Reductions to Prove \textbf{NP} Hardness

Nabil Mustafa

Computational Complexity
Reductions

- The class \textbf{NP} : languages with succinct membership certificates
Reductions

- The class **NP**: languages with succinct membership certificates
- **Cook-Levin**: All \( L \in \text{NP} \) can be reduced to SAT
  - \( \exists \) polynomial \( g(\cdot) \) such that \( x \in L \iff g(x) \in \text{SAT} \)
Reductions

- The class **NP**: languages with succinct membership certificates

- **Cook-Levin**: All $L \in \textbf{NP}$ can be reduced to SAT
  - $\exists$ polynomial $g(\cdot)$ such that $x \in L \iff g(x) \in \text{SAT}$
  - Very important to remember the implication goes both ways

- Gave a reduction from general CNF SAT to 3-CNF SAT
- 2-CNF SAT is in P through a poly-time algorithm.
- 3-CNF SAT then becomes a very important problem. Also, an elementary problem, very simple to understand
- Captures the 'pure' combinatorial choices of algorithms
- Given 3-CNF SAT, can construct a host of other reductions.
Reductions

- The class **NP**: languages with succinct membership certificates

- **Cook-Levin**: All $L \in \textbf{NP}$ can be reduced to SAT
  - $\exists$ polynomial $g(\cdot)$ such that $x \in L \iff g(x) \in \text{SAT}$
  - Very important to remember the implication goes both ways

- Gave a reduction from general CNF SAT to 3-CNF SAT
Reductions

- The class **NP**: languages with succinct membership certificates

- **Cook-Levin**: All $L \in \text{NP}$ can be reduced to SAT
  - $\exists$ polynomial $g(\cdot)$ such that $x \in L \iff g(x) \in \text{SAT}$
  - Very important to remember the implication goes both ways

- Gave a reduction from general CNF SAT to 3-CNF SAT
  - 2-CNF SAT is in **P** through a poly-time algorithm.
Reductions

- The class **NP**: languages with succinct membership certificates

- **Cook-Levin**: All \( L \in \textbf{NP} \) can be reduced to SAT
  - \( \exists \) polynomial \( g(\cdot) \) such that \( x \in L \iff g(x) \in \text{SAT} \)
  - Very important to remember the implication goes both ways

- Gave a reduction from general CNF SAT to 3-CNF SAT
  - 2-CNF SAT is in \( \textbf{P} \) through a poly-time algorithm.

- 3-CNF SAT then becomes a very important problem. Also,
Reductions

- The class **NP**: languages with succinct membership certificates

- **Cook-Levin**: All $L \in \text{NP}$ can be reduced to SAT
  - $\exists$ polynomial $g(\cdot)$ such that $x \in L \iff g(x) \in \text{SAT}$
  - Very important to remember the implication goes both ways

- Gave a reduction from general CNF SAT to 3-CNF SAT
  - 2-CNF SAT is in P through a poly-time algorithm.

- 3-CNF SAT then becomes a very important problem. Also,
  - An elementary problem, very simple to understand
Reductions

- The class **NP**: languages with succinct membership certificates

- **Cook-Levin**: All $L \in \text{NP}$ can be reduced to SAT
  - $\exists$ polynomial $g(\cdot)$ such that $x \in L \iff g(x) \in \text{SAT}$
  - Very important to remember the implication goes both ways

- Gave a reduction from general CNF SAT to 3-CNF SAT
  - 2-CNF SAT is in $\text{P}$ through a poly-time algorithm.

- 3-CNF SAT then becomes a very important problem. Also,
  - An elementary problem, very simple to understand
  - Captures the ‘pure’ combinatorial choices of algorithms
Reductions

- The class **NP**: languages with succinct membership certificates

- **Cook-Levin**: All $L \in \text{NP}$ can be reduced to SAT
  - $\exists$ polynomial $g(\cdot)$ such that $x \in L \iff g(x) \in \text{SAT}$
  - Very important to remember the implication goes both ways

- Gave a reduction from general CNF SAT to 3-CNF SAT
  - 2-CNF SAT is in $\text{P}$ through a poly-time algorithm.

- 3-CNF SAT then becomes a very important problem. Also,
  - An elementary problem, very simple to understand
  - Captures the ‘pure’ combinatorial choices of algorithms

- Given 3-CNF SAT, can construct a host of other reductions.
Claim

**INDSET**: Given a graph $G = (V, E)$ and a parameter $k$, does $G$ have an independent set of size $k$?

The independent set problem (**INDSET**) is **NP** complete.
INDSET

Claim

INDSET: Given a graph $G = (V, E)$ and a parameter $k$, does $G$ have an independent set of size $k$?

The independent set problem (INDSET) is NP complete.

- i) It is in NP
Claim

**INDSET**: Given a graph $G = (V, E)$ and a parameter $k$, does $G$ have an independent set of size $k$?

*The independent set problem (INDSET) is NP complete.*

- i) It is in NP
- ii) Reduce a NP hard problem to it
Claim

**INDSET**: Given a graph $G = (V, E)$ and a parameter $k$, does $G$ have an independent set of size $k$?

The independent set problem (**INDSET**) is **NP** complete.

- i) It is in **NP**
- ii) Reduce a **NP** hard problem to it

Show: If one can solve INDSET in polynomial time, then can solve 3-CNF SAT problem in polynomial time as well.
\[ \phi = C_1 \land C_2 \land \ldots \land C_k, \text{ where } C_i = (x_1^i \lor x_2^i \lor x_3^i) \]
\[ \phi = C_1 \land C_2 \land \ldots \land C_k, \text{ where } C_i = (x^i_1 \lor x^i_2 \lor x^i_3) \]

- Construct a graph \( G = (V, E) \) as follows:

  - Vertices: Each literal \( x^i_j \) in \( C_i \) corresponds to the vertex \( v^i_j \).
  - Edges: \((v^i_j, v^{i'}_{j'}) \in E\) if \( x^i_j, x^{i'}_{j'} \) are the same variable negated.
  - Add all possible edges between \( v^i_1, v^i_2 \) and \( v^i_3 \) for all \( i \).
\[ \phi = C_1 \land C_2 \land \ldots \land C_k, \text{ where } C_i = (x_1^i \lor x_2^i \lor x_3^i) \]

- Construct a graph \( G = (V, E) \) as follows:
  - Vertices: Each literal \( x_j^i \in C_i \) corresponds to the vertex \( v_j^i \).
\( \phi = C_1 \land C_2 \land \ldots \land C_k \), where \( C_i = (x^i_1 \lor x^i_2 \lor x^i_3) \)

- Construct a graph \( G = (V, E) \) as follows:
  - Vertices: Each literal \( x^i_j \in C_i \) corresponds to the vertex \( v^i_j \).
  - Edges: \( (v^i_j, v^i_{j'}) \in E \) if \( x^i_j, x^i_{j'} \) are the same variable negated.
\[ \phi = C_1 \land C_2 \land \ldots \land C_k, \text{ where } C_i = (x_1^i \lor x_2^i \lor x_3^i) \]

- Construct a graph \( G = (V, E) \) as follows:
  - Vertices: Each literal \( x_j^i \in C_i \) corresponds to the vertex \( v_j^i \).
  - Edges: \( (v_j^i, v_j'^i) \in E \) if \( x_j^i, x_j'^i \) are the same variable negated.
  - Add all possible edges between \( v_1^i, v_2^i \) and \( v_3^i \) for all \( i \).
\[ \phi = C_1 \land C_2 \land \ldots \land C_k, \] 
where \( C_i = (x_1^i \lor x_2^i \lor x_3^i) \)

