Exercise:

Show that CSAT remains NP-complete even if it is restricted to instances in which each variable appears at most three times. Let's call this variant 3VAR CSAT.

Solution: We show how to reduce CSAT to 3VAR CSAT.

Given a formula $F$ in CNF, we construct a formula $E$ in CNF where each variable appears at most three times and such that $E$ is satisfiable if and only if $F$ is satisfiable.

Let $x$ be a variable appearing in $F$ $k$ times, with $k > 3$.

Then we construct from $F$ a formula $F_x$ as follows:

1) We replace the $i$-th occurrence of $x$ in $F$ with $x_{i+1}$ for $i \in \{1, \ldots, k\}$, where each $x_{i+1}$ is a fresh variable.

2) We add the following clauses, ensuring that all variables $x_1, \ldots, x_k$ are assigned the same truth value

$$\left( \overline{x_1} \lor x_2 \right) \land \left( \overline{x_2} \lor x_3 \right) \land \cdots \land \left( \overline{x_{k-1}} \lor x_k \right) \land \left( \overline{x_k} \lor x_1 \right)$$

We have that $F_x$ can be constructed from $F$ in polynomial time, and $F_x$ is satisfiable if and only if $F$ is satisfiable.

Then, the formula $E$ is obtained from $F$ by repeating the above transformation for each variable $x$ occurring in $F$ more than three times.

We have that:

1) Each variable appears in $E$ at most three times
2) $E$ can be constructed from $F$ in polynomial time
3) $E$ is satisfiable if and only if $F$ is satisfiable.
Exercise:

Consider 2VARCSAT, i.e., the variant of CSAT in which each variable appears at most two times.

What is the complexity of 2VARCSAT?

Solution: 2VARCSAT is in P.

Let $F$ be an instance of 2VARCSAT.

Notice that for each variable $x$, we have one of 3 cases:

1. $x$ appears only positively in $F$ (one or two times)
2. $x$ appears only negatively in $F$ (one or two times)
3. $x$ appears one time positively and one time negatively in $F$

From this, we obtain the following algorithm to decide the satisfiability of $F$:

Input: set $F$ of clauses over variables $x_1, \ldots, x_n$, with each $x_i$ appearing in at most two clauses.

Output: YES, if $F$ is satisfiable, NO otherwise.

For each variable $x_i \in \{x_1, \ldots, x_n\}$:

- If $x_i$ appears only positively in $F$ (case 1), or
- If $x_i$ appears only negatively in $F$ (case 2), or
- Then remove from $F$ the clause(s) in which $x_i$ appears (case 3)

else let $C = x_i \lor C_{rest}$ and $C' = \neg x_i \lor C'_{rest}$

- Let the two clauses of $F$ in which $x_i$ appears

if $C_{rest}$ and $C'_{rest}$ are both empty
then answer NO
else remove from $F$ both $C$ and $C'$, and replace them with the clause $C_{rest} \lor C'_{rest}$

Answer YES
The algorithm runs in polynomial time, since the for-loop is executed \( m \) times, and each iteration is at most linear.

Note that a variable \( x_i \) might be removed from \( F \) before it is considered in the \( i \)-th iteration of the for loop. In this case, \( F \) is not changed in that iteration.

Moreover:

- For cases (1) and (2) the clauses containing \( x_i \) can be trivially satisfied by making \( x_i \) true/false.

- In case (3), the algorithm applies a resolution step to \( x_i \), and replaces the clauses \( C \) and \( C' \) with their resolvent.

- By applying a resolution step to a variable \( x_i \), for another variable \( x_j \), that has not yet been considered (i.e., \( j > i \)) and that occurred positively and negatively in two different clauses, the two occurrences of \( x_j \) might be merged into a single clause \( C \text{ rest } \lor C' \text{ rest} \). This clause can be removed, since it is trivially satisfied by every truth assignment.
Exercise: 2SAT is in P

Idea: we show that 2SAT can be encoded as a graph reachability problem, and then use an algorithm for graph reachability.

1) Encoding of 2SAT as a directed graph reachability problem

Let \( \Phi \) be an instance of 2SAT. We define a graph \( G(\Phi) \) as follows:
- one node for each variable and one node for each negated variable
- for each clause \( \alpha \lor \beta \) two edges \( \overline{\alpha} \rightarrow \beta \) \( \overline{\beta} \rightarrow \alpha \)

(Note: \( \alpha \lor \beta \equiv \overline{\alpha} \rightarrow \beta \equiv \overline{\beta} \rightarrow \alpha \))

Example: \( (x_1 \lor x_2), (x_1 \lor \overline{x}_3), (\overline{x}_4 \lor x_2), (x_2 \lor x_3) \)

\[ \begin{array}{c}
\text{Then } \Phi \text{ is unsatisfiable iff there is a variable } x \text{ such that } G(\Phi) \text{ contains two paths } \overline{x} \rightarrow \cdots \rightarrow \overline{x} \\
\end{array} \]

\[ \begin{array}{c}
\leq \text{ Suppose that } \Phi \text{ has a satisfying truth assignment } T. \\
\text{Assume that } T(x) = \text{true} \text{ (a similar argument holds for } T(x) = \text{false}). \\
\text{Since } T(x) = \text{true and } T(\overline{x}) = \text{false, and there is a path } \overline{x} \rightarrow \cdots \rightarrow \overline{x}, \\
\text{there must be an edge } \alpha \rightarrow \beta \text{ along this path with } T(\alpha) = \text{true} \text{ and } T(\beta) = \text{false}. \\
\end{array} \]
However, since \( x \rightarrow y \) is an edge of \( G(\phi) \), it follows that \( x + \bar{y} \) is a clause of \( \phi \). This clause is not satisfied by \( T \), which is a contradiction.

\( \Rightarrow \) Let \( G(\phi) \) be a graph that does not contain any node \( \bar{x} \) with \( \bar{x} \rightarrow \cdots \rightarrow \bar{x} \).

We construct from such a graph \( \tau \) satisfying truth assignment \( T \).

Repeat the following step as often as possible:

- Choose a node \( \bar{x} \) such that
  - \( T(\bar{x}) \) is not yet defined, and
  - there is no path \( \bar{x} \rightarrow \bar{x} \).

For every node \( \bar{y} \) that is reachable from \( \bar{x} \) (including \( \bar{x} \) itself):

1) set \( T(\bar{y}) = \text{true} \) (Note: 2) means to assign false to all predecessors of \( \bar{y} \)).

2) set \( T(\bar{y}) = \text{false} \).

Observe: if the truth assignment \( T \) is well defined, i.e.,

we never have both \( T(\bar{y}) = \text{true} \) and \( T(\bar{y}) = \text{true} \) or \( T(\bar{y}) = \text{false} \) and \( T(\bar{y}) = \text{false} \).

\( T \) would not be well defined, if we had both \( \bar{x} \rightarrow \bar{y} \) (for some \( \bar{y} \)).

But this cannot happen, since we would have

\[
\begin{array}{c}
\bar{x} \\
\downarrow \\
\bar{y} \\
\bar{z} \\
\end{array}
\]

Hence, we would have \( \bar{x} \rightarrow \bar{z} \).

2) We assign to all nodes a truth value, since there is no path \( \bar{x} \rightarrow \bar{x} \).

3) The truth assignment satisfies all clauses of \( F \), since each clause corresponds to an implication, and there is no \( \bar{x} \rightarrow \bar{z} \).