- polynomial time reduction
  - A \rightarrow B: spend \text{poly time} on input X of A to prepare an input Y for B, then return B(Y)
  - poly time: if there is a poly time reduction from A \rightarrow B if B has a poly time algorithm, then A has a poly time algorithm. (if B \in P, A \in P)
  - no post processing: related to definition of NP
    \begin{align*}
    \text{NP} & \quad \text{answer} = \text{YES} \quad \exists \text{ solution s.t. verifier accepts} \\
    \text{NP} & \quad \text{answer} = \text{NO} \quad \text{no solution that verifier accepts}
    \end{align*}

- \text{NP-hard problems}
  - Problem B is \underline{NP-hard}, if for any problem A \in \text{NP} there is a polynomial time reduction from A \rightarrow B
  - Problem B is \underline{NP-complete}, if B is NP-hard, and B is in NP.

- Polynomial time reduction from A to B

  \begin{align*}
  A \leq B \\
  A \leq B, B \leq C \Rightarrow A \leq C
  \end{align*}

  \begin{align*}
  \text{A(}\text{input}X) & \rightarrow B(\text{input}Y) \rightarrow C(\text{input}Z) \\
  \text{if NP-hard problem B is in P} \quad \text{then since every NP problem A \leq B, A \in P} \\
  \Rightarrow P = NP
  \end{align*}
- if both $A$ and $B$ are NP-complete, then if one of them has a poly-time algorithm, the other one also has a poly-time algorithm.

$$B \text{ is NP-complete } \Rightarrow B \text{ is NP hard } \quad A \leq B$$

$$A \text{ is NP-complete } \Rightarrow A \text{ is in NP } \quad B \leq A$$

- Prove a problem $B$ is NP-hard.
  - find an NP-hard problem $A$
  - do a poly-time reduction from $A$ to $B$

$$A \leq B$$

for any $C \in NP$  $C \leq A$

$$C \leq B$$

- Cook-Levin Theorem

- CIRCUIT-SAT problem is NP-hard.

- For any NP problem $A$, there is a poly-time answer to $A = \begin{cases} YES & \exists \text{ Solution} \\ NO & \forall \text{ Solution} \end{cases}$

\[
\begin{align*}
\text{Verifier}(\text{input } X, \text{ solution}) \\
\{ \text{ input } X, \text{ solution} \} \rightarrow \text{Verifier (solution)} \\
\ldots \text{return true if else,} \\
\text{return felse.} \\
\text{Compile} \\
\text{Circuit} \\
\text{Solution} \\
\text{return value} \\
\text{run CIRCUIT-SAT on this} \\
\end{align*}
\]

- reduction

- INDEPENDENT SET to CLIQUE
1. **transform X for INDEP. SET \[\Rightarrow Y \text{ for CLIQUE}**
   - Observation: INDEP. SET want vertices to have no edges
   - CLIQUE want vertices to be connected

   ![Diagram of X for INDEP. SET and Y for CLIQUE]

   \[K = 2\]

   - Idea: flip edges

   - In first step, cannot use the solution to the INDEP. SET problem

2. **X is YES \[\Rightarrow Y \text{ is YES}**
   - X has indep. set of size K
   - Y has clique of size K
   - Solution for X \[\Rightarrow\] Solution for Y
   - Claim: every independent set of the original graph is a clique in the new graph.

3. **X is NO \[\Rightarrow Y \text{ is NO}**
   - Y is YES \[\Rightarrow\] X is YES
   - Solution for Y \[\Rightarrow\] Solution for X
   - Claim: every clique in the new graph is an independent set in the original graph.

- 3-SAT

\[
(X_1 \lor X_2 \lor X_3) \land (\overline{X_2} \lor X_3 \lor X_4) \land (\overline{X_1} \lor X_2 \lor \overline{X_4})
\]

Answer: Yes \[X_1 = \text{true}, \ X_2 = \text{true}\]
\[ x_3 = \text{true} \quad x_4 = \text{anything} \]
\[
(x_1 \lor x_2) \land (\overline{x_1}) \land (\overline{x_2})
\]

Answer: No

- Reduction: 3-SAT \( \rightarrow \) INDEP. SET

\[
x_1 \lor x_2 \lor \overline{x_3} \quad \overline{x_2} \lor x_3 \lor x_4 \quad \overline{x_1} \lor x_2 \lor \overline{x_4}
\]

\[ K : \text{intuition: for a satisfying assignment } \]
\[ \text{can choose 1 vertex from every variable gadget} \]
\[ \text{choose 1 vertex from every clause gadget} \]

\[ K = \Pi + M \]

Step 2: 3-SAT YES \( \Rightarrow \) INDEP. SET YES