- Construct a graph \( G = (V, E) \) as follows:
  - Vertices: Each literal \( x_j^i \in C_i \) corresponds to the vertex \( v_j^i \).
  - Edges: \( (v_j^i, v_j'^i) \in E \) if \( x_j^i, x_j'^i \) are the same variable negated.
  - Add all possible edges between \( v_1^i \), \( v_2^i \) and \( v_3^i \) for all \( i \).

\[
(x_1 \lor x_2 \lor \overline{x_3}) \land (\overline{x_1} \lor x_3 \lor \overline{x_4}) \land (\overline{x_2} \lor x_4)
\]
\[ \phi = C_1 \land C_2 \land \ldots \land C_k, \text{ where } C_i = (x_1^i \lor x_2^i \lor x_3^i) \]

- Construct a graph \( G = (V, E) \) as follows:
  - Vertices: Each literal \( x_j^i \in C_i \) corresponds to the vertex \( v_j^i \).
  - Edges: \( (v_j^i, v_j^{i'}) \in E \) if \( x_j^i, x_j^{i'} \) are the same variable negated.
  - Add all possible edges between \( v_1^i, v_2^i \) and \( v_3^i \) for all \( i \).

\[
(x_1 \lor x_2 \lor \overline{x_3}) \land (\overline{x_1} \lor x_3 \lor x_4) \land (\overline{x_2} \lor x_4)
\]
\[ \phi = C_1 \land C_2 \land \ldots \land C_k, \text{ where } C_i = (x_1^i \lor x_2^i \lor x_3^i) \]

*Construct a graph* \( G = (V, E) \) *as follows:*

- **Vertices:** Each literal \( x_j^i \in C_i \) corresponds to the vertex \( v_j^i \).
- **Edges:** \((v_j^i, v_j'^i) \in E\) if \( x_j^i, x_j'^i \) are the same variable negated.
- **Add all possible edges between** \( v_1^i, v_2^i \) **and** \( v_3^i \) **for all** \( i \).

\[
(x_1 \lor x_2 \lor \overline{x_3}) \land (\overline{x_1} \lor x_3 \lor \overline{x_4}) \land (\overline{x_2} \lor x_4)
\]
\( \phi = C_1 \land C_2 \land \ldots \land C_k \), where \( C_i = (x_1^i \lor x_2^i \lor x_3^i) \)

- Construct a graph \( G = (V, E) \) as follows:
  - Vertices: Each literal \( x_{ij}^i \in C_i \) corresponds to the vertex \( v_{ij}^i \).
  - Edges: \( (v_{ij}^i, v_{ij'}^i) \in E \) if \( x_{ij}^i, x_{ij'}^i \) are the same variable negated.
  - Add all possible edges between \( v_1^i, v_2^i \) and \( v_3^i \) for all \( i \).

\[
(x_1 \lor x_2 \lor \overline{x_3}) \land (\overline{x_1} \lor x_3 \lor \overline{x_4}) \land (\overline{x_2} \lor x_4)
\]
Reduction for INDSET

- **Claim:** If $G$ has an independent set of size $k$, $\phi$ is satisfiable
Reduction for INDSET

**Claim**: If $G$ has an independent set of size $k$, $\phi$ is satisfiable

- The independent set, say $V'$, cannot have two vertices from the same clause
Reduction for INDSET

- **Claim:** If $G$ has an independent set of size $k$, $\phi$ is satisfiable
  - The independent set, say $V'$, cannot have two vertices from the same clause
  - Therefore, $V'$ has exactly one vertex from each of the $k$ clauses
Reduction for INDSET

- **Claim:** If $G$ has an independent set of size $k$, $\phi$ is satisfiable

  - The independent set, say $V'$, cannot have two vertices from the same clause
  - Therefore, $V'$ has exactly one vertex from each of the $k$ clauses
  - No pair maps to literals $x$ and $\overline{x}$
Claim: If $G$ has an independent set of size $k$, $\phi$ is satisfiable

- The independent set, say $V'$, cannot have two vertices from the same clause
- Therefore, $V'$ has exactly one vertex from each of the $k$ clauses
- No pair maps to literals $x$ and $\overline{x}$
- If $v^i_j$ corresponds to literal $x^i_j = x$, set $x^i_j = 1$
Reduction for INDSET

Claim: If $G$ has an independent set of size $k$, $\phi$ is satisfiable

- The independent set, say $V'$, cannot have two vertices from the same clause
- Therefore, $V'$ has exactly one vertex from each of the $k$ clauses
- No pair maps to literals $x$ and $\overline{x}$
- If $v^i_j$ corresponds to literal $x^i_j = x$, set $x^i_j = 1$
- If $v^i_j$ corresponds to literal $x^i_j = \overline{x}$, set $x^i_j = 0$
**Claim:** If $G$ has an independent set of size $k$, $\phi$ is satisfiable

- The independent set, say $V'$, cannot have two vertices from the same clause
- Therefore, $V'$ has exactly one vertex from each of the $k$ clauses
- No pair maps to literals $x$ and $\overline{x}$
- If $v^i_j$ corresponds to literal $x^i_j = x$, set $x^i_j = 1$
- If $v^i_j$ corresponds to literal $x^i_j = \overline{x}$, set $x^i_j = 0$
- Each clause satisfied, and no variable-setting conflicts between clauses
Reduction for INDSET

- **Claim:** If $G$ has an independent set of size $k$, $\phi$ is satisfiable
  
  - The independent set, say $V'$, cannot have two vertices from the same clause
  - Therefore, $V'$ has exactly one vertex from each of the $k$ clauses
  - No pair maps to literals $x$ and $\overline{x}$
  - If $v_j^i$ corresponds to literal $x_j^i = x$, set $x_j^i = 1$
  - If $v_j^i$ corresponds to literal $x_j^i = \overline{x}$, set $x_j^i = 0$
  - Each clause satisfied, and no variable-setting conflicts between clauses

- **Claim:** If $\phi$ is satisfiable, $G$ has an independent set of size $k$
Claim

**CLIQUE**: Given a graph $G = (V, E)$ and a parameter $k$, does $G$ have a clique of size $k$?

The clique problem (CLIQUE) is **NP complete**.
CLAIM

**Claim**

**CLAIM**: Given a graph $G = (V, E)$ and a parameter $k$, does $G$ have a clique of size $k$?

*The clique problem (CLAIM) is NP complete.*

- Reduction from INDSET
**Claim**

**CLIQUE** : Given a graph $G = (V, E)$ and a parameter $k$, does $G$ have a clique of size $k$?

The clique problem (CLIQUE) is NP complete.

- Reduction from INDSET
- Given $G$, consider the complement graph $\overline{G}$
Claim

**CLIQUE**: Given a graph $G = (V, E)$ and a parameter $k$, does $G$ have a clique of size $k$?

*The clique problem (CLIQUE) is NP complete.*

- Reduction from INDSET
- Given $G$, consider the complement graph $\overline{G}$
- Claim: $G$ has an indset of size $k$ iff $\overline{G}$ has clique of size $k$
**Claim**

**CLIQUE** : Given a graph $G = (V, E)$ and a parameter $k$, does $G$ have a clique of size $k$?

The clique problem (CLIQUE) is **NP complete**.

