- **Cook-Levin Theorem**
- **Example of Reductions**

- **Cook-Levin Theorem**: For any problem \( L \) in NP, there is a polynomial time reduction from \( L \) to \( \text{CIRCUIT-SAT} \) (SAT, 3-SAT).

- **CIRCUIT-SAT** (circuit satisfiability)
  - Boolean circuits
  - 3 basic operations: \( \wedge \) and \( \lor \) or \( \neg \) not

- Circuit: Directed acyclic graph whose nodes are "gates"

  - **CIRCUIT-SAT**: Given a circuit as the input, decide if there is a set of assignments to the input variables that makes the circuit output 1.

  - **CIRCUIT-SAT \( \in \) NP**
    - Easy. The "proof" is just one satisfying assignment, verifier will evaluate the circuit and output 1 if the circuit outputs 1.

  - **Proof idea of Cook-Levin Theorem**:
    - If \( L \) is an NP problem, then there is a poly-time verifier \( V \)
- If the verifier $V$ is actually implemented by a boolean circuit:

  ![Circuit Diagram]

  (instance of $L$) (proof $C$)

- For any instance $x \in L$, fix the input for $x$, try to see if the circuit is still satisfiable.

  If circuit is sat. $\implies$ answer to $x$ is yes
  Circuit is not sat. $\implies$ answer to $x$ is no.

- Reduction from $L$ to CIRCUIT-SAT.

- Claim: All polynomial-time algorithm can be implemented by a circuit of polynomial size.

- Reductions

  - To prove $L$ is NP-hard, only need to reduce CIRCUIT-SAT to $L$.
    $$L \supseteq \text{CIRCUIT-SAT} \supseteq \text{any NP problem}$$

  - Common starting point: 3-SAT problem

  - A 3-SAT instance has $m$ clauses, each clause is an or of (at most) 3 literals. A literal is a variable or its negation.

    $$X_1 \lor X_2 \lor X_3$$

    $$\rightarrow$$

    Clause

    Literals

    $$\rightarrow$$

    Clause is satisfied if $X_1 = 1$ or $X_2 = 1$ or $X_3 = 1$.

    - Answer to 3-SAT is yes if all $m$ clauses can be satisfied simultaneously.

    (Another way to write is $C_1 \land C_2 \land \ldots \land C_m$ is satisfiable)

    $\uparrow$

    First clause

    $\uparrow$

    Last clause
3-SAT \rightarrow CIRCUIT-SAT easy

this reduction does not show 3-SAT is NP-hard.

CIRCUIT-SAT \geq 3-SAT

in order to show 3-SAT is NP-hard, need

CIRCUIT-SAT \rightarrow 3-SAT

3-SAT \geq CIRCUIT-SAT \geq \text{any NP problem}

- Example: INDEPENDENT-SET is NP-complete.

- IND-SET: Given a graph G (undirected), S \subseteq V is an independent set if no two vertices in S are connected by an edge.

| \includegraphics[width=0.5\textwidth]{independent_set_diagram.png} |

IND-SET: \((G, k)\) Decide whether \(G\) has an ind-set of size \( \geq k \).

- Reduction: 3-SAT \rightarrow IND-SET

- Idea: Use "gadgets"

  for each object in 3-SAT \rightarrow \text{map to some group of objects in IND-SET}

  literals \((x_i, \overline{x}_i)\) \text{ vertices}

  clauses \((C_1, C_2, \ldots)\) \text{ edges.}

  - Literals: For each literal in each clause \rightarrow \text{map to a vertex}

    \[x_1 \lor \overline{x_3} \lor \overline{x}_5 \rightarrow \bigcirc \quad \bigcirc\]

    in solution, one of these three is satisfied.

    \[\text{the satisfied literal will be in ind-set}\]

    \[\text{edges: } u, v \text{ are connected } \iff \text{ cannot choose both } u, v, \]

    \[\text{connect all vertices labeled } x_i \text{ to all vertices labeled } \overline{x}_i.\]
- Connect all literals within the same clause.

(want each clause to contribute 1 vertex to \( \text{IND-SET} \))

\[(x_1, \overline{x_2}, x_3) \land (x_2, \overline{x_3}, x_4) \land (x_3, \overline{x_2}, x_4)\]

Claim: The 3-SAT instance is satisfiable iff the graph has an \( \text{ind-} \) set of size \( m \).