lor a_2 \lor \ldots \lor a_k)$.
$\geq 1$ of $a_1, \ldots, a_k$ is satisfied & this satisfies $\geq 1$ clause in $S_i$. Use $y_i$'s to satisfy other $k-3$ clauses in $S_i$.

$\Leftarrow$ For setting of $y_i$'s & $a_1, \ldots, a_k$ that satisfies $S_i$.
Note $y_i$'s satisfy $\leq k-3$ clauses of $S_i$.
Thus $\geq 1$ of $a_1, \ldots, a_k$ is satisfied & this also satisfies $C_i$. $\Box$