- Reduction from INDSET
- Given $G$, consider the complement graph $\overline{G}$
- Claim: $G$ has an indset of size $k$ iff $\overline{G}$ has clique of size $k$
Reduce 3-CNF SAT to CLIQUE

\[ \phi = C_1 \land C_2 \land \ldots \land C_k, \text{ where } C_i = (x_1^i \lor x_2^i \lor x_3^i) \]
Same CLIQUE Reduction ‘Unrolled’

Reduce 3-CNF SAT to CLIQUE

\[ \phi = C_1 \land C_2 \land \ldots \land C_k, \text{ where } C_i = (x_i^1 \lor x_i^2 \lor x_i^3) \]

Construct a graph \( G = (V, E) \) as follows:
Same CLIQUE Reduction ‘Unrolled’

Reduce 3-CNF SAT to CLIQUE

\[ \phi = C_1 \land C_2 \land \ldots \land C_k, \text{ where } C_i = (x_1^i \lor x_2^i \lor x_3^i) \]

- Construct a graph \( G = (V, E) \) as follows:
  - **Vertices**: Each literal corresponds to a vertex.
Same CLIQUE Reduction ‘Unrolled’

Reduce 3-CNF SAT to CLIQUE

\[ \phi = C_1 \land C_2 \land \ldots \land C_k, \text{ where } C_i = (x_i^1 \lor x_i^2 \lor x_i^3) \]

Construct a graph \( G = (V, E) \) as follows:

- **Vertices**: Each literal corresponds to a vertex.
- **Edges**: All vertices are connected with an edge except the vertices of the same clause and vertices with negated literals.
Same CLIQUE Reduction ‘Unrolled’

Reduce 3-CNF SAT to CLIQUE

\[ \phi = C_1 \land C_2 \land \ldots \land C_k, \text{ where } C_i = (x_1^i \lor x_2^i \lor x_3^i) \]

Construct a graph \( G = (V, E) \) as follows:

- **Vertices**: Each literal corresponds to a vertex.
- **Edges**: All vertices are connected with an edge except the vertices of the same clause and vertices with negated literals.
  - \( x_i, x_j \in C_k \) does not have an edge.
Same CLIQUE Reduction ‘Unrolled’

Reduce 3-CNF SAT to CLIQUE

\[ \phi = C_1 \land C_2 \land \ldots \land C_k, \text{ where } C_i = (x_i^1 \lor x_i^2 \lor x_i^3) \]

- Construct a graph \( G = (V, E) \) as follows:
  - **Vertices**: Each literal corresponds to a vertex.
  - **Edges**: All vertices are connected with an edge except the vertices of the same clause and vertices with negated literals.
    - \( x_i, x_j \in C_k \) does not have an edge.
    - \( x_i \in C_k, \bar{x}_i \in C_j \) does not have an edge.
Same CLIQUE Reduction ‘Unrolled’

Reduce 3-CNF SAT to CLIQUE

\[ \phi = C_1 \land C_2 \land \ldots \land C_k, \text{ where } C_i = (x^i_1 \lor x^i_2 \lor x^i_3) \]

- **Construct a graph** \( G = (V, E) \) as follows:
  - **Vertices**: Each literal corresponds to a vertex.
  - **Edges**: All vertices are connected with an edge except the vertices of the same clause and vertices with negated literals.
    - \( x_i, x_j \in C_k \) does not have an edge.
    - \( x_i \in C_k, \bar{x}_i \in C_j \) does not have an edge.

- Clique of size \( k \) in \( G \) iff 3-CNF satisfiable.
Example Clique Reduction

\[ \phi = (x_1 \lor x_2 \lor \overline{x_3}) \land (\overline{x_1} \lor x_3 \lor x_4) \land (x_2 \lor x_4) \]
A Vertex Cover of an undirected graph \( G = (V, E) \) is a subset \( V' \) of vertex set \( V \) of the vertices of \( G \) which contains at least one of the two end points of each edge.

\[
V' \subseteq V : \forall (v_i, v_j) \in E : v_i \in V' \lor v_j \in V'
\]
Claim

**VERTEX-COVER** : Given a graph $G = (V, E)$ and a parameter $k$, does $G$ have a vertex cover of size $k$?

The vertex cover problem (**VERTEX-COVER**) is **NP** complete.
vertex-cover

Claim

vertex-cover: Given a graph \( G = (V, E) \) and a parameter \( k \), does \( G \) have a vertex cover of size \( k \)?

The vertex cover problem (vertex-cover) is \( \text{NP} \) complete.

- Note that vertex-cover is in \( \text{NP} \)
Claim

**VERTEX-COVER**: Given a graph $G = (V, E)$ and a parameter $k$, does $G$ have a vertex cover of size $k$?

*The vertex cover problem (VERTEX-COVER) is NP complete.*

- Note that VERTEX-COVER is in NP
- We reduce 3-CNF SAT to VERTEX-COVER
Reduction for VERTEX-COVER

\[ \phi = C_1 \land C_2 \land \ldots \land C_m, \text{ where } C_i = (x_1^i \lor x_2^i \lor x_3^i) \]
Reduction for VERTEX-COVER

\[ \phi = C_1 \land C_2 \land \ldots \land C_m, \text{ where } C_i = (x_1^i \lor x_2^i \lor x_3^i) \]

- Construct a graph \( G = (V, E) \) as follows:

1. **Variable vertices:** Each variable \( x_i \) creates two vertices \( v_1^i, v_0^i \).
2. **Edge vertices:** Each clause \( C_j \) creates a set \( V(C_j) \) of \( n_j \) vertices.
3. **Edges:** For all \( x_i \), \((v_1^i, v_0^i) \in E\).
4. **Edges:** Add all possible edges between each set \( V(C_j) \).
5. **Edges:** For each literal \( x_i \in C_j \), connect its vertex in \( C_j \) to the corresponding literal vertex.

Question: Does \( G \) have a vertex-cover of size \( k = n + 2m \)?
Reduction for VERTEX–COVER

\[
\phi = C_1 \land C_2 \land \ldots \land C_m, \text{ where } C_i = (x_1^i \lor x_2^i \lor x_3^i)
\]

- Construct a graph \( G = (V, E) \) as follows:
  - Variable vertices: Each variable \( x_i \) creates two vertices \( v_1^i, v_0^i \)
  - Edge vertices: Each clause \( C_j \) creates a set \( V(C_j) \) of \( n_j \) vertices
  - Edges: For all \( x_i \), \((v_1^i, v_0^i) \in E\)
  - Edges: Add all possible edges between each set \( V(C_j) \)
  - Edges: For each literal \( x_i \in C_j \), connect its vertex in \( C_j \) to the corresponding literal vertex.

\( k = n + 2m \)

Question: Does \( G \) have a vertex-cover of size \( k \)?
Reduction for VERTEX-COVER

\[ \phi = C_1 \land C_2 \land \ldots \land C_m, \text{ where } C_i = (x_1^i \lor x_2^i \lor x_3^i) \]

Construct a graph \( G = (V, E) \) as follows:

- Variable vertices: Each variable \( x_i \) creates two vertices \( v_1^i, v_0^i \)
- Edge vertices: Each clause \( C_j \) \( n_j \) creates a set \( V(C_j) \) of \( n_j \) vertices
Reduction for VERTEX–COVER

\[ \phi = C_1 \land C_2 \land \ldots \land C_m, \text{ where } C_i = (x_1^i \lor x_2^i \lor x_3^i) \]

Construct a graph \( G = (V, E) \) as follows:

- **Variable vertices**: Each variable \( x_i \) creates two vertices \( v_1^i, v_0^i \)
- **Edge vertices**: Each clause \( C_j n_j \) creates a set \( V(C_j) \) of \( n_j \) vertices
- **Edges**: For all \( x_i \), \( (v_1^i, v_0^i) \in E \)
Reduction for VERTEX-COVER

\[ \phi = C_1 \land C_2 \land \ldots \land C_m, \text{ where } C_i = (x_1^i \lor x_2^i \lor x_3^i) \]

Construct a graph \( G = (V, E) \) as follows:

- **Variable vertices:** Each variable \( x_i \) creates two vertices \( v_1^i, v_0^i \)
- **Edge vertices:** Each clause \( C_j \) \( n_j \) creates a set \( V(C_j) \) of \( n_j \) vertices
- **Edges:** For all \( x_i \), \((v_1^i, v_0^i) \in E\)
- **Edges:** Add all possible edges between each set \( V(C_j) \)
Reduction for VERTEX-COVER

\[ \phi = C_1 \land C_2 \land \ldots \land C_m, \text{ where } C_i = (x_{i1}^i \lor x_{i2}^i \lor x_{i3}^i) \]

Construct a graph \( G = (V, E) \) as follows:

- Variable vertices: Each variable \( x_i \) creates two vertices \( v_{i1} \) and \( v_{i0} \)
- Edge vertices: Each clause \( C_j \) creates a set \( V(C_j) \) of \( n_j \) vertices
- Edges: For all \( x_i \), \( (v_{i1}, v_{i0}) \in E \)
- Edges: Add all possible edges between each set \( V(C_j) \)
- Edges: For each literal \( x_i \in C_j \), connect its vertex in \( C_j \) to the corresponding literal vertex.
Reduction for VERTEX-COVER

\[ \phi = C_1 \land C_2 \land \ldots \land C_m, \text{ where } C_i = (x_1^i \lor x_2^i \lor x_3^i) \]

- Construct a graph \( G = (V, E) \) as follows:
  - Variable vertices: Each variable \( x_i \) creates two vertices \( v_1^i, v_0^i \)
  - Edge vertices: Each clause \( C_j \) \( n_j \) creates a set \( V(C_j) \) of \( n_j \) vertices
  - Edges: For all \( x_i \), \( (v_1^i, v_0^i) \in E \)
  - Edges: Add all possible edges between each set \( V(C_j) \)
  - Edges: For each literal \( x_i \in C_j \), connect its vertex in \( C_j \) to the corresponding literal vertex.
  - \( k = n + 2m \)
Reduction for VERTEX-COVER

\[ \phi = C_1 \land C_2 \land \ldots \land C_m, \text{ where } C_i = (x_1^i \lor x_2^i \lor x_3^i) \]

- Construct a graph \( G = (V, E) \) as follows:
  - Variable vertices: Each variable \( x_i \) creates two vertices \( v_1^i, v_0^i \)
  - Edge vertices: Each clause \( C_j \) \( n_j \) creates a set \( V(C_j) \) of \( n_j \) vertices
  - Edges: For all \( x_i \), \( (v_1^i, v_0^i) \in E \)
  - Edges: Add all possible edges between each set \( V(C_j) \)
  - Edges: For each literal \( x_i \in C_j \), connect its vertex in \( C_j \) to the corresponding literal vertex.
  - \( k = n + 2m \)

- Question: Does \( G \) have a vertex-cover of size \( k \)?
Reduction for VERTEX-COVER

$$(x_1 \lor x_2 \lor \overline{x_3}) \land (\overline{x_1} \lor x_3 \lor \overline{x_4}) \land (\overline{x_2} \lor x_4 \lor x_1), \, n = 4, \, m = 3$$
Reduction for VERTEX-COVER

\[(x_1 \lor x_2 \lor \overline{x_3}) \land (\overline{x_1} \lor x_3 \lor \overline{x_4}) \land (\overline{x_2} \lor x_4 \lor x_1), n = 4, m = 3\]
Reduction for VERTEX-COVER

$$(x_1 \lor x_2 \lor \overline{x_3}) \land (\overline{x_1} \lor x_3 \lor \overline{x_4}) \land (\overline{x_2} \lor x_4 \lor x_1), \ n = 4, \ m = 3$$
Reduction for VERTEX-COVER

$$(x_1 \lor x_2 \lor \overline{x_3}) \land (\overline{x_1} \lor x_3 \lor \overline{x_4}) \land (\overline{x_2} \lor x_4 \lor x_1), \ n = 4, \ m = 3$$
Reduction for VERTEX-COVER

Claim: If $\phi$ is satisfiable, then $\exists$ a vertex cover of size $k = n + 2m$
Reduction for VERTEX-COVER

Claim: If $\phi$ is satisfiable, then $\exists$ a vertex cover of size $k = n + 2m$

- If variable $x_i = 0$, pick $v_i^0$ to be in the vertex cover
Reduction for VERTEX-COVER

Claim: If $\phi$ is satisfiable, then $\exists$ a vertex cover of size $k = n + 2m$

- If variable $x_i = 0$, pick $v_i^0$ to be in the vertex cover
- If variable $x_i = 1$, pick $v_i^1$ to be in the vertex cover

Clearly, the number of vertices picked is $n + 2m$.

An edge between two variable vertices is covered

For each $i$, exactly one of the variable vertices picked.

For each $C_j$, edges to the picked vertices covered.

Crucial: What about the edges incident upon $x_j^2$?

Edges within its clause covered by the other two picked vertices

Remaining edge of the form: $(x_j^i, v_i^1)$ if $x_j^i$ is un-negated.

We know that $v_i^1$ is picked, since $v_i = 1$. 
Reduction for VERTEX–COVER

Claim: If \( \phi \) is satisfiable, then \( \exists \) a vertex cover of size \( k = n + 2m \)

- If variable \( x_i = 0 \), pick \( v^0_i \) to be in the vertex cover
- If variable \( x_i = 1 \), pick \( v^1_i \) to be in the vertex cover
- Each clause \( C_j \) has at least one literal, say \( x^j_2 \), set to \textit{true}. 

- Don't pick edge vertex of \( x^j_i \), pick other two edge vertices

Claim: The picked vertices cover all the edges of \( G \).

Clearly, the number of vertices picked is \( n + 2m \).

An edge between two variable vertices is covered

For each \( i \), exactly one of the variable vertices picked.

For each \( C_j \), edges to the picked vertices covered.

Crucial: What about the edges incident upon \( x^j_2 \)?

Edges within its clause covered by the other two picked vertices

Remaining edge of the form: \( (x^j_i, v^1_i) \) if \( x^j_i \) is un-negated.

We know that \( v^1_i \) is picked, since \( v^1_i = 1 \).
Reduction for VERTEX-COVER

Claim: If $\phi$ is satisfiable, then $\exists$ a vertex cover of size $k = n + 2m$

- If variable $x_i = 0$, pick $v_i^0$ to be in the vertex cover
- If variable $x_i = 1$, pick $v_i^1$ to be in the vertex cover
- Each clause $C_j$ has at least one literal, say $x_{2j}^j$, set to true.
  - Don’t pick edge vertex of $x_{i}^j$, pick other two edge vertices
Reduction for VERTEX-COVER

Claim: If $\phi$ is satisfiable, then $\exists$ a vertex cover of size $k = n + 2m$

- If variable $x_i = 0$, pick $v_i^0$ to be in the vertex cover
- If variable $x_i = 1$, pick $v_i^1$ to be in the vertex cover
- Each clause $C_j$ has at least one literal, say $x_{j}^{2}$, set to $true$.
  - Don’t pick edge vertex of $x_{j}^{2}$, pick other two edge vertices

Claim: The picked vertices cover all the edges of $G$. 
Reduction for VERTEX-COVER

Claim: If $\phi$ is satisfiable, then $\exists$ a vertex cover of size $k = n + 2m$

- If variable $x_i = 0$, pick $v_i^0$ to be in the vertex cover
- If variable $x_i = 1$, pick $v_i^1$ to be in the vertex cover
- Each clause $C_j$ has at least one literal, say $x_j^2$, set to true.
  - Don’t pick edge vertex of $x_j^i$, pick other two edge vertices

Claim: The picked vertices cover all the edges of $G$.

- Clearly, the number of vertices picked is $n + 2m$. 
Reduction for VERTEX-COVER

Claim: If $\phi$ is satisfiable, then $\exists$ a vertex cover of size $k = n + 2m$

- If variable $x_i = 0$, pick $v_i^0$ to be in the vertex cover
- If variable $x_i = 1$, pick $v_i^1$ to be in the vertex cover
- Each clause $C_j$ has at least one literal, say $x_j^2$, set to $true$.
  - Don’t pick edge vertex of $x_j^i$, pick other two edge vertices

Claim: The picked vertices cover all the edges of $G$.

- Clearly, the number of vertices picked is $n + 2m$.
- An edge between two variable vertices is covered
Reduction for VERTEX-COVER

Claim: If $\phi$ is satisfiable, then $\exists$ a vertex cover of size $k = n + 2m$

- If variable $x_i = 0$, pick $v_i^0$ to be in the vertex cover
- If variable $x_i = 1$, pick $v_i^1$ to be in the vertex cover
- Each clause $C_j$ has at least one literal, say $x_{ij}^j$, set to $true$.
  - Don’t pick edge vertex of $x_{ij}^j$, pick other two edge vertices

Claim: The picked vertices cover all the edges of $G$.

- Clearly, the number of vertices picked is $n + 2m$.
- An edge between two variable vertices is covered
  - For each $i$, exactly one of the variable vertices picked.
Reduction for VERTEX-COVER

Claim: If $\phi$ is satisfiable, then $\exists$ a vertex cover of size $k = n + 2m$

- If variable $x_i = 0$, pick $v_i^0$ to be in the vertex cover
- If variable $x_i = 1$, pick $v_i^1$ to be in the vertex cover
- Each clause $C_j$ has at least one literal, say $x_{2j}^i$, set to $true$.
  - Don’t pick edge vertex of $x_{2j}^i$, pick other two edge vertices

Claim: The picked vertices cover all the edges of $G$.

- Clearly, the number of vertices picked is $n + 2m$.
- An edge between two variable vertices is covered
  - For each $i$, exactly one of the variable vertices picked.
- For each $C_j$, edges to the picked vertices covered.
Reduction for VERTEX-COVER

Claim: If $\phi$ is satisfiable, then $\exists$ a vertex cover of size $k = n + 2m$

- If variable $x_i = 0$, pick $v_i^0$ to be in the vertex cover
- If variable $x_i = 1$, pick $v_i^1$ to be in the vertex cover
- Each clause $C_j$ has at least one literal, say $x_{2j}^j$, set to true.
  - Don’t pick edge vertex of $x_{2j}^j$, pick other two edge vertices

Claim: The picked vertices cover all the edges of $G$.

- Clearly, the number of vertices picked is $n + 2m$.
- An edge between two variable vertices is covered
  - For each $i$, exactly one of the variable vertices picked.
- For each $C_j$, edges to the picked vertices covered.

Crucial: What about the edges incident upon $x_{2j}^j$?
Reduction for VERTEX-COVER

Claim: If $\phi$ is satisfiable, then $\exists$ a vertex cover of size $k = n + 2m$

- If variable $x_i = 0$, pick $v_i^0$ to be in the vertex cover
- If variable $x_i = 1$, pick $v_i^1$ to be in the vertex cover
- Each clause $C_j$ has at least one literal, say $x_{2j}^j$, set to true.
  - Don’t pick edge vertex of $x_{2j}^j$, pick other two edge vertices

Claim: The picked vertices cover all the edges of $G$.

- Clearly, the number of vertices picked is $n + 2m$.
- An edge between two variable vertices is covered
  - For each $i$, exactly one of the variable vertices picked.
- For each $C_j$, edges to the picked vertices covered.
- Crucial: What about the edges incident upon $x_{2j}^j$?
  - Edges within its clause covered by the other two picked vertices
Reduction for VERTEX–COVER

Claim: If \( \phi \) is satisfiable, then \( \exists \) a vertex cover of size \( k = n + 2m \)

- If variable \( x_i = 0 \), pick \( v_i^0 \) to be in the vertex cover
- If variable \( x_i = 1 \), pick \( v_i^1 \) to be in the vertex cover
- Each clause \( C_j \) has at least one literal, say \( x_i^j \), set to \( true \).
  - Don’t pick edge vertex of \( x_i^j \), pick other two edge vertices

Claim: The picked vertices cover all the edges of \( G \).

- Clearly, the number of vertices picked is \( n + 2m \).
- An edge between two variable vertices is covered
  - For each \( i \), exactly one of the variable vertices picked.
- For each \( C_j \), edges to the picked vertices covered.
- Crucial: What about the edges incident upon \( x_2^j \)?
  - Edges within its clause covered by the other two picked vertices
  - Remaining edge of the form: \( (x_i^j, v_i^1) \) if \( x_i^j \) is un-negated.
Reduction for VERTEX-COVER

Claim: If $\phi$ is satisfiable, then $\exists$ a vertex cover of size $k = n + 2m$

- If variable $x_i = 0$, pick $v_i^0$ to be in the vertex cover
- If variable $x_i = 1$, pick $v_i^1$ to be in the vertex cover
- Each clause $C_j$ has at least one literal, say $x_{2j}^j$, set to $true$.
  - Don’t pick edge vertex of $x_{2j}^j$, pick other two edge vertices

Claim: The picked vertices cover all the edges of $G$.

- Clearly, the number of vertices picked is $n + 2m$.
- An edge between two variable vertices is covered
  - For each $i$, exactly one of the variable vertices picked.
- For each $C_j$, edges to the picked vertices covered.

Crucial: What about the edges incident upon $x_{2j}^j$?

- Edges within its clause covered by the other two picked vertices
- Remaining edge of the form: $(x_{2j}^j, v_i^1)$ if $x_{2j}^j$ is un-negated.
- We know that $v_i^1$ is picked, since $v_i = 1$.  
  

Reduction for VERTEX-COVER

Claim: If \( \exists \) a vertex cover of size \( k = n + 2m \), then \( \phi \) satisfiable
Reduction for VERTEX-COVER

Claim: If $\exists$ a vertex cover of size $k = n + 2m$, then $\phi$ satisfiable

- For each $i$, at least one of the variable vertices in vertex cover.

In each set $V(C_j)$, at least 2 vertices in vertex cover. This already makes $n + 2m$ vertices!

Exactly one vertex not picked in each clause

Consider the clause $(x_1 \lor x_2 \lor x_3)$

Let's say $x_1$ and $x_2$ were picked in vertex cover

The edge $(x_3, v_3)$ must be covered

Therefore $v_3$ must be in the vertex cover.

Then set the boolean variable $x_3 = 0$

Similarly, whichever vertex not picked in cover, set corresponding literal to 1

Thus, setting variables as above, each clause satisfied
Reduction for VERTEX-COVER

Claim: If \( \exists \) a vertex cover of size \( k = n + 2m \), then \( \phi \) satisfiable

- For each \( i \), at least one of the variable vertices in vertex cover.
- In each set \( V(C_j) \), at least 2 vertices in vertex cover
Reduction for VERTEX-COVER

Claim: If $\exists$ a vertex cover of size $k = n + 2m$, then $\phi$ satisfiable

- For each $i$, at least one of the variable vertices in vertex cover.
- In each set $V(C_j)$, at least 2 vertices in vertex cover
- This already makes $n + 2m$ vertices!

Consider the clause $(x_1 \lor x_2 \lor x_3)$

Let's say $x_1$ and $x_2$ were picked in vertex cover

The edge $(x_3, v_3)$ must be covered

Therefore $v_3$ must be in the vertex cover.

Then set the boolean variable $x_3 = 0$

Similarly, whichever vertex not picked in cover, set corresponding literal to 1

Thus, setting variables as above, each clause satisfied
Reduction for VERTEX-COVER

Claim: If $\exists$ a vertex cover of size $k = n + 2m$, then $\phi$ satisfiable
- For each $i$, at least one of the variable vertices in vertex cover.
- In each set $V(C_j)$, at least 2 vertices in vertex cover.
- This already makes $n + 2m$ vertices!
- Exactly one vertex not picked in each clause.
Reduction for VERTEX-COVER

Claim: If $\exists$ a vertex cover of size $k = n + 2m$, then $\phi$ satisfiable

- For each $i$, at least one of the variable vertices in vertex cover.
- In each set $V(C_j)$, at least 2 vertices in vertex cover.
- This already makes $n + 2m$ vertices!
- Exactly one vertex *not* picked in each clause.
- Consider the clause $(x_1 \lor x_2 \lor \overline{x}_3)$
Reduction for VERTEX-COVER

Claim: If $\exists$ a vertex cover of size $k = n + 2m$, then $\phi$ satisfiable

- For each $i$, at least one of the variable vertices in vertex cover.
- In each set $V(C_j)$, at least 2 vertices in vertex cover.
- This already makes $n + 2m$ vertices!
- Exactly one vertex not picked in each clause.
- Consider the clause $(x_1 \lor x_2 \lor \overline{x_3})$
- Lets say $x_1$ and $x_2$ were picked in vertex cover.

Therefore, $v_3$ must be in the vertex cover.

Then set the boolean variable $x_3 = 0$.

Similarly, whichever vertex not picked in cover, set corresponding literal to 1.

Thus, setting variables as above, each clause satisfied.
Reduction for \textsc{VERTEX-COVER}

**Claim:** If \( \exists \) a vertex cover of size \( k = n + 2m \), then \( \phi \) satisfiable

- For each \( i \), at least one of the variable vertices in vertex cover.
- In each set \( V(C_j) \), at least 2 vertices in vertex cover
- This already makes \( n + 2m \) vertices!
- Exactly one vertex \textit{not} picked in each clause
- Consider the clause \((x_1 \lor x_2 \lor \overline{x}_3)\)
- Lets say \( x_1 \) and \( x_2 \) were picked in vertex cover
- The edge \((\overline{x}_3, \overline{v}_3)\) must be covered
Reduction for VERTEX-COVER

Claim: If $\exists$ a vertex cover of size $k = n + 2m$, then $\phi$ satisfiable

- For each $i$, at least one of the variable vertices in vertex cover.
- In each set $V(C_j)$, at least 2 vertices in vertex cover
- This already makes $n + 2m$ vertices!
- Exactly one vertex not picked in each clause
- Consider the clause $(x_1 \lor x_2 \lor \overline{x}_3)$
- Lets say $x_1$ and $x_2$ were picked in vertex cover
- The edge $(\overline{x}_3, \overline{v}_3)$ must be covered
- Therefore $\overline{v}_3$ must be in the vertex cover.
Reduction for VERTEX-COVER

Claim: If $\exists$ a vertex cover of size $k = n + 2m$, then $\phi$ satisfiable

- For each $i$, at least one of the variable vertices in vertex cover.
- In each set $V(C_j)$, at least 2 vertices in vertex cover.
- This already makes $n + 2m$ vertices!
- Exactly one vertex not picked in each clause.
- Consider the clause $(x_1 \lor x_2 \lor \overline{x_3})$
- Let's say $x_1$ and $x_2$ were picked in vertex cover.
- The edge $(\overline{x_3}, \overline{v_3})$ must be covered.
- Therefore $\overline{v_3}$ must be in the vertex cover.
- Then set the boolean variable $x_3 = 0$
Reduction for VERTEX-COVER

Claim: If $\exists$ a vertex cover of size $k = n + 2m$, then $\phi$ satisfiable

- For each $i$, at least one of the variable vertices in vertex cover.
- In each set $V(C_j)$, at least 2 vertices in vertex cover
- This already makes $n + 2m$ vertices!
- Exactly one vertex not picked in each clause
- Consider the clause $(x_1 \lor x_2 \lor \overline{x}_3)$
- Lets say $x_1$ and $x_2$ were picked in vertex cover
- The edge $(\overline{x}_3, \overline{v}_3)$ must be covered
- Therefore $\overline{v}_3$ must be in the vertex cover.
- Then set the boolean variable $x_3 = 0$
- Similarly, whichever vertex not picked in cover, set corresponding literal to 1
Reduction for VERTEX-COVER

Claim: If $\exists$ a vertex cover of size $k = n + 2m$, then $\phi$ satisfiable

- For each $i$, at least one of the variable vertices in vertex cover.
- In each set $V(C_j)$, at least 2 vertices in vertex cover.
- This already makes $n + 2m$ vertices!
- Exactly one vertex not picked in each clause.
- Consider the clause $(x_1 \lor x_2 \lor \overline{x}_3)$
- Let's say $x_1$ and $x_2$ were picked in vertex cover.
- The edge $(\overline{x}_3, \overline{v}_3)$ must be covered.
- Therefore $\overline{v}_3$ must be in the vertex cover.
- Then set the boolean variable $x_3 = 0$.
- Similarly, whichever vertex not picked in cover, set corresponding literal to 1.
- Thus, setting variables as above, each clause satisfied.
Another reduction for VERTEX-COVER

Claim

$V'$ is a vertex cover of size $k$ in $G = (V, E)$ iff $V \setminus V'$ of size $(n - k)$ is an independent set.
Another reduction for VERTEX–COVER

Claim

$V'$ is a vertex cover of size $k$ in $G = (V, E)$ iff $V \setminus V'$ of size $(n - k)$ is an independent set.

- Let $I = \{v_1, v_2, \ldots, v_{n-k}\}$ be INDSET of size $n - k$
Another reduction for VERTEX-COVER

**Claim**

$V'$ is a vertex cover of size $k$ in $G = (V, E)$ iff $V \setminus V'$ of size $(n - k)$ is an independent set.

- Let $I = \{v_1, v_2, \ldots, v_{n-k}\}$ be INDSET of size $n - k$
  - There is no edge in $G$ with both endpoints in $I$
Another reduction for VERTEX-COVER

Claim
$V'$ is a vertex cover of size $k$ in $G = (V, E)$ iff $V \setminus V'$ of size $(n - k)$ is an independent set.

- Let $I = \{v_1, v_2, \ldots, v_{n-k}\}$ be INDSET of size $n - k$
  - There is no edge in $G$ with both endpoints in $I$
  - So if we pick the remaining $k$ vertices, all edges covered
Another reduction for VERTEX-COVER

**Claim**

$V'$ is a vertex cover of size $k$ in $G = (V, E)$ iff $V \setminus V'$ of size $(n - k)$ is an independent set.

- Let $I = \{v_1, v_2, \ldots, v_{n-k}\}$ be INDSET of size $n - k$
  - There is no edge in $G$ with *both* endpoints in $I$
  - So if we pick the remaining $k$ vertices, all edges covered

- $V' = \{v_1, v_2, \ldots, v_k\}$ is a vertex cover of size $k$
Another reduction for VERTEX-COVER

Claim

$V'$ is a vertex cover of size $k$ in $G = (V, E)$ iff $V \setminus V'$ of size $(n - k)$ is an independent set.

- Let $I = \{v_1, v_2, \ldots, v_{n-k}\}$ be INDSET of size $n - k$
  - There is no edge in $G$ with both endpoints in $I$
  - So if we pick the remaining $k$ vertices, all edges covered

- $V' = \{v_1, v_2, \ldots, v_k\}$ is a vertex cover of size $k$
  - Each edge in $G$ has at least one endpoint in $V'$
Another reduction for VERTEX-COVER

Claim

$V'$ is a vertex cover of size $k$ in $G = (V, E)$ iff $V \setminus V'$ of size $(n - k)$ is an independent set.

- Let $I = \{v_1, v_2, \ldots, v_{n-k}\}$ be INDSET of size $n - k$
  - There is no edge in $G$ with both endpoints in $I$
  - So if we pick the remaining $k$ vertices, all edges covered

- $V' = \{v_1, v_2, \ldots, v_k\}$ is a vertex cover of size $k$
  - Each edge in $G$ has at least one endpoint in $V'$
  - So the remaining $n - k$ vertices form an independent set
HITTING-SET

Claim

**HITTING-SET**: A collection \( C \) of subsets of a set \( V \), and parameter \( k \), find a hitting set \( V' \subseteq V \) of size \( k \).

The hitting set problem (**HITTING-SET**) is \( \text{NP} \) complete.
**Claim**

**HITTING-SET**: A collection $C$ of subsets of a set $V$, and parameter $k$, find a hitting set $V' \subseteq V$ of size $k$.

The hitting set problem (HITTING-SET) is $\textbf{NP}$ complete.
HITTING-SET

Claim

**HITTING-SET**: A collection $C$ of subsets of a set $V$, and parameter $k$, find a hitting set $V' \subseteq V$ of size $k$.

*The hitting set problem (HITTING-SET) is NP complete.*

Reduction from?
HITTING-SET

Claim

**HITTING-SET** : A collection $C$ of subsets of a set $V$, and parameter $k$, find a hitting set $V' \subseteq V$ of size $k$.

The hitting set problem (**HITTING-SET**) is **NP** complete.

Reduction from? Consider each set having size two!
Claim

**INTEGER-PROGRAMMING**: Given a set of linear inequalities over variables $v_1, \ldots, v_n$, does there exist an satisfying assignment of $v_i$ to positive integers.

The integer programming problem is **NP complete**.
**Claim**

**INTEGER-PROGRAMMING** : Given a set of linear inequalities over variables $v_1, \ldots, v_n$, does there exist an satisfying assignment of $v_i$ to positive integers

The integer programming problem is **NP** complete.

\[ u_1 + 4u_2 - 32u_3 \geq 34 \]
**Claim**

*INTEGER-PROGRAMMING*: Given a set of linear inequalities over variables \( v_1, \ldots, v_n \), does there exist an satisfying assignment of \( v_i \) to positive integers

The integer programming problem is **NP** complete.

\[
\begin{align*}
  u_1 + 4u_2 - 32u_3 & \geq 34 \\
  2u_2 - 2u_4 + 7u_3 & \leq 239
\end{align*}
\]
INTEGER-PROGRAMMING

Claim

**INTEGER-PROGRAMMING**: Given a set of linear inequalities over variables $v_1, \ldots, v_n$, does there exist an satisfying assignment of $v_i$ to positive integers

The integer programming problem is $\textbf{NP}$ complete.

\[ u_1 + 4u_2 - 32u_3 \geq 34 \]
\[ 2u_2 - 2u_4 + 7u_3 \leq 239 \]
\[ 43u_4 - 2u_1 + 17u_2 \geq 17 \]
Claim

**INTEGER-PROGRAMMING** : Given a set of linear inequalities over variables $v_1, \ldots, v_n$, does there exist an satisfying assignment of $v_i$ to positive integers

The integer programming problem is **NP** complete.

\[
\begin{align*}
  u_1 + 4u_2 - 32u_3 & \geq 34 \\
  2u_2 - 2u_4 + 7u_3 & \leq 239 \\
  43u_4 - 2u_1 + 17u_2 & \geq 17
\end{align*}
\]

Note that **INTEGER-PROGRAMMING** is in **NP**
Claim

**INTEGER-PROGRAMMING**: Given a set of linear inequalities over variables $v_1, \ldots, v_n$, does there exist an satisfying assignment of $v_i$ to positive integers?

The integer programming problem is **NP** complete.

\[
\begin{align*}
    u_1 + 4u_2 - 32u_3 & \geq 34 \\
    2u_2 - 2u_4 + 7u_3 & \leq 239 \\
    43u_4 - 2u_1 + 17u_2 & \geq 17
\end{align*}
\]

- Note that **INTEGER-PROGRAMMING** is in **NP**
- Reduction from 3-CNF SAT
Reduction for INTEGER-PROGRAMMING

\[ \phi = C_1 \land C_2 \land \ldots \land C_m, \text{ where } C_i = (x_1^i \lor x_2^i \lor x_3^i) \]
Reduction for INTEGER-PROGRAMMING

\[ \phi = C_1 \land C_2 \land \ldots \land C_m, \text{ where } C_i = (x_1^i \lor x_2^i \lor x_3^i) \]

Given \( \phi \), construct the set of inequalities as following:
Reduction for INTEGER-PROGRAMMING

\[ \phi = C_1 \land C_2 \land \ldots \land C_m, \text{ where } C_i = (x_1^i \lor x_2^i \lor x_3^i) \]

Given \( \phi \), construct the set of inequalities as following:

- Each boolean variable \( x_i \) becomes integer variable \( v_i \)
Reduction for INTEGER-PROGRAMMING

\[ \phi = C_1 \land C_2 \land \ldots \land C_m, \text{ where } C_i = (x_1^i \lor x_2^i \lor x_3^i) \]

Given \( \phi \), construct the set of inequalities as following:

- Each boolean variable \( x_i \) becomes integer variable \( v_i \)
- Each clause becomes an equation set to be greater than 1:
Reduction for INTEGER-PROGRAMMING

\[ \phi = C_1 \land C_2 \land \ldots \land C_m, \text{ where } C_i = (x^i_1 \lor x^i_2 \lor x^i_3) \]

Given \( \phi \), construct the set of inequalities as following:

- Each boolean variable \( x^i \) becomes integer variable \( v^i \)
- Each clause becomes an equation set to be greater than 1:
  \[ x^i \rightarrow v^i, \quad \bar{x}^i \rightarrow (1 - v^i), \quad \lor \rightarrow + \]
Reduction for INTEGER-PROGRAMMING

\[ \phi = C_1 \land C_2 \land \ldots \land C_m, \text{ where } C_i = (x_i^1 \lor x_i^2 \lor x_i^3) \]

- Given \( \phi \), construct the set of inequalities as following:
  - Each boolean variable \( x_i \) becomes integer variable \( v_i \)
  - Each clause becomes an equation set to be greater than 1:
    \[ x_i \rightarrow v_i, \quad \overline{x_i} \rightarrow (1 - v_i), \quad \lor \rightarrow + \]
    \[ (x_1 \lor x_2 \lor \overline{x_3}) \]
Reduction for INTEGER-PROGRAMMING

\[ \phi = C_1 \land C_2 \land \ldots \land C_m, \text{ where } C_i = (x^i_1 \lor x^i_2 \lor x^i_3) \]

Given \( \phi \), construct the set of inequalities as following:
- Each boolean variable \( x_i \) becomes integer variable \( v_i \)
- Each clause becomes an equation set to be greater than 1:
  \[ x_i \rightarrow v_i, \quad \overline{x}_i \rightarrow (1 - v_i), \quad \lor \rightarrow + \]

\[ (x_1 \lor x_2 \lor \overline{x}_3) \]
\[ v_1 + v_2 + (1 - v_3) \geq 1 \]
Reduction for INTEGER-PROGRAMMING

\[ \phi = C_1 \land C_2 \land \ldots \land C_m, \text{ where } C_i = (x_1^i \lor x_2^i \lor x_3^i) \]

- Given \( \phi \), construct the set of inequalities as following:
  - Each boolean variable \( x_i \) becomes integer variable \( v_i \)
  - Each clause becomes an equation set to be greater than 1:
    \[ x_i \rightarrow v_i, \quad \overline{x}_i \rightarrow (1 - v_i), \quad \lor \rightarrow + \]

\[ (x_1 \lor x_2 \lor \overline{x}_3) \land (\overline{x}_1 \lor x_3 \lor \overline{x}_4) \]
Reduction for INTEGER-PROGRAMMING

\[ \phi = C_1 \land C_2 \land \ldots \land C_m, \text{ where } C_i = (x_1^i \lor x_2^i \lor x_3^i) \]

**Given** \( \phi \), **construct the set of inequalities as following:**

- Each boolean variable \( x_i \) becomes integer variable \( v_i \)
- Each clause becomes an equation set to be greater than 1:
  \[ x_i \rightarrow v_i, \quad \overline{x}_i \rightarrow (1 - v_i), \quad \lor \rightarrow + \]

\[ (x_1 \lor x_2 \lor \overline{x}_3) \land (\overline{x}_1 \lor x_3 \lor \overline{x}_4) \]

\[ v_1 + v_2 + (1 - v_3) \geq 1, \quad (1 - v_1) + v_3 + (1 - v_4) \geq 1 \]
Reduction for INTEGER-PROGRAMMING

\[ \phi = C_1 \land C_2 \land \ldots \land C_m, \text{ where } C_i = (x_1^i \lor x_2^i \lor x_3^i) \]

- Given \( \phi \), construct the set of inequalities as following:
  - Each boolean variable \( x_i \) becomes integer variable \( v_i \)
  - Each clause becomes an equation set to be greater than 1:
    \[ x_i \rightarrow v_i, \quad \overline{x}_i \rightarrow (1 - v_i), \quad \lor \rightarrow + \]

\[ (x_1 \lor x_2 \lor \overline{x}_3) \land (\overline{x}_1 \lor x_3 \lor \overline{x}_4) \land (\overline{x}_2 \lor x_4 \lor x_1) \]
Reduction for INTEGER-PROGRAMMING

\[ \phi = C_1 \land C_2 \land \ldots \land C_m, \text{ where } C_i = (x_1^i \lor x_2^i \lor x_3^i) \]

- Given \( \phi \), construct the set of inequalities as following:
  - Each boolean variable \( x_i \) becomes integer variable \( v_i \)
  - Each clause becomes an equation set to be greater than 1:
    \[ x_i \to v_i, \quad \overline{x_i} \to (1 - v_i), \quad \lor \to + \]

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

\[ v_1 + v_2 + (1 - v_3) \geq 1, \quad (1 - v_1) + v_3 + (1 - v_4) \geq 1, \quad (1 - v_2) + v_4 + v_1 \geq 1 \]
Reduction for INTEGER-PROGRAMMING

\[ \phi = C_1 \land C_2 \land \ldots \land C_m, \text{ where } C_i = (x^i_1 \lor x^i_2 \lor x^i_3) \]

Given \( \phi \), construct the set of inequalities as following:

- Each boolean variable \( x_i \) becomes integer variable \( v_i \)
- Each clause becomes an equation set to be greater than \( 1 \):
  \[ x_i \rightarrow v_i, \quad \overline{x}_i \rightarrow (1 - v_i), \quad \lor \rightarrow + \]

\[
(x_1 \lor x_2 \lor \overline{x}_3) \land (\overline{x}_1 \lor x_3 \lor \overline{x}_4) \land (\overline{x}_2 \lor x_4 \lor x_1) \\
\]
\[
v_1 + v_2 + (1 - v_3) \geq 1, \quad (1 - v_1) + v_3 + (1 - v_4) \geq 1, \quad (1 - v_2) + v_4 + v_1 \geq 1
\]
Reduction for INTEGER-PROGRAMMING

\[ \phi = C_1 \land C_2 \land \ldots \land C_m, \text{ where } C_i = (x_i^1 \lor x_i^2 \lor x_i^3) \]

Given \( \phi \), construct the set of inequalities as following:

- Each boolean variable \( x_i \) becomes integer variable \( v_i \)
- Each clause becomes an equation set to be greater than 1:
  \[
  x_i \rightarrow v_i, \quad \overline{x}_i \rightarrow (1 - v_i), \quad \lor \rightarrow +
  \]

\[
(x_1 \lor x_2 \lor \overline{x}_3) \land (\overline{x}_1 \lor x_3 \lor \overline{x}_4) \land (\overline{x}_2 \lor x_4 \lor x_1) \\
v_1 + v_2 + (1 - v_3) \geq 1, \ (1 - v_1) + v_3 + (1 - v_4) \geq 1, \ (1 - v_2) + v_4 + v_1 \geq 1
\]

\[
0 \leq v_1 \leq 1, \quad 0 \leq v_2 \leq 1, \quad 0 \leq v_3 \leq 1, \quad 0 \leq v_4 \leq 1
\]
Claim: Solve INTEGER-PROGRAMMING to get a valid solution. Set boolean variable $x_i = v_i$. This satisfies $\phi$. 
Reduction for INTEGER-PROGRAMMING

**Claim:** Solve INTEGER-PROGRAMMING to get a valid solution. Set boolean variable \( x_i = v_i \). This satisfies \( \phi \).

- Each \( v_i \) can be either 0 or 1
Reduction for INTEGER-PROGRAMMING

Claim: Solve INTEGER-PROGRAMMING to get a valid solution. Set boolean variable $x_i = v_i$. This satisfies $\phi$.

- Each $v_i$ can be either 0 or 1
- A clause $C_j$ satisfied $\iff$ corresponding inequality satisfied
  
  $$(x_1 \lor x_2 \lor \overline{x_3}) = 1 \iff v_1 + v_2 + (1 - v_3) \geq 1$$
Reduction for INTEGER-PROGRAMMING

Claim: Solve INTEGER-PROGRAMMING to get a valid solution. Set boolean variable $x_i = v_i$. This satisfies $\phi$.

- Each $v_i$ can be either 0 or 1
- A clause $C_j$ satisfied $\iff$ corresponding inequality satisfied

\[ (x_1 \lor x_2 \lor \overline{x_3}) = 1 \iff v_1 + v_2 + (1 - v_3) \geq 1 \]

- Likewise in the other direction, set $v_i = x_i$. 